## Contents

*About This Book* ............................................................... xiii  
*What's New in SAS/GRAPH 9.4: Reference* ............................ xvii

### PART 1  Getting Started  1

**Chapter 1 • Get Started Using SAS/GRAPH** .......................... 3  
What is SAS/GRAPH? ...................................................... 3  
Components of SAS/GRAPH Software ................................... 4  
What You Can Create Using SAS/GRAPH .............................. 5  
What You Need to Know to Get Started Using SAS/GRAPH ............ 6  
Learning By Example: Create Your First Chart Using SAS/GRAPH ... 9  
Where to Go from Here .................................................. 12

**Chapter 2 • Gallery of SAS/GRAPH Plots and Charts** ............... 13  
A Quick Look at the Gallery ............................................ 13  
Gallery of Basic Charts and Plots ...................................... 14  
Gallery of Additional Graph Features .................................. 27

**Chapter 3 • Common Tasks Associated with Developing SAS/GRAPH Programs** .......................... 31  
About the SAS/GRAPH Tasks ............................................ 31  
Output Environment Tasks ............................................... 31  
Graph Creation Tasks ................................................... 32  
Graph Enhancement Tasks ............................................... 33

**Chapter 4 • Additional Resources to Help You Develop Your SAS/GRAPH Programs** .................. 35  
SAS/GRAPH Sample Programs ........................................... 35  
Examples and Resources on the Web ................................... 37  
Books ............................................................................. 38

### PART 2  SAS/GRAPH Concepts  39

**Chapter 5 • Elements of a SAS/GRAPH Program** .................... 41  
Overview .......................................................................... 41  
A Typical SAS/GRAPH Program ....................................... 41  
SAS/GRAPH PROC Step .................................................. 42  
Global Statements .......................................................... 43  
Annotate DATA Set ......................................................... 44  
ODS Statements ............................................................. 45  
Base SAS Language Elements ......................................... 46  
Other Resources ............................................................ 48

**Chapter 6 • Overview of Devices, ODS Destinations, and ODS Styles** .................................. 49  
Introduction to Devices, ODS Destinations, and ODS Styles .......... 49  
Working with ODS Destinations and Styles ............................ 50  
Generating Output with SAS/GRAPH Procedures ..................... 53
Contents

Chapter 7 • Using Run-Group Processing ........................................ 67
Using Run-Group Processing ........................................ 67
RUN-group Processing with Global and Local Statements ............... 67
RUN-group Processing with BY Statements ................................ 68
RUN-group Processing with the WHERE Statement ....................... 68

Chapter 8 • The Graphics Output Environment .............................. 69
Overview ........................................ 69
The Graphics Output and Device Display Areas ......................... 70
Controlling Dimensions ........................................ 70
Controlling Display Area Size and Image Resolution ................... 72
Units ........................................ 72
Maintaining the Quality of Your Image across Devices ................. 75
How Graphics Elements Are Placed in the Graphics Output Area .......... 76
How Errors in Sizing Are Handled ......................................... 77

Chapter 9 • Using Graphics Devices ........................................... 79
Overview: Using Graphics Devices ........................................ 79
What Is a SAS/GRAPH Device? ........................................ 80
Commonly Used Devices ........................................ 80
Default Devices for ODS Destinations ................................... 82
Viewing the List of All Available Devices .............................. 83
Displaying Information with the QDEVICE Procedure ................ 84
Deciding Which Device to Use ........................................ 85
Overriding the Default Device ........................................ 86
Device Categories and Modifying Default Output Attributes ........... 87
Using Universal Printer Shortcut Devices ............................... 89
Creating Universal Printers and Shortcut Devices ...................... 91
Viewing and Modifying Device (DEV) Entries ........................... 91
Creating a Custom Device ........................................ 93
Compatibility Device Drivers (Z Drivers) ................................ 93
Additional Considerations for Devices .................................. 93
Related Topics ........................................ 94
Dictionary ........................................ 94

Chapter 10 • Managing Your Graphics with ODS............................ 97
Introduction ........................................ 97
Managing ODS Destinations ........................................ 97
Specifying a Destination ........................................ 98
ODS Destination Statement Options ................................... 99
ODS and Procedures That Support RUN-Group Processing ............ 100
Controlling Titles and Footnotes ......................................... 101

Chapter 11 • SAS/GRAPH Output ............................................ 105
Overview of SAS/GRAPH Output ....................................... 106
Specifying the Graphics Output File Type for Your Graph ............. 109
The SAS/GRAPH Output Process .................................... 112
Setting the Size of Your Graph ....................................... 113
Setting the Resolution of Your Graph .................................. 114
Controlling Where Your Output Is Stored ................................ 116
<table>
<thead>
<tr>
<th>Chapter 12 • Plotting a Cloud Analytic Services (CAS) In-Memory Table</th>
<th>135</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing In-Memory Tables</td>
<td>135</td>
</tr>
<tr>
<td>Preprocessing Data in CAS and Plotting the Results Using SAS/GRAPH</td>
<td>135</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 3 Generating Output with SAS/GRAPH</th>
<th>139</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 13 • Using SVG Graphics</td>
<td>141</td>
</tr>
<tr>
<td>Introduction to Scalable Vector Graphics</td>
<td>141</td>
</tr>
<tr>
<td>Advantages of SVG Graphics</td>
<td>142</td>
</tr>
<tr>
<td>Universal Printers versus SAS/GRAPH Devices</td>
<td>142</td>
</tr>
<tr>
<td>The SVG Devices and the Output That They Create</td>
<td>142</td>
</tr>
<tr>
<td>Example: Creating an SVG File</td>
<td>143</td>
</tr>
<tr>
<td>Example: Generating Stand-Alone SVG Files in HTML5</td>
<td>144</td>
</tr>
<tr>
<td>Example: Placing Images behind SVG Files</td>
<td>145</td>
</tr>
<tr>
<td>Example: Generating a Single SVG File with Multiple Pages and Page Controls</td>
<td>147</td>
</tr>
<tr>
<td>Web Server Content Type for SVG Files</td>
<td>149</td>
</tr>
<tr>
<td>Height, Width, and Scaling of SVG Files</td>
<td>149</td>
</tr>
<tr>
<td>Browser Support for Viewing SVG Files</td>
<td>149</td>
</tr>
<tr>
<td>Controlling Graph Resolution with the SVG Devices</td>
<td>151</td>
</tr>
<tr>
<td>Controlling Graph Size with the SVG Devices</td>
<td>152</td>
</tr>
<tr>
<td>SAS System Options and SVG Output</td>
<td>152</td>
</tr>
<tr>
<td>Overview of Animating Output with the SVG Device</td>
<td>152</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 14 • Generating SVG, PNG, GIF, and TIFF Graphics</th>
<th>153</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is a Non-Interactive Graph?</td>
<td>153</td>
</tr>
<tr>
<td>Creating a Non-Interactive Graph</td>
<td>154</td>
</tr>
<tr>
<td>SVG, PNG, GIF, and TIFF Devices Compared to ACTXIMG and JAVAIMG Devices</td>
<td>156</td>
</tr>
<tr>
<td>Developing Web Presentations with the PNG, SVG, and GIF Devices</td>
<td>156</td>
</tr>
<tr>
<td>Developing Web Presentations with the JAVAIMG and ACTXIMG Devices</td>
<td>157</td>
</tr>
<tr>
<td>Adding Drill-Down Links to Web Presentations Generated with a Static-Graphic Device</td>
<td>159</td>
</tr>
<tr>
<td>Implementing Data Tips and Drill-Down Links with the SVG Devices</td>
<td>159</td>
</tr>
<tr>
<td>Examples: Generating Static Graphics</td>
<td>159</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 15 • Generating Animations</th>
<th>167</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animating Images in SAS/GRAPH</td>
<td>167</td>
</tr>
<tr>
<td>Controlling Animations with System Options</td>
<td>171</td>
</tr>
<tr>
<td>Animating Annotation Text Labels</td>
<td>173</td>
</tr>
<tr>
<td>Examples: Generating Animated Images</td>
<td>176</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16 • Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality</th>
<th>187</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Enhancing Web Presentations</td>
<td>188</td>
</tr>
<tr>
<td>Chart Descriptions for Web Presentations</td>
<td>189</td>
</tr>
<tr>
<td>Data Tips for Web Presentations</td>
<td>191</td>
</tr>
</tbody>
</table>
Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options ........................................... 192
Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes .................................................. 198
Generating Drill-Down Links with the Annotate Facility ................................................................. 201
Examples ............................................................................................................................................. 202

Chapter 17 • Using SAS/GRAPH Output with Microsoft Office Products .................................................. 233
What to Consider When Choosing an Output Format ........................................................................ 233
Comparison of the Graphics Output ................................................................................................. 235
Enhancing Your Graphs .................................................................................................................. 238
Importing Your Graphs into Microsoft Office .................................................................................... 239

Chapter 18 • Writing Your Graphs to a PDF File ................................................................................. 241
Overview of Writing Graphs to a PDF File ........................................................................................ 241
Specifying Fonts ............................................................................................................................ 242
Creating an Archivable PDF File ....................................................................................................... 243
Changing the Page Layout .............................................................................................................. 244
Adding Metadata to Your PDF File ................................................................................................. 244
Adding Bookmarks for Your Graphs ............................................................................................... 244
Adding Drill-Down Graphs in Your PDF File .................................................................................... 245
Changing the Default Compression Level for Your PDF File .......................................................... 246
Examples: Writing Graphs to a PDF File ........................................................................................................ 246

Chapter 19 • Troubleshooting Web Output ......................................................................................... 263
Overview ............................................................................................................................................. 263
Solutions to Common Rendering Problems ...................................................................................... 263
Checking Browser Permissions ........................................................................................................ 267
Using HTML Character Entities .................................................................................................... 267
Connecting to Web Servers That Require Authentication ............................................................... 267
Removing CLASSPATH Environment Variables .............................................................................. 267
Setting the SAS_ALT_DISPLAY Variable for X Window Systems on UNIX .................................. 268
Correcting Text Fonts ........................................................................................................................ 268
Resolving Differences Fonts............................................................................................................. 268

PART 4  Controlling the Appearance of Your Output 271

Chapter 20 • Using ODS Styles, Device Parameters, and Options ..................................................... 273
Overview ............................................................................................................................................. 274
Style Attributes versus Device Entry Parameters ........................................................................ 274
About Style Templates ..................................................................................................................... 275
Specifying a Style .......................................................................................................................... 279
Overriding Style Attributes with SAS/GRAPH Statement Options .............................................. 280
Precedence of Appearance Option Specifications ...................................................................... 281
Output from Different Devices and the GSTYLE/NOGSTYLE System Options ....................... 281
Additional Considerations with ODS Styles ...................................................................................... 283
Viewing the List of Styles Provided by SAS .............................................................................. 284
Modifying a Style .......................................................................................................................... 284
Graphical Style Element Reference for Device-Based Graphics .................................................. 287
Turning Off Styles .......................................................................................................................... 288
Changing the Appearance of Output to Match That of Earlier SAS Releases .................................. 298
Related Topics .................................................................................................................................. 298

Chapter 21 • Specifying Fonts in SAS/GRAPH Programs ................................................................. 299
Introduction: Specifying Fonts in SAS/GRAPH Programs .............................................................. 299
## Contents

SAS/GRAPH, System, and Device-Resident Fonts ........................................... 300
TrueType Fonts Supplied by SAS ................................................................. 300
Determining What Fonts Are Available .................................................... 300
Default Fonts ......................................................................................... 301
Viewing Font Specifications in the SAS Registry ..................................... 302
Specifying a Font .................................................................................... 303
Methods for Specifying Fonts ................................................................... 308

### Chapter 22 • Using Colors in SAS/GRAPH Programs ................................. 313
Colors, Styles, and Device Parameters ....................................................... 313
Specifying Colors in SAS/GRAPH Programs ........................................... 314
Using Transparency ............................................................................... 316
Color-Naming Schemes .......................................................................... 317
Using the Color Utility Macros ............................................................... 327
Processing Limitations for Colors .......................................................... 328

### Chapter 23 • Adding Images to SAS/GRAPH Output ................................. 331
Introduction .......................................................................................... 331
Image File Types Supported by SAS/GRAPH ........................................... 331
Displaying an Image in a Graph Background .......................................... 333
Displaying an Image in Graph Frame ...................................................... 334
Displaying Images on Data Elements ..................................................... 336
Displaying Images Using Annotate .......................................................... 338
Displaying Pop-Up Images Using the SVG Graphics Devices ................... 339
Disabling and Enabling Image Output ....................................................... 340

### PART 5  Statements and Options Used by SAS/GRAPH ............................... 341

#### Chapter 24 • SAS/GRAPH Statements .................................................. 343
Overview of Global Statements ............................................................... 343
Specifying Units of Measurement ........................................................... 344
Dictionary .............................................................................................. 345
Examples .................................................................................................. 465

#### Chapter 25 • Graphics Options and Device Parameters Dictionary ............ 515
Introduction to Graphics Options and Device Parameters ....................... 518
Specifying Graphics Options and Device Parameters ............................ 518
Options Used in Both GOPTIONS and OPTIONS Statements .................... 519
Dictionary .............................................................................................. 520

#### Chapter 26 • SAS System Options Used by SAS/GRAPH ......................... 629
Introduction to System Options ............................................................... 629
Dictionary .............................................................................................. 629

### PART 6  The Annotate Facility 633

#### Chapter 27 • Using Annotate Data Sets .................................................. 635
Overview: The Annotate Facility ............................................................... 636
The Annotate Function, Variable, and Macro Dictionaries ....................... 638
About the Annotate Data Set ................................................................. 638
About Annotate Graphics ................................................................. 645
Creating an Annotate Data Set ...................................................... 650
Producing Graphics Output from Annotate Data Sets ....................... 651
Annotate Processing Details .......................................................... 652
List of Annotate Examples ............................................................ 655
Annotate Error Messages ................................................................ 656
Examples ....................................................................................... 660

Chapter 28 • Annotate Function Dictionary ........................................ 667
About the Annotate Functions .......................................................... 667
Dictionary ...................................................................................... 668

Chapter 29 • Annotate Variable Dictionary .......................................... 705
About the Annotate Variables .......................................................... 706
Annotate Internal Coordinates ......................................................... 706
Dictionary ...................................................................................... 706

Chapter 30 • Annotate Macro Dictionary ............................................. 749
About the Annotate Macros ............................................................. 750
Using Annotate Macros .................................................................. 750
Dictionary ...................................................................................... 752

PART 7 SAS/GRAPH Procedures ...................................................... 775
Chapter 31 • SAS Mapping Procedures Documented in Another SAS Publication .............................................................. 777
SAS Non-Mapping Procedures .......................................................... 777

Chapter 32 • GANNO Procedure ...................................................... 779
Overview: GANNO Procedure .......................................................... 779
Syntax: GANNO Procedure .............................................................. 780
Examples: GANNO Procedure ........................................................... 784

Chapter 33 • GAREABAR Procedure ................................................ 795
Overview: GAREABAR Procedure .................................................... 795
Concepts: GAREABAR Procedure .................................................... 796
Syntax: GAREABAR Procedure ........................................................ 796
Examples: GAREABAR Procedure .................................................... 803

Chapter 34 • GBARLINE Procedure .................................................. 815
Overview: GBARLINE Procedure ..................................................... 816
About Bar-Line Charts ..................................................................... 817
About the Chart Variable ............................................................... 819
About Midpoints ............................................................................ 819
About Response Variables ............................................................. 821
About Chart Statistics ..................................................................... 821
Missing Values .............................................................................. 823
Plotting Variable Values Out of Range ............................................. 824
Controlling Patterns, Outlines, Colors, and Images ......................... 824
Syntax: GBARLINE Procedure ........................................................ 827
Examples: GBARLINE Procedure .................................................... 861

Chapter 35 • GCHART Procedure ..................................................... 871
Overview: GCHART Procedure ........................................................ 872
Chart Terminology .......................................................................... 877
Chapter 36 • GCONTOUR Procedure ........................................... 1007
Overview: GCONTOUR Procedure ........................................... 1007
CONTOUR Plot Terms ......................................................... 1008
Input Data ............................................................................. 1009
Data Ranges ........................................................................... 1009
Missing Values ....................................................................... 1010
Interpolating Data .................................................................. 1010
Syntax: GCONTOUR Procedure .............................................. 1010
Examples: GCONTOUR Procedure ........................................... 1031
References ............................................................................. 1043

Chapter 37 • GDEVICE Procedure ............................................. 1045
Overview: GDEVICE Procedure .............................................. 1046
Device Catalogs ....................................................................... 1046
Ways to Use the GDEVICE Procedure .................................... 1047
Syntax: GDEVICE Procedure .................................................. 1049
Using the GDEVICE Windows ............................................... 1055
Creating or Modifying Device Entries ..................................... 1060
Example: Creating a Custom Device Entry with Program Statements ........................................................................... 1062

Chapter 38 • GFONT Procedure ................................................ 1067
Overview: GFONT Procedure .................................................. 1067
Font Terminology and Characteristics ....................................... 1068
Storing or Displaying User-Created Fonts: GFONT0 Libref ............. 1069
Specifying Units for Creating Fonts .......................................... 1070
Syntax: GFONT Procedure ...................................................... 1070
Font Data Sets and the GFONT Procedure ............................... 1081
The Font Data Set .................................................................... 1081
The Kern Data Set ..................................................................... 1089
The Space Data Set ................................................................... 1091
Examples: GFONT Procedure .................................................. 1093

Chapter 39 • GKPI Procedure .................................................... 1099
Overview: GKPI Procedure ...................................................... 1099
Specifying Qualitative Range Boundaries and Actual KPI Values .... 1102
Controlling the Display of Boundary and Tick Mark Values .......... 1103
Controlling Qualitative Range Colors ....................................... 1104
Syntax: GKPI Procedure ......................................................... 1108
Examples: GKPI Procedure ...................................................... 1117

Chapter 40 • GOPTIONS Procedure ............................................ 1125
Overview: GOPTIONS Procedure .............................................. 1125
Syntax: GOPTIONS Procedure .................................................. 1126
Examples: GOPTIONS Procedure .............................................. 1129

Chapter 41 • GPLOT Procedure .................................................. 1133
Overview: GPLOT Procedure ................................................... 1134
Parts of a Plot ......................................................................... 1138
About This Book

Using This Document

Prerequisites

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

<table>
<thead>
<tr>
<th>To learn how to</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke SAS at your site</td>
<td>instructions provided by the on-site SAS support personnel</td>
</tr>
<tr>
<td>use Base SAS software</td>
<td>Base SAS documentation:</td>
</tr>
<tr>
<td>use the DATA step to create and manipulate SAS data sets</td>
<td>• SAS Language Reference: Concepts</td>
</tr>
<tr>
<td>allocate SAS libraries and assign librefs</td>
<td>• SAS Data Set Options: Reference</td>
</tr>
<tr>
<td>create external files and assign filerefs</td>
<td>• SAS Formats and Informats: Reference</td>
</tr>
<tr>
<td>manipulate SAS data sets using SAS procedures</td>
<td>• SAS Functions and CALL Routines: Reference</td>
</tr>
<tr>
<td></td>
<td>• SAS DATA Step Statements: Reference</td>
</tr>
<tr>
<td></td>
<td>• SAS System Options: Reference</td>
</tr>
<tr>
<td></td>
<td>• Base SAS Utilities: Reference</td>
</tr>
<tr>
<td></td>
<td>documentation for using SAS in your operating environment:</td>
</tr>
<tr>
<td></td>
<td>• SAS Companion for Windows</td>
</tr>
<tr>
<td></td>
<td>• SAS Companion for UNIX Environments</td>
</tr>
<tr>
<td></td>
<td>• SAS Companion for z/OS</td>
</tr>
</tbody>
</table>

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

**Mapping Procedures and Map Data Sets**

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

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

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

**Annotate Macros SAS Programs**

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

**About Examples, Their Output, and Sample Data Sets**

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

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

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

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

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

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

- **BORDER**
  - draws a border around the graphics output area.

Many examples process sample data contained in a SAS data set stored in the SASHELP library. The documentation for each example provides the data set name used. To see a brief description of any SAS data set in the SASHELP library, as well as output displaying the first five observations in each data set, please refer to [SASHELP Data Sets](#).
What's New in SAS/GRAPH 9.4: Reference

Overview


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

- Many procedures have significant enhancements and new options. See “Procedures” on page xx for a complete list.
- There are three new Universal Printer shortcut devices: EMFDUAL, TIFF, and TIFFK.
- The SVG and GIF devices now support animation. See “Devices and Universal Printers” on page xviii for more information.
- The Annotate Facility has several enhancements. See “The Annotate Facility” on page xix for a complete list.
- The Tile Chart applet supports adding one or more custom menu items to the tile chart pop-up menu. See “The Tile Chart Applet” on page xxii.
- The Treeview applet supports assigning multiple drill-down links for each node in the diagram. See “The Treeview Applet” on page xxii.

SAS 9.4M2:

- TITLE and FOOTNOTE statements now support the ALT="text-string" option. See “Global Statements” on page xxii.
- Information about the use of the JAVA and ActiveX devices was moved from the SAS/GRAPH Reference document to a separate document, SAS/GRAPH: Java Applets and ActiveX Control User’s Guide. Information about the level of support that SAS/GRAPH statements and options have for Java and ActiveX devices remain, including the Appendix Appendix 1, “Summary of ActiveX and Java Support,” on page 1441.
- The first and second parts of this reference were reorganized to consolidate device and output topics.

SAS 9.4M3:

- A topic in the introduction consolidates links for examples and various resources on the SAS website.
Two charts of the predefined colors in SAS are added to the appendix. Each chart lists the color name, the CX color code, the HLS color code, and a sample color swatch for each of the predefined colors. The colors are sorted by name in the first chart and by hue in the second chart.

SAS 9.4M4: Information about creating maps was moved from the SAS/GRAPH Reference to a separate document, SAS/GRAPH and Base SAS: Mapping Reference. This document includes the GEOCODE, GINSIDE, GMAP, GPROJECT, GREduce, GREMOVE, and MAPIMPORT procedures, and their supporting documentation.

SAS 9.4M5:

- Information about using SAS/GRAPH to plot data that is processed in SAS Cloud Analytic Services is added to the SAS/GRAPH Reference. See Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.
- SAS/GRAPH Network Visualization Workshop is discontinued and is no longer supported.

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

SAS 9.4M1: Descriptions of each data set in the SASHELP library are available online. To see these descriptions, as well as output displaying the first five observations in each data set, see SASHELP Data Sets.

SAS 9.4M3:

- The DEVICE=, GSTYLE, and GWINDOW system options are now documented in the SAS/GRAPH 9.4: Reference. The MAPS= system option is now documented in the SAS/GRAPH 9.4: Mapping Reference.
- HTMLBlue is the default style for the HTML ODS destination when using SAS both in batch mode and in the window environment. Previously, HTMLBlue was the default style only when using SAS in the window environment.
- The following ODS styles have been removed from suggested use: Astronomy, Banker, Curve, Education, Electronics, Gears, Magnify, Money, RSVP, Science, Sketch, Solutions, Torn, and Watercolor. These styles still function when used in SAS programs, but they no longer appear in the list of available styles.

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

Starting with SAS 9.4M6, the %CENTROID macro is available with Base SAS. The macro is no longer part of SAS/GRAPH, nor is SAS/GRAPH required to be installed to use this macro. This macro is useful when adding labels to a map, or when computing distances with the Base SAS GEODIST function. See the “%CENTROID Macro” on page 754, and the “GEODIST Function” in SAS Functions and CALL Routines: Reference for more information.

Devices and Universal Printers

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

Also, the GIF device now supports animation, and using this device is the preferred method for doing GIF animations. Documentation for the SAS/GRAPH GIFANIM device has been removed from SAS/GRAPH: Reference.

- The new TIFF and TIFFK devices produce TIFF images and support the RGBA and CMYK color schemes, respectively.
- The EMF device is now a Universal Printer shortcut device for the EMF Universal Printer. The default format for the EMF Universal Printer is now Enhanced Metafile Format Plus Extensions (EMF Plus). It supports RGBA colors (transparency).
- The new EMFDUAL device is a Universal Printer shortcut device that produces graphics file containing both EMF and EMF Plus records. It supports RGBA colors (transparency).
- The new %SHORTCUT autocall macro creates a shortcut device that is linked to the existing Universal Printers of the same name. It can also define a new Universal Printer and shortcut device and link the device to the Universal Printer.

SAS 9.4M2:
- The ZPNG device is disabled.
- The SVG, SVGT, and SVGView devices now add the ROLE and ARIA-LABEL attributes in the HTML output. The attributes are available when custom data tips are specified for hot-spots and the ODS HTML5 destination is used. These attributes make these data tips accessible to iPad users who are using the Voiceover screen reader.
- The PowerPoint destination supports both the JAVA IMG device and the ACTXIMG device.

SAS 9.4M5: The IMGPNG and IMGGIF devices are available. These devices can be used to improve the performance of SAS jobs that generate a large number of graphs. When used to render a large number of graphs, the performance of these devices is better than their corresponding Universal Printer shortcut devices, PNG and GIF.

Support for Transparency

The GIF and EMF devices and the new EMFDUAL device support RGBA color mode. RGBA colors enable you to specify transparency. (The RGBA color mode supports alpha channel blending. Overlapping colors of varying opacity are blended together.) See “Using Transparency” on page 316 for more information.

The Annotate Facility

- The HTML= option has been added to the ARROW function. The drill-down link or tooltip is applied to the line arrowhead.
- The IMGPATH variable now accepts a URL location for graphics that are on web pages.
• The HTML= option enables you to animate text labels that are created with the \texttt{LABEL} function. When you specify the HTML= variable, you can include the animation action and the parameters that control the animation. The animated label feature is available only when you use an SVG graphics device.

• The \%CENTROID macro has been enhanced to return centroid locations with a higher degree of accuracy.

• SAS 9.4M3: The maximum length for the XC= and YC= annotate variables is increased to 256 characters. In prior releases, the maximum length is 32 characters.

• SAS 9.4M5: The Style= variable for images supports SINGLE, which centers a single instance of the image on the specified coordinates.

\section*{Procedures}

\textbf{GBARLINE Procedure}

Starting with SAS 9.4M5, the \texttt{GBARLINE} procedure provides a new frequency format option \texttt{STATFMT=} on its BAR and the PLOT statements. This option applies a specified format to a calculated statistical value such as that specified with the frequency (FREQ=) option of the TYPE= option. The \texttt{STATFMT=} option overrides the GBARLINE procedure’s default format of the displayed statistical value. Use this option to change the default format that might contain decimal points, percentages, or commas. The \texttt{STATFMT=} option does not control the format of the response axis tick marks.

\textbf{GCHART Procedure}

The \texttt{GCHART} procedure has the following new options on the PIE, PIE3D, and DONUT statements:

• The PPERCENT= option modifies the font, height, and color of the percentages displayed in pie slice labels. When used with the PLABEL= option (which controls the text attributes of pie slice labels), you can easily differentiate between percentages and text labels.

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

Starting with SAS 9.4M5, the \texttt{GCHART} procedure has the following new options:

• The new GROUPREF option on the HBAR and VBAR statements provides the ability to draw a dividing line between bars or groups of bars. In addition, three new options enable you to manage the appearance of this reference line: option CGROUPREF manages line color, option LGROUPREF manages line type, and option WGROUPREF manages line width.

• Four new options enable you to draw a reference line between the midpoints of bars or groups of bars. Option MIDPOINTREF on the HBAR and VBAR statements provides the ability to draw a dividing line between the midpoints of bars or groups of bars. Three additional options enable you to manage the appearance of the midpoint reference line: option CMIDPOINTREF manages line color, option Lomidpointref manages line type, and option WMIDPOINTREF manages line width.
• The new frequency format option STATFMT= on the BLOCK, VBAR, VBAR3d, HBAR, HBAR3d, PIE, PIE3D, Donut, and STAR statements applies a specified format to a calculated statistical value such as that specified with the frequency (FREQ=) option of the TYPE= option. This option overrides the GCHART procedure’s default format of the displayed statistical value. Use this option to change the default format that might contain decimal points, percentages, or commas. The STATFMT= option does not control the format of the response axis tick marks.

**GKPI Procedure**

In the **GKPI procedure**, the FORMAT= option now accepts user-defined formats.

**GPLOT Procedure**

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

- Both the HZERO and VZERO requests on a PLOT, BUBBLE, PLOT2, or BUBBLE2 statement are ignored if the associated variable contains any time–formatted values, such as date, time, or datetime.
- The PLOT and PLOT2 statements in the GPLOT procedure support the new CBASELINE= option. This specifies the color of the horizontal baseline for reference lines of a plot. This option compensates for a reference line obscuring the axis.
- **SAS 9.4M1**: The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.
- **SAS 9.4M3**: The CTEXT= option in the GPLOT procedure’s PLOT statement is updated. It can now affect the color of POINTLABEL symbols.

**GRADAR Procedure**

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

**GTILE Procedure**

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

- **SAS 9.4M2**: The NOLEGEND option suppresses the legend that is automatically generated by default.
- **SAS 9.4M5**: If a device other than JAVA, JAVAIMG, ACTIVEX, or ACTXIMG is specified, the JAVAIMG device is used to generate the tile chart instead. In prior releases, the tile chart is not generated in that case.
The Tile Chart Applet

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

You can add custom pop-up menu items by specifying parameters in the PARAMETERS= option in the ODS HTML statement. The \texttt{MENU\_n\_LABEL=} and \texttt{MENU\_n\_VALUE=} parameters add custom menu item \textit{n} to all levels of the chart. The \texttt{MENU\_m\_LABEL=} and \texttt{MENU\_m\_VALUE=} parameters add custom menu item \textit{n} to the level \textit{m} pop-up menu only. In the ODS HTML statement, you can specify parameters \texttt{MENU\_n\_LABEL} and \texttt{MENU\_n\_VALUE}, or parameters \texttt{MENU\_m\_LABEL} and \texttt{MENU\_m\_VALUE}, but not both.

The Treeview Applet

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

Global Statements

The global statements have the following changes and enhancements:

- \textbf{SAS 9.4M2}: The \texttt{TITLE} and \texttt{FOOTNOTE} statements now support the \texttt{ALT=}``text-string'' option. This option enables you to specify descriptive text for the title or footnote. If you use \texttt{ALT=} in conjunction with the \texttt{LINK=} option, you can specify descriptive text for the URL to which the title or footnote links. The ``text-string'' can also contain occurrences of the variables named in a \texttt{BY} statement.

- \textbf{SAS 9.4M3}:
  - The new \texttt{GraphTitle1Text} style element controls and reduces the font size of the output of a \texttt{TITLE1} statement in order to scale better with the graphs.
  - The \texttt{SYMBOL} statement’s \texttt{POINTLABEL COLOR=} option now shows the sequential order of its color selection. The default color selection now aligns with the color specified for the axis label.
Part 1

Getting Started

Chapter 1
Get Started Using SAS/GRAPH ................................................. 3

Chapter 2
Gallery of SAS/GRAPH Plots and Charts .............................. 13

Chapter 3
Common Tasks Associated with Developing SAS/
GRAPH Programs ............................................................... 31

Chapter 4
Additional Resources to Help You Develop Your
SAS/GRAPH Programs ..................................................... 35
Chapter 1
Get Started Using SAS/GRAPH

What is SAS/GRAPH?

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

- organizes the presentation of your data and visually represents the relationship between data values as two- and three-dimensional graphs, including charts, plots, and maps.
- enhances the appearance of your output by enabling you to select text fonts, colors, patterns, and line styles, and control the size and position of many graphics elements.
- creates presentation graphics. SAS/GRAPH can create text slides, display several graphs at one time, combine graphs and text in one display, and create automated presentations.
• generates a variety of graphics output that you can display on your screen or in a web browser, store in catalogs, or review. You can send the graphics output to a hard copy graphics output device such as a laser printer.
• provides utility procedures and statements to manage the output.

Components of SAS/GRAPH Software

There are several components to SAS/GRAPH software:

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

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

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

The Annotate Facility enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an Annotate data set. You can use it to generate custom graphics or to enhance graphics output from many device-based SAS/GRAPH procedures, including GCHART, GMAP, GPLOT, GBARLINE, GCONTOUR, and G3D, as well as others. Annotation data sets are described in \textit{Chapter 27, “Using Annotate Data Sets,” on page 635}.

Network Visualization (NV) Workshop enables you to visualize and investigate the patterns and relationships hidden in network data (node-link data). Some common applications that use network data include supply chains, communication networks, websites, database schema, and software module dependencies. NV Workshop is designed for visualizing large networks. Using a combination of data tables, statistical graphs, and network graphs, NV Workshop enables you to extract information that would otherwise remain hidden. Help is available from the menu within the product. Network Visualization Workshop runs in Windows operating environments only. The Network Visualization Workshop is described in \textit{SAS/GRAPH: Network Visualization Workshop User’s Guide}.

\textit{Note:} Starting with SAS 9.4M5, SAS/GRAPH Network Visualization Workshop is discontinued and is no longer supported.
What You Can Create Using SAS/GRAPH

Charts and Plots

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

Table 1.1  SAS/GRAPH Charts and Plots

<table>
<thead>
<tr>
<th>Category</th>
<th>Chart or Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charts</td>
<td>Area Bar Charts, Bar Charts, Bar-line Charts, Block Charts, Donut Charts, KPI Charts, Pie Charts, Radar Charts, Star Charts, Tile Charts</td>
</tr>
<tr>
<td>2-D Plots</td>
<td>Bubble Plots, High-Low Plots, Line Plots, Needle Plots, Regression Plots, Scatter Plots, Spline Plots, Step Plots</td>
</tr>
<tr>
<td>3-D Plots</td>
<td>Contour Plots, Scatter Plots, Surface Plots</td>
</tr>
</tbody>
</table>

Multicell Graphs

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

Interactive Graphs

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

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

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

SAS/GRAPH Procedures

Procedures That Generate Single-Cell Plots

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

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

<table>
<thead>
<tr>
<th>Chart or Plot</th>
<th>Procedure</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Bar Chart</td>
<td>GAREABAR</td>
<td>GAREABAR procedure statement HBAR, HBAR3D, VBAR, or VBAR3D</td>
</tr>
<tr>
<td>Bar Chart</td>
<td>GCHART</td>
<td>GCHART procedure statement HBAR, HBAR3D, VBAR, or VBAR3D</td>
</tr>
<tr>
<td>Bar-line Chart</td>
<td>GBARLINE</td>
<td>GBARLINE procedure statements BAR for a vertical bar-line chart and PLOT for the line-plot overlay</td>
</tr>
<tr>
<td>Block Chart</td>
<td>GCHART</td>
<td>GCHART procedure statement BLOCK</td>
</tr>
<tr>
<td>Box Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=BOX option</td>
</tr>
<tr>
<td>Bubble Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure BUBBLE statement</td>
</tr>
<tr>
<td>Contour Plot</td>
<td>GCONTOUR</td>
<td>GCONTOUR procedure statement PLOT</td>
</tr>
<tr>
<td>Chart or Plot</td>
<td>Procedure</td>
<td>Statement</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Donut Chart</td>
<td>GCHART</td>
<td>GCHART procedure statement DONUT</td>
</tr>
<tr>
<td>High-Low Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=HILO option</td>
</tr>
<tr>
<td>KPI Chart</td>
<td>GKPI</td>
<td>GKPI procedure statement DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, or VSLIDER</td>
</tr>
<tr>
<td>Line Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=JOIN option</td>
</tr>
<tr>
<td>Map</td>
<td>GMAP</td>
<td>The GMAP procedure statements and the supporting data-preparation procedures GEOCODE, GINSIDE, GPROJECT, GREduce, GREMOVE, and MAPIMPORT are described in SAS/GRAph and Base SAS: Mapping Reference.</td>
</tr>
<tr>
<td>Needle Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=NEEDLE option</td>
</tr>
<tr>
<td>Pie Chart</td>
<td>GCHART</td>
<td>GCHART procedure statement PIE or PIE3D</td>
</tr>
<tr>
<td>Radar Chart</td>
<td>GRADAR</td>
<td>GRADAR procedure CHART statement</td>
</tr>
<tr>
<td>Regression Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the regression interpolation option</td>
</tr>
<tr>
<td>Scatter Plot (2-D)</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td>Chart or Plot</td>
<td>Procedure</td>
<td>Statement</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Scatter Plot (3-D)</td>
<td>G3D</td>
<td>G3D procedure statement SCATTER</td>
</tr>
<tr>
<td>Series Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=JOIN option</td>
</tr>
<tr>
<td>Spline Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=SPLINE option</td>
</tr>
<tr>
<td>Star Chart</td>
<td>GCHART</td>
<td>GCHART procedure statement STAR</td>
</tr>
<tr>
<td>Step Plot</td>
<td>GPLOT</td>
<td>GPLOT procedure statement PLOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global statement SYMBOL with the INTERPOL=STEP option</td>
</tr>
<tr>
<td>Surface Plot</td>
<td>G3D</td>
<td>G3D procedure statement PLOT</td>
</tr>
<tr>
<td>Tile Chart</td>
<td>GTILE</td>
<td>GTILE procedure statement FLOW, TILE, or TOGGLE</td>
</tr>
</tbody>
</table>

An example of each of these plots is in Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” on page 13.

**Procedure That Generates Multicell Graphs**

SAS/GRAPH provides the GREPLAY procedure, which you can use to create various multicell graphs using a predefined layout or your own custom layout. The GREPLAY procedure also has other capabilities that you might find useful. The GREPLAY procedure is described in Chapter 43, “GREPLAY Procedure,” on page 1280.

**Global Statements**

A global statement is a statement that can be specified anywhere in your SAS program. It remains in effect until it is explicitly changed, overridden, canceled, or until the SAS session is terminated. TITLE, FOOTNOTE, and OPTIONS are examples of global statements. These and other global statements can be used with SAS/GRAPH. The global statements are described in “Overview of Global Statements” on page 343. SAS/GRAPH provides additional global statements that affect only the graphics output that is generated by the SAS/GRAPH procedures and by the SAS/GRAPH Annotate facility.
**ODS Destinations and Styles**

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

**Graphics Options**

SAS/GRAPH provides several graphics options that control various aspects of SAS/GRAPH output such as size, format, fonts, and so on. These options are listed in Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515. The GOPTIONS statement sets one or more graphics options. The GOPTIONS statement is described in GOPTIONS Statement on page 375. The GOPTIONS procedure displays the current value of the graphics options. The GOPTIONS procedure is described in GOPTIONS Procedure on page 1125.

**Output Devices**

Graphics output in SAS/GRAPH is generated by a graphics device. A graphics device generates output in a specific format such as PNG, SVG, PDF, and so on. The device is specified with the DEVICE= graphics option in a GOPTIONS statement or with the DEVICE= system option in an OPTIONS statement. The SAS/GRAPH devices are described in Chapter 9, “Using Graphics Devices,” on page 79. A list of the commonly used graphics devices is in “Commonly Used Devices” on page 80.

By default, SAS/GRAPH output is determined by the ODS destination. The default output for the ODS destinations is described in “Default Devices for ODS Destinations” on page 82. There are some exceptions, which are indicated in this document, where applicable.

---

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

**About the Scenario in This Chapter**

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

- Use the GOPTIONS statement to set graphics options for your graph.
- Use ODS statements to specify an output document format and style for your output.
- Use SAS/GRAPH AXIS global statements to specify various attributes of your graph axes.
• Use global statements TITLE and FOOTNOTE to add a title and a footnote to your graph.
• Use the SAS/GRAPH GCHART procedure to generate a vertical bar chart.

Creating a Bar Chart Using SAS/GRAPH

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

goptions reset=all border xpixels=460 ypixels=500;            /* 1 */
ods _all_ close;                                              /* 2 */
ods html style=Daisy path="." (url=none) file="mychart.html";  /* 3 */
axis1 label=(angle=90 "Models Produced");                     /* 4 */
axis2 label=('"Vehicle Type"');
title "Models Produced By Vehicle Type and Origin";           /* 5 */
footnote "Data: SASHELP.CARS";
proc gchart data=sashelp.cars;                                /* 6 */
vbar3d type /                                               /* 7 */
   subgroup=origin
   type=freq freq
   shape=cylinder space=1
   axis=axis1 maxaxis=axis2 autoref clipref;
run;
quit;                                                        /* 8 */
ods html close;                                              /* 9 */
title;                                                       /* 10 */
footnote;
ods html;                                                    /* 11 */

Here is a description of each step in the program.

1. The GOPTIONS statement on page 375 specifies the graphics options. Option RESET=ALL sets all of the options to their default values. The XPIXELS=460 option specifies a width of 460 pixels, and the YPIXELS=500 specifies a height of 500 pixels for the graph.

2. The ODS _ALL_ CLOSE statement closes all currently open ODS destinations in order to conserve resources.

3. The ODS HTML statement opens the HTML destination. Option STYLE=DAISY specifies the style. Option FILE= specifies filename Mychart.html as the name of the output file, and option PATH= specifies a location for the output file. The URL=NONE option specifies that information in the PATH= option is not to appear in the links and references in the HTML output.

   Note: You must have Write access to the path that you specify in the PATH= option.

4. The AXIS statements on page 345 specify the attributes of the chart axes. The AXIS1 statement specifies a label for the response axis. The ANGLE=90 option rotates the label 90 degrees to display it vertically along the response axis. By default, the label is displayed horizontally at the top of the axis. The AXIS2 statement specifies a label for the midpoint axis.

5. The TITLE global statement specifies the title for the graph, and the FOOTNOTE global statement specifies the footnote. See “TITLE, FOOTNOTE, and NOTE Statements” on page 447.

6. The PROC GCHART statement on page 887 starts the GCHART procedure step. The DATA= option specifies the input data set, Sashelp.Cars.
7 The VBAR3D statement generates a 3-D vertical bar chart. Variable Type is specified as the category variable, and variable Origin is specified as the subgroup variable. Option TYPE=FREQ specifies frequency as the summary statistic. Option FREQ displays the frequency at the top of each bar. The SHAPE=CYLINDER and SPACE=1 specify the bar shape and spacing. The AXIS=AXIS1 option assigns the AXIS1 statement to the response axis, and the MAXIS=AXIS2 assigns the AXIS2 statement to the midpoint axis. The AUTOREF option draws reference lines at the major tick marks while the CLIPREF option clips the reference lines at the bars.

8 The QUIT statement terminates the GCHART procedure step.

9 The ODS HTML CLOSE statement closes the HTML destination.

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

11 The ODS HTML statement opens the HTML destination for subsequent procedure executions. This step is not required if you are using SAS Studio.

Here is the program output.

![Models Produced By Vehicle Type and Origin](image)

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

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

Table 1.3  Next Steps

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>If necessary, review the basic concepts of programming in the SAS language.</td>
<td>“Prerequisites” (p. xiii)</td>
</tr>
<tr>
<td>Learn more about the basic elements of a SAS/GRAPH program.</td>
<td>Chapter 5, “Elements of a SAS/GRAPH Program,” (p. 41)</td>
</tr>
<tr>
<td>See other charts and plots that you can create using SAS/GRAPH.</td>
<td>Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” (p. 13)</td>
</tr>
<tr>
<td>Learn how to perform common tasks that are associated with developing SAS/GRAPH programs.</td>
<td>Chapter 3, “Common Tasks Associated with Developing SAS/GRAPH Programs,” (p. 31)</td>
</tr>
<tr>
<td>Identify resources that you can use to help you develop your SAS/GRAPH programs.</td>
<td>Chapter 4, “Additional Resources to Help You Develop Your SAS/GRAPH Programs,” (p. 35)</td>
</tr>
</tbody>
</table>
Chapter 2

Gallery of SAS/GRAPH Plots and Charts

A Quick Look at the Gallery

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

<table>
<thead>
<tr>
<th>Area Bar Charts</th>
<th>Geographical Maps</th>
<th>Scatter Plots, 3-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Plot</td>
<td>High-Low Plots</td>
<td>Series Plots</td>
</tr>
<tr>
<td>Bar Charts</td>
<td>KPI Charts</td>
<td>Spline Plots</td>
</tr>
<tr>
<td>Bar-Line Charts</td>
<td>Line Plots</td>
<td>Star Charts</td>
</tr>
<tr>
<td>Block Charts</td>
<td>Needle Plots</td>
<td>Step Plots</td>
</tr>
<tr>
<td>Box Plots</td>
<td>Pie Charts</td>
<td>Surface Plots</td>
</tr>
<tr>
<td>Bubble Plots</td>
<td>Radar Charts</td>
<td>Tile Charts</td>
</tr>
<tr>
<td>Contour Plots</td>
<td>Regression Plots</td>
<td></td>
</tr>
<tr>
<td>Donut Charts</td>
<td>Scatter Plots</td>
<td></td>
</tr>
</tbody>
</table>

---

### Gallery of Basic Charts and Plots

#### Area Bar Charts

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

Use the GAREABAR procedure VBAR statement to generate an area bar chart. The GAREABAR procedure syntax is described in Chapter 33, “GAREABAR Procedure,” on page 795. The code that generates this area bar chart is in “Example 1: Generating an Area Bar Chart” on page 803. For more examples, see:

- “Example 2: Generating an Area Bar Chart with a Numeric Chart Variable” on page 805
- “Example 3: Generating an Area Bar Chart with Subgroups” on page 808
- “Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages” on page 810

#### Area Plot

An area plot is a line, step, or spline plot where the area between the line and the category axis is filled with a color, pattern, or both. They are typically used to visualize quantitative data over time. Here is a sample.
Use the SAS/GRAPH GPLOT procedure PLOT statement with the AREAS= option and a SYMBOL statement that specifies JOIN, SPLINE, or STEP interpolation. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The code that generates this area plot is in “Example 7: Filling Areas in an Overlay Plot” on page 1223.

### Bar Charts

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

Use the GCHART procedure VBAR statement for a vertical bar chart or HBAR statement for a horizontal bar chart. The GCHART procedure syntax is described in Chapter 35, “GCHART Procedure,” on page 872. For examples, see:

- “Example 3: Specifying the Sum Statistic in Bar Charts” on page 979
- “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982
- “Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985
- “Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

### Bar-Line Charts

A bar-line chart is a vertical bar chart with one or more plot overlays. This chart graphically represents the value of a statistic calculated for one or more variables in an input SAS data set. Here is a sample.
Use the GBARLINE procedure BAR and PLOT statements to generate a bar-line chart. The GBARLINE procedure syntax is described in Chapter 34, “GBARLINE Procedure,” on page 816. For examples, see:

- “Example 1: Producing a Basic Bar-Line Chart” on page 861
- “Example 2: Calculating Weighted Statistics” on page 863
- “Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 865

**Block Charts**

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

Use the GCHART procedure BLOCK statement to generate a block chart. The GCHART procedure syntax is described in Chapter 35, “GCHART Procedure,” on page 872. The code that generates this block chart is in “Example 1: Specifying the Sum Statistic in a Block Chart” on page 975. For another example, see “Example 2: Grouping and Subgrouping a Block Chart” on page 977.

**Box Plots**

A box plot summarizes the data and indicates the median, upper and lower quartiles, and minimum and maximum values. The plot provides a quick visual summary that easily shows center, spread, range, and any outliers. Here is a sample.
Use the GCHART procedure PLOT statement and the SYMBOL statement with the INTERPOL=BOX option to generate a box plot. A HILO interpolation can be used instead of the BOX interpolation. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412. For an example, see “Example 4: Creating and Modifying Box Plots” on page 476.

**Bubble Plots**

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

Use the GPLOT procedure BUBBLE statement to generate a bubble plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. For examples, see:

- “Example 1: Generating a Simple Bubble Plot” on page 1206
- “Example 2: Labeling and Sizing Plot Bubbles” on page 1208
- “Example 3: Adding a Right Vertical Axis” on page 1211

**Contour Plots**

A contour plot represents a three-dimensional surface by plotting Z values on a two-dimensional format. The data provides the X, Y, and Z values. Here is a sample.
Use the GCONTOUR procedure PLOT statement to generate a contour plot. The GCONTOUR procedure syntax is described in Chapter 36, “GCONTOUR Procedure,” on page 1007. For examples, see:

- “Example 1: Simple Contour Plot” on page 1031
- “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033
- “Example 3: Specifying Contour Levels” on page 1036
- “Example 4: Using Patterns and Joins” on page 1038

**Donut Charts**

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

Use the GCHART procedure DONUT statement to generate a donut chart. The GCHART procedure syntax is described in Chapter 35, “GCHART Procedure,” on page 872. For an example, see “Example 8: Subgrouping a Donut or Pie Chart” on page 994.

**Geographical Maps**

A geographical map displays values or attributes related to areas on the map. Map types include block, choropleth, prism, and surface. Maps can be generated by SAS/GRAPH. Here is a sample.
Use the GMAP procedure BLOCK, CHORO, PRISM, or SURFACE statement to generate a geographical map. See *SAS/GRAPH and Base SAS: Mapping Reference*.

**High-Low Plots**

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

Use the GPLOT procedure PLOT statement and the SYMBOL statement with a HILO interpolation option to generate a high-low plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412.

**KPI Charts**

Creates graphical key performance indicators (KPIs), which show metrics that help a business monitor its performance and measure its progress toward specific goals. You can create five KPI chart types, as displayed left to right in the graphic following this list:

- vertical slider
- speedometer
- radial dial
- vertical traffic light
- horizontal bullet graph (bottom)
Use the GKPI procedure to create a KPI chart. The GKPI procedure syntax is described in Chapter 39, “GKPI Procedure,” on page 1099. For examples, see:

- “Example 1: Using the Default Colors as the Active Colors” on page 1117
- “Example 2: Creating a Gray Scale Bullet Graph” on page 1118
- “Example 3: Creating a Dial KPI Chart” on page 1119
- “Example 4: Defining a Speedometer” on page 1121
- “Example 5: Defining a Speedometer with Reversed Colors” on page 1122
- “Example 6: Creating a Traffic Light” on page 1123

Line Plots

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

Use the GPLOT procedure PLOT statement and the SYMBOL statement with the INTERPOL=JOIN option to create a simple line chart. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412. The code that generates this line plot is in “Example 5: Connecting Plot Data Points” on page 1217. For more examples, see:

- “Example 6: Generating an Overlay Plot” on page 1219
Needle Plots

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

Use the GPLOT procedure PLOT statement and the SYMBOL statement with the NEEDLE interpolation option to generate a needle plot. The GPLOT procedure syntax is described in Chapter 41, "GPLOT Procedure," on page 1134. The SYMBOL statement syntax is described in "SYMBOL Statement" on page 412. For an example, see "Example 9: Plotting with Different Scales of Values" on page 1231.

Pie Charts

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

Use the GCHART procedure PIE statement to generate a pie chart. Use the PIE3D statement to generate a 3-D pie chart. The GCHART procedure syntax is described in Chapter 35, "GCHART Procedure," on page 872. For examples, see:

- “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991
- “Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996
- “Example 10: Grouping and Arranging Pie Charts” on page 998
- “Example 13: Creating a Detail Pie Chart” on page 1004
Radar Charts

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

![Radar Chart Example](image)

Use the GRADAR procedure CHART statement to generate a radar chart. The GRADAR procedure syntax is described in Chapter 42, “GRADAR Procedure,” on page 1243. The code that generates this radar chart is in “Example 1: Producing a Basic Radar Chart” on page 1266. For more examples, see:

- “Example 2: Overlaying Radar Charts” on page 1267
- “Example 3: Tiling Radar Charts” on page 1268
- “Example 4: Using Multiple Classification Variables in Radar Charts” on page 1269
- “Example 5: Modifying the Appearance of Radar Charts” on page 1271
- “Example 6: Modifying Chart Axes” on page 1272
- “Example 7: Creating a Wind Rose Chart” on page 1275
- “Example 8: Creating a Calendar Chart” on page 1276

Regression Plots

A regression plot includes a scatter plot of two numeric variables along with an overlaid linear or nonlinear fit line that enables you to perform a regression analysis. You can specify with the DEGREE option range of 1–10 a certain type of regression equation. Examples are linear (DEGREE=1), quadratic (DEGREE=2), or cubic (DEGREE=3). You can display confidence limits for mean predicted values or individual predicted values. Here is a sample.
Use the GPLOT procedure PLOT statement and the SYMBOL statement with a regression interpolation option to generate a regression plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412. The code that generates this regression plot is in “Example 4: Plotting Two Variables” on page 1214.

**Scatter Plots**

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

Use the GPLOT procedure PLOT statement to generate a scatter plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The code that generates this scatter plot is in “Example 4: Plotting Two Variables” on page 1214.

**Scatter Plots, 3-D**

A 3-D scatter plot generates a marker for each observation that has nonmissing values for three variables. Markers can be symbols or character strings. Symbol markers can be labeled. Here is a sample.
Use the G3D procedure SCATTER statement to generate a 3-D scatter plot. The G3D procedure syntax is described in Chapter 46, “G3D Procedure,” on page 1381. For examples, see:

- “Example 4: Generating a Scatter Plot” on page 1409
- “Example 5: Generating a Scatter Plot with Modified Shapes” on page 1410
- “Example 6: Generating a Scatter Plot with Modified Shapes and a Grid” on page 1412
- “Example 7: Generating a Rotated Scatter Plot with Modified Axes” on page 1414

**Series Plots**

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

Use the GPLOT procedure PLOT statement and the SYMBOL statement with the JOIN interpolation option to generate a series plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412. For examples, see:

- “Example 5: Connecting Plot Data Points” on page 1217
- “Example 6: Generating an Overlay Plot” on page 1219
- “Example 6: Generating an Overlay Plot” on page 1219
- “Example 8: Plotting Three Variables” on page 1226

**Spline Plots**

A spline plot is a line plot that uses a cubic spline method with continuous second derivatives to produce smooth curves. Here is a sample.
Use the GPLOT procedure PLOT statement and the SYMBOL statement with the INTERPOL=SPLINE option to create a spline plot. The GPLOT procedure syntax is described in Chapter 41, “GPLOT Procedure,” on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412. For an example, see “Example 8: Plotting Three Variables” on page 1226.

**Star Charts**

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

Use the GCHART procedure STAR statement to generate a star chart. The GCHART procedure syntax is described in Chapter 35, “GCHART Procedure,” on page 872. The code that generates this start chart is in “Example 11: Specifying the Sum Statistic in a Star Chart” on page 999. For another example, see “Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1001.

**Step Plots**

A step plot displays a series of horizontal and vertical line segments that connect observations of input data. The plots use a step function to connect the data points. The vertical line can change at each step. Here is a sample.
Use the GPLOT procedure PLOT statement and the SYMBOL statement with the STEP interpolation option to generate a step plot. The GPLOT procedure syntax is described in GPLOT Procedure on page 1134. The SYMBOL statement syntax is described in “SYMBOL Statement” on page 412.

Surface Plots

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

Use the G3D procedure PLOT statement to generate a surface plot. The G3D procedure syntax is described in Chapter 46, “G3D Procedure,” on page 1381. The code that generates this surface plot is in “Example 1: Generating a Surface Plot” on page 1404. For more examples, see:

- “Example 2: Generating a Rotated Surface Plot” on page 1405
- “Example 3: Generating a Tilted Surface Plot” on page 1407

Tile Charts

A tile chart consist of rectangles or squares that are divided into tile-shaped segments. The chart is sometimes referred to as a rectangular tree map. The chart represents the relative sizes of the tiles to one another and to the whole. Here is a sample.
Use the GTILE procedure TILE statement to generate a tile chart. The GTILE procedure syntax is described in Chapter 45, “GTILE Procedure,” on page 1345. The code that generates this tile chart is in “Example 1: Simple GTILE with the COLORVAR= Option” on page 1365. For more examples, see:
- “Example 2: Specifying the COLORRAMP= and COLORVAR= Options” on page 1367
- “Example 3: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range” on page 1370
- “Example 4: Specifying Discrete Tile Colors” on page 1373
- “Example 5: Specifying Custom Items in the Chart Pop-Up Menu” on page 1374

Gallery of Additional Graph Features

Annotations

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

In this sample, labels are drawn inside the bar segments using text attributes that differ from the rest of the graph. The SAS/GRAPHS/GRAPH Annotate facility is described in Chapter 27, “Using Annotate Data Sets,” on page 635. The code that generates this annotated bar
Multicell Graphs

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

A layout template provides the layout for the multicell graph. Several predefined templates are provided. You can also create your own template. The GREPLAY procedure syntax and usage are described in Chapter 43, “GREPLAY Procedure,” on page 1280 and “Example 3: Replaying Graphs into a Template” on page 1323. The code that generates this multicell graph is in “Example 3: Replaying Graphs into a Template” on page 1323. For another example, see “Example 3: Replaying Graphs into a Template” on page 1323.

Interactive Graphs

Drill-Down Graphs

You can use SAS/GRAPH to create drill-down graphs that enable your users to explore your data. Elements in the graph such as bars, markers, and so on, can link to additional resources that show more detail. You can create multiple drill-down levels. When a user clicks a linked element in the graph, a new resource with more detail opens in the browser.
Drill-down graphs are described in Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 187. The code that generates this drill-down graph is in “Example 4: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 225. For more examples, see:

- “Example 3: Creating a Drill-Down Java Presentation for the Web” on page 220
- “Example 4: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 225

**Animated Graphs**

You can use SAS/GRAPH to create animated graphs in the GIF or SVG format. An animated graph shows a sequence of graphs automatically. It is useful for showing how data changes over time. You can control the amount of time that each graph is displayed and how the graphs transition. Animated graphs are described in Chapter 15, “Generating Animations,” on page 167. For examples, see “Examples: Generating Animated Images” on page 176.

**Java and ActiveX Graphs**

SAS/GRAPH provides Java applets and an ActiveX control that enable you to create interactive graphs where Java applets and ActiveX controls are supported. The JAVA graphics device generates a SAS/GRAPH Java applet that displays your graph in a web browser that supports Java applets. The ACTIVEX graphics device generates the SAS/GRAPH ActiveX Control, which displays your graph, and embeds it in an HTML document or in an RTF document. You can embed your ACTIVEX output in Microsoft documents such as Microsoft Word and Microsoft PowerPoint. Both Java and ActiveX
graphs provide pop-up menus that enable your users to change various attributes of the graph dynamically.

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

![Horizontal Bar Chart]

Java and ActiveX graphs are described in *SAS/GRAPH: Java Applets and ActiveX Control User’s Guide*.

**T I P** You can use the JAVA IMG and ACTX IMG graphics devices to generate PNG images of Java and ActiveX graphs for use where Java applets and ActiveX controls are not supported.
Chapter 3

Common Tasks Associated with Developing SAS/GRAPH Programs

About the SAS/GRAPH Tasks

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

- Output environment
- Graph creation
- Graph enhancement

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

Output Environment Tasks

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapter 11, “SAS/GRAPH Output,” on page 105</td>
</tr>
<tr>
<td>Learn about the SAS/GRAPH graphics devices.</td>
<td>Chapter 9, “Using Graphics Devices,” (p. 79)</td>
</tr>
</tbody>
</table>
Graph Creation Tasks

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
</table>
| Create a custom SAS/GRAPH graphics device.                          | “Creating Universal Printers and Shortcut Devices” on page 91  
|                                                                    | “Creating a Custom Device” on page 93               |
| Specify the graph size.                                             | “Setting the Size of Your Graph” on page 113        |
| Specify the image resolution.                                       | “Setting the Resolution of Your Graph” on page 114  |
| Use ODS to specify the output document format.                      | Chapter 10, “Managing Your Graphics with ODS,” (p. 97) |
| Determine the SAS/GRAPH language elements that you need to use to create your graph. | Chapter 2, “Gallery of SAS/GRAPH Plots and Charts,” (p. 13) |
| Create your SAS/GRAPH program.                                      | Chapter 5, “Elements of a SAS/GRAPH Program,” (p. 41) |
| Add titles, footnotes, and notes to your graph.                     | “TITLE, FOOTNOTE, and NOTE Statements” (p. 447)     |
| Specify fonts in your SAS/GRAPH program.                            | Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” (p. 299) |
| Specify colors in your SAS/GRAPH program.                           | Chapter 22, “Using Colors in SAS/GRAPH Programs,” (p. 313) |
|                                                                    | “BY Statement” on page 370                          |
| Generate a graph in the SVG format.                                 | Chapter 13, “Using SVG Graphics,” (p. 141)           |
| Generate a graph in a bitmap format.                                | Chapter 14, “Generating SVG, PNG, GIF, and TIFF Graphics,” (p. 153) |
Graph Enhancement Tasks

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add data tips and a description to your graph.</td>
<td>Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” (p. 187)</td>
</tr>
<tr>
<td>Add one or more images to your graph.</td>
<td>Chapter 23, “Adding Images to SAS/GRAPH Output,” (p. 331)</td>
</tr>
<tr>
<td>Annotate your graph.</td>
<td>Chapter 22, “Using Colors in SAS/GRAPH Programs,” (p. 313)</td>
</tr>
<tr>
<td>Animate your graph.</td>
<td>Chapter 15, “Generating Animations,” (p. 167)</td>
</tr>
<tr>
<td>Add drill-down links to your graph.</td>
<td>Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” (p. 187)</td>
</tr>
</tbody>
</table>
Chapter 4
Additional Resources to Help You Develop Your SAS/GRAPH Programs

SAS/GRAPH Sample Programs .............................................................. 35
Examples and Resources on the Web .................................................. 37
Books ......................................................................................... 38

SAS/GRAPH Sample Programs

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

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

• In most operating environments, you can access the sample code through the SAS Help and Documentation. Select Help ⇒ SAS Help and Documentation. On the Contents tab, select Learning to Use SAS ⇒ Sample SAS Programs ⇒ SAS/GRAPH Samples.

• In other operating environments, the SAS Sample Library might be installed in your file system. Ask your on-site SAS support personnel where it is located on your system.

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

This document includes the name of each sample file that it references (for example, GBLWTSTA or GCHBRMID). The naming convention for SAS/GRAPH samples is Gpcxxxxx, where pc is the product code and xxxxx is an abbreviation of the example title. The product code can be a code for a procedure, a statement, or in the case of Java and ActiveX examples, WB for “web graphs.” To illustrate, the code for an example in the GCHART procedure chapter, “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991, is stored in sample member GCHPISUM.

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

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

**Table 4.1  Product Codes for SAS/GRAPH Procedures**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ganno</td>
<td>AN</td>
</tr>
<tr>
<td>gareabar</td>
<td>AB</td>
</tr>
<tr>
<td>gbarline</td>
<td>BL</td>
</tr>
<tr>
<td>gchart</td>
<td>CH</td>
</tr>
<tr>
<td>gcontour</td>
<td>CT</td>
</tr>
<tr>
<td>gfont</td>
<td>FO</td>
</tr>
<tr>
<td>gkpi</td>
<td>KP</td>
</tr>
<tr>
<td>goptions</td>
<td>OP</td>
</tr>
<tr>
<td>gplot</td>
<td>PL</td>
</tr>
<tr>
<td>gradar</td>
<td>RR</td>
</tr>
<tr>
<td>greplay</td>
<td>RE</td>
</tr>
<tr>
<td>gslide</td>
<td>SL</td>
</tr>
<tr>
<td>gtile</td>
<td>TL</td>
</tr>
<tr>
<td>g3d</td>
<td>TD</td>
</tr>
<tr>
<td>g3grid</td>
<td>TG</td>
</tr>
</tbody>
</table>

**Table 4.2  Product Codes for SAS/GRAPH Statements**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>AX</td>
</tr>
<tr>
<td>by</td>
<td>BY</td>
</tr>
<tr>
<td>footnote</td>
<td>FO</td>
</tr>
<tr>
<td>goptions</td>
<td>ON</td>
</tr>
</tbody>
</table>
Examples and Resources on the Web

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

- The SAS Training Post is a blog that provides tutorials, tips, and practical information about SAS. Dr. Robert Allison frequently contributes to the blog. His posts use SAS/GRAPH and ODS Graphics for data visualization.
  
  http://blogs.sas.com/content/sastraining/author/robertallison/

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


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

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


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

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


In addition, SAS offers instructor-led training and self-paced e-learning courses to help you get started with platform graphics software. For more information about the courses available, see sas.com/training.
Books

For a list of books that might help you develop your SAS/GRAPH programs, see “Recommended Reading” on page 1557.
Part 2

SAS/GRAPH Concepts

Chapter 5
Elements of a SAS/GRAPH Program ............................................ 41

Chapter 6
Overview of Devices, ODS Destinations, and ODS Styles ............... 49

Chapter 7
Using Run-Group Processing ..................................................... 67

Chapter 8
The Graphics Output Environment ............................................. 69

Chapter 9
Using Graphics Devices ............................................................ 79

Chapter 10
Managing Your Graphics with ODS ............................................. 97

Chapter 11
SAS/GRAPH Output .............................................................. 105

Chapter 12
Plotting a Cloud Analytic Services (CAS) In-Memory Table ........... 135
Overview

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

A Typical SAS/GRAPH Program

Most SAS/GRAPH programs have Base SAS statements, ODS statements, and SAS/GRAPH statements. Annotate DATA steps are also used in many SAS/GRAPH programs. The sample program below identifies the basic parts of a typical SAS/GRAPH program. Each element is described in more detail in the following sections.
The following is an ODS destination statement that defines the output file name and location, and changes the style from the default of HTMLBlue to Analysis.

```sql
ods html file="c:\regression.htm" style=analysis;
```

The following four statements are Global statements:

```sql
goptions reset=all device=png;
title "Study of Height vs. Weight";
footnote j=r h=2 "Data: SASHELP.CLASS";
symbol interpol=rclm95 value=circle;
```

The next five lines represent the PROC step:

```sql
proc gplot data=sashelp.class;
plot height*weight /
    haxis=45 to 155 by 10;
run;
quitting;
```

The following is an ODS destination statement. It resets the default style.

```sql
ods html close;
```

For information about ODS destinations, see “Destination Statements” on page 45.

For information about the GOPTIONS statement, see “Global Statements” on page 43.

For information about the TITLE and FOOTNOTE statements, see “Global Statements” on page 43.

For information about the SYMBOL statement, see “Global Statements” on page 43.

For information about the procedure statement, see “Procedure Statement” on page 43.

For information about the plot (action) statement, see “Subordinate Statement” on page 43.

For information about the RUN statement, see “Shared Language Statements” on page 46.

For information about the QUIT statement, see “Shared Language Statements” on page 46.

For information about specifying the style for an ODS destination, see “Destination Statements” on page 45.

For information about the HAXIS option, see “Other Statements and Options” on page 43.

For information about ODS styles, see “ODS Statement Options” on page 45.

---

**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. They can generate the graphics output and control the appearance of the output. Statements can define variables and perform other operations.
on your data. You can also specify global statements and options within the PROC step to customize the appearance of your graph. However, it is often more efficient to specify global statements before the PROC step.

**Procedure Statement**

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

**Subordinate Statement**

Subordinate statements 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 that you can specify within the PROC step to control your graphics output. PROC step options always follow the forward slash (/) following the action statement of the procedure. These options might control such things as axis characteristics, midpoint values, statistics, catalog entry descriptions, or appearance elements of your graph. An example is the SUBGROUP= option in the BLOCK statement of the GCHART procedure. The option tells the procedure to divide the graph's bars into segments according to the values of the SUBGROUP= variable. The HAXIS option in the PLOT statement of the GPLOT procedure, as shown in “A Typical SAS/GRAPH Program” on page 41, specifies where to draw the major tick mark values for the horizontal axis.

**Global Statements**

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

Below is a list of all the SAS/GRAPH statements along with their brief descriptions. See Chapter 24, “SAS/GRAPH Statements,” on page 343 for a more detailed description of each of these statements.

**AXIS**

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

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

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

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

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

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

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

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

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

---

**Annotate DATA Set**

An Annotate DATA set is a data set containing graphics commands that can be applied to SAS/GRAPH output. See Chapter 27, “Using Annotate Data Sets,” on page 635 for information about building and using Annotate data sets. The Annotate facility can be used to create a completely new graph or to annotate existing PROC output. See Chapter 28, “Annotate Function Dictionary,” on page 667 and Chapter 29, “Annotate Variable Dictionary,” on page 705 for a complete description of all Annotate functions and variables. Below is an example of how the Annotate facility can be used to annotate numbers on bars in a graph that was created using the GCHART procedure.
Like Base SAS, SAS/GRAPH uses ODS destination statements to control where the output goes and how it looks. Although ODS statements are not required in every SAS/GRAPH program, they are necessary if you want to generate graphs for destinations other than the default HTML destination. Some other destinations include LISTING, RTF, and PDF. For more information about ODS destinations, see “Understanding ODS Destinations” in SAS Output Delivery System: User’s Guide.

As shown in “A Typical SAS/GRAPH Program” on page 41, the ODS destination statement is used at the beginning and end of the program to modify the default destination. The beginning statement modifies the default path and filename of the output file as well as the default style. The end statement sets the HTML style back to its default of HTMLBlue. The ODS destination statement can also be used to open a different destination. If you do choose to use a destination other than the default and need to use the ODS destination statement, you should always open the destination before calling the procedure. Opening a non-default destination results in output being sent both to HTML by default as well as to the additional specified destination. Conserve system resources by using the ODS destination statement at the end of the SAS program to close a destination that was opened in that program.

**ODS Statement Options**

You can use the STYLE= option on the ODS destination statement to change the style that is applied to your output. Starting with SAS 9.4, the default style is...
Base SAS Language Elements

Shared Language Statements

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

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

**FILENAME**
associates a SAS fileref with an external file or output device. See “FILENAME Statement” on page 47 for a more detailed description of this statement. For a complete description of this statement, see *SAS Global Statements: Reference*.

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

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

**LIBNAME**
associates a libref with a SAS library. See “LIBNAME Statement” on page 47 for a more detailed description of this statement. For a complete description of this statement, see *SAS Global Statements: Reference*.

**ODS statements**
control the output of SAS/GRAPH procedures, where the output is sent (destination), the appearance of the output (STYLE=), and the output file type (DEVICE=). Starting with SAS 9.4, the default style is HTMLBlue. See Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 49 for information about using ODS with SAS/GRAPH procedures.

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

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

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

HTMLBlue. For more information about the STYLE= option, see Chapter 20, “Using ODS Styles, Device Parameters, and Options,” on page 273.
**FILENAME Statement**

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

- point to a text file that you want to use for data input or output.
- assign the destination of a graphics stream file (GSF). This destination can be either a single, specific file or an aggregate file storage location, such as directory or PDS. See “Exporting Your Output” on page 133 for information about creating graphics stream files.

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

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

```
FILENAME fileref='external-file';
```

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

**LIBNAME Statement**

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

- SAS files for data input and output.
- SAS catalogs that contain SAS/GIS maps, fonts, GRSEG, CMAP, TEMPLATE, or device entries.
- SAS catalogs that contain graphics output. These catalogs are often stored in permanent libraries. See “Controlling Where Your Output Is Stored” on page 116 for information about storing graphics output in a permanent catalog.

The LIBNAME statement has this general form:

```
LIBNAME libref='SAS-library';
```

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

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

- For more information about using and managing SAS/GRAPH programs to create graphics output, see Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 49.


- For information about using and managing SAS/GRAPH output, see Chapter 11, “SAS/GRAPH Output,” on page 105.
Chapter 6
Overview of Devices, ODS Destinations, and ODS Styles

Introduction to Devices, ODS Destinations, and ODS Styles

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

SAS/GRAPH device

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

Note: For information about ActiveX control, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide
ODS destination
Like other SAS procedures, the output from SAS/GRAPH procedures is controlled by ODS (Output Delivery System). An ODS destination controls where your output is sent, which could be to an HTML file, a PDF file, to the GRAPH window, directly to a printer, and so on. The ODS destination is specified by the ODS destination statement.

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

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

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

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

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

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

---

Working with ODS Destinations and Styles

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

Opening and Closing Destinations

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

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

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

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

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

For more information about ODS destinations, see “Managing ODS Destinations” on page 97 and “ODS Destination Statement Options” on page 99.

**The LISTING Destination**

The LISTING destination is somewhat different from the other ODS destinations. For the LISTING destination, if you do not specify a SAS/GRAPH device, then your output is sent to the GRAPH window. However, if you specify a device, then the device determines where your output is sent in addition to determining the graphics format. For example, the PNG device sends output to a PNG file instead of the GRAPH window. Your company might have device drivers specific to your site that send output directly to a certain printer. Where your output is sent is controlled by the device entry in the SASHELP.DEVICES catalog. For more information about devices, see “Controlling the Graphics Output Format with the DEVICE= Option” on page 61 and Chapter 9, “Using Graphics Devices,” on page 79.

The LISTING destination is not open by default. You must open the LISTING destination at the beginning of your SAS program if you want to send output to it. You should close the LISTING destination at the end of your program to conserve resources. See “Closing Destinations to Save System Resources” on page 64 for more information.

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

**Introduction to Styles**

By default, ODS applies a style to all output. A style is a template, or set of instructions, that determines the colors, font face, font sizes, and other presentation aspects of your output. SAS ships many predefined styles in the STYLES item store, such as Analysis, Statistical, and Journal. Examples of some of these predefined styles are shown in Table 6.1 on page 52. Many additional styles (see “Viewing the List of Styles Provided by SAS” on page 284) are available in the STYLES item store in SASHELP.TMPLMST.

Each destination has a default style that is associated with it. For example, the default style for the PDF destination is Pearl, and the default style for the HTML destination is HTMLBlue. See “ODS Destinations and Default Styles” on page 275 and “Recommended Styles” on page 276 for more information.
Table 6.1 Examples of Styles Available in SASHELP.TMPLMST

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistical</td>
<td>Style for statistical output</td>
</tr>
<tr>
<td>Analysis</td>
<td>Style for analysis output</td>
</tr>
<tr>
<td>Ocean</td>
<td>Style for ocean output</td>
</tr>
<tr>
<td>Journal</td>
<td>Style for journal output</td>
</tr>
</tbody>
</table>

Specifying a Style

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

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

See “About Style Templates” on page 275 and “Specifying a Style” on page 279 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...”
Generating Output with SAS/GRAPH Procedures

Overview of Generating Output

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

Each ODS destination is associated with a default style and a default SAS/GRAPH device to optimize your output for that destination. For more information, see “Introduction to Devices, ODS Destinations, and ODS Styles” on page 49.

The rules change when you are running SAS without a terminal. In this case you must either specify the DEVICE= option in the GOPTIONS statement or specify the DEVICE= or PRINTER= option in the ODS destination statement. A terminal is not associated with your SAS session if you are running in batch mode on z/OS. This is also true if you are running SAS on UNIX without an X Server or if you have specified the NOTERMINAL system option. See the global option “DEVICE” on page 540, and Chapter 9, “Using Graphics Devices,” on page 79 for more information.

Output in the SAS Windowing Environment

Sending Output to the GRAPH Window (LISTING Destination)

When working in an interactive environment such as Windows, the default LISTING destination is the GRAPH window. Use the ODS destination statement at the beginning and end of the SAS program to open and close the LISTING destination, respectively.

The following example is a simple GCHART program that produces the output shown in Figure 6.1 on page 54.

ods html close;
ods listing;
gooptions reset=all border hsize=6 vsize=4.5;
title "US Electric Power - Revenue and Generation Sources";
proc gchart data=sashelp.electric (where=(year >= 2000)) ;
  vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;
ods listing close;
The default style applied to output sent to the LISTING destination is LISTING in a Display Manager Session and in batch mode. When you send output to the LISTING destination, SAS/GRAPH uses a default device driver that generates output for the GRAPH window. This device driver does not write an image file to disk.\(^1\) For the LISTING destination, the default device driver varies by operating environment. In a Display Manager Session, the default device driver on Windows systems is WIN. On UNIX systems, the default device driver is XCOLOR, and on z/OS systems, the default device driver is IBMPCGX.

### Sending Output to a File

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

To create a GIF file with the graph shown in Figure 6.1 on page 54, 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.

```sas
ods html close;
ods listing;
goptions reset=all border hsize=6 vsize=4.5;
filename gout "./revgensrcs.gif";
goptions device=gif gsfname=gout;
title "US Electric Power - Revenue and Generation Sources";
proc gchart data=sashelp.electric (where=(year >= 2000)) ;
```

\(^1\) SAS/GRAPH procedures create GRSEG catalog entries when you send output to the LISTING destination, but the GRSEG file format is an internal file format specific to SAS/GRAPH. It cannot be used as if it were an image file such as a PNG or GIF file.
vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;

ods listing close;
The LISTING style is applied to the graph as shown in Figure 6.1 on page 54. In the FILENAME statement, the current directory is the default SAS output directory.

For more information about sending graphics output to a file, see “Controlling Where Your Output Is Stored” on page 116.

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

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

To create a web page using the style of Default with the graph shown in Figure 6.1 on page 54, add the ODS HTML statements around the procedure code. The first style= option overrides the default style of HTMLBlue. The second style= option resets that style. Submit the following SAS program:

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

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

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

**Sending Output to an RTF File**

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

To create an RTF document that contains the graph shown in Figure 6.1 on page 54, add the ODS RTF statements around the procedure code and submit the following SAS program. To conserve system resources, close the default HTML destination with another ODS destination statement.

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

proc gchart data=sashelp.electric (where=(year >= 2000)) ;
vbar year / discrete sumvar=Revenue subgroups=Customer;
run;
quit;
ods rtf close;
```
By default, SAS/GRAPH creates an RTF file with the graph embedded in it and, in some cases, displays the RTF file in a viewer. When you send output to the RTF destination, SAS/GRAPH does not write a separate image file to disk. The default device driver is the SASEMF driver, and the default style is RTF.

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

To create a PDF document that contains the graph shown in Figure 6.1 on page 54, add the ODS PDF statements around the procedure code and submit the following SAS program:

```sas
ods html close;
ods pdf;
goptions reset=all border;
title "US Electric Power - Revenue and Generation Sources";
proc gchart data=sashelp.electric (where=(year >= 2000)) ;
  vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;
ods pdf close;
```
Figure 6.4  PDF Output Using the Printer Style

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

Output in SAS Studio

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

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

To see which edition of SAS Studio you are using, select About SAS Studio in the SAS Studio Help menu. For more information about SAS Studio, see SAS Studio: User’s Guide.

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

• Generate HTML5, PDF, and RTF output by default, and view the HTML5 output in the RESULTS tab.

• Download your generated output using the download buttons for the three default output types.
• Change the default output style for each destination using the Preferences window.

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

**Viewing the Default HTML5 Results**

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

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

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

**Customizing the SAS Studio Output Environment**

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

• generate results for another output destination, such as HTML, PowerPoint, LISTING, and so on
• send your results directly to another location
• use a custom style such as a corporate style for your results
• use an image format other than the default for your graphics
• create a drill-down graph
• create an animated graph

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

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

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

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

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

   *Note:* The directories that you specify must already exist, and you must have Write access to the directories.
2. To conserve system resources, disable the default output environment using the following statement:
   
   ods _all_ close;

3. Open the desired ODS destination. Use the PATH= option to specify the file reference that you created for your ODS output. If you created a separate file reference for your image files, use the GPATH= option to specify the image output file reference. Here is an example.
   
   ods html path=odsout gpath=odsiout file="saleschart.html";

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

5. Close your ODS destination.

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

**Generate Output for Other Output Destinations**

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

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

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

**Send Your Results to Another Location**

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

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

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

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

**Use a Custom Style for Your Output**

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

Overview of Devices and Destinations

SAS/GRAPH procedures use device drivers to generate graphics output. Device drivers determine the format of your graphics output. For example, the GIF device driver generates GIF image files. The SASPRTC device generates graphics output for the current printer as determined by the PRINTPATH= system option (or the SYSPRINT= system option on Windows).
Every ODS destination has a default device driver that is associated with it. For example, the default device driver for the HTML destination is PNG. By default, when you send output to the HTML destination, your graphics output is rendered as a PNG file. (An HTML file is also generated. This HTML file contains any non-graphical output generated by your application plus an `<IMG>` tag that links to the PNG output that was generated.)

Each destination supports several devices. For example, the HTML destination supports the SVG, PNG, and GIF devices, in addition to several others. “Viewing the List of All Available Devices” on page 83 describes how to display the entire list of devices that are available. Table 6.2 on page 63 lists the default and supported devices for the LISTING, EPUB, HTML, HTML5, RTF, PDF, PowerPoint, and PRINTER destinations.

### Specifying the DEVICE= Graphics Option

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

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

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

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

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

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

For details, see “GOPTIONS Statement” on page 375. “Summary of Default Destinations, Styles, and Devices” on page 62 describes the supported devices for the HTML, LISTING, RTF, and PDF destinations. See also Chapter 9, “Using Graphics Devices,” on page 79.

### Summary of Default Destinations, Styles, and Devices

Each destination has a default device and default style that are used if you do not specify otherwise. Also, each destination has a set of recommended devices. Table 6.2 on page 63 summarizes this information for the HTML, LISTING, RTF, and PDF destinations.

You can use any style with any destination. If you specify a device with the GOPTIONS DEVICE= option, you should specify a device that is compatible with all of the destinations that you have open.
### Table 6.2 Default Devices and Styles for Commonly Used ODS Destinations

<table>
<thead>
<tr>
<th>ODS destination</th>
<th>Default device</th>
<th>Default style</th>
<th>Default output</th>
<th>Recommended devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB</td>
<td>PNG</td>
<td>Daisy</td>
<td>EPUB file</td>
<td>SVG, PNG, or GIF</td>
</tr>
<tr>
<td>HTML</td>
<td>PNG</td>
<td>HTMLBlue</td>
<td>HTML and PNG files</td>
<td>SVG, PNG, GIF, JAVA, JAVAIMG, ACTIVEX, or ACTXIMG***†</td>
</tr>
<tr>
<td>HTML5</td>
<td>SVG</td>
<td>HTMLBlue</td>
<td>HTML with inline SVG</td>
<td>SVG, PNG, GIF, JAVAIMG, or ACTXIMG**</td>
</tr>
<tr>
<td>LISTING</td>
<td>WIN (Windows)</td>
<td>Listing</td>
<td>Graphics output is displayed in the GRAPH window.*</td>
<td>All devices except JAVA and ACTIVEX</td>
</tr>
<tr>
<td></td>
<td>XCOLOR (UNIX)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBMPCGX (z/OS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDF</td>
<td>SASPRTC</td>
<td>Pearl</td>
<td>PDF file</td>
<td>SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)</td>
</tr>
<tr>
<td>PowerPoint</td>
<td>PNG</td>
<td>PowerPoint Light</td>
<td>PPTX file</td>
<td>PNG, GIF, EMF, BMP, TIFF, JAVAIMG, ACTXIMG</td>
</tr>
<tr>
<td>PRINTER</td>
<td>SASPRTC</td>
<td>Pearl</td>
<td>Controlled by the PRINTERPATH= system option (and by the SYSPRINT= system option on Windows)**</td>
<td>SASPRTC (color) SASPRTG (gray scale) SASPRTM (monochrome)</td>
</tr>
<tr>
<td>RTF</td>
<td>EMF</td>
<td>Rtf</td>
<td>RTF file (with embedded metafile)</td>
<td>EMF, SASEMF, PNG, PNG300, ACTIVEX, ACTXIMG***†</td>
</tr>
</tbody>
</table>

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

** 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.

*** The ACTXIMG device is available only on Windows platforms.

† For more information about the JAVA and ACTIVEX devices, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.

**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

---

*The content is formatted to align with the guidelines for readability and natural interpretation.*
HTML destination. Web browsers cannot display images in high resolution, so high resolution images appear very large.

Sending Output to Multiple Open Destinations

**Tips for Sending Output to Multiple Destinations**

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

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

**Closing Destinations to Save System Resources**

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

**Specifying Devices and Styles with Multiple Open Destinations**

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

For example, suppose your SAS program specifies the following:

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

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

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

```sas
goptions device=activex;
ods html style=analysis;
ods rtf style=analysis;
```
If you specify a device that is not supported for an open destination, SAS/GRAPH switches to the default device for that destination and prints a warning to the SAS log.

**Related Topics**

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

- Chapter 11, “SAS/GRAPH Output,” on page 105
- “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 156 describes the devices that can generate non-interactive graphs that contain interactive capabilities such as drill-down links or data tips. These graphs are often used in web presentations. “Developing Web Presentations with the JAVA IMG and ACTX IMG Devices” on page 157 describes the options available for creating a web presentation with interactive graphs. For details about displaying web presentations using either the ActiveX control or the Java applet, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide

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

- Chapter 10, “Managing Your Graphics with ODS,” on page 97
- Chapter 20, “Using ODS Styles, Device Parameters, and Options,” on page 273

For more information about using the graphics devices, see Chapter 9, “Using Graphics Devices,” on page 79.
Chapter 7

Using Run-Group Processing

Using Run-Group Processing

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

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

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

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

RUN-group Processing with Global and Local Statements

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

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

### RUN-group Processing with BY Statements

BY statements persist in exactly the same way as global and local statements. Suppose you submit a BY statement within a RUN-group. Then the BY-group processing produces a separate graph for each value of the BY variable for the RUN-group in which you submit it. This continues for all subsequent RUN-groups until you cancel the BY statement or exit the procedure. Thus, as you submit subsequent action statements, you continue to get multiple graphs (one for each value of the BY variable). For more information, see the “BY Statement” on page 370.

### RUN-group Processing with the WHERE Statement

The WHERE statement enables you to graph only a subset of the data in the input data set. If you submit a WHERE statement with a RUN-group, the WHERE definition remains in effect for all subsequent RUN-groups until you exit the procedure or reset the WHERE definition.

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

- The ANNOTATE= option must be included in the action statement.
- The WHERE variable must occur in both the input data set and the Annotate data set.
The result of most SAS/GRAPH procedures is the graphic display of data in the form of graphics output. **Graphics output** consists of commands that tell a graphics device how to draw graphics elements. A **graphics element** is a visual element of graphics output. Examples include a plot line, a bar, a footnote, the outline of a map area, or a border.

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

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

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

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

This section describes how you can change the dimensions of the output and display areas. It discusses how these changes in dimension affect the output. It will also describe the types of units that you can specify for your output. For a description of the graphics options and device parameters referred to in this section, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

Controlling Dimensions

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

Since the dimensions of the device display area are typically the same as the dimensions of the graphics output area, the default value of HSIZE and VSIZE is 0. However, for hard-copy devices, because the XMAX, YMAX values represent the outer boundaries of the output medium (such as a sheet of paper), these devices might need a margin. Therefore, HSIZE, VSIZE, HORIGIN, and VORIGIN are assigned default values and the default graphics output area is somewhat smaller than the device's display area. Figure 8.1 on page 71 illustrates such a device.

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

**Figure 8.1  Default Dimensions of the PSCOLOR Device**

For additional discussion of how the default values for HSIZE and HORIGIN are determined using the value of the LEFTMARGIN option, see “HSIZE” on page 574 and “HORIGIN” on page 572.

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:

\[
\text{right-margin} = \text{XMAX} - (\text{HSIZE} + \text{HORIGIN}) \\
\text{top-margin} = \text{YMAX} - (\text{VSIZE} + \text{VORIGIN})
\]

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

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

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

\[
x\text{-resolution} = \frac{\text{XPIXELS}}{\text{XMAX}}
\]

Therefore, the X resolution of the PSCOLOR device illustrated in Figure 8.1 on page 71 is 300dpi (dots per inch).

Ordinarily, you do not want to change the image resolution because changing it might distort your image. However, you might want to change the size of the display area. To do so without changing the resolution, use the GOPTIONS statement to change either the values of XPIXELS= and YPIXELS=, or the values of XMAX= and YMAX=.

SAS/GRAPH automatically calculates the correct value for the unspecified parameters so that the device retains the default resolution.

For information about controlling the resolution of your image see “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” on page 115.

Units

Cells

Within the graphics output area, SAS/GRAPH defines an invisible grid of rows and columns. This grid consists of character cells as shown in Figure 8.2 on page 73.

The size and shape of these cells affect the size and appearance of your graph since each graphics element is drawn using units of cells. The size and shape of the cells are determined by the size of the graphics output area. They are also determined by the number of rows and columns that SAS/GRAPH has defined in the grid. You can control the number of rows by specifying the LROWS device parameter (for a landscape orientation) or the PROWS device parameter (for a portrait orientation). Similarly, the number of columns is controlled by the LCOLS (landscape) or PCOLS (portrait) device parameter.

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

Figure 8.2 on page 73 illustrates how device parameter settings for the size of the output area relate to the parameter settings for the number of character cells in the output area.
Figure 8.2  Rows, Columns, and Cells in the Graphics Output Area

Changing only the outer dimensions of the graphics output area (HSIZE= and VSIZE=) retains the cell size. However, this causes SAS/GRAPH to automatically recalculate the number of rows and columns, as illustrated in Figure 8.3 on page 73.

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

Changing only the number of rows and columns (HPOS and VPOS) changes the size of the cells without altering the overall size of the output. Figure 8.4 on page 73 shows how increasing the number of rows and columns reduces the size of the individual cells.

Figure 8.4  Changing HPOS= and VPOS= Changes Cell Size

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

To change all the attributes of the graphics output area, specify values for all four options, as shown in Figure 8.5 on page 74.

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

<table>
<thead>
<tr>
<th>Options Specified</th>
<th>Options Not Specified</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSIZE= and VSIZE=</td>
<td>HPOS= and VPOS=</td>
<td>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.</td>
</tr>
<tr>
<td>(or specify HPOS=0 and VPOS=0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPOS= and VPOS=</td>
<td>HSIZE= and VSIZE=</td>
<td>Keeps the external dimensions but changes the cell size according to the number of rows and columns.</td>
</tr>
<tr>
<td>HSIZE=, HPOS=, VSIZE=, and VPOS=</td>
<td></td>
<td>Changes the dimensions of the graphics output area, the number of rows and columns, and recalculates the cell size.</td>
</tr>
</tbody>
</table>

**Other Units**

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

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

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

The percent unit specification is often a good choice because it changes the height of the graphics elements in proportion to the size of the graphics output area. You can specify the unit for individual graphics elements, or you can use the GUNIT= graphics option to set the units for most graphics element heights.

---

**Maintaining the Quality of Your Image across Devices**

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

**Maintaining Proportions**

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

```sas
goptions gunit=pct;
```

You can also specify PCT anywhere you specify a size:

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

See a complete description of the graphics option “GUNIT” on page 568.

**Getting the Colors That You Want**

ODS styles are designed to provide optimal results for a variety of devices. Therefore, you use the STYLE= option in the ODS statement to specify a style most appropriate for your device. For example, you might want to specify the ODS style `Journal` since it works well with black and white devices. You can also set a different style for each ODS output destination. For information about ODS styles and destinations, see “Specifying Devices and Styles with Multiple Open Destinations” on page 64. You can compare colors and patterns for different devices and choose the device that has the fewest colors. A slide camera, for example, offers more than 16 million colors from which to chose, but some graphics monitors display significantly fewer colors.

**Previewing Your Output**

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

```sas
goptions targetdevice=pscolor;
```
By default, SAS/GRAPH software positions certain graphics elements in predefined locations in the graphics output area.

Figure 8.6 on page 76 shows the graphics output area and the areas within it that are used by the following graphics elements:

- Titles are placed in the title area at the top of the graphics output area.
- Footnotes are placed in the footnote area at the bottom of the graphics output area.
- The graph itself uses the *procedure output area*, which is the area left after the titles and footnotes have been drawn.
- Legends use the procedure output area and can affect the amount of space available for the graph. By default, space is reserved for the legend below the axis area of a graph and above the footnote area. However, you can position the legend in the part of the procedure output area that is reserved for the graph. For details, see the “LEGEND Statement” on page 377.

**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 447 for details. For destinations other than the LISTING destination, some graphics elements, such as the title and footnote, can appear in the graphics output instead of the procedure output area.

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

**Note:** If the titles, footnotes, and legend are very large, they can make the procedure output area too small for the graph. You can control the size of title and footnote text and of most legend elements with statement options. For details, see Chapter 24, “SAS/GRAPH Statements,” on page 343 for a description of the appropriate statement. In addition, the “GOPTIONS Statement” on page 375 lists the graphics options that control the size of various graphics elements.
Sometimes SAS/GRAPH cannot fit one or more graphics elements on the graph. This can happen if an element is too big for the available space. An example is when the title is too long. Or perhaps you have too many elements to fit in a given space. An example of this is a bar chart with too many bars. In these cases, SAS/GRAPH does one of the following:

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

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

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Using Graphics Devices</td>
<td>79</td>
</tr>
<tr>
<td>What Is a SAS/GRAPH Device?</td>
<td>80</td>
</tr>
<tr>
<td>Commonly Used Devices</td>
<td>80</td>
</tr>
<tr>
<td>Default Devices for ODS Destinations</td>
<td>82</td>
</tr>
<tr>
<td>Viewing the List of All Available Devices</td>
<td>83</td>
</tr>
<tr>
<td>Displaying Information with the QDEVICE Procedure</td>
<td>84</td>
</tr>
<tr>
<td>Deciding Which Device to Use</td>
<td>85</td>
</tr>
<tr>
<td>Overriding the Default Device</td>
<td>86</td>
</tr>
<tr>
<td>Device Categories and Modifying Default Output Attributes</td>
<td>87</td>
</tr>
<tr>
<td>Using Universal Printer Shortcut Devices</td>
<td>89</td>
</tr>
<tr>
<td>Creating Universal Printers and Shortcut Devices</td>
<td>91</td>
</tr>
<tr>
<td>Viewing and Modifying Device (DEV) Entries</td>
<td>91</td>
</tr>
<tr>
<td>Viewing the Contents of a Device Entry</td>
<td>92</td>
</tr>
<tr>
<td>Modifying Device Entry Parameters</td>
<td>92</td>
</tr>
<tr>
<td>Creating a Custom Device</td>
<td>93</td>
</tr>
<tr>
<td>Compatibility Device Drivers (Z Drivers)</td>
<td>93</td>
</tr>
<tr>
<td>Additional Considerations for Devices</td>
<td>93</td>
</tr>
<tr>
<td>Related Topics</td>
<td>94</td>
</tr>
<tr>
<td>Dictionary</td>
<td>94</td>
</tr>
<tr>
<td>%SHORTCUT Autocall Macro</td>
<td>94</td>
</tr>
</tbody>
</table>
What Is a SAS/GRAPH Device?

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

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

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

See also “Viewing and Modifying Device (DEV) Entries” on page 91.

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

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

Commonly Used Devices

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

Table 9.1 SAS/GRAPH Devices and the Output They Generate

<table>
<thead>
<tr>
<th>Device</th>
<th>External files</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVEX</td>
<td>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 SAS/GRAPH ActiveX control. When the HTML or RTF file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive ActiveX control. The user must install the ActiveX control to view the output of the ACTIVEX device.</td>
</tr>
<tr>
<td>Device</td>
<td>External files</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ACTXIMG</td>
<td>A PNG file that contains an image of the graph that is generated with the ACTIVEX device. The user does not need to install the SAS/GRAPH ActiveX control to view the output from the ACTXIMG device.</td>
</tr>
<tr>
<td>BMP</td>
<td>A BMP file that contains the graph.</td>
</tr>
<tr>
<td>CGM</td>
<td>A CGM file that contains the graph.</td>
</tr>
<tr>
<td>CGMOF97L</td>
<td>A CGM file suitable for inserting into Microsoft Word or PowerPoint presentations.</td>
</tr>
<tr>
<td>EMF</td>
<td>An EMF file that contains the graph. The file contains EMF Plus language extensions.</td>
</tr>
<tr>
<td>EMFDUAL</td>
<td>An EMF file that contains the graph. The file contains both traditional EMF and EMF Plus language extensions.</td>
</tr>
<tr>
<td>GIF</td>
<td>A GIF file that contains the graph.</td>
</tr>
<tr>
<td>IBMPCGX</td>
<td>Display device. This device is available on z/OS hosts only.</td>
</tr>
<tr>
<td>JAVA</td>
<td>This device is used with the ODS HTML destination. It generates 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.</td>
</tr>
<tr>
<td>JAVAIMG</td>
<td>A PNG file that contains a graph that is generated with the JAVA device.</td>
</tr>
<tr>
<td>JPEG</td>
<td>A JPG file that contains the graph. You can use the JPEGQUALITY= system option to control the image quality and compression of JPEG images. See SAS System Options: Reference for more information.</td>
</tr>
<tr>
<td>PCL5</td>
<td>A PCL file that contains one or more graphs and tables.</td>
</tr>
<tr>
<td>PDF</td>
<td>A PDF file that contains one or more graphs and tables.</td>
</tr>
<tr>
<td>PNG</td>
<td>A PNG file that contains the graph. This is often the default driver when using SAS both in Batch mode and in the window environment.</td>
</tr>
<tr>
<td>PSCOLOR</td>
<td>A PostScript file that contains one or more graphs.</td>
</tr>
<tr>
<td>PSL</td>
<td>A PostScript file that contains the graph in gray scale.</td>
</tr>
<tr>
<td>SASEMF</td>
<td>An EMF file that contains the graph. The file contains traditional EMF language.</td>
</tr>
<tr>
<td>SVG</td>
<td>An SVG file that contains the graph.</td>
</tr>
<tr>
<td>TIFF</td>
<td>A TIFF file that contains the graph in color. This device uses RGBA colors.</td>
</tr>
</tbody>
</table>
Device | External files
---|---
TIFFK | A TIFF file that contains the graph in color. This device uses CMYK colors.
WIN | Display device. This device is available on Windows hosts only.
XCOLOR | Display device. This device is available on UNIX hosts only.

* For detailed information about JAVA and ACTIVEX devices, see "SAS/GRAPH: Java Applets and ActiveX Control User’s Guide."

Note: “The SAS/GRAPH ACTIVEX, JAVA, and JAVAMETA Graphics Output Devices” in "SAS/GRAPH: Java Applets and ActiveX Control User’s Guide" describes any requirements or limitations associated with using the ActiveX and Java devices.

Note: Chapter 13, “Using SVG Graphics,” on page 141 describes any requirements or limitations associated with using the SVG device.

### Default Devices for ODS Destinations

Each ODS destination has a default device. Table 9.2 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 86.

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

<table>
<thead>
<tr>
<th>ODS destination</th>
<th>Default device</th>
<th>Default style</th>
<th>Default output</th>
<th>Recommended devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB</td>
<td>PNG</td>
<td>Daisy</td>
<td>EPUB file</td>
<td>SVG, PNG, or GIF</td>
</tr>
<tr>
<td>HTML</td>
<td>PNG</td>
<td>HTMLBlue</td>
<td>HTML and PNG files</td>
<td>SVG, PNG, GIF, JAVA, JAVAIMG, ACTIVEX, or ACTXIMG***</td>
</tr>
<tr>
<td>HTML5</td>
<td>SVG</td>
<td>HTMLBlue</td>
<td>HTML with inline SVG</td>
<td>SVG, PNG, GIF, JAVAIMG, or ACTXIMG***</td>
</tr>
<tr>
<td>LISTING</td>
<td>WIN (Windows)</td>
<td>Listing</td>
<td>Graphics output is displayed in the GRAPH window.*</td>
<td>All devices except JAVA and ACTIVEX</td>
</tr>
<tr>
<td></td>
<td>XCOLOR (UNIX)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBMPCGX (z/OS)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Graphics output is displayed in the GRAPH window.
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 1055 for details.
Use GDEVICE procedure statements to write the list of device drivers to the Output window. For example:

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

The “NOFS” on page 1050 option in the PROC GDEVICE statement causes the procedure not to use the GDEVICE window.

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

- save the contents of the Output window.
- use the PRINTTO procedure to redirect the GDEVICE procedure output to an external file. See Base SAS Procedures Guide for a description of the PRINTTO procedure.
- Use the QDEVICE procedure to write the list of all device drivers to a SAS data set. Specify the keyword _ALL_ in the DEVICE statement. See “Displaying Information with the QDEVICE Procedure” on page 84 for more information.

### Displaying Information with the QDEVICE Procedure

The QDEVICE procedure produces reports about graphics devices and Universal Printers. You can use the information in these reports to determine the best device or printer to use for a specific application. See “QDEVICE Procedure” in Base SAS Procedures Guide for more information.

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

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

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

The following code prints specific information from WORK.MYREPORT.

```sas
proc print data=myreport;
    var name desc type clrspace
    height width units;
run;
```
You can display information about all of the available device drivers by specifying the keyword **_ALL_** in the DEVICE statement. For example, the following code produces a GENERAL report for all devices and sends this report to WORK. ALLDEV.

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

---

### Deciding Which Device to Use

The default device for each ODS destination generates optimal results for that destination. It is recommended that you use the default device whenever possible. If you do not specify a device, then SAS/GRAPH automatically uses the default device listed in Table 9.1 on page 80 for each open destination.

*Note:* If you are working with multiple open destinations, see “Specifying Devices and Styles with Multiple Open Destinations” on page 64.

If you need to specify a different device, you should specify one of the recommended devices in the table in Table 9.1 on page 80. If you specify a device that cannot be used with an open destination, SAS/GRAPH switches to a device that produces similar results as the device that you specified.

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

When you are sending output to the HTML destination, there are several devices that you can specify. See “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 156 for information and recommendations on which device to use.

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

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

---

<table>
<thead>
<tr>
<th>Obs</th>
<th>NAME</th>
<th>DESC</th>
<th>TYPE</th>
<th>CLRSPACE</th>
<th>HEIGHT</th>
<th>WIDTH</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PNG</td>
<td>Portable Network Graphics</td>
<td>Shortcut Device</td>
<td>RGBA</td>
<td>6.25</td>
<td>8.33</td>
<td>IN</td>
</tr>
<tr>
<td>2</td>
<td>IMGPNG</td>
<td>Portable Network Graphics Format -- Perf</td>
<td>Graph Device</td>
<td>RGBA</td>
<td>6.25</td>
<td>8.33</td>
<td>IN</td>
</tr>
<tr>
<td>3</td>
<td>GIF</td>
<td>Graphics Interchange Format RGB Color/Alpha Blending</td>
<td>Shortcut Device</td>
<td>RGBA</td>
<td>6.25</td>
<td>8.33</td>
<td>IN</td>
</tr>
<tr>
<td>4</td>
<td>IMGGIF</td>
<td>Graphics Interchange Format -- Performan</td>
<td>Graph Device</td>
<td>RGBA</td>
<td>6.25</td>
<td>8.33</td>
<td>IN</td>
</tr>
<tr>
<td>5</td>
<td>SVG</td>
<td>Scalable Vector Graphics 1.1</td>
<td>Universal Printer</td>
<td>RGBA</td>
<td>6.25</td>
<td>8.33</td>
<td>IN</td>
</tr>
<tr>
<td>6</td>
<td>PDF</td>
<td>Portable Document Format Version 1.4</td>
<td>Universal Printer</td>
<td>RGBA</td>
<td>10.50</td>
<td>8.00</td>
<td>IN</td>
</tr>
</tbody>
</table>
Starting with SAS 9.4M5, for SAS jobs that generate a large number of graphs, using the IMGPNG or IMGGIF device might improve job performance. When used to render a large number of graphs, the performance of these devices is better than their corresponding Universal Printer shortcut devices, PNG and GIF.

Overriding the Default Device

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

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

  ```
goptions device=gif;  
  ```

  For details, see “GOPTIONS Statement” on page 375.

- Specify the name of a device entry with the DEVICE= option in an OPTIONS statement. For details, see “DEVICE= System Option” on page 629.

- Specify the name of the device entry with the DEVICE= option in an ODS destination statement. You can specify this option on the ODS HTML, ODS LISTING, ODS MARKUP, and ODS RTF statements. When you specify a device in an ODS destination statement, do not specify the ACTIVEX, ACTXIMG, JAVA, or JAVAIMG devices. For additional information about these ODS destination statements, see *SAS Output Delivery System: User’s Guide*.

- Enter OPTIONS on the SAS command line, or select Tools ⇒ Options ⇒ System to open the SAS System Options window. Expand Graphics, and select Driver settings. Right-click Device, select Modify value, and specify the name of the graphics device that you want to use.

- Enter the device name in the DEVICE prompt window. This window appears automatically if you submit a SAS/GRAPH program that produces graphics output, no device has been specified, and you are running outside of the SAS windowing system environment.

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

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

You can also specify a device for previewing or printing your output with the TARGETDEVICE= graphics option. For details, see “Printing Your Graph” on page 131.

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

Note: Chapter 20, “Using ODS Styles, Device Parameters, and Options,” on page 273 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 OPTIONS statement is not sufficient. If you need to modify a device entry, consider contacting SAS Technical Support for assistance first.

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

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

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

In output from the QDEVICE procedure, a traditional SAS/GRAPH device is identified as a “Graph Device”. See Figure 9.1 on page 85 for more information.

With traditional SAS/GRAPH devices, you can specify default attributes for your output by customizing the device entry (the DEV catalog entry). For example, by editing the DEV catalog entry for the device, you can change several attributes. You can change the default size and resolution of your output and the default colors and fonts that are used when styles are turned off.

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

Note: For more information about the JAVA and ACTIVEX devices, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide

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

Universal Printer shortcut devices can generate output in the following formats: PDF, PostScript, PCL, PNG, GIF, EMF, TIFF, and SVG. For example, TIFFK, EMFDUAL, PNG, and SVG are Universal Printer shortcut devices. The description of a Universal Printer shortcut device frequently says “Universal Printer” when you view the contents of the SASHELP.DEVICES catalog. In output from the QDEVICE procedure, a Universal Printer shortcut device is identified as a “Shortcut Device”. See Figure 9.1 on page 85 for an example. See “Viewing the List of All Available Devices” on page 83. The list of all Universal Printer shortcut devices is shown in Table 9.4 on page 90.

Universal Printer shortcut devices are designed to emulate a traditional SAS/GRAPH device, which means that these devices behave as much as possible like traditional...
SAS/GRAPH devices. For example, these devices set the value of PRINTERPATH= so that you need only specify the device name with the GOPTIONS statement. However, for these devices there are some attributes of your output, such as default resolution, that cannot be changed by modifying the DEV catalog entry. See “Using Universal Printer Shortcut Devices” on page 89 for more information.

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

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

Table 9.3 on page 88 summarizes the device categories and describes how to control output attributes based on the device category. For more information about the JAVA and ACTIVEX devices, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide. See also “Viewing and Modifying Device (DEV) Entries” on page 91.

Table 9.3  Device Categories, GOPTIONS, and DEV Entries

<table>
<thead>
<tr>
<th>Device category</th>
<th>Examples</th>
<th>Honor GOPTIONS specifications?</th>
<th>Honor specifications in the device (DEV) catalog entry?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut devices</td>
<td>EMF, GIF, PNG, SASEMF, SVG</td>
<td>Yes*, except for resolution†</td>
<td>Yes, except for size, resolution, and fonts</td>
</tr>
<tr>
<td>Interface devices</td>
<td>SASPRTC, SASPRTG, SASPRTM</td>
<td>Yes, except for resolution††‡</td>
<td>Yes, except for size, resolution, and fonts</td>
</tr>
<tr>
<td>Printer**</td>
<td>WIN, XCOLOR</td>
<td>Yes, except for resolution‡‡</td>
<td>Yes, except for size‡, resolution‡‡, and fonts</td>
</tr>
<tr>
<td>Display</td>
<td>BMP**</td>
<td>Yes</td>
<td>Yes, except for resolution‡‡ and fonts</td>
</tr>
<tr>
<td>Device category</td>
<td>Examples</td>
<td>Honor GOPTIONS specifications?</td>
<td>Honor specifications in the device (DEV) catalog entry?</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Traditional SAS/GRAPH</td>
<td>BMP**</td>
<td>Yes***</td>
<td>Yes</td>
</tr>
<tr>
<td>devices</td>
<td>CGM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBMPCGX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IMGGIF‡‡‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IMGPNG‡‡‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JAVAMETA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIFFP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java and ActiveX devices</td>
<td>ACTIVEX</td>
<td>Yes, except as noted in the documentation for specific graphics options. Also, resolution is controlled by the operating environment.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ACTXIMG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JAVA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JAVAIMG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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.

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

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

† 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.

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

†† 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.

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

‡‡ 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.

‡‡‡ Valid starting with SAS 9.4M5.

---

**Using Universal Printer Shortcut Devices**

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

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

- convert SAS/GRAPH GRSEG output into instructions understood by Universal Printers.
Using a Universal Printer shortcut device requires that there is a Universal Printer with the same name in the SAS registry. Universal printers have already been defined for all of the shortcut devices that are shipped with SAS. You can create your own device by copying one of the Universal Printer shortcut device entries. However, you must make sure that you define a Universal Printer with the same name as your new device entry. For information about creating a new SAS/GRAPH device, see “Creating a Custom Device” on page 93. For information about defining a new Universal Printer, see “Define a New Printer” in 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 9.4.

Table 9.4 Differences in Using Shortcut Devices and Universal Printers

<table>
<thead>
<tr>
<th>Using a shortcut device</th>
<th>Using a Universal Printer directly</th>
</tr>
</thead>
<tbody>
<tr>
<td>**goptions device=PNG; <em>/ procedure step <em>/</em></em></td>
<td>The following two sets of code are equivalent.</td>
</tr>
<tr>
<td>**ods printer printer=PNG; <em>/ procedure step <em>/</em></em></td>
<td>**ods printer close; <em>/ procedure step <em>/</em></em></td>
</tr>
<tr>
<td>options printerpath=PNG;</td>
<td>options printerpath=PNG;</td>
</tr>
<tr>
<td>ods printer;</td>
<td>ods printer;</td>
</tr>
<tr>
<td>ods printer close;</td>
<td>ods printer close;</td>
</tr>
</tbody>
</table>

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, saspot.png).

Table 9.5 lists all of the Universal Printer shortcut devices that are provided by SAS.

Table 9.5 Universal Printer Shortcut Devices

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>GIF Universal Printer</td>
</tr>
<tr>
<td>EMF</td>
<td>Enhanced Metafile Format Plus Extensions (EMF Plus)*</td>
</tr>
<tr>
<td>EMFDUAL</td>
<td>EMF and EMF Plus Format Support*</td>
</tr>
<tr>
<td>PCL5</td>
<td>PCL5 Universal Printer</td>
</tr>
<tr>
<td>PCL5C</td>
<td>PCL5c Universal Printer</td>
</tr>
<tr>
<td>PCL5E</td>
<td>PCL5e Universal Printer</td>
</tr>
<tr>
<td>PDF</td>
<td>PDF Version 1.3 — color</td>
</tr>
<tr>
<td>PDFA</td>
<td>Archive PDF - ISO-19005-1/b</td>
</tr>
</tbody>
</table>
SAS can produce EMF files in three different formats: EMF, EMF Plus, and EMF Dual. EMF Dual files contain both EMF and EMF Plus records. The EMF viewer determines which set of records to display. See “EMF Graphics in SAS” in SAS Language Reference: Concepts for more information.

Creating Universal Printers and Shortcut Devices

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

For information about this macro, see “%SHORTCUT Autocall Macro” on page 94.

Viewing and Modifying Device (DEV) Entries

As described in “What Is a SAS/GRAPH Device?” on page 80, 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFC</td>
<td>PDF Version 1.3 — color</td>
</tr>
<tr>
<td>PNG</td>
<td>PNG Universal Printer</td>
</tr>
<tr>
<td>PNG300</td>
<td>PNG Universal Printer-300 dpi</td>
</tr>
<tr>
<td>PNGT</td>
<td>PNG Universal Printer with Transparency</td>
</tr>
<tr>
<td>PSCOLOR</td>
<td>PostScript Level 1 (Color)</td>
</tr>
<tr>
<td>PSL</td>
<td>PostScript Level 1 (Gray Scale)</td>
</tr>
<tr>
<td>PSLEPSF</td>
<td>PostScript EPS (Gray Scale)</td>
</tr>
<tr>
<td>PSLEPSFC</td>
<td>PostScript EPS (Color)</td>
</tr>
<tr>
<td>SASEMF</td>
<td>Enhanced Metafile Format*</td>
</tr>
<tr>
<td>SVG</td>
<td>SVG Universal Printer</td>
</tr>
<tr>
<td>SVGZ</td>
<td>SVG Compressed Universal Printer</td>
</tr>
<tr>
<td>SVGVIEW</td>
<td>SVG Printer w/ Control Buttons</td>
</tr>
<tr>
<td>TIFF</td>
<td>TIFF RGBA Universal Printer</td>
</tr>
<tr>
<td>TIFFK</td>
<td>TIFF CMYK Universal Printer</td>
</tr>
</tbody>
</table>

* SAS can produce EMF files in three different formats: EMF, EMF Plus, and EMF Dual. EMF Dual files contain both EMF and EMF Plus records. The EMF viewer determines which set of records to display. See “EMF Graphics in SAS” in SAS Language Reference: Concepts for more information.
it is not always used. This is especially true for interface and shortcut devices, which derive many attributes from the operating environment and the Universal Printer definition in the SAS registry. See “Device Categories and Modifying Default Output Attributes” on page 87 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 1046 or ask your on-site SAS support personnel.

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

- In the SAS windowing environment, use the SAS Explorer window to display the contents of the DEVICES catalog in the SASHELP library. Double-click a device entry to display the contents of the device entry in the Output window.

- Run the GDEVICE procedure in program mode. For example, the following statements list the contents of the PSCOLOR device entry in the Output window:

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

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

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

  From the GDEVICE Directory window, select the device name to open the GDEVICE Detail window. From there you can move to the other GDEVICE windows for the entry, either by selecting windows from the Tools menu or entering commands on the command line. For details, see “Using the GDEVICE Windows” on page 1055.

**Modifying Device Entry Parameters**

Use the GDEVICE procedure to modify the properties of an existing device entry. See Chapter 37, “GDEVICE Procedure,” on page 1046.

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

**Note:** The device driver that is associated with a device entry is shown in the Module field in the device entry. It is recommended that you do not change the device driver associated with a device entry. Please contact SAS Technical Support before changing the device driver associated with a device entry.
Note: If you run SAS/GRAPH software in a multi-user environment, you should not change the device entries in the SASHELP.DEVICES catalog. Exceptions to this rule apply if you are the system administrator or other on-site SAS support personnel.

If you need to change a device entry in SASHELP.DEVICES, copy it into a personal catalog named DEVICES, and then modify the copy. To use the new device, assign the libref GDEVICE0 to the library that contains the modified copy. See “Creating or Modifying Device Entries” on page 1060 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 GDEVICEn.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 92 and Chapter 37, “GDEVICE Procedure,” on page 1046 for more information.

Compatibility Device Drivers (Z Drivers)

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

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

These devices ignore the FONTRENDERING= system option and force host font rendering. See “Changing the Appearance of Output to Match That of Earlier SAS Releases” on page 298 for more information.

Additional Considerations for Devices

• Some ODS styles, such as BlockPrint, include background images. Not all devices support these images. You can suppress the display of these images by specifying the NOIMAGEPRINT graphics option. See “Disabling and Enabling Image Output” on page 340.

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

• When you are replaying a graph with the GREPLAY procedure, it is recommended that you replay the graph with the same device that it was originally created with. Replaying the graph with other devices can produce undesirable results. See “Replaying Your Output Using the GREPLAY Procedure” on page 128 for additional information.
Related Topics

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

- Chapter 11, “SAS/GRAPH Output,” on page 105 provides general information about graphics output formats and the SAS/GRAPH output process. Tasks discussed include, setting the size and resolution of your graphics output and previewing on one device how output looks on another device. Tasks also include sending output directly to a printer or other hardcopy device, and replaying output.

- “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 156 describes the devices that can generate non-interactive graphs that contain interactive capabilities such as drill-down links or data tips. These graphs are often used in web presentations.“Developing Web Presentations with the JAVAIMG and ACTXIMG Devices” on page 157 describes the options available for creating a web presentation with interactive graphs. For details about displaying web presentations using either the ActiveX control or the Java applet, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.

- Chapter 17, “Using SAS/GRAPH Output with Microsoft Office Products,” on page 233 describes how to choose a device for output that you want to use in Microsoft Office products.

- Chapter 37, “GDEVICE Procedure,” on page 1046 describes how to create and modify devices.

Dictionary

%SHORTCUT Autocall Macro

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

Syntax

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

Required Argument

NAME=new-name

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

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

**Interactions**

If you do not also specify the PROTOTYPE= option, then the macro assumes that you want to create a shortcut device to an existing Universal Printer. The *new-name* must match the name of an existing Universal Printer. The list of valid printer names is available in the SAS registry at CORE\PRINTING\PRINTERS. To view the SAS registry, issue the REGEDIT command.

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

**Optional Arguments**

**PROTOTYPE=** *prototype-name*

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

**DESCRIPTION=** *description*

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

**LIBREF=** *libref*

specifies the library where the new device driver is to be created. You can specify any of the GDEVICE\* librefs, from GDEVICE0 to GDEVICE9. You can also specify the SASHELP libref, but you must have Update access to SASHELP.

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

See "Device Catalogs" on page 1046

**REGISTRY=SASUSER | SASHELP**

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

Default SASUSER

See "The SAS Registry" in *SAS Language Reference: Concepts*

**Details**

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

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

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

**Examples**

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

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

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

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

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

```sas
%shortcut(name=newEMF,prototype=EMF Plus, description=Custom EMF Plus, libref=gdevice0,registry=sasuser)
```
Chapter 10

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 sends output to each destination. The HTML destination is open by default.

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

Note: For more information about ODS destinations, see SAS Output Delivery System: User’s Guide.
The following table lists the ODS destinations and the default type of output that results from each destination.

Table 10.1 Relevant Destination Table

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

---

**Specifying a Destination**

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

¹ LATEX is an experimental tagset. Do not use this tagset in production jobs.
ODS Destination Statement Options

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

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

This ODS statement does the following:

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

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

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

The HTML output is written to the file specified by the BODY= option, `barGraph.html`. At start-up, the default folder is the WORK folder. When running SAS with the windowing environment in the Windows operating system, the current folder is displayed in the status bar at the bottom of the main SAS window. In SAS Studio, the default folder is your working directory. For more information, see “Output in SAS Studio” on page 58.

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

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

**GPATH= **

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 WORK directory. For complete
documentation on PATH= option, see the ODS MARKUP or TAGSET.RTF

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

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

STYLE= style-definition
specifies a style to be used for the output. Each ODS destination has a default style
for the formatting of output. The style specifies a collection of visual attributes that
are used for the rendering of the output. The STYLE= option is valid for all ODS
destinations except the Document destination and the Output destination. For
complete documentation on the STYLE= option, see the ODS statements in SAS
Output Delivery System: User’s Guide. For more information about using the
STYLE= option with SAS/GRAPH output, see Chapter 20, “Using ODS Styles,
Device Parameters, and Options,” on page 273.

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

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

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

---

**ODS and Procedures That Support RUN-Group Processing**

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

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

Controlling Where Titles and Footnotes Are Rendered

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

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

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

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

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

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

Using Graphics Options with ODS (USEGOPT)

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

- CTEXT=
- CTITLE=
- FTEXT=
- FTITLE=
- HTEXT=
- HTITLE=
When titles and footnotes are rendered as part of the graphic image, these options are honored regardless of whether you specify the ODS USEGOPTS statement.

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

```sas
/* Specify an ODS output path */
filename odsout ".";
ods _all_ close;
ods html path=odsout file="myout.htm" style=analysis;
goptions reset=all dev=activex htitle=8 ctext="black";
ods nousegopt;
title "My title";
footnote "My footnote";
proc gchart data=sashelp.class;
   pie age / discrete legend;
run;
odds usegopt;
   pie age / discrete legend;
run;
quit;
ods nousegopt;
odds html close;
odds html; /* Not required in SAS Studio */
```

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

The default setting is ODS NOUSEGOPT.

Controlling the Text Font, Size, and Color

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

1. SAS looks for options on the TITLE and FOOTNOTE statement. For example, you can specify BOLD, ITALIC, FONT=, or HEIGHT= options on these statements.

2. When titles and footnotes are rendered as part of the ODS output file, and the ODS USEGOPTS statement is used, SAS looks for global options in the GOPTIONS statement. Such options are CTEXT= and FTITLE=. When titles and footnotes are rendered as part of the graphic image, SAS looks for options in the GOPTIONS statement. In this case using the ODS USEGOPTS statement has no effect. For more information, see “Using Graphics Options with ODS (USEGOPT)” on page 101.

3. SAS looks for information specified in the current style.

There are occasions when titles and footnotes are rendered as part of the ODS output file, and font sizes are specified as a percentage. These font sizes are interpreted as a percentage of the size specified by the current style. When titles and footnotes are rendered as part of the graphic file, fonts sizes that are specified as a percentage are interpreted as a percentage of the graphics output area. For more information about specifying fonts and font sizes, refer to the following topics:
• “FTEXT” on page 555 and “FTITLE” on page 556
• “HTEXT” on page 575 and “HTITLE” on page 575
• “GUNIT” on page 568
• “TITLE, FOOTNOTE, and NOTE Statements” on page 447
Chapter 11
SAS/GRAPH Output

Overview of SAS/GRAPH Output ........................................... 106
SAS/GRAPH Output Terminology ........................................ 106
Supported Graphics Formats ............................................. 106
Output Types .................................................................... 107
About GRSEGs .................................................................. 108
What You Can Do with SAS/GRAPH Output ......................... 109

Specifying the Graphics Output File Type for Your Graph .......... 109
About the Output Delivery System (ODS) ............................. 109
About the Graphics Output Devices ................................... 110

The SAS/GRAPH Output Process ......................................... 112
All Devices except JAVA, JAVAIMG, ACTIVEX, and ACTXIMG .... 112
JAVAIMG or ACTXIMG Device ........................................ 112

Setting the Size of Your Graph ........................................... 113
Using the HSIZE= and VSIZE= Graphics Options to Set the Size of Your Graphics Area ........................................ 113
Using the XPIXELS= and YPIXELS= Graphics Options to Set the Size of Your Graph ........................................ 113

Setting the Resolution of Your Graph .................................. 114
About Setting the Resolution ............................................. 114
Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices .................. 115
Using a Device Variant to Set the Size or Resolution of Your Graph ......................................................... 116

Controlling Where Your Output Is Stored ............................. 116
About the ODS Default Output Directory ............................. 116
Specifying the Name and Location of Your ODS Output .......... 117
Specifying the Name and Location of Your Graphics Output Files ................................................................. 118
About Filename Indexing ................................................ 119
Specifying the Catalog Name and Entry Name for Your GRSEGs ......................................................... 120
Summary of How Output Filenames and GRSEG Names Are Handled ......................................................... 123

Replacing an Existing Graphics Output File Using the GSFMODE= Graphics Option ............................................. 125

Storing Multiple Graphs in a Single Graphics Output File ........ 126
Using Graphics Options to Store Multiple Graphs in One Graphics Output File ........................................... 126
Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File ........................................ 126

Replaying Your SAS/GRAPH Output .................................... 128
Overview of SAS/GRAPH Output

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

SAS/GRAPH Output Terminology

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

Graphics output file
A file that contains bitmapped or vector graphic information. See “Supported Graphics Formats” on page 106.

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

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

Supported Graphics Formats

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

BMP
Windows Bitmap

GIF
Graphics Interchange Format

JPEG
Joint Photographic Experts Group

PNG
Portable Network Graphics

TIFF
Tagged Image Format File

SAS/GRAPH supports the following vector file formats:
The vector-based formats:
• are usually smaller than image files
• can be edited with third–party software (except for EPS)
• support system fonts
• support font embedding with the PDF, SVG, and PostScript devices
• provide a clear image on high-resolution devices.

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

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

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

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

Output Types

The SAS graphics procedures can generate the following types of output:
• a GRSEG (except for procedures GKPI, GTILE, and GAREABAR)
• a graphics output file that contains the graph (BMP, JPG, GIF, PNG, and so on)
• an HTML file that contains XML code that is consumed by the ActiveX control 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 one or more graphs
• a PCL file that contains one or more graphs
• a PDF file that contains one or more graphs
• a PostScript file that contains one or more graphs
• an RTF file that contains one or more graphs
• an SVG file that contains one or more graphs

About GRSEGs

A GRSEG is a SAS catalog entry that contains graphics commands in a generic, device-independent format. There are few cases in which you would be concerned with the GRSEGs. One case for using the GRSEGs is when combining multiple graphs into a single graphics output file using the GREPLAY procedure. (See “Using the GREPLAY Procedure to Store Multiple Graphs in One Graphics Output File” on page 126.) 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 GKIPI, GTILE, and GAREABAR, do not support GRSEGs.

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

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

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

A GRSEG is stored in a catalog in the SAS temporary directory. The graphics instructions that are contained in the GRSEG are understood only by the SAS/GRAPH software. You cannot use third-party graphics applications to view the graphic in a GRSEG. The SAS/GRAPH software provides devices that enable you to convert a GRSEG to standard graphics formats such as GIF, PNG, and PDF, which you can view using third-party applications.

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

By default, SAS/GRAPH appends each new GRSEG to the catalog. If you create more than one graph with a procedure during a SAS session and the GRSEGs are stored in the same catalog, SAS/GRAPH software appends a number to the end of the name of subsequent GRSEGs. This number makes the names unique within the catalog. For example, if you create three graphs with the GCHART procedure during the same SAS session, the GRSEGs are named GCHART, GCHART1, and GCHART2. SAS/GRAPH software uses this naming convention whether GRSEGs are being stored in a temporary or permanent catalog.
You can supply a name and description when you create the graph by using the NAME= and DESCRIPTION= options. If you create more than one graph of the same name, the SAS/GRAPH software increments the specified name just as it does the default names.

**What You Can Do with SAS/GRAPH Output**

By default, output from SAS/GRAPH procedures that produce graphics output is sent to the ODS HTML destination using the HTMLBlue ODS style. In SAS Studio, the graphics output is sent to a special instance of the ODS HTML5, ODS RTF, and ODS PDF destinations. (See “Output in SAS Studio” on page 58.) 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 131.
- save it in a temporary or permanent SAS catalog for later replay. See “Replaying Your SAS/GRAPH Output” on page 128.
- 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 133.

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 545.

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 9, “Transporting and Converting Graphics Output,” on page 1551.
- 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 1554.
- export them to graphics output files using different graphics file formats. For details, see “Exporting Your Output” on page 133.

### 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.
About the Graphics Output Devices

The Output That Each Device Generates
By default, the SAS/GRAPH ODS writes to the HTML destination, which displays your graph in a web browser window and creates a GRSEG in the catalog. You can specify a graphics output device other than the default PNG device for the ODS HTML destination, or you can specify a different ODS destination and device. For information about using ODS to manage your graphics output, see Chapter 10, “Managing Your Graphics with ODS,” on page 97.

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

### Table 11.1 SAS/GRAPH Devices and the Output They Generate

<table>
<thead>
<tr>
<th>Device</th>
<th>External files</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVEX</td>
<td>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 SAS/GRAPH ActiveX control. When the HTML or RTF file is viewed in a browser, the SAS/GRAPH output is displayed as an interactive ActiveX control. The user must install the ActiveX control to view the output of the ACTIVEX device.</td>
</tr>
<tr>
<td>ACTXIMG</td>
<td>A PNG file that contains an image of the graph that is generated with the ACTIVEX device. The user does not need to install the SAS/GRAPH ActiveX control to view the output from the ACTXIMG device.</td>
</tr>
<tr>
<td>BMP</td>
<td>A BMP file that contains the graph.</td>
</tr>
<tr>
<td>CGM</td>
<td>A CGM file that contains the graph.</td>
</tr>
<tr>
<td>CGMOF97L</td>
<td>A CGM file suitable for inserting into Microsoft Word or PowerPoint presentations.</td>
</tr>
<tr>
<td>EMF</td>
<td>An EMF file that contains the graph. The file contains EMF Plus language extensions.</td>
</tr>
<tr>
<td>EMFDUAL</td>
<td>An EMF file that contains the graph. The file contains both traditional EMF and EMF Plus language extensions.</td>
</tr>
<tr>
<td>GIF</td>
<td>A GIF file that contains the graph.</td>
</tr>
<tr>
<td>IBMPCGX</td>
<td>Display device. This device is available on z/OS hosts only.</td>
</tr>
<tr>
<td>JAVA</td>
<td>This device is used with the ODS HTML destination. It generates 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.</td>
</tr>
</tbody>
</table>
Graphics Output Files

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

You can also use the ODS to generate SAS/GRAPH output as HTML that you can view with a web browser. Details are discussed in Chapter 10, “Managing Your Graphics with ODS,” on page 97.

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

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

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

About File Extensions
When you send SAS/GRAPH output to an aggregate file storage location, SAS/GRAPH generates the name of the graphics output file. This is done by taking the GRSEG name and adding the appropriate file extension. Most devices provide a default extension. If a device does not generate an extension, then SAS/GRAPH uses the default extension .gsf. To specify a different extension from the one SAS/GRAPH provides, use the EXTENSION= graphics option. (For details, see “EXTENSION” on page 549).

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.

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).

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

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

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

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

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

The ODS PRINTER, ODS PDF, ODS PS, and ODS PCL destinations do not honor the XPIXELS= and YPIXELS= graphics options. For these destinations, you must use the HSIZE= and VSIZE= graphics options to set the graph size. See “Using the HSIZE= and VSIZE= Graphics Options to Set the Size of Your Graphics Area” on page 113.

Here is an example that creates a 600 pixel wide by 800 pixel high graph.

```sas
options reset=all xpixels=600 ypixels=800;
proc gchart data=sashelp.cars;
  vbar Make;
  where MPG_Highway >= 37;
run;
quit;
```

Notice that XMAX= and YMAX= are not set. In this example, the SAS/GRAPH software recomputes the XMAX= and YMAX= values to retain the original resolution for the new graph size.

---

**Setting the Resolution of Your Graph**

**About Setting the Resolution**

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

- Use the IMAGE_DPI= option in an ODS LISTING or ODS HTML statement to set the image resolution, and specify one of the following Universal Printer shortcut devices: PNG, PNGT, PNG300, SVG, SVGT, SVGVIEW, or SVGZ. The default resolution for the PNG300 device is 300 DPI while the default resolution for the remaining devices that are listed is 96 DPI.
  
  Note: Devices other than those listed do not honor the IMAGE_DPI= option.

- Use the IMAGE_DPI= option in an ODS RTF statement to set the image resolution, and specify one of the following Universal Printer shortcut devices: PNG, PNGT, or PNG300.

- Use the DPI= option in an ODS PDF, ODS PCL, ODS PS, or ODS PRINTER statement to set the image resolution. The default resolution for the ODS PDF, ODS PCL, and ODS PS destinations is 150 DPI. The default resolution for the ODS PRINTER destination is the default resolution of the printer that is used. For example, the ODS PRINTER destination produces output at 300 DPI by default when the PRINTER=PDF option is used, and it produces output at 96 DPI by default when the PRINTER=PNG option is used.

- Use a traditional SAS/GRAPH device and specify the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution for graphics formats that support variable resolution. See “Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices” on page 115.

- Use a device variant to set the resolution of your graph to a specific resolution. See “Using a Device Variant to Set the Size or Resolution of Your Graph” on page 116.
Using the XPIXELS=, XMAX=, YPIXELS=, and YMAX= Graphics Options to Set the Resolution for the Traditional Devices

For the traditional SAS/GRAPH devices, you can use the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options to set the resolution of your graph. Note the following:

- The XPIXELS=, YPIXELS=, XMAX=, and YMAX= graphics options are not supported by the default display devices and the Universal Printer devices, including the shortcut devices. These graphics options are partially supported by the ACTIVEX and JAVA devices.

- The resolution of GIF and BMP images is fixed and cannot be changed using this method.

The XPIXELS= and YPIXELS= graphics options set the number of pixels for the X and Y axes respectively. The XMAX= and YMAX= graphics options set the maximum boundaries of the output on the X and Y axes respectively. The SAS/GRAPH software computes the resolution as follows:

X-resolution = XPIXELS / XMAX
Y-resolution = YPIXELS / YMAX

Table 11.2 on page 115 summarizes the effect of the XPIXELS=, XMAX=, YPIXELS=, and YMAX= graphics options have on the image resolution.

<table>
<thead>
<tr>
<th>Options Specified</th>
<th>Options Not Specified</th>
<th>SAS/GRAPH Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPIXELS= and YPIXELS=</td>
<td>XMAX= and YMAX=</td>
<td>Changes the dimensions and recalculates the value of XMAX= and YMAX= in order to retain the resolution.</td>
</tr>
<tr>
<td>XMAX= and YMAX=</td>
<td>XPIXELS= and YPIXELS=</td>
<td>Changes the dimensions and recalculates the value of XPIXELS= and YPIXELS= in order to retain the resolution.</td>
</tr>
<tr>
<td>XMAX= and XPIXELS=</td>
<td></td>
<td>Changes the horizontal dimension and recalculates the resolution.</td>
</tr>
<tr>
<td>YMAX= and YPIXELS=</td>
<td></td>
<td>Changes the vertical dimension and recalculates the resolution.</td>
</tr>
</tbody>
</table>

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.

```plaintext
ods _all_ close;
ods listing style=seaside;
```
goptions reset=all device=tiffp xpixels=1000 ypixels=1200 xmax=5in ymax=6in;
proc gchart data=sashelp.cars;
    vbar Make;
    where MPG_Highway >= 37;
run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */

Using a Device Variant to Set the Size or Resolution of Your Graph

Some of the graphics output devices have variants that produce graphics of a specific size or resolution for a given format. For example, the PNG300 and JPEG300 device variants produce 300 DPI images in the PNG and JPEG format respectively.

Note: The PNG300 and JPEG300 devices are not appropriate for use with the ODS HTML destination. These devices are used when a high-resolution graph (300 DPI) in the PNG or JPEG format is required for printing purposes. Because most browsers do not use the resolution value stored in the PNG or JPEG file, images produced by the PNG300 and JPEG300 devices appear very large when they are viewed in the browser.


Controlling Where Your Output Is Stored

About the ODS Default Output Directory

SAS Windowing Environment

With the exception of the ODS HTML destination, by default, ODS output is stored in the current SAS output directory, which is displayed on the status bar of the SAS window. A special case applies to the ODS HTML destination. When you start a new SAS session, the ODS HTML destination is open by default. This initial instance of the ODS HTML destination sends its output to the SAS WORK directory instead of the current directory. To find the path to the WORK directory, in the SAS Explorer window, you can right-click Work in the Libraries folder, and then select Properties from the pop-up menu.

When you close the initial instance of the ODS HTML destination and open a new instance, the output of the new instance and all other instances that are opened in that SAS session is sent to the current directory instead of the WORK directory. It is important that you keep this behavior in mind so that you can locate your ODS output, especially when using the ODS HTML destination.

SAS Studio

In SAS Studio, by default, output is generated in the HTML5, PDF, and RTF formats. You can download the output from SAS Studio to a file on your local machine or to a specified location. If you open your own ODS destination in SAS Studio, by default, the output is stored in your working directory. See “Output in SAS Studio” on page 58 for details.
Specifying the Name and Location of Your ODS Output

You can use the FILE= option in your ODS statement to specify where your ODS output files are stored. For the HTML destination, you can also use the PATH=, GPATH=, and the BODY= options to specify a different location for the HTML output file and the graphics output files. Here is an example that uses the FILE= ODS option with the PDF destination to send the PDF output to file mygraph.pdf.

```sas
/* Specify the ODS output path */
filename odsout ".";
goptions reset=all;
ods _all_ close;
ods pdf path=odsout file="mygraph.pdf";
proc gchart data=sashelp.prdsale;
  vbar Product / sumvar=actual;
  title1 "First Quarter Sales in Canada";
  where Quarter=1 and Country="CANADA";
  run;
quit;
ods pdf close;
ods html; /* Not required in SAS Studio */
```

Here is an example that uses the PATH=, GPATH=, and the BODY= ODS options with the HTML destination to send the HTML output to file mygraph.html in the output directory, and the graphics output file to the images subdirectory. You can open the resulting HTML output file in your web browser.

```sas
/* Specify the ODS output path */
filename odsout ".";
goptions reset=all;
ods _all_ close;
ods html                  /* The PNG device is used by default */
  path=odsout (url=none) /* HTML output directory */
  gpath="./images"       /* Graphics output file location */
  body="mygraph.html";   /* HTML filename */
proc gchart data=sashelp.prdsale; /* Run procedure to generate graphs */
  vbar Product / sumvar=actual;
  title1 "First Quarter Sales in Canada";
  where Quarter=1 and Country="CANADA";
  run;
quit;
ods html close; /* Close the output file */
ods html;       /* Not required in SAS Studio */
```

You can use BY statements to create multiple graphs.

You can specify a device other than PNG with the DEVICE= option in the GOPTIONS statement.

*Note:* The ./images subdirectory must already exist.

*Note:* With the SVG, PNG, GIF, and TIFF device, footnotes and titles are stored in the graphics output file by default. To move footnotes and titles out of the graphics output file and into the HTML file, specify the ODS HTML options NOGTITLE or NOGFOOTNOTE or both. See “Controlling Titles and Footnotes” on page 101.
For more information about the PATH=, GPATH=, and BODY= options, see SAS Output Delivery System: User’s Guide.

Specifying the Name and Location of Your Graphics Output Files

When Graphics Output Files Are Created

Many of the graphics devices such as PNG and GIF write their graphics output to a disk file when used with the ODS LISTING or ODS HTML destination. (The ODS HTML destination uses the PNG device by default.) For the remaining ODS destinations, the graphics output is embedded in the ODS output. In that case, no graphics output file is created. For more information, see “About the Graphics Output Devices” on page 110.

For the ODS LISTING and ODS HTML destinations, options are available that enable you to control the location and name of the graphics output files.

Controlling Graphics Output for ODS LISTING

For the ODS LISTING destination, you can use the GSFNAME= graphics option to specify the location of the output file. The GSFNAME= option requires a FILENAME statement that creates a file reference that points to a file or an aggregate file storage location. The syntax of the FILENAME statement is as follows:

FILENAME RefName "DirectoryOrFile";

If the file reference points to an aggregate file storage location, the graphics output files are named according to the NAME= option, if specified, or the default naming convention. If the file reference points to a file, the file specified in the FILENAME statement is used, even if the NAME= option is specified. See “Summary of How Output Filenames and GRSEG Names Are Handled” on page 123.

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

filename graphout "MyGraphs";
goptions reset=all device=png gsfname=graphout;
ods _all_ close;
ods listing;
proc gchart data=sashelp.cars;
  pie Make / name="MYCHART";
  where MSRP <= 15000;
run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */

Note: The target directory MyGraphs should already exist. Otherwise, SAS creates a file named MyGraphs.

If a MYCHART GRSEG entry does not already exist in the temporary catalog, the device sends the output to file mychart.png in the Mygraphs directory. If a MYCHART GRSEG entry already exists, the device uses an incremented name such as MYCHART1. In the previous example, you can replace the aggregate file location with a 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 112.
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`.

```sas
filename grafout ftp "'/public/sas/graphs" dir=host="unixhost73" fileext user="anonymous";
ods _all_ close;
ods listing;
goptions reset=all device=png gsfname=grafout;

/* Create our data set by sorting sashelp.cars by type */
proc sort data=sashelp.cars out=work.cars;
  by type;
run;

/* Generate the graphs */
proc gchart data=work.cars;
  vbar Make;
  title1 "30 MPG or Better";
  where MPG_Highway >= 30;
  by type;
run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */
```

This example creates four PNG files in directory `/public/sas/graphs` on host `unixhost73`. Since the GCHART procedure uses BY-group processing, the FILENAME statement includes the DIR option, which defines an aggregate file storage location. If you need to create only one graph, remove the DIR option and specify the absolute path to your graphics output file in your FILENAME statement.

### Controlling Graphics Output for ODS HTML

For the ODS HTML destination, you can use the GPATH= option in the ODS HTML statement to specify the location of the graphics output files and the plot statement NAME= option to specify the base name. Here is an example that shows how to send the output of the GCHART procedure to file `mychart.svg` in the `MyGraphs` directory.

```sas
/* Specify the ODS output path */
filename odsout ".";
goptions reset=all device=svg;
ods _all_ close;
ods html path=odsout file="mychart.html" gpath="MyGraphs";
proc gchart data=sashelp.cars;
  pie Make / name="MYCHART";
  where MSRP <= 15000;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

### About Filename Indexing

When duplicate names occur in graphics output 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 Graphics indexing and catalog-based indexing. ODS Graphics indexing is used in all ODS Graphics output and by the procedures listed in Table 11.3 on page 120. All of the other procedures use catalog-based indexing.

**Table 11.3 Filename Indexing Systems Used by SAS/GRAPH Procedures**

<table>
<thead>
<tr>
<th>Indexing System</th>
<th>How To Control Graphics Filenames</th>
<th>Procedure Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog-based</td>
<td>NAME= option in the procedure action statement</td>
<td>All procedures not listed below.</td>
</tr>
<tr>
<td>ODS Statistical Graphics</td>
<td></td>
<td>GAREABAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GKPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GTILE</td>
</tr>
</tbody>
</table>

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

The current index value for both catalog-based indexing and ODS Graphics indexing can be reset. To reset the index value for catalog-based indexing, delete the entries from the GRSEG catalog. To reset the index value for ODS Graphics indexing, use the following statement:

```latex
ods graphics / reset=index;
```

In both cases, after the index value is reset, the new output overwrites output files that already exist.

**Specifying the Catalog Name and Entry Name for Your GRSEGs**

**Using the Default Catalog and Entry Name**

If you omit the NAME= and GOUT= options, the SAS/GRAPH software uses the default naming convention to name the GRSEG entry and stores the entry in the default WORK.GSEG catalog. The GRSEG naming convention uses up to eight characters of the default name for the procedure as the base name for the GRSEG. If the name generated by the procedure duplicates an existing GRSEG, the name is incremented such as GCHART, GCHART1, GCHART2, and so on. For details, see the description of the NAME= option for a specific procedure.

If you specify a 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 .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.
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.

```sas
filename outfile "/";
goptions reset=all device=png gsfname=outfile;
ods _all_ close;
ods listing;
proc gchart data=sashelp.cars;
  pie Make / name="MYCHART";
  where MSRP <= 15000;
run;
quit;
ods listing close;
ods html; /* Not required in SAS Studio */
```

This example creates the file mychart.png in the SAS default output directory, and it creates the GRSEG Mychart in the SAS temporary catalog.

See “Summary of How Output Filenames and GRSEG Names Are Handled” on page 123 for additional information about output naming.

Specifying the Catalog and GRSEG Name with the GOUT= and NAME= Options
By default, GRSEGs are stored in the WORK.GSEG temporary catalog under the default name of the procedure that was used to generate the graph. The GRSEG name can be specified using the NAME= option, and the output catalog can be changed using the GOUT= procedure option. GRSEG names are limited to eight characters. If the NAME= option is set to a name that is more than eight characters in length, the GRSEG name is truncated to eight characters.

The name of the library and catalog in which the GRSEG is stored can be changed with the GOUT= procedure option. The GOUT= procedure option is assigned the catalog name in the format `libref.catalog` for the desired catalog. The name can be a one-level or two-level name. If a one-level name is used, the GRSEG is stored in the temporary WORK library under the specified catalog name. A two-level name can be used to specify a permanent catalog.

Here is an example that shows how to store a GRSEG generated by the GCHART procedure under entry MYCHART in the MYGRAPHS.CARS catalog.

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

Table 11.4 on page 122 summarizes the location of the GRSEG based on the NAME= and GOUT= procedure using the GCHART procedure as an example.
Table 11.4  How NAME= and GOUT= Affect the GRSEG Location

<table>
<thead>
<tr>
<th>NAME=</th>
<th>GOUT=</th>
<th>GRSEG Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not specified</td>
<td>Not specified</td>
<td>Gchart in WORK.GSEG</td>
</tr>
<tr>
<td>Not specified</td>
<td>CARS</td>
<td>Gchart in WORK.CARS</td>
</tr>
<tr>
<td>Not specified</td>
<td>MYGRAPH.CARS</td>
<td>Gchart in MYGRAPH.CARS</td>
</tr>
<tr>
<td>MYCHART</td>
<td>Not specified</td>
<td>Mychart in WORK.GSEG</td>
</tr>
<tr>
<td>MYCHART</td>
<td>CARS</td>
<td>Mychart in WORK.CARS</td>
</tr>
<tr>
<td>MYCHART</td>
<td>MYGRAPH.CARS</td>
<td>Mychart in MYGRAPH.CARS</td>
</tr>
</tbody>
</table>

Where GRSEGs Are Stored When Multiple ODS Destinations Are Used

When you send output to multiple ODS destinations, a catalog is created for the GRSEGs for each of the destinations. If the GOUT= procedure option is not specified, by default, the GRSEGs for the first destination that was opened are sent to the WORK.GSEG catalog. The GRSEGs for the subsequently opened ODS destinations are sent to a catalog that is named after the destination itself. For example, if you open the ODS LISTING, HTML, and RTF destinations, in that order, the GRSEGs are stored in the catalogs that are shown in the following table.

Table 11.5  Catalog Names for Default Catalog and Multiple ODS Destinations

<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.GSEG</td>
<td>The GRSEGs for ODS LISTING</td>
</tr>
<tr>
<td>WORK.HTML</td>
<td>The GRSEGs for ODS HTML</td>
</tr>
<tr>
<td>WORK.RTF</td>
<td>The GRSEGs for ODS RTF</td>
</tr>
</tbody>
</table>

In the default case, the GRSEGs for the first destination that is opened are stored in the WORK.GSEG catalog, regardless of the destination.

If you use the GOUT= procedure option to specify a catalog name, the GRSEGs for the first destination that you opened are sent to the catalog that is specified by the GOUT= procedure option. The GRSEGs for the subsequently opened ODS destinations are sent to a catalog that is named after the destination itself. For example, if you open the ODS HTML, LISTING, and RTF destinations, and you use the GOUT=MyGraphs.Sales procedure option, the GRSEGs are stored in the catalogs that are shown in the following table.
<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYGRAPHS.SALES</td>
<td>The GRSEGs for ODS HTML</td>
</tr>
<tr>
<td>MYGRAPHS.LISTING</td>
<td>The GRSEGs for ODS LISTING</td>
</tr>
<tr>
<td>MYGRAPHS.RTF</td>
<td>The GRSEGs for ODS RTF</td>
</tr>
</tbody>
</table>

The GRSEGs for the first destination are stored in the catalog that is specified by the GOUT= procedure option.

**Summary of How Output Filenames and GRSEG Names Are Handled**

Table 11.7 on page 123 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>
<thead>
<tr>
<th>NAME=</th>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME=&quot;FRED&quot;</td>
<td>GSFNAME= points to a file named &quot;MYGRAPH.GIF&quot; and the catalog is empty.</td>
<td>GRSEG name: FRED external filename: MYGRAPH.GIF</td>
</tr>
<tr>
<td>NAME=&quot;FRED&quot;</td>
<td>GSFNAME= points to an aggregate file storage location and the catalog is empty.</td>
<td>GRSEG name: FRED external filename: FRED.GIF</td>
</tr>
<tr>
<td>NAME=&quot;MYIMAGES&quot;</td>
<td>The NAME= value is exactly eight characters in length. GSFNAME= points to an aggregate file storage location and the catalog is empty.</td>
<td>GRSEG name:MYIMAGES external filename:MYIMAGES.GIF</td>
</tr>
<tr>
<td>NAME=&quot;WEATHEROBS&quot;</td>
<td>GSFNAME= points to an aggregate file storage location and the catalog is empty.</td>
<td>GRSEG name:WEATHEROBS external filename:WEATHEROBS.GIF</td>
</tr>
<tr>
<td>NAME= (not specified)</td>
<td>GSFNAME= points to a file named &quot;MYGRAPH.GIF&quot; and the catalog is empty.</td>
<td>GRSEG name: GCHART external filename: MYGRAPH.GIF</td>
</tr>
</tbody>
</table>
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 11.7 on page 123.

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 using the same number that is used for the GRSEG name. The way in which the incremented graphics output filename is formed depends on the length of the NAME= value as follows:

- If the length of the NAME= value is less than eight characters or more than eight characters, the number that is used in the GRSEG name is appended to the NAME= value.
- If the length of the NAME= value is exactly eight characters, the number that is used in the GRSEG name is appended to the first seven characters of the NAME= value.

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

Table 11.8 on page 124 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>
<thead>
<tr>
<th>NAME=</th>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME= &quot;FRED&quot;</td>
<td>GSFNAME= points to a file named &quot;MYGRAPH.GIF&quot; and GRSEG FRED already exists.</td>
<td>GRSEG name: FRED1, external filename: MYGRAPH.GIF</td>
</tr>
</tbody>
</table>
### 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

<table>
<thead>
<tr>
<th>NAME=</th>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
</table>
| "FRED" | GSFNAME= points to an aggregate file storage location and GRSEG FRED already exists. | GRSEG name: FRED1  
external filename: FRED1.GIF |
| "MYIMAGES" | The NAME= value is exactly eight characters in length. GSFNAME= points to an aggregate file storage location and GRSEG MYIMAGES already exists. | GRSEG name:MYIMAGE1  
external filename:MYIMAGE1.GIF |
| "WEATHEROBS" | GSFNAME= points to an aggregate file storage location and GRSEG WEATHERO already exists. | GRSEG name:WEATHER1  
external filename:WEATHEROBS1.GIF |
| (not specified) | GSFNAME= points to a file named "MYGRAPH.GIF" and GRSEG GCHART already exists. | GRSEG name:GCHART1  
external filename:MYGRAPH.GIF |
| (not specified) | GSFNAME= points to an aggregate file storage location and GRSEGs GCHART and GCHART1 already exist. | GRSEG name:GCHART2  
external filename:GCHART2.GIF |

Notice that in the NAME="MYIMAGES" example the incremented external filename is formed by appending the number to the first seven characters of the NAME= value, while in the NAME="WEATHEROBS" and NAME="FRED" examples, the incremented external filename is formed by appending the number to the full NAME= value. Be aware that the way in which the incremented graphics output filename is formed is different when the length of the NAME= value is exactly eight characters than when it is more or less than eight characters.

You cannot replace individual GRSEGs in a catalog. To replace a GRSEG, you must delete the GRSEG, and then re-create it. Therefore, even though the contents of the graphics output file are replaced, the GRSEG is not. Each time you submit the program, a new GRSEG is created, and the GRSEG name is incremented.
GSFMODE=REPLACE is used, the existing graphics output file is replaced with the new graph. However, a unique GRSEG is still generated each time you run the procedure.

See “GSFMODE” on page 565.

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.

The GSFMODE=APPEND option is useful only with the GIF graphics output device, and the PCL and PostScript Universal Printer devices. When used with other devices, such as PNG, SVG, or PDF, only the first graph is displayed in the viewer. A common application of the GSFMODE=APPEND option is in the production of animated GIFs. See “Animating GIF Images in SAS/GRAPH” on page 167.

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:
   ```plaintext
   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:
   ```plaintext
   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.

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

/* Close the currently open destinations */
ods _all_ close;

/* Specify the graphics options */
goptions reset=all device=pscolor gsfname=psout nodisplay;

/* Open the LISTING destination */
ods listing;

/* 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;

/* Close the LISTING destination and open the HTML destination */
ods listing close;
```
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 that you select. When you replay your graphs, use the same device that you used when you generated the original graphs. If you use a different device, your replayed graphs might be distorted.

You can replay your graphs to the GRAPH window for viewing or to a graphics output file. Here is an example that replays all of the graphs in the WORK.GSEG catalog for viewing:

```sas
ods html; /* Not required in SAS Studio */

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 126 and Chapter 15, “Generating Animations,” on page 167.

For information about the GREPLAY procedure, see Chapter 43, “GREPLAY Procedure,” on page 1280.

Replaying Output Using the DOCUMENT Procedure

About the DOCUMENT Procedure

For all of the SAS/GRAPH procedures, you can use the DOCUMENT procedure to replay output that you created. Use the ODS DOCUMENT destination, without having to rerun your DATA step and procedures. The ODS DOCUMENT destination creates ODS output objects for your output. You can replay the output objects at any time to your monitor or to a different device.

Creating Your ODS Document

To create an ODS document for your output, do the following in your SAS program:

1. Open ODS DOCUMENT and specify the name of the output catalog with Write permissions.
2. Close ODS HTML.
3. Open the ODS destinations that you want to send your output to.
4. Specify the device that you want to use using the DEVICE= graphics option.
5. Generate your chart.
6. Close the ODS destinations that you opened in step 3.
7. Close ODS DOCUMENT.
8. Open ODS HTML.

Here is an example that shows how to create an ODS document containing three pie charts and how to store it in catalog Mygraphs.Mydocs. The pie charts are generated with the JAVAIMG device.

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

/* Close the currently open ODS destinations */
ods _all_ close;

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

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

/* 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 HTML destination (not required in SAS Studio) */
ods html;
```

**Replaying Your ODS Document**

After you create your ODS document, use the DOCUMENT procedure to replay it. You can replay all of the graphs in your document or only those that you select. To see a list of the graphs in an ODS document, use a LIST statement with the DOCUMENT procedure. Here is an example that shows how to list the graphs in Mygraphs.Mydocs.

```sas
proc document name=Mygraphs.Mydocs;
   list / levels=all;
run;
quit;
```
A list of the graphs in the document is displayed in the Output window as shown in the following figure.

<table>
<thead>
<tr>
<th>Listing of: \Mygraphs.Mydocs\</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Order by: Insertion</strong></td>
</tr>
<tr>
<td><strong>Number of levels: All</strong></td>
</tr>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

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 currently open ODS destinations.
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 HTML destination (not required in SAS Studio).

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

```sas
options reset=all device=actximg;
ods _all_ close;
ods rtf;
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 html; /* Not required in SAS Studio */
```

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

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

```sas
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 616 for a complete description of TARGETDEVICE=.

Printing Your Graph

About Printing Graphs

You can print your SAS/GRAPH output on hard-copy devices such as a printer. Regardless of the destination, you can create a hard copy of your graph in one of the following ways:

- Print the SAS/GRAPH program output directly to a hard-copy device.
- Print the SAS/GRAPH program output by creating a graphics output file, HTML file, or PDF file, and then printing the file using host commands or host application commands.
- Print the displayed graph directly from the GRAPH or Results Viewer window or the Graphics Editor window.
- Print the displayed graph directly from a browser that supports the SVG format.
Sending Your Graph Directly to a Printer

You can send graphics output directly to a hard-copy device by sending the graphics commands directly to the device. On most systems, you can use any of the following methods to print directly to a device:

• Use the ODS PRINTER destination to send your output directly to the default printer. Use the PRINTER= option if you want to direct your output to a printer other than the default printer or if a default printer is not defined.
  

  See the SAS Language Reference: Concepts for information about how to define a default printer for the Universal Printer.

• Use a FILENAME statement, a GOPTIONS statement, and a SAS/GRAPH device. The FILENAME statement defines a file reference that points to the print commands to send your output to any available hard-copy device. The GOPTIONS statement references the file reference, assigns the device, and specifies any additional parameters.

• Use the GDEVICE procedure to modify a SAS/GRAPH device entry to spool output directly to a printer. See Chapter 37, “GDEVICE Procedure,” on page 1046 for information about adding host commands to a device entry.

• Use the Universal Printing interface.

For detailed instructions on each of these methods, refer to the SAS Help facility for SAS/GRAPH.

Saving and Printing Your Graph

You can save your graph to a graphics output file, and then print the file using host commands. You can perform these two steps separately or combine them by incorporating the host printing commands into your program or graphics output device. In any case, you must choose a graphics file format that is compatible with your printer. For example, if you are using a PostScript printer, be sure to create a PostScript file using the appropriate device for the printer.

You can use any of the following methods to create and print a graphics output file:

• Use FILENAME and GOPTIONS statements to create the graphics output file, and then use a host command to spool the file to a spooler for the device.

• Use an ODS PRINTER statement to produce a PostScript, PDF, PCL, SVG, PNG, or GIF file. Then use a host command or a host application command to send the file to the printer.

• Use the GDEVICE procedure to modify a SAS/GRAPH device to save the output to a graphics output file and spool the output directly to a printer. See Chapter 37, “GDEVICE Procedure,” on page 1046 for information about modifying device entries.

• Use the Universal Printing interface.

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

For detailed instructions on each of these methods, refer to the SAS Help facility for SAS/GRAPH.
Exporting Your Output

You can export your SAS/GRAPH output to other formats or to other software applications such as Microsoft Office. See the following topics for more information.

- “Replaying Output Using the DOCUMENT Procedure” on page 128
- Chapter 18, “Writing Your Graphs to a PDF File,” on page 241
- Chapter 17, “Using SAS/GRAPH Output with Microsoft Office Products,” on page 233
Chapter 12

Plotting a Cloud Analytic Services (CAS) In-Memory Table

Accessing In-Memory Tables

The SAS/GRAPH procedures are not included in SAS Viya. However, starting with SAS 9.4M5, the SAS/GRAPH procedures can access an in-memory table through a CAS engine libref. In that case, the data is downloaded from SAS Cloud Analytic Services to the SAS client through the CAS engine libref, and then it is processed on the SAS client by the SAS/GRAPH procedure. The amount of data that is read from an in-memory table is limited by system option CASDATALIMIT=. See “CASDATALIMIT= System Option” in SAS Cloud Analytic Services: User’s Guide. It can also be limited by the DATALIMIT= CAS engine LIBNAME statement option or the DATALIMIT= data set option. See the DATALIMIT= LIBNAME statement option and the DATALIMIT= data set option.

If the data limit is exceeded, the procedure step is terminated, and an error message is written to the SAS log. Although you can use one of these options to increase the limit, processing a large amount of data locally might degrade SAS performance. For a very large in-memory table, use a procedure that is supported by the CAS server or use a CAS action to summarize or otherwise reduce the data, and then plot the results using SAS/GRAPH. See “Preprocessing Data in CAS and Plotting the Results Using SAS/GRAPH” on page 135.

For information about how to use a CAS engine libref to access in-memory tables, see “CAS LIBNAME Engine” in SAS Cloud Analytic Services: User’s Guide. For information about SAS Cloud Analytic Services programming in SAS, see An Introduction to SAS Viya Programming.

Preprocessing Data in CAS and Plotting the Results Using SAS/GRAPH

Here is an example that shows how to summarize an in-memory table in CAS and how to plot the results using the GCHART procedure. For large in-memory tables, it is recommended that you preprocess the data in CAS to reduce the size of the data before
you plot it using a SAS/GRAPH procedure. This example uses the data in SAS data set Sashelp.Cars as sample data. The MDSUMMARY procedure is used to summarize the data before it is plotted. The MDSUMMARY procedure is supported by the CAS server and can process a large amount of data. For information about the MDSUMMARY procedure, see “MDSUMMARY Procedure” in SAS Cloud Analytic Services: User’s Guide.

Here is the SAS code for this example.

```sas
*options cashost="cloud.example.com" casport=5570; /* 1 */
cas casauto sessopts=(caslib=casuser); /* 2 */
libname mycas cas; /* 3 */
proc casutil; /* 4 */
  load data=sashelp.cars casout="cars"
  quit;

proc mdssummary data=mycas.cars; /* 5 */
  var mpg_highway;
  groupby type;
  output out=mycas.cars_summary;
run;

proc print data=mycas.cars_summary; /* 5 */
  var type _mean_
  run;

goptions reset=all border xpixels=420 ypixels=340; /* 7 */
  axis1 label=(angle=90 "Average Highway MPG");
  axis2 label="Vehicle Type"
  title "Average Highway MPG and Vehicle Type"
proc gchart data=mycas.cars_summary;
  vbar3d type / sumvar=_mean_sum
    raxis=axis1 maxis=axis2;
    format _mean_ 5.2;
  run;
quit;

cas casauto terminate;
```

1 System options CASHOST= and CASPORT= specify the host name and port for the CAS server. You must set these options if they are not already set. See system options CASHOST= and CASPORT= in SAS Cloud Analytic Services: User’s Guide.

2 The CAS statement establishes a connection with the CAS server. Session option CASLIB=CAUSER in the CAS statement specifies CASUSER as the active caslib. See CAS Statement and “CASLIB= Session Option” in SAS Cloud Analytic Services: User’s Guide.

3 The LIBNAME statement creates CAS engine libref MYCAS, which enables the SAS client to access in-memory tables in the active caslib, CASUSER. See “CAS Statement” in SAS Cloud Analytic Services: User’s Guide.


5 The MDSUMMARY procedure summarizes variable Mpg_Highway, groups the results by variable Type, and then stores the results in in-memory table.

Optional: The PRINT procedure prints columns Type and _Mean_ in table Cars_Summary.

The GCHART procedure creates a bar chart from the data in in-memory table Cars_Summary. The VBAR3D statement creates a vertical bar chart of variable Type using variable _Mean_ as the sum variable and display the sum at the top of each bar.

The summarized data generated by the MDSUMMARY procedure contains only 6 observations as shown in the following image.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Type</th>
<th><em>Mean</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUV</td>
<td>20.5</td>
</tr>
<tr>
<td>2</td>
<td>Hybrid</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>Sedan</td>
<td>28.629770992</td>
</tr>
<tr>
<td>4</td>
<td>Truck</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>Sports</td>
<td>25.489795918</td>
</tr>
<tr>
<td>6</td>
<td>Wagon</td>
<td>27.9</td>
</tr>
</tbody>
</table>

Here is the resulting bar chart.

![Average Highway MPG and Vehicle Type chart](chart.png)

For more information about SAS Cloud Analytic Services programming in SAS, see *An Introduction to SAS Viya Programming*. 
Part 3

Generating Output with SAS/GRAPH

Chapter 13
Using SVG Graphics ............................................. 141

Chapter 14
Generating SVG, PNG, GIF, and TIFF Graphics ................. 153

Chapter 15
Generating Animations .......................................... 167

Chapter 16
Enhancing Web Presentations with Chart
Descriptions, Data Tips, and Drill-Down Functionality ........ 187

Chapter 17
Using SAS/GRAPH Output with Microsoft Office Products ..... 233

Chapter 18
Writing Your Graphs to a PDF File .............................. 241

Chapter 19
Troubleshooting Web Output .................................... 263
Chapter 13
Using SVG Graphics

Introduction to Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an XML language for describing two-dimensional vector graphics.

SAS creates SVG files based on the World Wide Web Consortium (W3C) recommendation for SVG files. SAS SVG files are created using the UNICODE standard encoding.
Advantages of SVG Graphics

SVG graphics are vector graphics, so they can be resized without losing quality. A single SVG file can be scaled to any size or transformed to any resolution without compromising the clarity of the graphic. Bitmap images such as PNG and GIF lose quality anytime they are resized.

Also, if you need to display the same graphic at multiple sizes or resolutions, you would need multiple bitmap images, but only one SVG file. SVG files are displayed clearly at any size in any viewer or browser that supports SVG. Using browser controls, the user can zoom in to view details in a complicated SVG graphic.

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

SVG files are ideal for producing documents of any size. The documents can be displayed on a computer monitor, PDA, or cell phone, or be printed or delivered as PDF files.

Universal Printers versus SAS/GRAPH Devices

You can create SVG files by using either the SAS/GRAPH SVG devices or the Base SAS SVG universal printers. The SVG devices enable you to generate SAS/GRAPH output using the Universal Printing system without specifying ODS statements or an OPTIONS PRINTERPATH= statement.

The SAS/GRAPH SVG devices are primarily for use with the LISTING and HTML destinations. With the PRINTER destination, it is required that you use the SVG universal printers directly.

The information provided here is limited to creating SVG files using SAS/GRAPH devices in the LISTING and HTML destinations. For information about creating SVG documents in the PRINTER destination using the SVG universal printers, see “Creating SVG (Scalable Vector Graphics) Files Using Universal Printing” in SAS Language Reference: Concepts.

See also Table 9.4 on page 90 for examples showing the difference in using a shortcut device and a universal printer.

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

The SVG Devices and the Output That They Create

There are five SVG devices:

SVG

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

When you also specify ANIMATION=START in the OPTIONS statement, the SVG device produces animated output.

SVG

produces SVG 1.1 files that are transparent (no background). These files are useful when you want to overlay several graphs on top of each other and you want all of the graphs to be visible. The SVG 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.

SVGT

produces SVG 1.1 files that are transparent (no background). These files are useful when you want to overlay several graphs on top of each other and you want all of the graphs to be visible. The SVGT device is intended for use when a procedure produces multiple graphs and is best used in conjunction with the ODS PRINTER destination. See “Creating Overlaid Transparent SVG Documents” in SAS Language Reference: Concepts for more information.

SVGZ

produces compressed SVG 1.1 files, which are useful when file size is an issue. However, some browsers do not support compressed SVG files, and you cannot view these files in a text editor. (See also “Browser Support for Viewing SVG Files” on page 149.)

SVGVIEW

produces SVG 1.1 files with navigational controls when the SVG file contains multiple pages. This device is primarily for use in the LISTING destination with procedures that produce multiple graphs. The navigational controls enable you to page through the graphs. See “Example: Generating a Single SVG File with Multiple Pages and Page Controls” on page 147. When used in the HTML destination, the SVGVIEW device produces separate SVG files for each graph, just like the SVG device.

SVGANIM

produces SVG 1.1 animated output. You can modify the animation timing by specifying animation system options.

Note: You can also produce animated output by using the SVG device along with the ANIMATION=START system option. This documentation uses the SVG device in its examples.

Example: Creating an SVG File

The following code produces an SVG file named carstype.svg and an HTML file named cars.htm:

```sas
/* Specify the output path */
filename svgout ".";

/* Specify the graphics options */
goptions reset=all device=svg
   hsize=5.75in vsize=4.5in
gsfname=svgout;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the ODS HTML destination */
ods html path=svgout file="cars.htm";
```
/* Generate the bar chart. The NAME= option */
/* specifies the name of the SVG file. */
title h=2 "Types of Vehicles Produced Worldwide";
axis1 label=none major=none minor=none;
proc gchart data=sashelp.cars;
   vbar type / raxis=axis1 outside=freq
       name="carType";
run;
quit;

/* Close the ODS HTML destination */
ods html close;
ods html; /* Not required in SAS Studio */

You can view the SVG graphic by opening cars.htm in your SVG-enabled browser. If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 149.

You can view the SVG coding by opening the SVG file, cartype.svg, in a text editor.

---

**Example: Generating Stand-Alone SVG Files in HTML5**

In the HTML destination, SVG graphics are created as stand-alone files by default. In the HTML5 destination, SVG graphics are created as inline graphics in the HTML file. You can generate a stand-alone SVG image file in the ODS HTML5 destination by specifying `svg_mode='embed'` in the ODS HTML5 statement. This option embeds the SVG image file into the HTML file using the `<embed>` tag.
In the following example, the code shown in “Example: Creating an SVG File” on page 143 is modified to use this option.

**T I P** When using the ODS HTML5 destination, it’s important to close the HTML destination first. The following code closes all open destinations before opening the HTML5 destination.

```sas
/* Specify the output path */
filename svgout ":."

/* Specify the graphics options */
goptions reset=all device=svg
   hsize=5.75in vsize=4.5in
gsfname=svgout;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the ODS HTML5 destination */
ods html5 options(svg_mode="embed")
   path=svgout file="cars.htm";

/* Generate the bar chart. The NAME= option */
/* specifies the name of the SVG file. */
title h=2 "Types of Vehicles Produced Worldwide";
axis1 label=none major=none minor=none;
proc gchart data=sashelp.cars;
   vbar type / raxis=axis1 outside=freq
      name="carType";
run;
quit;

/* Close the ODS HTML5 destination */
ods html5 close;
ods html; /* Not required in SAS Studio */
```

In the SAS windowing environment, the map appears in the SAS Results viewer. The code produces an SVG image file named `cartype.svg` and an HTML file named `cars.htm`. You can view the SVG graphic by opening `cars.htm` in your web browser.

---

**Example: Placing Images behind SVG Files**

You can use the IBACK= or IFRAME= graphics options in the GOPTIONS statement to specify the graphics file that you want to be placed behind the SVG graphic. From the file that you specify with either of these options, SAS/GRAPH embeds a base64-encoded PNG image directly into the SVG file. For more information, see “Images in SVG Documents” in *SAS Language Reference: Concepts*.

```sas
/* Specify the output path */
filename svgout ":."

/* 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
gsfname=svgout
    imagestyle=fit iback="external-image-file";

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the LISTING destination */
ods listing;

/* 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 LISTING destination */
ods listing close;
ods html; /* Not required in SAS Studio */

The program creates a file named cartype.svg. To view the results, open the SVG file in your browser.

For additional information, see “Displaying an Image in a Graph Background” on page 333.

If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 149.
Example: Generating a Single SVG File with Multiple Pages and Page Controls

The SVGVIEW device is designed to be used when in the LISTING destination. It is useful when a single procedure produces multiple graphs, such as with BY-group processing. When used in the LISTING destination, the SVGVIEW device creates a single SVG file with multiple pages. Each graph produced by the procedure is on a different page. The SVG file, by default, has control buttons that enable you to navigate forward and backward through the graphs. You can also display an index page that shows a thumbnail image of each page.

For example, the following display shows the initial graph that is produced by the program in Example Code 13.1 on page 148. The program produces six graphs. You can page through them by clicking on the Prev and Next buttons.

The Index button displays a page of thumbnail images. There is one thumbnail for each page in the SVG file that, when clicked, take you to that page.
The program that generates this SVG file is as follows:

**Example Code 1  Program Code: Using the SVGVIEW Device with BY-Group Processing**

```sas
/* Specify the output path and filename */
filename mysvg './productView.svg';

/* Subset the data set SASHELP.PRDSALE. */
/* Output the subset to WORK.PRODSUB. */

data prodsub;
  set sashelp.prdsale;
  where year=1994 and
    (country = "U.S.A." or country = "CANADA")
    and region="EAST" and division="CONSUMER" and
    (product in {"SOFA", "TABLE", "BED"});
run;

/* Sort WORK.PRODSUB. */
proc sort data=prodsub;
  by country product;
run;

/* Close the currently open ODS destinations, and */
/* open the LISTING destination. */
ods _all_ close;
ods listing;

/* Use the GSFNAME= option to send the */
/* output of the LISTING destination to */
/* the MYSVG fileref. */

goptions reset=all device=svgview
  gsfmode=replace gsfname=mysvg;

/* Join the data points and change the */
/* line style for the predicted sales */
/* to a dashed line. */
symbol1 interpol=join line=1 color=_style_;
symbol2 interpol=join line=2 color=_style_;
legend1 label=none;

/* Generate a graph for each unique */
/* combination of country and product. */
proc gplot data=work.prodsub;
  by country product;
  plot actual*month predict*month /
    overlay legend=legend1;
run;
quit;

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

Chapter 13 • Using SVG Graphics
The program creates a file named productView.svg. To view the results, open the SVG file in your browser.

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


If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 149.

---

### Web Server Content Type for SVG Files

If the mime content type setting for your web server does not have the correct setting for SVG files, the results can be unpredictable. Your web browser might render SVG files as text files, or SVG files might be unreadable.

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

```
image/svg+xml
```

---

### Height, Width, and Scaling of SVG Files

In HTML5 output, SVG and SVGZ file height and width attributes are set to the size of the SVG. The reason is that many browsers do not scale an SVG file to 100% of the container. The Google Chrome and Safari browsers do scale the SVG file to 100% of the container.

To enable scaling of SVG files that can be scaled to the size of the container, you set the SVGHEIGHT= and SVGWIDTH= system options to 100%.

```ods html; /* Not required in SAS Studio */
options svgheight="100%" svgwidth="100%";
```

---

### Browser Support for Viewing SVG Files

**Browsers That Support SVG Files**

In order to view SVG files, you need a viewer or browser that supports Scalable Vector Graphics. Some browsers, such as Mozilla Firefox, have built-in support for SVG files.

**Note:** SVGZ graphics that you create in the ODS HTML5 destination can be viewed only with the Google Chrome or Opera web browsers.

The following table lists some browsers and viewers that support SVG files.
Table 13.1  SVG Browser Support

<table>
<thead>
<tr>
<th>Browser or Viewer</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batik SVG Toolkit</td>
<td>Apache Software Foundation</td>
</tr>
<tr>
<td>eSVG Viewer and IDE</td>
<td>eSVG Viewer for PC, PDA, Mobile</td>
</tr>
<tr>
<td>Google Chrome ***</td>
<td>Google Inc.</td>
</tr>
<tr>
<td>GPAC Project</td>
<td>GPAC</td>
</tr>
<tr>
<td>Internet Explorer 9*</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Mozilla Firefox *</td>
<td>Mozilla Foundation</td>
</tr>
<tr>
<td>Opera</td>
<td>Opera Software ASA</td>
</tr>
<tr>
<td>Safari, including iPad *****</td>
<td>Apple, Inc.</td>
</tr>
<tr>
<td>TinyLine</td>
<td>TinyLine</td>
</tr>
</tbody>
</table>

* This browser is supported by SAS.
** See “Height, Width, and Scaling of SVG Files” on page 149.
*** When an animated SVG file is opened in Safari, the Refresh button does not work. You need to refresh the browser.

Using Mozilla Firefox to View SVG Graphics

- Compressed SVG files using the SVGZ Universal Printer are not supported.
- Zooming and panning features are not currently implemented.
- If you select View ⇒ Page Style ⇒ No Style, all graphs appear as a black rectangle.
- Firefox does not support font embedding. To avoid font mapping problems in your SVG file, you can set the NOFONTEMBEDDING system option. If the FONTEMBEDDING option is set when an SVG file is created and the SVG file is subsequently viewed in Firefox, Firefox uses the default font setting that is defined on the Contents tab in the Firefox Options dialog box.

Using Internet Explorer 9 to View SVG Graphics

Internet Explorer 9 does not display SVG graphics that were created using the ODS HTML destination. Use the ODS HTML5 destination to create SVG graphics that are to appear in Internet Explorer 9.

Internet Explorer 9 does not support animated SVG graphics. For animated SVG graphics, use Internet Explorer 8 or another browser.

Certain settings are required for Inline SVG files in HTML5 output to display in Internet Explorer 9. If the SVG file looks similar to the following output and does not display any graphic, you must set the Browser Mode and the Document Mode to properly view the SVG file.
To set the Browser Mode and the Document Mode in Internet Explorer, follow these steps:

1. Select **Tools ➤ F12 developer tools**.
2. Above the search box, click **Browser Mode** and select **IE9**.
3. Click **Document Mode** and select **IE9 standards**.

### Using an iPad to View SVG Graphics

To be able to view SVG graphics on a iPad and rotate the graphic to fit within the screen, you might need to do the following:

- Set the SVGHEIGHT= and SVGWIDTH= options. See “Height, Width, and Scaling of SVG Files” on page 149 for more information.
- Create an HTML file that embeds an SVG file and view the HTML file. See “Example: Generating Stand-Alone SVG Files in HTML5” on page 144 for more information.

Starting with SAS 9.4M2, the SVG, SVGT, and SVGView devices add the ROLE and ARIA-LABEL attributes in the HTML output when custom data tips are specified for hot-spots and the ODS HTML5 destination is used. These attributes make these custom data tips accessible to iPad users who are using the Voiceover screen reader. The SVG devices do not generate data tips by default. For information about how to add data tips to your SVG graphs, see “Data Tips in PNG, GIF, JAVAMETA, SVG, SVGT, SVGView, and SVGZ Presentations” on page 191.

### See Also

“Browser Support for Viewing SVG Documents” in *SAS Language Reference: Concepts*

### Controlling Graph Resolution with the SVG Devices

The default resolution for the SVG devices is 96 DPI. Because the SVG devices are Universal Printer shortcut devices, you cannot change the resolution using options in the GOPTIONS statement.

If you are sending output to the HTML or LISTING destinations, you can control the resolution with the IMAGE_DPI= option on the ODS destination statement. For example:

```sas
ods listing image_dpi=600;
```
Controlling Graph Size with the SVG Devices

The default graph size for the SVG output is 800 x 600 pixels. You can change the size of your graph with the HSIZE= and VSIZE= graphics options. You can change the paper size by specifying the XMAX= and YMAX= or the XPIXELS= and YPIXELS= graphics options. Specifying a value for the XMAX=, YMAX=, XPIXELS=, or YPIXELS= graphics options changes the setting of the PAPERSIZE= system option. See “SAS System Options and SVG Output” on page 152 and Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

SAS System Options and SVG Output

Because the SVG devices are Universal Printer shortcut devices, there are several SAS system options that affect how 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 under “Creating SVG (Scalable Vector Graphics) Files Using Universal Printing” in SAS Language Reference: Concepts. Before reviewing the topics that deal with the various system options, you should review the topic “SVG Terminology” in SAS Language Reference: Concepts.

Topics dealing primarily with SAS system options are as follows:

• “SAS System Options That Affect Stand-alone SVG Documents” in SAS Language Reference: Concepts
• “Scaling an SVG Document to the Viewport” in SAS Language Reference: Concepts
• “Setting the ViewBox” in SAS Language Reference: Concepts
• “Creating a Static viewBox” in SAS Language Reference: Concepts
• “Preserving the Aspect Ratio” in SAS Language Reference: Concepts
• “Adding a Title to an SVG Document” in SAS Language Reference: Concepts
• “Interaction between SAS SVG System Options and the SVG Element Attributes” in SAS Language Reference: Concepts

Overview of Animating Output with the SVG Device

You can use the SVG device to create a single SVG file that displays a sequence of images. Various SAS system options enable you to affect how the SVG devices generate output. Step-by-step details about creating an animated SVG image file are described in “Main Steps for Animating SVG Output” on page 170.
What is a Non-Interactive Graph?

A non-interactive graph is a graph that is permanently fixed after it is displayed. You can view a non-interactive graph but you cannot manipulate it as you view it in a browser. Examples of non-interactive graphs include GIF and PNG images. To generate a non-interactive graph, in your SAS program, run a SAS graphics procedure and specify with the DEVICE= graphics option on one of the following devices:

- SVG
- TIFF
- GIF
- ACTXIMG
Creating a Non-Interactive Graph

You can use a GOPTIONS statement with a device type of SVG, PNG, GIF, and TIFF to create a non-interactive graphic output file from one or more SAS/GRAPH procedures. SAS first creates a GRSEG entry in a graphics catalog in your WORK library, and then creates a graphics output file of the specified type from the GRSEG entry.

Follow these steps to generate one or more non-interactive graphs using the ODS LISTING destination:

1. Add a FILENAME statement to create a file reference for the location of the output files. To generate only one output file, specify the file reference, 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 123.

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:

```sas
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 123.

3. Close the currently open ODS destinations, and then open the ODS LISTING destination.

```sas
ods _all_ close;
ods listing;
```

4. Run the procedure that generates the graph. For example:

```sas
proc gchart data=sashelp.class;
   hbar3d sex / sumvar=height type=mean;
run;
quit;
```

The output is stored in the format specified by the DEVICE= graphics option, and in the output location specified by the GSFNAME= graphics option. For example, `C:\mysas\images\barchart.gif` is an output file for a bar chart.

5. Close the ODS LISTING destination, and then open ODS HTML (not required in SAS Studio).

```sas
ods listing close;
ods html; /* Not required in SAS Studio */
```

To create an HTML file that links to the image, use the ODS HTML destination with the following options:

- **BODY=**
  - The filename of the output HTML file (FILE= is a synonym for BODY=).

- **PATH=**
  - The location (path, directory, or file reference) of the HTML file and non-interactive graph file.

- **GPATH=**
  - The location of the graphics output file that is created.

  **Note:** You must specify a value for the GPATH= option only if you specify the FILE= option as a complete path and filename, and you do not specify the PATH= option. You can specify FILE= as just a filename (and extension). If you also specify PATH=, then both the HTML file and the graphics output file are written to the same location (as specified by PATH=).

- **STYLE=**
  - The style to be applied. If you do not specify a style, the default style is applied.

For samples, see “Examples: Generating Static Graphics” on page 159.

For complete information about these options, see *SAS Output Delivery System: User’s Guide*. 
SVG, PNG, GIF, and TIFF Devices Compared to ACTXIMG and JAVAIMG Devices

SVG, PNG, GIF, and TIFF Devices

When you use the graphics option DEVICE=SVG, PNG, GIF, and TIFF with a SAS/GRAPH procedure, an ODS style is applied to your graph by default. In other words, the GSTYLE system option is on by default. You can apply any of the ODS styles to your graph when the GSTYLE system option is on. You cannot apply an ODS style if the NOGSTYLE system option is on.

For complete information about the GSTYLE system option, see SAS System Options: Reference.

When you send your output to the ODS HTML destination, you can add data tips to your graph that are displayed when the cursor is over a portion of the image. You can also add drill-down links to other images or to other URLs. See “Links in SVG, PNG, and GIF Presentations” on page 197.

ACTXIMG and JAVAIMG Devices

When you use the graphics option DEVICE=ACTXIMG or JAVAIMG with a SAS/GRAPH procedure, an ODS style is always applied to your graph by default. This is the same behavior as with the SVG, PNG, GIF, and TIFF devices. You can apply any of the ODS styles to your graph. Unlike the SVG, PNG, GIF, and TIFF devices, the ACTXIMG and JAVAIMG style is not affected by the GSTYLE and NOGSTYLE system options. (See “Output from Different Devices and the GSTYLE/NOGSTYLE System Options” on page 281.)

You can also add data tips to your graph (see “Data Tips in ACTIVEX, ACTXIMG, JAVA, and JAVAIMG Presentations” on page 192) and drill-down links to other URLs.

Developing Web Presentations with the PNG, SVG, and GIF Devices

You can use the SVG, PNG, and GIF devices to create web presentations that consist of non-interactive graphs. You can also add data tips and drill-down links to your graphs.

About the SVG, PNG, and GIF Devices

The SVG, PNG, and GIF devices enable you to generate non-interactive graphs for your web presentation. You can use these devices when you are sending output to the ODS HTML destination in order to generate an HTML file to display one or more graphs. For details, see “Controlling Graphics Output for ODS HTML” on page 119.

Enhancements that are available to SVG, PNG, and GIF 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:
Developing Web Presentations with the JAVAIMG and ACTXIMG Devices

About the JAVAIMG and the 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. However, these graphs do not have the interactivity that JAVA and ACTIVEX graphs provide. You can also add data tips and drill-down links to your graphs.

The JAVAIMG and ACTXIMG devices enable you to generate web presentations that display a snapshot of one or more graphs in the PNG format. 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 about installing the ActiveX control, see “Installing the SAS/GRAPH ActiveX Control” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide. 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 “Procedures and Statements That Generate Output for the SAS/GRAPH ActiveX Control” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide. The procedures that can be used with the JAVAIMG device are listed in “Graph, Map, Tile Chart, and Contour Applets” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.

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. These latter graphs are interactive and can be manipulated by a user viewing them in a browser. However, the PNG files are non-interactive and their appearance cannot be changed after they are created.

Note: With the JAVA IMG and ACTX IMG devices, the titles and footnotes are always stored in the HTML file. They are not stored in the graphics output files regardless of whether the GTITLE and GFOOTNOTE options are set. See “Controlling Titles and Footnotes” on page 101.

**When to Use the JAVA IMG or ACTX IMG Device**

You might not need interactivity such as changing the chart type or style. In this case, the JAVA IMG and ACTX IMG 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.

Some of the SAS/GRAPH procedures, such as the GKPI and GAREABAR procedures, support only the JAVA, JAVA IMG, ACTIVEX, and ACTX IMG devices. For these procedures, you must use the JAVA IMG device or the ACTX IMG device if you need a non-interactive image.

Finally, in some cases such as plots generated with the G3D procedure, the ACTX IMG and JAVA IMG devices provide a better non-interactive image than the other devices.

Note: When SAS is installed on a server, the ACTX IMG and JAVA IMG devices are limited by the display capabilities of the server on which they run. The number of colors that the server is capable of displaying is just such a limitation. Consequently, the ACTX IMG or JAVA IMG 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 JAVA IMG in the z/OS Environment**

If you are running SAS in the z/OS operating environment with the DEVICE=JAVA IMG graphics option, then you must specify FILESYSTEM=HFS. This is because HFS file space is needed to write the graphics output files. You might also need to increase the amount of memory that is allotted for your session so that SAS can run Java in the background. The suggested region size is 400 megabytes. For a batch job, add either REGION=400M or REGION=409600K to the JOB card. For a TSO session, specify SIZE(409600). For more information, refer to your JCL reference manual.

**Generating an HTML Output File Using the JAVA IMG or the ACTX IMG Device**

The procedure for generating an HTML output file for viewing JAVA IMG or ACTX IMG device output is similar to the procedure for generating the same file for the SVG, PNG, or GIF devices. For an example, see "Example: Using the ACTX IMG Device” on page 160.
Adding Drill-Down Links to Web Presentations
Generated with a Static-Graphic Device

You can add drill-down links to web presentations that are generated with an SVG, PNG, GIF, JAVA IMG, or an ACTX IMG device. For information about the default configurations of these web presentations, see “Example: GIF Output with Drill-Down Links” on page 163.

You can add drill-down links to the following elements:

• graph elements or legend elements or both. See “Example: GIF Output with Drill-Down Links” on page 163.

• graph elements specified in an Annotate data set. See “Generating Drill-Down Links with the Annotate Facility” on page 202.

• titles and footnotes using the LINK= option in the TITLE or FOOTNOTE statement.

Implementing Data Tips and Drill-Down Links with the SVG Devices

You can implement data tips and drill-down links in SVG files that are generated in the HTML and LISTING destinations. In both cases, you use the HTML= option or the HTML_LEGEND= option (or both options) to specify variables in your input data that define the drill-down URLs. See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192 for information about implementing drill-down links, including defining link variables. For information about adding data tips, see “Data Tips for Web Presentations” on page 191.

Implementing data tips and drill-down links in SVG files that are generated in the LISTING destination has an additional requirement: you must specify the IMAGEMAP= option in the PROC statement. This option makes the image map generated by the procedure available to the SVG device. For example:

```plaintext
proc gchart data=sashelp.prdsale imagemap=myimgmap;
```

Note: You can also implement drill-down links and data tips in SVG output when you are using the ODS PRINTER destination. However, when you are using the ODS PRINTER destination, you are using a universal printer, not a device.

Examples: Generating Static Graphics

About These Examples

The examples in this section demonstrate how to create non-interactive images for a web presentation. The examples include:

• “Example: Using the ACTX IMG Device” on page 160
Example: Using the ACTXIMG Device

Here is an example that uses the ODS HTML destination to create an HTML file. That file references four PNG files that are created by the GCHART procedure with the DEVICE=ACTXIMG graphics option. Because the ACTXIMG device invokes a SAS/GRAPH ActiveX Control, you can run this example only in a Windows environment.

The GCHART procedure in this example uses BY-group processing to display the results of each of the four quarters of the year. Consequently, the procedure produces four separate PNG files. Only the first graph is shown here. To see all of the PNG images in the output, you must scroll down the page in your browser.

Figure 14.1 Using ODS with the ACTXIMG Device

Here is the complete SAS code for this example. In this example, the output files are sent to the default working directory. If you want to send the output files to a different location, change the path in the ODS HTML statement PATH= option. If you want to send the PNG files to a separate location, add the GPATH= option to the ODS HTML statement to specify the new location for the PNG files.


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

/* Create data set from sashelp.prdsale */
```
data prdsummary;
  set sashelp.prdsale;
  where year=1993 and (country = "GERMANY" or country = "CANADA")
    and region="EAST" and division="CONSUMER" and
    (product="SOFA" or product="TABLE" or product="BED");
run;

/* Sort the data set by quarter */
proc sort data=work.prdsummary;
  by quarter;
run;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Specify the SEASIDE style for the HTML output */
ods html path=odsout file="sales.htm" style=seaside;

/* Specify device as actximg */
goptions reset=all device=actximg border;
title1 "1993 Sales";

/* Chart total 1993 sales for each country by quarter */
proc gchart data=work.prdsummary;
  hbar country / sumvar=actual subgroup=product sum;
  by quarter;
run;
quit;

/* Close ODS HTML and then reopen it to restore the default style. */
ods html close;
ods html; /* Not required in SAS Studio */

Example: Generating PNG Output

Here is an example that uses ODS to create an HTML file that references four PNG files that are created by a SAS/GRAPH procedure. The GCHART procedure in this example uses BY-group processing to display the results of each of the four quarters of the year. Consequently, the procedure produces four separate PNG files. Only the first graph is shown here. To see all of the graphs, you must scroll down the page in your browser.
Here is the complete SAS code for this example. In this example, the output files are sent to the default working directory. If you want to send the output files to a different location, change the path in the ODS HTML statement PATH= option. If you want to send the PNG files to a separate location, add the GPATH= option to the ODS HTML statement to specify the new location for the PNG files. See “ODS HTML Statement” in SAS Output Delivery System: User’s Guide for details.

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

/* Create data set from sashelp.prdsale */
data prdsummary;
  set sashelp.prdsale;
  where year=1993 and (country = "GERMANY" or country = "CANADA")
  and region="EAST" and division="CONSUMER" and
  (product="SOFA" or product="TABLE" or product="BED");
run;

/* Sort the data set by quarter */
proc sort data=work.prdsummary;
  by quarter;
run;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Generate the graph */
ods html path=odsout file="sales.htm" style=seaside;
goptions reset=all border;
title1 "1993 Sales";
proc gchart data=prdsummary(where=(year=1993));
  vbar3d country / sumvar=actual subgroup=product sum;
```
by quarter;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

Notice that a device is not specified in the GOPTIONS statement in this example. ODS uses the PNG device as the default device for the HTML destination.

Example: GIF Output with Drill-Down Links

Here is an example that generates web output with drill-down functionality using the GIF device.

(See also Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 187.)

In this example, the DEVICE=GIF graphics option generates image output files and the ODS HTML statement generates an HTML output file. The URL= option identifies a link variable that provides drill-down URLs. The values of the link variables are added to the data set with IF/THEN statements. ODS inserts the drill-down URLs into an image map that it generates in the HTML output file. Because the link variable contains the URL string only, the URL= option must be used to identify the link variable. The HTML= option cannot be used in this case.

When you display the HTML output file in a web browser, the following chart is displayed.
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.

<table>
<thead>
<tr>
<th>State</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL</td>
<td>$18,038</td>
</tr>
<tr>
<td>IN</td>
<td>$13,611</td>
</tr>
<tr>
<td>OH</td>
<td>$11,084</td>
</tr>
<tr>
<td>MI</td>
<td>$19,660</td>
</tr>
</tbody>
</table>

Here is the example code.

```plaintext
/* Specify the ods output path. */
filename odsout "./";

/* Close the currently open ODS destinations. */
```
ods _all_ 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 URL variable. */
  if Region="Central" then
    rpt="central.html";
  else if Region="South" then
    rpt="south.html";
  else if Region="West" then
    rpt="west.html";

datalines;
  West CA 13636
  West OR 18988
  West WA 14523
  Central IL 18038
  Central IN 13611
  Central OH 11084
  Central MI 19660
  South FL 14541
  South GA 19022
;
/* Open the HTML destination for ODS output. Specify the */
/* path and filename. */
ods html path=odsout body="company.html" style=statistical;
/* Create a chart that uses the link variable. */
  title1 "Company Sales";
  proc gchart data=regsales;
    vbar region / sumvar=sales
      patternid=midpoint
      url=rpt; /* Set the URL variable to rpt */
    run;
  quit;
/* Create the Central sales page */
  ods html path=odsout body="central.html";
  title1 "Central Sales";
  proc print data=regsales noobs;
    var state sales;
    where region="Central";
  run;
quit;

/* Create the Southern sales page */
title1 "Southern Sales";
ods html path=odsout body="south.html";
proc print data=regsales noobs;
    var state sales;
    where region="South";
run;
quit;

/* Create the Western sales page */
title1 "Western Sales";
ods html body="west.html";
proc print data=regsales noobs;
    var state sales;
    where region="West";
run;
quit;

/* Close ODS HTML to close the output file, and then reopen ODS HTML. */
ods html close;
ods html; /* Not required in SAS Studio */

The code for this example is also available in the SAS Sample Library under the name GWBDRILL. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
Chapter 15
Generating Animations

Animating Images in SAS/GRAPH

You can create either an animated GIF or SVG image output file. The following sections describe what an animated image is, the steps involved in animating your output, the system options available for altering animation attributes, and examples of creating animations.

Animating GIF Images in SAS/GRAPH

Overview of GIF Animation
Using the GIF image format, you can create an animated image that displays a sequence of images automatically from a single GIF file. These animated sequences are commonly referred to as slide shows. The display sequence repeats until the web user selects Stop in the web browser, until the user displays another web page, or until the animation completes the number of iterations that it is configured to run. Animated images are useful for presentations that do not need to be controlled by the web user. Finite looping is appropriate for most cases such as demonstrating trends in data over time. Infinite looping is appropriate for unattended kiosk displays.
The information provided here is limited to creating animated GIF files using the SAS/GRAPH animation device in the HTML destination. For information about creating animated GIF files in the PRINTER destination using the GIF universal printers, see “Creating Animated GIF Images and SVG Documents” in SAS Language Reference: Concepts.

**Animating GIF Image Output**

Use the GIF graphics output device with the ODS HTML destination to generate the animated GIF file and an HTML file that displays the animation in a web browser.

*Note:* The generated HTML file can also contain HTML tags for drill-down links and data tips. See Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 187.

**Main Steps for Animating GIF Image Output**

To create an animated GIF image, follow these general steps in your SAS program:

1. Use an OPTIONS statement to set the appropriate system options for the animated GIF image:

   ```
   OPTIONS <ANIMDURATION=duration | MIN> <ANIMLOOP=number> <ANIMOVERLAY | NOANIMOVERLAY>;
   ```

   For information about the animation options, see “Specifying System Options to Control Animation Attributes” on page 171.

2. At the point in your program where you want your animation output to begin, use the following OPTIONS statement to start the animation output:

   ```
   OPTIONS ANIMATE=START;
   ```

3. Open the ODS HTML destination. After you have set the options and opened the ODS destination, proceed with your SAS code to create each frame in your file. The animation frames are created when you run SAS procedures.

   **Tip** You can use the ANIMDURATION= animation option in between procedures to change the duration that a frame is held in view.

4. To stop your animation output temporarily in your program, use the following OPTIONS statement at the point where you want to stop:

   ```
   OPTIONS ANIMATE=STOP;
   ```

   To resume your animation output later in your program, use the following OPTIONS statement:

   ```
   OPTIONS ANIMATE=START;
   ```

5. When all of your animation output has been generated, close the ODS HTML destination:

6. Use the following OPTIONS statement to stop the animation output:

   ```
   OPTIONS ANIMATE=STOP;
   ```
Animating Image Output with the SVG Device

Overview of SVG Animation
You can use the SVG device to create a single SVG file that displays a sequence of images. The system options enable you to configure these animation attributes:

- the amount of time that a frame is in view
- whether frames are overlaid or are played sequentially
- the number of times an animation loop is repeated
- whether to immediately start the animation when the file is loaded in the web page
- whether a frame fades in and out of view and whether the frames are overlaid or played sequentially during the fade-in and fade-out time

The information provided here is limited to creating animated SVG files using the SAS/GRAPH animation device in the LISTING and HTML destinations. For information about creating animated SVG files in the PRINTER destination using the SVG universal printers, see “Creating Animated GIF Images and SVG Documents” in SAS Language Reference: Concepts.

The following display shows one frame of an SVG animation. This animation was created by the example code shown in “Example: Creating an SVG Animation” on page 183.

By default, the SVG device adds three buttons to the top of the display.

The [SVG Controls] button turns on and off the display of the buttons across the top of the SVG image.
resets the animation to the initial image.

pauses the animation. When the animation is paused, the button changes to 🔄.

By default, the display sequence repeats until you click 🔄 or until you display another web page.

**Main Steps for Animating SVG Output**

To create an animated SVG image, follow these general steps in your SAS program:

1. Use the following GOPTIONS statement to specify the SVG device:
   
   ```
   GOPTIONS DEVICE=SVG
   ```

2. Use an OPTIONS statement to set the graphics options that you want for your animation:
   
   ```
   OPTIONS <ANIMDURATION=duration | MIN>
   <ANIMLOOP=YES | NO | number>
   <ANIMOVERLAY | NOANIMOVERLAY>
   <SVGAUTOPLAY | NOSVGAUTOPLAY>
   <SVGFADEIN=number>
   <SVGFADEOUT=number>
   <SVGFADEMODE=OVERLAP | SEQUENTIAL>;
   ```

Here is an example:

```sas
options animduration=5 animloop=yes noanimoverlay;
```

In this OPTIONS statement, the ANIMDURATION option specifies that each frame is held for 5 seconds. The ANIMLOOP option specifies to continuously repeat the animation loop. The NOANIMOVERLAY option specifies that each frame is played sequentially.

For information about the animation options, see “Specifying System Options to Control Animation Attributes” on page 171.

3. At the point in your program where you want your animation output to begin, use the following OPTIONS statement to begin generating the animation output:
   
   ```
   OPTIONS ANIMATE=START;
   ```

4. Open the ODS HTML5, ODS HTML, or the ODS LISTING destination.

   **TIP** See “Browser Support for Viewing SVG Files” on page 149 to determine whether to use HTML5 or HTML.

5. After you have set the options and opened the ODS destination, proceed with your SAS code to create each frame in your file. The animation frames are created when you run SAS procedures.

   **TIP** You can use animation options in between procedures to change the duration that a frame is held in view and the fade-in and fade-out times. For example, you might want to hold a particular frame in view for a longer period of time. You would use the OPTIONS ANIMDURATION= statement before a procedure to increase the time that the frame is held in view.

6. To stop your animation output temporarily in your program, use the following OPTIONS statement at the point where you want to stop:
OPTIONS ANIMATE=STOP;
To resume your animation output later in your program, use the following OPTIONS statement:

OPTIONS ANIMATE=START;

7. When your animation output has been generated, close the ODS HTML5, ODS HTML, or the ODS LISTING destination.

8. Use the following OPTIONS statement to stop the animation output:

OPTIONS ANIMATION=STOP;

Adding Data Tips and Drill-down Links

Note: Animated GIF images and animated SVG images do not support data tips and drill-down links. However, using the ODS HTML output destination in combination with certain device drivers and any SAS/GRAPH procedure that supports the HTML= option enables you to use these functions. If you need to add data tips and drill-down links to your images, see the following:

• Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” on page 187
• “Developing Web Presentations for the Metaview Applet” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide

For animated GIF images, the GIF graphics output device supports alpha transparency and anti-aliasing for lines.

Controlling Animations with System Options

Starting and Stopping Animation Output in Your SAS Program

The system option ANIMATE starts and stops the animation output in a SAS program. In your SAS program, to start your animation output, add an OPTIONS ANIMATE=START statement at the point where you want your animation to start. Add an OPTIONS ANIMATE=STOP statement at the point where you want your animation output to stop. You can start and stop your animation output multiple times in your SAS program, if necessary. This feature enables you to span procedure boundaries and include in your animation output selected graphics produced from multiple procedures in your SAS program.

For more information about the ANIMATE system options, see SAS System Options: Reference.

Specifying System Options to Control Animation Attributes

You can use system options to control various attributes of your animated image. Attributes include such things as the amount of time each image is displayed or the number of times the image sequence is repeated. You set the system options using the OPTIONS statement before opening the ODS destination. The following table summarizes the system options that specify various attributes of animated images.
<table>
<thead>
<tr>
<th>Image Output Supported</th>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF, SVG</td>
<td>ANIMATION=</td>
<td>Starts or stops the creation of an animation file.</td>
</tr>
<tr>
<td>GIF, SVG</td>
<td>ANIMDURATION=</td>
<td>Specifies the amount of time that each frame in an animation is held in view.</td>
</tr>
<tr>
<td>GIF, SVG</td>
<td>ANIMLOOP=</td>
<td>Specifies whether the animation loop is played continuously or is played once. For SVG files, if ANIMLOOP=0 or YES, the animation loops continuously.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All other values (NO and any number other than 0) display the animation only once.</td>
</tr>
<tr>
<td>GIF, SVG</td>
<td>ANIMOVERLAY</td>
<td>NOANIMOVERLAY</td>
</tr>
<tr>
<td>SVG</td>
<td>SVGAUTOPLAY</td>
<td>NOSVGAUTOPLAY</td>
</tr>
<tr>
<td>SVG</td>
<td>SVGFADEIN=</td>
<td>Specifies the number of seconds for an SVG frame to fade into view.</td>
</tr>
<tr>
<td>SVG</td>
<td>SVGFADEMODE=</td>
<td>Specifies whether an SVG frame overlaps the previous frame or if each frame is played sequentially when a frame is fading in and out.</td>
</tr>
<tr>
<td>SVG</td>
<td>SVGFADEOUT=</td>
<td>Specifies the number of seconds for an SVG frame to fade out of view.</td>
</tr>
</tbody>
</table>

For more information about these system options, see “Dictionary of System Options” in SAS System Options: Reference.

To control the transparency of the background, use the CBACK= graphics option to specify an RGBA color code. See “RGBA Color Codes” on page 319.
Animating Annotation Text Labels

When you create text labels using the Annotate facility, you can animate those labels if you use one of the SAS/GRAPH SVG device drivers, except SVGANIM. Various animation actions are available, including moving text on a path, growing or shrinking text, moving text, and blinking text.

To animate labels, add the HTML variable to the LABEL function in the annotate data set. The HTML variable accepts attributes that specify the animation action and the parameters that control the animation.

In the following example, the HTML variable specifies that the text “ALERT: Here is a moving news ticker” moves continuously across the bottom of the screen. The code for this example can be found in Example Code 15.1.

The figure captures a still frame of the output.

**Example: Adding a Moving Text Annotation to Charts**

Here is the HTML variable statement that created the moving text.

```html
html="animate=move('ALERT: Here is a moving news ticker', 100, +0, 0, 5, repeat)";
```
The following items describe the parts of the statement:

1. `ANIMATE=MOVE(parameters)` specifies the type of animation to perform. In this example, the MOVE animation is performed. The HTML variable specified must be enclosed in double quotation marks.

2. `'ALERT: Here is a moving news ticker'` specifies the text that is moved. The text must be enclosed in single quotation marks. In addition, the text must match the text string that is provided for the TEXT variable in the LABEL function.

3. `100` specifies the X coordinate that the text moves to in percentage units of the display or the SVG space.

4. `+0` specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space. The `+` indicates that the text moves the specified amount relative to the Y value that is specified for the label.

5. `0, 5` specifies that the animation begins 0 seconds after the SVG page has been loaded and has a duration of 5 seconds.

6. `REPEAT` specifies that the animation repeats continuously.

The attributes that determine the text color, background color, and so on, are specified separately in the data set.

Here are the main steps for producing the graph in the example.

1. In your annotation data set, after you specify `FUNCTION='LABEL'`, add the HTML variable that specifies the animation action and associated parameters. For complete syntax, see “HTML Variable” on page 715.

2. Use the GOPTIONS statement to specify one of the SVG device drivers, except SVGANIM.

   ```
goptions dev=svgview;
   ```

3. Specify the ODS HTML5, ODS HTML, or the ODS LISTING destination.

   **TIP** If using one of the ODS HTML destinations, see “Browser Support for Viewing SVG Files” on page 149 to determine whether to use HTML5 or HTML.

4. After you have set the options and opened the ODS destination, proceed with your SAS code to create the graph. The graph code should be designed to display the annotation. For more information, see “Producing Graphics Output from Annotate Data Sets” on page 651.

SAS/GRAPH generates the output as an SVG graphic. To view the output, open the SVG file in a browser.

**Note:** If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 149.

The following code produces the output shown in the previous example.

**Example Code 1  Code for Adding a Moving Text Annotation to Charts**

```sas
/* Create a file reference for the ODS output */
filename odsout ".";

/* Specify the graphics options */
options nodate nonumber nofontembedding;
goptions reset=all device=svgview;
ods _all_ close;
```
ods html path=odsout file="animlabel.html";

/* Create the annotation data set */
/* that includes animation for the label */
data moving;
length color $ 8;
xsys='3'; ysys='3'; hsys='D'; position='6'; function='label';
x=0; y=5; color='black'; style='Arial'; size=18;
text='ALERT: Here is a moving news ticker';
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
output;
function='move'; x=0; y=3; hsys='3';
text=''; html='';  output;
function='bar'; x=100; y=5; hsys='3'; style='solid'; line=0;
color='yellow';
output;
run;

/* Create the graph data by extracting information for */
/* Canada and Germany from sashelp.prdsale. */
data work.qsales;
set sashelp.prdsale(where=(country="CANADA" or country="GERMANY")
keep=Actual Country Product Quarter Year);
run;
/* Sort the data by quarter */
proc sort data=work.qsales;
by quarter;
run;

/* Generate the first set of graphs */
title1 "1993 Sales";
footnote h=2 ' ';
axis1 order=(0 to 30000 by 5000);
proc gchart data=work.qsales(where=(year=1993)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

/* Generate the second set of graphs */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

ods html close;
ods html; /* Not required in SAS Studio */
You can also use Base SAS Universal Printing to generate SVG output. For more information, see “Creating SVG (Scalable Vector Graphics) Files Using Universal Printing” in *SAS Language Reference: Concepts*.

For another example of label text animation, see “Example: Moving Text on a Path” on page 687.

**See Also**

- Chapter 27, “Using Annotate Data Sets,” on page 635
- “LABEL Function” on page 681

---

**Examples: Generating Animated Images**

**About These Examples**

This section provides examples of how to generate animated GIF images using the GIF graphics output device or animated SVG images using the SVG graphics output device. The examples include:

- “Example: Creating an Animated GIF Using BY-Group Processing” on page 176
- “Example: Creating an Animated GIF Using Run-Group Processing and Transparency” on page 178
- “Example: Using the GREPLAY Procedure to Control the Image Order in a GIF Animation” on page 181
- “Example: Creating an SVG Animation” on page 183

You can find an additional example, GWBANIMA, in the SAS Sample Library.

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

**Example: Creating an Animated GIF Using BY-Group Processing**

This example uses the GIF graphics output device to generate an animated GIF image from a SAS data set and two invocations of the GCHART procedure. Each invocation of the GCHART procedure uses BY-group processing to create a series of graphs. This example also generates an HTML file that displays the animation.

The following figure shows the first image of the animated GIF image sequence.
Here is the example program code.

```sas
/* Define a fileref for the ODS output */
filename odsout ";

/* Specify a base name for the output */
%let name=gifa;

/* Specify a name for the HTML output file */
%let htmlout=gifanima.html; /* HTML output */

/* Close all open destinations */
ods _all_ close;

/* Sort our sales data by year and quarter */
proc sort data=sashelp.prdsale out=work.sales;
  by year quarter;
run;

/* Set the graphics options */
goptions reset=all
  device=gif /* Specify the GIF shortcut device */
  border; /* Specify a border around the graphs */

/* Set the system options for the animation */
options nobyline /* Suppress the BY-line */
  animduration=2 /* Wait 2 seconds between graphs */
  animloop=yes; /* Repeat the sequence continuously */

/* Create a consistent response axis */
axis1 order=(0 to 25000 by 5000);

/* Start the animation output */
goptions animation=start;
```

Examples: Generating Animated Images
/* Open the ODS HTML destination */
ods html path=odsout file="&htmlout" style=harvest;

/* Generate the graphs by year and quarter */
title1 "byval(year) Sales" lspace=1;
title2 "Quarter byval(quarter)*";
proc gchart data=work.sales;
  vbar3d country / name="&name" sumvar=actual subgroup=product
    sum raxis=axis1;
  where product in ("BED" "TABLE" "CHAIR");
  by year quarter;
  format actual dollar8.0;
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animation=stop;
ods html; /* Not required in SAS Studio */

The animation displays the sales figures for each quarter of each of the two years. It waits 1.5 seconds between each graph. The sequence plays once initially, and then repeats two times for a total of three iterations. After the last iteration, it stops on the last graph in the sequence (1994 Q4). You can change the amount of time that each graph is displayed by changing the ANIMDURATION= system option value. You can change the number of times the sequence repeats by changing the ANIMLOOP= system option value.

Example: Creating an Animated GIF Using Run-Group Processing and Transparency

This example uses the GIF graphics output device to generate an animated GIF image file. RUN-group processing is used to show the 1993 sales data in a specific product order: desks, tables, chairs, sofas, and beds. This example also shows how to create an animated GIF image with a transparent background.

The following display shows the first image of the animated GIF image sequence.
Here is the example program code.

```sas
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a name for the graphics output */
%let name=gifb;

/* Specify a name for the HTML output file */
%let htmlout=gifanimb.html; /* HTML output */

/* Close the ODS HTML destination */
ods _all_ close;

/* Specify a border and a transparent background */
goptions reset=all border cback=aFFFFFF00 gsfname=gout;

/* Set the system animation options */
options
   device=gif /* Specify the GIF shortcut device */
   noanimoverlay /* Display the images sequentially */
   animduration=2 /* Wait 2 seconds between graphs */
   animloop=4; /* Repeat the sequence 4 times */

/* Delete the previously created graphs before creating new ones */
proc greplay igout=work.gseg nofs;
   delete _all_
run; quit;

/* Create a consistent response axis */
axis1 order=(0 to 40000 by 10000);

/* Start the animation output */
options animate=start;
```
/* Open the ODS PRINTER destination */
ods html path=odsout file="xhtmlout" style=analysis;

/* Generate the graphs using RUN-group processing */
title1 "1993 Sales lspace=1;
proc gchart data=sashelp.prdsale(where=(year=1993));
title2 "Desks";
vbar3d country / name="&name" sumvar=actual sum
raxis=axis1;
where product="DESK";
format actual dollar8.0;
run;
title2 "Tables";
vbar3d country / name="&name" sumvar=actual sum
raxis=axis1;
where product="TABLE";
run;
title2 "Chairs";
vbar3d country / name="&name" sumvar=actual sum
raxis=axis1;
where product="CHAIR";
run;
title2 "Sofas";
vbar3d country / name="&name" sumvar=actual sum
raxis=axis1;
where product="SOFA";
run;
title2 "Beds";
vbar3d country / name="&name" sumvar=actual sum
raxis=axis1;
where product="BED";
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animate=stop;
ods html; /* Not required in SAS Studio */

The animation displays the sales data for Canada, Germany, and the U.S.A. It waits two seconds between each image and repeats the image sequence four times. The animation stops after the fourth iteration and displays the last graph in the sequence (beds). The images are generated using the GCHART procedure with RUN-group processing and WHERE clauses to select individual products. The CBACK= option specifies a transparent background for the graph, which allows the page background to show through. Because of the transparent graph background, the NOANIMOVERLAY option is used to display the frames sequentially. Otherwise, the previous frames can show through the current frame.

You can change the amount of time that each graph is displayed by changing the ANIMDURATION= system option value. You can change the number of iterations by changing the ANIMLOOP= system option value. You can make the background opaque
by removing the CBACK= option from the GOPTIONS statement or you can change the transparency value of the RGBA color code to vary the degree of transparency.

**Example: Using the GREPLAY Procedure to Control the Image Order in a GIF Animation**

This example uses the ODS HTML destination and the GIF graphics output device to create an animated GIF web presentation. The GCHART procedure, with BY-group processing, generates the graphs and stores them in a catalog. The GREPLAY procedure is then used with the GIF graphics output device to create an animated GIF image that plays the graphs in a specific product order: desks, tables, chairs, sofas, and beds.

The following display shows the first image of the animated GIF image sequence.

![](image)

**Note:** When the GREPLAY procedure and the GIF graphics output device are used to create an animated GIF image, anti-aliasing for lines is not supported. In order to produce smoother lines in graphs that contain angled or curved lines, use RUN-group processing with WHERE statements to arrange the images instead of the GREPLAY procedure. See “Example: Creating an Animated GIF Using Run-Group Processing and Transparency” on page 178.

Here is the example program code.

```plaintext
/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a base name for the output */
%let name=gifc;

/* Specify a name for the HTML output file */
%let htmout=gifanimc.html; /* HTML output */

/* Close the currently open ODS destinations */
ods _all_ close;
```
/* Create our data set by extracting the 1993 data from sashelp.prdsale */
data sales;
  set sashelp.prdsale;
  where year=1993;
run;

/* Sort the data by year and product */
proc sort data=sales;
  by product;
run;
quit;

/* Set graphics options */
goptions reset=all device=gif noborder nodisplay;

/* Delete the previously created graphs before creating new ones */
proc greplay igout=work.gseg nofs;
  delete _all_;
run; quit;

/* Create a consistent response axis */
axis1 order=(0 to 700 by 100);

/* Disable the BY-line */
options nobyline;

/* Open ODS LISTING and specify the graph style */
ods listing style=analysis;

/* Generate the graphs using BY-group processing */
title1 "1993 Sales" lspace=1;
title2 "Product Category: &BYVAL(product)";
proc gchart data=work.sales;
  hbar country / sumvar=actual type=mean name="&name" raxis=axis1
    mean meanlabel="Mean";
  by product;
  format actual dollar8.0;
run;
quit;

/* Close ODS LISTING */
ods listing close;

/* Reset the graphics options and set the device to GIF */
goptions reset=all device=gif;

/* Set the system animation options */
options animduration=2 /* Wait 2 seconds between graphs */
  animloop=yes; /* Repeat the sequence continuously */

/* Start the animation output */
options animate=start;

/* Open the ODS HTML destination */
ods html path=odsout file="&htmout" style=analysis;
/* The graphs are to be replayed in the following product order: */
/* desks, tables, chairs, sofas, and beds */
/* This means that we have to replay the GRSEGs in the following order: */
/* Gifc2, Gifc4, Gifc1, Gifc3, and Gifc. */
proc greplay igout=work.gseg nofs;
    replay &name.2 &name.4 &name.1 &name.3 &name;
run;
quit;

/* Close the HTML destination */
ods html close;

/* Stop the animation output */
options ANIMATE=STOP;

/* Open ODS HTML (not required in SAS Studio) */
ods html;

The animation iterates through the 1993 product sales data for Canada, Germany, and the U.S.A. The animation waits two seconds between each image and repeats the animation sequence continuously.

**Example: Creating an SVG Animation**

This example animation shows sales figures for each of the four quarters in the years 1993 and 1994.

The following display shows one frame of the example SVG animation.

*Note:* If you are using Internet Explorer 9, you need to specify the ODS HTML5 destination instead of the ODS HTML destination. See “Using Internet Explorer 9 to View SVG Graphics” on page 150.
**Example Code 2  Creating an Animation Using the SVG Device**

/* Define a fileref for the ODS output */
filename odsout ".";

/* Specify a name for the HTML output file */
%let htmout=svgAnim.html; /* HTML output */

/* Close the currently open ODS destinations */
ods _all_ close;

/* Set the graphic environment. Specify */
/* the SVG device and start the animation output. */
goptions reset=all device=svg htext=1.5 htitle=2;
options animation=start animduration=5 noanimoverlay
       nodate nonumber;

/* Open the ODS HTML destination, specify */
/* the filename. */
ods html path=odsout file="&htmout";

/* Create a data set by extracting data */
/* from sashelp.prdsale. Keep only the */
/* columns needed for the graphs. */
data work.qsales;
   set sashelp.prdsale (keep=Actual Country Product Quarter Year);
   run;

/* Sort the data by quarter. When using */
/* BY groups, the data must be sorted */
/* according to the BY variable. */
proc sort data=work.qsales;
   by quarter;
   run;

/* Set the major tick mark interval */
/* on the vertical axis. */
axis1 order=(0 to 25000 by 5000);

/* Generate graphs for 1993. */
title1 "1993 Sales";
proc gchart data=work.qsales(where=(year=1993));
vbar3d country / sumvar=actual subgroup=product sum
        raxis=axis1 shape=cylinder name='sales93';
   where product in ("BED" "TABLE" "CHAIR");
   by quarter;
   run;
   quit;

/* Generate graphs for 1994. */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994));
vbar3d country / sumvar=actual subgroup=product sum
   raxis=axis1 shape=cylinder name='sales94';
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

/* Close the ODS HTML destination */
ods html close;

/* Stop the animation output */
options animation=stop;

/* Reset the title to null */
title1;

/* Open ODS HTML (not required in SAS Studio) */
ods html;
Chapter 16
Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality

Overview of Enhancing Web Presentations .............................................. 188

Chart Descriptions for Web Presentations ............................................. 189
What Is a Chart Description? ................................................................. 189
Example: Adding Custom Chart Descriptions ........................................ 189
Chart Descriptions in PNG, GIF, ACTXIMG, and JAVAIMG Presentations .... 190
Chart Descriptions in SVG, SVGT, SVGView, and SVGZ Presentations ........ 190

Data Tips for Web Presentations ............................................................ 191
What Is a Data Tip? .................................................................................. 191
Adding Custom Data Tips with the HTML= Option ................................... 191
Data Tips in PNG, GIF, JAVA META, SVG, SVGT, SVGView, and SVGZ Presentations ................................................................. 191
Data Tips in ACTIVEX, ACTXIMG, JAVA, and JAVA IMG Presentations .... 192

Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options ............................................................. 192
About the URL=, HTML=, and HTML_LEGEND= Options ......................... 192
Link Variables and Enhancement Variables ............................................. 193
Working with Link and Enhancement Variables ....................................... 193
Links in SVG, PNG, and GIF Presentations ............................................. 197
Links in ACTXIMG and JAVA IMG Presentations ..................................... 198
Links in ACTIVEX Presentations ............................................................ 198
Links in JAVA Presentations ................................................................. 198
Links in Metaview Applet Presentations ................................................ 198
Links in Animated GIF Presentations ..................................................... 198

Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes 198
Additional Link Enhancements That Are Available with the SVG Graphics Devices ................................................................. 198
Using the ONMOUSEOVER= HTML Attribute ......................................... 199
Using the ONCLICK= HTML Attribute .................................................. 200

Generating Drill-Down Links with the Annotate Facility ......................... 201
When to Use PROC GANNO to Generate Drill–Down Links ..................... 201
When to Apply Annotate Data Sets to Web Presentations ......................... 201
Generating Drill–Down Links with the Annotate Facility .......................... 202
Examples of Using Annotated Data Sets ................................................ 202

Examples ............................................................................................... 202
Example 1: Adding Custom Data Tips to a Graph in a Web Presentation ....... 202
Example 2: Creating a Drill-Down HTML Presentation for the Web .......... 204
Example 3: Creating a Drill-Down Java Presentation for the Web ............... 220
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 189.
- Displaying pop-up text when the mouse pointer is over a portion of the diagram. See “Data Tips for Web Presentations” on page 191.
- Adding drill-down functionality that enables you to link to other web pages. See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192 and “Generating Drill-Down Links with the Annotate Facility” on page 201.

### Table 16.1 Support for Chart Descriptions, Data Tips, and Drill-Down Functionality

<table>
<thead>
<tr>
<th>Device</th>
<th>Chart Descriptions</th>
<th>Data Tips</th>
<th>Drill-down Links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Generated by default</td>
<td>Can be customized</td>
<td>Generated by default</td>
</tr>
<tr>
<td>GIF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PNG</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SVG family</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ACTIV EX*</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>JAVA*</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>JAVAM ETA*</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ACTXI MG</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>JAVAIM G</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* For more information about JAVA, ACTIVEX, and JAVAMETA devices, see *SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.*
Chart Descriptions for Web Presentations

What Is a Chart Description?

A chart description is the text that describes the entire chart. Default chart descriptions are generated when you use the HTML output destination, in combination with certain device driver entries. A description of your graphics output is created and stored in the HTML IMG tag ALT= attribute in your output file. You can suppress any chart description by specifying the NOALTDESC graphics option. You can display the chart description again by using the ALTDESC graphics option.

Chart descriptions are one way to meet Section 508 standards that require text equivalents for graphics elements. See “ACCESSIBLE” on page 520 for an alternate technique.

To supply your own text that describes the chart, use the DESCRIPTION= option with your procedure statement. The maximum length for a custom chart description is 256 characters. See individual procedure statements for DESCRIPTION= option details.

Example: Adding Custom Chart Descriptions

The following code generates a plot chart with a custom chart description. The custom chart description is created using the DESCRIPTION= option.

goptions reset=all border cback="white";
axis1 label=("Population Est.") minor=(n=2);
symbol v=dot;
proc gplot data=sashelp.citiyr;
    format pan comma7.;
    plot pan*date/ vaxis=axis1 autovref
        description="This is the DESCRIPTION= option text.";
run;
quit;
Chart Descriptions in PNG, GIF, ACTXIMG, and JAVAIMG Presentations

For output generated with the PNG, GIF, ACTXIMG, and JAVAIMG device drivers, using the ALTDESC graphics option displays the chart description, and is set by default. The NOALTDESC graphics option suppresses the display of the chart description. Specifying DESCRIPTION= provides content to the ALT attribute of your HTML file and replaces the default chart description content. The description value is limited to 256 characters. Chart descriptions are not supported in presentations generated by the 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 are created by default, and custom data tips are supported when using the HTML output destination in combination with certain device drivers.

Note: The information displayed in a data tip can often be read by screen reader software applications used by the visually impaired.

Adding Custom Data Tips with the HTML= Option

You can add custom data tips to the output of any SAS/GRAPH procedure that supports the HTML= option. The default device for the HTML destination is PNG, so the output of the following code is an HTML file that references a PNG image file.

To add custom data tips:

1. Add a data tip variable to the data set. In the example that follows, the data tip variable is named `rpt`.

2. Assign values to the data tip variable using the following form:

   `'title="data tip"'`

   Note: Some web browsers display text that is specified with the ALT= attribute as data tips. In that case, you can use the ALT= attribute instead, but be aware that the data tips might not be displayed in other browsers. Refer to the documentation for your preferred web browser to determine how the TITLE= and ALT= attribute text is displayed.

   Note: The JAVAMETA device does not support the TITLE= option.

3. Add HTML=data-tip-variable to your procedure's statement.

When the mouse pointer is positioned over a graph data element, the browser displays the data tip. For an example, see “Example 1: Adding Custom Data Tips to a Graph in a Web Presentation” on page 202.

Data Tips in PNG, GIF, JAVAMETA, SVG, SVGT, SVGView, and SVGZ Presentations

For output generated with the PNG, GIF, JAVAMETA, and SVG, SVGT, SVGView, and SVGZ device drivers, data tips are not generated by default. For SAS/GRAPH procedures that support the HTML= option, you can add custom data tips to the output as described in “Adding Custom Data Tips with the HTML= Option” on page 191. To determine whether a procedure supports the HTML= option, refer to the description for that procedure in Part 6: SAS/GRAPH Procedures.


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. The data tips are displayed when the mouse pointer is positioned over a graph data element. Use the TIPS=NONE parameter to suppress data tips for ActiveX and Java. For example:

```ods html parameters="TIPS="NONE"
```

For SAS/GRAPH procedures that support the HTML= option, you can add custom data tips to the output as described in “Adding Custom Data Tips with the HTML= Option” on page 191. To determine whether a procedure supports the HTML= option, refer to the description for that procedure in Part 6: SAS/GRAPH Procedures.

Note: The ACTIVEX and ACTXIMG devices do not display custom data tips when the number of observations exceeds 5000.

Note: Terminals set to use 16-bit colors or 32-bit colors are not supported when specifying data tips for output generated using DEVICE=ACTXIMG

---

Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options

About the URL=, HTML=, and HTML_LEGEND= Options

The URL=, HTML=, and HTML_LEGEND= options can be used in a number of statements that generate graphs. These options are used to add drill-down links, data tips, and enhancements to web presentations that are generated with the following device drivers:

- SVG, PNG, or GIF
- JAVA, JAVAIMG, ACTIVEX, and ACTXIMG
- JAVAMETA

The following table shows what each of these options can add to a graph.

<table>
<thead>
<tr>
<th>Option</th>
<th>What It Can Add to a Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL=</td>
<td>drill-down links to the data elements</td>
</tr>
<tr>
<td>HTML=</td>
<td>drill-down links, data tips, and enhancements to the data elements</td>
</tr>
<tr>
<td>HTML_LEGEND=</td>
<td>drill-down links, data tips, and enhancements to the graph legend values</td>
</tr>
</tbody>
</table>

As shown in the table, both the URL= option and the HTML= option can be used to add drill-down links to the data elements of a graph. You must use either the URL= option or the HTML= option, but not both. If you just want to link your graph data elements to a URL, use the URL= option or the HTML= option. However, if you want to add data tips or other enhancements to your graph data elements, you must use the HTML= option.
Add the Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options

Note: If you specify both the URL= and HTML= option in your SAS/GRAPH procedure statement, the URL= option is ignored. A warning message appears in the SAS log.

Link Variables and Enhancement Variables

The URL=, HTML=, and HTML_LEGEND= options identify a variable in the data set that contains the link information. The variable can contain a link URL only or it can contain other functionality, such as a data tip or a link target. If the variable contains only a link URL, it is referred to as a link variable. If the variable contains a data tip, link target, or other functionality, because it does more than establish a link, it is referred to as an enhancement variable. Only link variables can be used with the URL= option. Both link variables and enhancement variables can be used with the HTML= and HTML_LEGEND= options.

The syntax of the link variable value depends on the link option with which it is used. For the URL= option, the link value syntax is as follows:

"URL<anchor name>"

For the HTML= and HTML_LEGEND= options, the link value syntax is as follows:

"href='URL<anchor name>'"

An enhancement variable is used when other attributes such as TITLE= or TARGET= are included in the link value in order to provide additional functionality. The enhancement value syntax is as follows:

"attribute='value' <...attribute='value'>"

where attribute can be an HTML <A> tag attribute such as HREF=, TITLE=, TARGET=, or ALT=, or an SVG device driver attribute such as ONMOUSEOVER. See “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198.

The URL= option provides the simplest means of adding drill-down links to your presentation. You specify only the URL string, and the SAS/GRAPH software codes the URL into an HREF= attribute-value pair for you. The HTML= and HTML_LEGEND= options enable you to add one or more HTML <A> tag or SVG device driver attribute-value pairs to your link value. However, you must code the attribute-value pairs yourself, which can be complicated when multiple attribute-value pairs are included.

Note: Values created for the URL= option, and values created for the HTML= and HTML_LEGEND= options are not interchangeable.

Working with Link and Enhancement Variables

Creating Link and Enhancement Variables in a Web Presentation

To create link and enhancement variables for your web presentation, you must do the following:

- define the link variable or enhancement variable, or both
- add data to the link variable or enhancement variable, or both
- for a link variable, in your SAS/GRAPH procedure statement, do one of the following:
  - identify the variable in the URL= option
• identify the variable in the HTML= option, the HTML_LEGEND= option, or both

• for an enhancement variable, in your SAS/GRAPH procedure statement, identify the variable in the HTML= option, the HTML_LEGEND= option, or both

For example, assume that you have a web-based sales presentation that consists of a total sales bar chart in file sales.htm and a detailed sales chart for each region in file report.htm. Each bar in the total sales chart indicates the total sales for one of three corporate regions: central, southern, and western. You want to link each bar in your total sales chart to the detailed sales report for that region in file report.htm as shown in the following figure.

In addition, when a bar in the total sales chart is clicked, you want the drill-down chart in file reports.htm to be opened in a separate browser window named REGIONSALES.

To create the drill-down links for your presentation, perform the following steps:

1. Determine the link value that you need to implement your link behavior. For drill-down graphs to open in a separate browser window, your link value must include the URL and the TARGET= attribute as shown in the following example:

   href="reports.html#west" target="regionsales"

   See “Considerations When Using Anchored Links and Link Targets” on page 195.

2. Determine the type of variable that you need to create and the syntax for the link value. Because the value in this case includes a URL and the TARGET= attribute, you must use an enhancement variable. Therefore, the syntax for the link value is as follows:

   "href='reports.html#west' target='regionsales'"

   See “Link Variables and Enhancement Variables” on page 193.

3. Define your link variable in your data set. The following code fragment defines a link variable named RPT and assigns that variable a length of 80 characters.

   data regsales;
     input Region State Sales;
     length RPT $80;

   Note: When you create your link variable, be sure to define it with a length that is sufficient to contain your link value. There is no limit on the length of the variable.

4. In your DATA step, assign a value to your link variable as shown in the following example:

   RPT="href='reports.html#west' target='regionsales'";

   For information about assigning values to your link variable, see “Assigning Values to Link and Enhancement Variables” on page 195.
5. In your SAS/GRAPH procedure statement, use the HTML= option to identify the enhancement variable RPT as shown in the following code fragment.

```sas
   title "Regional Sales";
   proc gchart data=regsales;
      vbar region / sumvar=sales subgroup=region
         html=RPT;
   run;
```

**Considerations When Using Anchored Links and Link Targets**

When you link to an anchor in the same HTML file, you can omit the HTML filename. Instead, include only the anchor name in the URL as shown in the following example:

```html
   RPT="href='#west' target='regionsales'"
```

However, if you want to use an SVG device to generate your web presentation, to link from the SVG document to an anchor in the parent HTML file, you must include the HTML filename and the anchor name in the link URL. If you omit the HTML filename in that case, the link will not work.

When you use the TARGET= attribute to open a drill-down link in a separate browser window, be aware of the following:

- If a pop-up blocker is active in your web browser, it might prevent the new window from opening. In that case, you must disable or temporarily override the pop-up blocker, or you must modify the pop-up blocker settings to allow the window to open. Refer to your pop-up blocker documentation for more information.
- An SVG document might not be permitted to open a new window in the Microsoft Internet Explorer 7 and Internet Explorer 8 browsers.

**Assigning Values to Link and Enhancement Variables**

The most obvious method of adding these variables to your data set is to manually add them to the desired observations in your data set. This method is not practical or feasible in many cases. In those cases, you can use IF/THEN statements or variable substitution in your DATA step.

The following diagram shows how link variables are assigned to a bar chart. The three bars represent regional sales for a company's central, southern, and western regions.

**Figure 16.2  Links in Drill-Down Graphs**

Each bar in the chart links to an anchor tag in an HTML file named reports.htm. The anchor names in the linked file are “Central,” “South,” and “West.” In the following DATA step, IF/THEN/ELSE statements are used to assign values to the link variable, which is then identified using the HTML= option.

```sas
   /* Define a file reference for the ODS output */
   filename odsout ".";
```
/* create data set REGSALES */
data regsales;
  length Region State $ 8;
  format Sales dollar8.;
  input Region State Sales;
  length rpt $80; /* the link dest. variable */
datalines;
West CA 13636
West OR 18988
West WA 14523
Central IL 18038
Central IN 13611
Central OH 11084
Central MI 19660
South FL 14541
South GA 19022
;
/* assign HREF values to link dest. variable */
data regsales;
  set regsales;
  if Region="Central" then
    rpt="HREF='reports.htm#central' target='regionsales'";
  else if Region="South" then
    rpt="HREF='reports.htm#south' target='regionsales'";
  else if Region="West" then
    rpt="HREF='reports.htm#west' target='regionsales'";
run;

goptions reset=all device=ActiveX;
ods _all_ close;
ods html path=odsout 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;
```sas
ods html anchor="west";
title "Western Sales";
proc gchart data=regsales;
   where region="West";
   vbar state /sumvar=sales;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

The following code prints data set REGSALES to the SAS log.
proc print data=work.regsales noobs;
run;

Here is the output.

**Figure 16.3 REGSALES Data Set**

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>Sales</th>
<th>rpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>CA</td>
<td>$13,636</td>
<td>HREF='reports.htm#West' TARGET='regionsales'</td>
</tr>
<tr>
<td>West</td>
<td>OR</td>
<td>$18,988</td>
<td>HREF='reports.htm#West' TARGET='regionsales'</td>
</tr>
<tr>
<td>West</td>
<td>WA</td>
<td>$14,523</td>
<td>HREF='reports.htm#West' TARGET='regionsales'</td>
</tr>
<tr>
<td>Central</td>
<td>IL</td>
<td>$18,038</td>
<td>HREF='reports.htm#Central' TARGET='regionsales'</td>
</tr>
<tr>
<td>Central</td>
<td>IN</td>
<td>$13,611</td>
<td>HREF='reports.htm#Central' TARGET='regionsales'</td>
</tr>
<tr>
<td>Central</td>
<td>OH</td>
<td>$11,084</td>
<td>HREF='reports.htm#Central' TARGET='regionsales'</td>
</tr>
<tr>
<td>Central</td>
<td>MI</td>
<td>$19,680</td>
<td>HREF='reports.htm#Central' TARGET='regionsales'</td>
</tr>
<tr>
<td>South</td>
<td>FL</td>
<td>$14,541</td>
<td>HREF='reports.htm#South' TARGET='regionsales'</td>
</tr>
<tr>
<td>South</td>
<td>GA</td>
<td>$19,022</td>
<td>HREF='reports.htm#South' TARGET='regionsales'</td>
</tr>
</tbody>
</table>

You could use variable substitution to simplify the DATA step. The URLs used in the preceding program all use the same HTML filename, 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 filename to generate the drill-down URLs.

data regsales;
   set regsales;
   rpt="HREF='reports.htm#'|Region ||" TARGET='regionsales'"
   run;

Links in SVG, PNG, and GIF Presentations

To add drill-down functionality to images generated with the SVG, SVGT, SVGZ, SVGVIEWPNG, and GIF drivers, do one of the following:

- Use the URL= option with a SAS/GRAPH procedure to add drill-down functionality to the graph data elements.
- Use the HTML= option with a SAS/GRAPH procedure to add drill-down functionality to the graph data elements.
- Use the HTML_LEGEND= option with a SAS/GRAPH procedure to add drill-down functionality to the legend entries.
Use both the HTML= option and the HTML_LEGEND= option with a SAS/GRAPH procedure to add drill-down functionality to the legend entries.

Additional functionality is available through the use of HTML attributes with the SVG graphics devices. For more information, see “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198.

**Links in ACTXIMG and JAVAIMG Presentations**

To add drill-down functionality to an image created with the ACTXIMG or JAVAIMG device drivers, use the HTML= option as described in “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192.

**Links in ACTIVEX Presentations**

To add drill-down functionality to the ActiveX control created with the ACTIVEX device driver, use the URL= option or the HTML= option as described in “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192.

**Links in JAVA Presentations**

To generate drill-down presentations using Java, specify the DEVICE=JAVA graphics option to generate your graphs using the Graph applet. The Graph applet is a Java applet that provides drill-down functionality by default. For an example, see “Example 3: Creating a Drill-Down Java Presentation for the Web” on page 220.

**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 and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192.

**Links in Animated GIF Presentations**

SAS/GRAPH does not directly support drill-down functionality for animated GIFs. To enable drill-down functionality from an animated GIF, use any third-party tools that are available to you. You can make the entire image a hotspot by including the <IMG> tag inside an <A HREF=> tag.

---

### Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes

#### Additional Link Enhancements That Are Available with the SVG Graphics Devices

The SVG graphics devices provide additional functionality that you can use to enhance your drill-down links. These devices include SVG, SVGT, SVGView, and SVGZ. For information about these devices, see Chapter 13, “Using SVG Graphics,” on page 141.
The additional functionality that the SVG graphics devices provide is based on the mouse events that are shown in Table 16.2 on page 199.

**Table 16.2 Mouse Events and Available Functionality**

<table>
<thead>
<tr>
<th>Mouse Event</th>
<th>Available Functionality</th>
</tr>
</thead>
</table>
| The mouse pointer is positioned over a linked shape. | You can add one or more of the following effects for the duration of the event:  
  • display a pop-up image in a designated area of the graph  
  • change the fill color of the linked shape  
  • change the fill opacity of the linked shape |
| A linked shape is clicked. | In addition to opening a drill-down link, you can add one or more of the following effects:  
  • change the fill color of the linked shape  
  • change the fill opacity of the linked shape |

The ONMOUSEOVER= and ONCLICK= HTML attributes control the mouse event effects with an SVG graphics device. These attributes are added to the HTML string with other attributes, such as HREF and ALT. The ONMOUSEOVER= HTML attribute controls the effects that occur when the mouse pointer is positioned over a linked shape. The ONCLICK= HTML attribute controls the effects that occur when a linked shape is clicked.

*Note:* The ONMOUSEOVER= and ONCLICK= HTML attributes are supported only by the SVG graphics devices. These attributes are ignored when other graphics devices are used.

**Using the ONMOUSEOVER= HTML Attribute**

Use the ONMOUSEOVER= HTML attribute in your HTML= option value to display a pop-up image, change the shape fill color, and change the shape fill opacity when the mouse pointer is positioned over a linked shape. The effects that are provided for a linked shape by the ONMOUSEOVER= HTML attribute persist only as long as the mouse pointer is positioned on that shape. The syntax of the ONMOUSEOVER= HTML attribute is as follows:

ONMOUSEOVER=function(s)

To achieve the effects that you want, you can specify one or more of the functions shown in Table 16.3 on page 199 for functions().

**Table 16.3 ONMOUSEOVER= HTML Attribute Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showImage</td>
<td>This function displays a pop-up image in a specified area of the graph. The pop-up image is stored in a separate file.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>changeColor</td>
<td>This function changes the color of the linked shape to a specified color.</td>
</tr>
<tr>
<td>changeOpacity</td>
<td>This function changes the fill opacity of the linked shape to a specified</td>
</tr>
<tr>
<td></td>
<td>value.</td>
</tr>
</tbody>
</table>

The syntax of the `ShowImage` function is as follows:

`'showImage("image-file", xorigin, yorigin, image-width, image-height')`  

The `image-file` parameter specifies the name of the file that contains the image to be displayed. The filename must be enclosed in quotation marks. The `xorigin` and `yorigin` parameters specify the X and Y coordinates of the origin of the image, which is the lower left corner of the image. The image can be placed anywhere in the graphics output area. However, any portion of the image that extends beyond the graphics output area is clipped. The `image-height` and `image-width` parameters specify the size of the image in pixels.

The syntax of the `changeColor` and `changeOpacity` functions are as follows:

`changeColor(color)`  
`changeOpacity(factor)`  

The `color` parameter can be a color name, such as red, green, or blue, or an RGB specification, such as #55CC22. The `factor` parameter can be a value between 0 and 1, which specifies the desired degree of opacity. A value of 0 makes the shape 100% opaque.

When you specify more than one function in your `ONMOUSEOVER= HTML attribute, you must separate each function with a semicolon. Here is an example of an `ONMOUSEOVER= HTML attribute that displays image myimage.gif and changes the color of the linked shape to blue when the mouse pointer is positioned over the shape.

`'ONMOUSEOVER=ShowImage("./myimage.gif",300,550,200,250); changeColor{blue}'`  

In this example, the `showImage` function displays image myimage.gif at coordinates (300, 550), and scales the image to 200 pixels wide by 250 pixels high. The `changeColor` function changes the color of the linked shape to blue when the mouse pointer is moved off of the linked shape, the image disappears, and the shape fill color reverts to its original color.

For an additional example, see “Example 4: Enhancing an SVG Drill-Down Presentation Using HTML Attributes” on page 225.

**Using the `ONCLICK= HTML Attribute`**

Use the `ONCLICK= HTML attribute in your HTML= option value to change the shape fill color and the shape fill opacity when a linked shape is clicked. The syntax of the `ONCLICK= HTML attribute is as follows:

`'ONCLICK=function(s)'`  

To achieve the effects that you want, you can specify one or both of the `changeColor` and `changeOpacity` functions that are used with the `ONMOUSEOVER= HTML attribute. (See Table 16.3 on page 199.) When you specify more than one function, separate each function with a semicolon. Here is an example of an `ONCLICK= HTML attribute that changes the fill opacity of a linked shape to 50% when it is clicked.
Generating Drill-Down Links with the Annotate Facility

You can use the Annotate facility to generate web presentations with drill-down links in two ways: you can use PROC GANNO and an Annotate data set to generate an entire web presentation with drill-down links, or you can apply an Annotate data set to add drill-down functionality to a web presentation that is generated with the PNG, GIF, or ACTXIMG device driver.

When to Use PROC GANNO to Generate Drill–Down Links

You can use ODS, the GANNO procedure, an Annotate data set, and a PNG or GIF device driver to generate a web presentation with drill-down links. This method of generating a drill-down presentation is preferred if you do not need to use an image from another SAS/GRAPH procedure in your web presentation. For example, you could use PROC GANNO to generate an HTML output file that showed a GIF image, with accompanying text, and a selectable label containing the text “Click Here”. Larger presentations with multiple drill-down links are also entirely feasible.

Note: Although you can use the Annotate facility to enhance ACTXIMG device-generated output, you cannot use the GANNO procedure to create a web presentation with the ACTXIMG device driver.

For information about the GANNO procedure, see Chapter 32, “GANNO Procedure,” on page 779. To generate a drill-down graph with PROC GANNO, see “Generating Drill–Down Links with the Annotate Facility” on page 202.

When to Apply Annotate Data Sets to Web Presentations

You can use Annotate data sets to add drill-down links to web presentations generated by any procedure that uses the ANNOTATE= option. The web presentation must be generated with the PNG, GIF, or ACTXIMG device driver.

Using an Annotate data set to add drill-down links is preferable in the following circumstances:

• when you cannot add drill-down functionality by other means. Some SAS/GRAPH statements do not support the HTML= option, which SAS/GRAPH needs to generate an image map in the HTML output file. If the procedure does support the ANNOTATE= option, then you can use that procedure as the basis of a web presentation with drill-down functionality.

• when you do not want web users to drill down by selecting graph elements. For example, if you did not want your web users to drill down by selecting the bars in a bar chart, you could define graphics elements with drill-down links using the Annotate facility.

To learn how to use the Annotate facility to add drill-down links to a web presentation, see “Generating Drill–Down Links with the Annotate Facility” on page 202.

Reference information about generating and applying Annotate data sets is provided in “The Annotate Function, Variable, and Macro Dictionaries” on page 638. Usage information is provided in Chapter 27, “Using Annotate Data Sets,” on page 635.
Generating Drill–Down Links with the Annotate Facility

Follow these steps if you are adding drill-down links to a web presentation or if you are generating an entire web presentations with PROC GANNO:

1. Plan your web presentation so that you know how and where you want to apply Annotate graphical elements with drill-down links. Also determine your drill-down URLs.

2. Generate an Annotate data set. Elements that can be defined as drill-down hot zones are generated by Annotate functions that use the HTML variable. To see which functions use the HTML variable, refer to Figure 27.3 on page 643. To generate the Annotate data set, see Chapter 27, “Using Annotate Data Sets,” on page 635.

3. Specify the PNG, GIF, or ACTXIMG device driver using the DEVICE= option in a GOPTIONS statement.

4. Close the currently open ODS destinations, and then open an HTML output file in ODS.

   ods _all_ close;
   ods html file="annodril.htm"
   style=analysis;

5. Generate a PNG or GIF image and identify the Annotate data set. Use the GANNO procedure or another SAS/GRAPH procedure that uses the ANNOTATE= option.


7. Generate any additional HTML files or images as needed to provide files that are named in drill-down URLs.

Examples of Using Annotated Data Sets

For examples of applying Annotate data sets to output, see “List of Annotate Examples” on page 655.

Examples

Example 1: Adding Custom Data Tips to a Graph in a Web Presentation

Features:

- VBAR statement
- VBAR options
  - GROUP=
  - HTML=

Other features:

- ODS HTML statement options
  - FILE=
  - STYLE=
- GOPTIONS statement option
  - BORDER
This example uses the HTML= procedure option to add custom data tips to the vertical bar chart that is shown in the following figure. It generates the data tip **North Carolina** and **Massachusetts** for the East bars and **California** and **Oregon** for the West bars.

**Output 16.1 Bar Chart with Custom Data Tips**

![Company Sales, Mid Year](image)

**Program**

```plaintext
/* 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
East NC 17638 1998 1
East MA 12811 1998 1
East NC 12184 1998 2
East MA 12760 1998 2
;
/* Use an IF/THEN statement to assign values to the variable RPT. */
```
data sales;
set sales;
if state in ("NC" "MA") then
   RPT="title='North Carolina and Massachusetts'";
if state in ("CA" "OR") then  RPT="title='California and Oregon'";
run;

/* Reset the graphics options to their defaults and specify a border. */
goptions reset=all border;

/* Generate the bar chart. Add the HTML= option to associate */
/* the custom data tips with each graph element. */
title "Company Sales, Mid Year";
proc gchart data=sales;
   vbar region / sumvar=sales
      group=year
      html=RPT;
run;
quit;

Example 2: Creating a Drill-Down HTML Presentation for the Web

Features:
- VBAR statement
  - VBAR options
    - AUTOREF
    - DESCRIPTION=
    - DISCRETE
    - GAXIS=
    - GROUP=
    - SPACE=
    - HTML=
    - HTML_LEGEND=
    - NAME=
    - OUTSIDE=
    - RAXIS=
    - SHAPE=
    - SPACE=
    - SUBGROUP=

Other features:
- ODS HTML statement options
  - ANCHOR=
  - BODY=
  - CONTENTS=
  - FRAME=
  - NEWFILE
  - PATH=
  - GPATH=
  - AXIS statement
  - BY statement
  - FORMAT statement
  - GOPTIONS statement option
  - XPIXELS=
  - YPIXELS=
About This Example

This example shows how to create 2-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 VBAR 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 PNG files created by the ODS HTML statement and the PNG device driver
• assign anchor names to the graphics output
• use the HTML= and HTML_LEGEND= procedure options to assign link targets

For more information, see “ODS HTML Statement” on page 397.

For creating 2-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
• create and use SAS macros for generating multiple charts

The program generates twelve linked bar charts that display data about the world's leading grain producers. The charts are described in “Output” on page 206. 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.
Output

The first chart, shown in Output 16.2 on page 206 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.

Output 16.2  Browser View of Overview Graph

The next two charts break down grain production by year. These charts are linked to the legend values in Output 16.2 on page 206. For example, when you select the legend value for 1995, the graph in Output 16.3 on page 206 appears.

Output 16.3  Browser View of Year Breakdown for 1995

Another group of charts breaks down the data by country. These charts are linked to the bars in Output 16.2 on page 206 and Output 16.3 on page 206. For example, when you
click the bar for China in either Output 16.2 on page 206 or Output 16.3 on page 206, the graph in Output 16.4 on page 206 appears.

**Output 16.4  Browser View of Breakdown for China**

Finally, the data is charted by grain type. These graphs are linked to the bars in Output 16.4 on page 206. If you click the bar for Rice, Output 16.5 on page 207 appears.

**Output 16.5  Browser View of Breakdown for Rice**

**About This Example SAS Program**

The SAS program for this example is divided into four parts:

- Part A on page 208 generates the overview graph shown in Output 16.2 on page 206.
- Part B on page 214 generates two year breakdown graphs similar to Output 16.3 on page 206.
- Part C on page 216 generates five country breakdown graphs similar to Output 16.4 on page 206.
Part D on page 218 generates the three top producer graphs similar to Output 16.5 on page 207.

The following sections describe the functionality and SAS code for each part of the example program.

Overview: Part A

In the program, Part A creates the overview graph shown in Output 16.2 on page 206. In addition to setting the graphics environment and creating the data set, Part A does the following:

- Adds three HTML variables to the data set. The variables contain the link targets and suitable data tip text for all of the graphs that support drill-down functionality. The HREF values for the HTML variables in the data set contain the following information about the link targets:
  - the name of the body file that is the target. The BODY= option in the ODS HTML statement assigns the 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= option in the ODS HTML statement.
  - Opens the HTML destination for the frame and contents files and the first body file.
- Creates one grouped 2-D vertical bar chart (shown in Output 16.2 on page 206), with drill-down on the bars and legend values. The bars, 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 unique 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 and data tip information for the bars and for the legend values to the HTML= option and HTML_LEGEND= option, respectively.
- Assigns the web path. The FILENAME statement defines the fileref ODSOUT.

Part A Program

```
filename odsout ".";
goptions reset=all xpixels=550 ypixels=500 cback=white;
data grainldr;
    length country $ 3 type $ 5;
    input year country $ type $ amount;
    format megtons comma5.0;
    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
;
proc format;
  value $country
    "BRZ" = "Brazil"
    "CHN" = "China"
    "IND" = "India"
    "INS" = "Indonesia"
    "USA" = "United States";
data newgrain;
  set grainldr;
  length yeardrill typedrill countrydrill $ 100;

  /* Assign targets for the YEAR values. */
  yeardrill=
    "TITLE="||quote(trim(left(year)))||' '||
    "HREF="||quote('year'||trim(left(year))||'_body.html')
;
  /* Assign targets for the COUNTRY values. */
  countrydrill=
    "TITLE="||quote(trim(left(put(country,$country.))))||' '||
    "HREF="||quote('country_body.html#'||trim(left(country)))
;
  /* Assign targets for the TYPE values. */
  typedrill=
    "TITLE="||quote(trim(left(type)))||' '||
    "HREF="||quote(trim(left(type))||'_body.html')
;
legend1 label=none
  shape=bar(.15in,.15in)
Program Description

Here is a detailed description of the SAS code for the Part A portion of the example program.

**ODSOUT specifies a destination for the HTML and PNG files.** The files are created in 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.

```sas
filename odsout ".";
```
Set the graphics environment. The XPIXELS= and YPIXELS= options specify the size of each graph as 550 pixels wide by 500 pixels high. The CBACK= option sets the graph background to white.

```plaintext
goptions reset=all xpixels=550 ypixels=500 cback=white;
```

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. Variable MEGTONS stores the value of AMOUNT in millions of metric tons.

```plaintext
data grainldr;
  length country $ 3 type $ 5;
  input year country $ type $ amount;
  format megtons comma5.0;
  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 49860
  1995 INS Rice 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 51165
  1996 INS Corn 8925
  1996 USA Wheat 62099
  1996 USA Rice 7771
  1996 USA Corn 236064
  
  create a format for the values of COUNTRY.
  proc format;
    value $country
      "BRZ" = "Brazil"
      "CHN" = "China"
      "IND" = "India"
      "INS" = "India"
      "USA" = "United States"
    run;
```
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. The HTML variable COUNTRYDRILL contains the targets for the values of the variable COUNTRY. The HTML variable TYPEDRILL contains the names of the files that are the targets for the values of the variable TYPE.

```sas
data newgrain;
  set grainldr;
  length yeardrill typedrill countrydrill $ 100;
  /* Assign targets for the YEAR values. */
  yeardrill=
    "TITLE="||quote(trim(left(year)))||' '||
    "HREF="||quote('year'||trim(left(year))||'_body.html');
  /* Assign targets for the COUNTRY values. */
  countrydrill=
    "TITLE="||quote(trim(left(put(country,$country.))))||' '||
    "HREF="||quote('country_body.html#'||trim(left(country)));
  /* Assign targets for the TYPE values. */
  typedrill=
    "TITLE="||quote(trim(left(type)))||' '||
    "HREF="||quote(trim(left(type))||'_body.html');
```

Define legend characteristics for all legends. SHAPE= specifies the shape and size of the legend entries. POSITION= specifies the position of the legend.

```sas
legend1 label=none
  shape=bar(.15in,.15in)
  position=(bottom center);
```

Close the currently open ODS destinations.

```sas
ods _all_ close;
```

Set the output device to PNG. DEVICE= generates the SAS/GRAPH output as PNG files.

```sas
goptions device=png;
```

Open the ODS HTML destination for the ODS graphics output. BODY= names the file for storing the HTML output. FRAME= names the HTML file that integrates the contents and body files. 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. GTITLE includes the graph title in the SAS/GRAPH output instead of the HTML output. PATH= specifies the ODSOUT fileref as the HTML destination for all the HTML and PNG files.

```sas
ods html body="grain_body.html"
  frame="grain_frame.html"
```
Suppress the label and values for the midpoint axis. The midpoint values 1995 and 1996 do not appear below each bar.

```sql
axis1 label=none value=none;
```

Modify the response axis. The LABEL= option specifies a new axis label and rotates it 90 degrees. The MINOR=NONE option suppresses the minor tick marks. The ORDER= option sets the response axis range to 0 to 500 in increments of 100. The OFFSET= option sets the offsets to zero.

```sql
axis2 label=(angle=90 "Metric Tons (millions)"
minor=none
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.

```sql
axis3 label=none
order=("China" "United States" "India" "Indonesia" "Brazil")
split=" " offset=(4,4);
```

Define the footnote.

```sql
title1 ls=1.5 "Corn, Rice, and Wheat Production";
title2 "Leading Producers for 1995 and 1996";
footnote1 ls=1.3 "Click the bars or legend colors to drill down."
```

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 YEAR. SPACE= controls the space between the bars. HTML= specifies COUNTRYDRILL as the variable that contains the targets for the bars. Because the COUNTRYDRILL variable contains the TITLE= and HREF= options, the URL= option cannot be used in this case. 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 filename, the catalog entry name GRAINALL is automatically assigned to the PNG file. DES= specifies the description that is stored in the graphics catalog and used in the Table of Contents.

```sql
proc gchart data=newgrain;
   format country $country.;
   vbar year / discrete type=sum sumvar=megtons
      group=country
      subgroup=year
      space=0
      maxis=axis1
      gaxis=axis3
      raxis=axis2
      autoref cref=graydd clipref
      legend=legend1
```
Overview: Part B

Part B of this program creates the two charts that show the grain production breakdown by year for 1995 and 1996. The chart for 1995 is shown in Output 16.3 on page 206. Each bar in these graphs 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 Output 16.2 on page 206, Output 16.5 on page 207, and Output 16.4 on page 206. 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. Here is an example of the HREF attribute that points to the graph of 1995 and is stored in the variable YEARDRILL:

HREF=year1995_body.html

YEARDRILL is assigned to the HTML_LEGEND= option in Part A on page 208.

Part B Program

axis4 label=none
    order=('China' 'United States' 'India' 'Indonesia' 'Brazil')
    split=' ' offset=(8,8);
%macro do_year(year);
ods html path=odsout body="year&year_.body.html"
    gtitle gfootnote
    path=odsout style=listing;
title1 ls=1.5 "Total Production for &year";
proc gchart data=newgrain (where=(year=&year));
    format country $country.;
    vbar country / type=sum sumvar=megtons
        subgroup=type
        legend=legend1
        raxis=axis2
        maxis=axis4
        width=8
        autoref cref=graydd clipref
        html=countrydrill
        html_legend=typedrill
    name="year_&year"
    des="Production Breakdown for &year";
run;
quit;
%mend do_year;
%do_year(1995);
Program Description

Here is a detailed description of the SAS code for the Part B portion of the example program.

**Suppress the label and order the values for the midpoint axis.** The ORDER= option specifies the new order for the midpoints. The SPLIT=" " option allows the labels to wrap on a blank space, if necessary.

```sas
axis4 label=none
  order=(*China* *United States* *India* *Indonesia* *Brazil*)
  split=" " offset=(8,8);
```

**Open the %DO_YEAR macro definition.** This macro accepts a four-digit year as a parameter.

```sas
%macro do_year(year);
```

**Open a new body file for the graph of production for the year specified.** Assigning a new body file closes GRAIN_BODY.HTML. The body filename includes the year that is passed into the macro so that it can be identified easily. The contents and frame files, which remain open, and provide links to all body files.

```sas
ods html path=odsout body="year&year._body.html"
  gtitle gfootnote
  path=odsout style=listing;
```

**Define the title for the chart.** The title includes the year that is passed into the macro.

```sas
title1 ls=1.5 "Total Production for &year";
```

**Subset the data for the year that is passed into the macro and generate the vertical bar chart for that year.** 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 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 PNG files use the catalog entry name specified by the NAME= option. The name and description include the year that is passed into the macro.

```sas
proc gchart data=newgrain (where=(year=&year));
  format country $country.;
  vbar country / type=sum sumvar=megtons
    subgroup=type
    legend=legend1
    raxis=axis2
    maxis=axis4
    width=8
    autoref cref=graydd clipref
    html=countrydrill
    html_legend=typedrill
    name="year_&year"
    des="Production Breakdown for &year";
run;
quit;
```

**Close the %DO_YEAR macro definition.**
Call the %DO_YEAR macro for the years 1995 and 1996 to generate the graphs.

```
%do_year(1995);
%do_year(1996);
```

**Overview: Part C**

Part C of this program produces the five graphs that show the breakdowns by country. The chart for China is shown in Output 16.4 on page 206. Macro %DO_COUNTRY is defined and used to generate the graphs for each country, and all of the graphs are stored in a one 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. This program assigns a name {COUNTRY} to the ANCHOR= variable value for each country. The graphs created by this part are referenced by the COUNTRYDRILL variable. The NAME= option specifies “country_country” as the name for the graphics output for each country. The DES= option specifies a generic description for the HTML Table of Contents.

**Part C Program**

```
axis5 label=none split=" " offset=(4,4);
axis6 label=(angle=90 "Metric Tons (millions)"
  minor=none
  order=(0 to 250 by 50)
  offset=(0,0);
ods html path=odsout body="country_body.html"
  gtitle gfootnote
  path=odsout style=listing;
options nobyline;
%macro do_country(country);
  ods html anchor="&country";
  title1 ls=1.5 "Breakdown for #byval(country)";
  title2 "(In Millions of Metric Tons)";
  proc gchart data=newgrain (where=(country="&country"));
    format country $country.;
    by country; /* Enables the use of #byval() in title and description */
    vbar year / discrete type=sum sumvar=megtons
      group=type
      subgroup=year
      legend=legend1
      outside=sum
      space=0
      maxis=axis1
      raxis=axis6
      gaxis=axis5
      autoref cref=graydd clipref
      html=typedrill
      html_legend=yeardrill
      name="country_&country"
```
Program Description

Here is a detailed description of the SAS code for the Part C portion of the example program.

**Suppress the group axis labels.**

```sas
axis5 label=none split=" " offset=(4,4);
```

**Modify the range axis label and range.**

```sas
axis6 label=(angle=90 "Metric Tons (millions)"
minor=none
order=(0 to 250 by 50)
offset=(0,0);
```

**Open a new body file.** Assigning a new body file closes YEAR1996_BODY.HTML.

```sas
ods html path=odsout body="country_body.html"
gtitle gfootnote
path=odsout style=listing;
```

**Suppress the BY line.**

```sas
options nobyline;
```

**Open the %DO_COUNTRY macro definition** This macro accepts a country name as a parameter.

```sas
%macro do_country(country);
```

**Set the anchor name to the specified country.**

```sas
ods html anchor="&country";
```

**Define the titles.**

```sas
title1 ls=1.5 "Breakdown for #byval(country)";
title2 "(In Millions of Metric Tons)";
```

**Generate the vertical bar chart of production for each country.** BY-group processing on variable COUNTRY is used here so that #BYVAL(COUNTRY) can be used in the TITLE1 statement and the DES= procedure option. Because the WHERE clause in the DATA= procedure option filters the data to include only one country, only one output file is generated per macro call in this case. The GROUP= option groups the bars by country. 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.
Because the TYPEDRILL variable contains the TITLE= and HREF= options, the URL= option cannot be used in this case. The RAXIS= option assigns the AXIS4 statement. The GAXIS= option assigns the AXIS5 statement. The MAXIS= option assigns the AXIS1 statement to the midpoint axis. The NAME= option specifies the name of the catalog entry. The graphics catalog entry name increments so that the PNG 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.

```plaintext
proc gchart data=newgrain (where=(country="&country"));
  format country $country.;
  by country; /* Enables the use of #byval() in title and description */
  vbar year / discrete type=sum sumvar=megtons
    group=type
    subgroup=year
    legend=legend1
    outside=sum
    space=0
    maxis=axis1
    raxis=axis6
    gaxis=axis5
    autoref cref=graydd clipref
    html=typedrill
    html_legend=yeardrill
    name="country_&country"
    des="Grain and Year Breakdown for #byval(country)";
run;
```

Close the %DO_COUNTRY macro definitions.

```plaintext
%mend do_country;
```

Call the %DO_COUNTRY macro for CHN, USA, IND, INS, and BRZ.

```plaintext
%do_country(CHN);
%do_country(USA);
%do_country(IND);
%do_country(INS);
%do_country(BRZ);
```

quit;

Overview: Part D

Part C of this program creates the charts that show the leading producers for each type of grain. The chart for rice is shown in Output 16.5 on page 207. This part of the program defines and uses macro %DO_TYPE to generate graphs that show the leading producers for each type of grain. A leading producer is defined as any producer that produces 30 metric megatons or more 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. Because each graph is stored in a separate file, the links to these graphs reference filename only and do not require an anchor name. The graphs created by this part are referenced by the TYPEDRILL variable.

Part D Program

```plaintext
%macro do_type(type);
```
Example 2: Creating a Drill-Down HTML Presentation for the Web

Program Description

Here is a detailed description of the SAS code for the Part D portion of the example program.

Open the %DO_TYPE macro definition. This macro accepts a type as a parameter.

```
%macro do_type(type);
```

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.

```
ods html path=odsout body="&type._body.html"
    newfile=table
    gtitle gfootnote
    path=odsout style=listing;
%let minamount=30;
title1 la=1.5 "Top Producers of &type";
title2 "(Producing &minamount Million Metric Tons or More)";
/* Produce the series of bar charts: country and type. */
proc gchart data=newgrain (where=(megtons ge &minamount and type="&type"));
    format country $country.
    vbar year / discrete type=sum sumvar=megtons
        group=country
        subgroup=year
        legend=legend1
        outside=sum
        space=0
        maxis=axis1
        raxis=axis6
        gaxis=axis5
        autoref cref=graydd clipref
        html=countrydrill
        html_legend=yeardrill
        name="type_&type"
        des="Top &type Producers (&minamount Million Metric Tons or More)";
    run;
quit;
%mend do_type;
%do_type(Corn);
%do_type(Rice);
%do_type(Wheat);
title;
footnote;
ods html close;
ods html; /* Not required in SAS Studio */
Define the lower threshold for the top producers. MINAMOUNT is the minimum amount of grain that a producer must produce in order to be considered a top producer. The value is expressed in metric megatons.

\%let minamount=30;

Define the titles.

title1 ls=1.5 "Top Producers of &type";
title2 "(Producing &minamount Million Metric Tons or More)";

Generate the vertical bar chart of leading producers for each grain type.

/* Produce the series of bar charts: country and type. */
proc gchart data=newgrain (where=(megtons ge &minamount and type="&type"));
format country $country.;
vbar year / discrete type=sum sumvar=megtons
group=country
subgroup=year
legend=legend1
outside=sum
space=0
maxis=axis1
raxis=axis6
gaxis=axis5
autoref cref=graydd clipref
html=countrydrill
html_legend=yeardrill
name="type_&type"

des="Top &type Producers (&minamount Million Metric Tons or More)";
run;
quit;

Close the %DO_TYPE macro definition.

\%mend do_type;

Call the %DO_TYPE macro for Corn, Rice, and Wheat.

\%do_type(Corn);
\%do_type(Rice);
\%do_type(Wheat);

Clear all of the titles and footnotes.

title;
footnote;

Close ODS HTML to close the output file, and then reopen ODS HTML (not required in SAS Studio).

ods html close;
ods html; /* Not required in SAS Studio */

---

**Example 3: Creating a Drill-Down Java Presentation for the Web**

**Features:**

- VBAR3D Statement
VBAR3D options
   GROUP=
   SUBGROUP=

Other features: ODS HTML statement options
   FILE=
   STYLE=

GOPTIONS statement options
   DEVICE=
   BORDER

The Graph applet is a Java applet that provides drill-down functionality by default. You do not need to add linking variables to your chart data to get drill-down links when using the Graph applet. To use the Graph applet to generate your graph, use the DEVICE=JAVA graphics option. When the web page is displayed, the drill-down functionality is enabled by default. The Graph applet retains the type and style of the initial graph for all the graphs in the presentation. This example creates a drill-down sequence of three-dimensional, vertical bar charts that use the ODS style LISTING.

Note:

If you are using SAS Studio, note the following:

- The ODS HTML5 destination does not support the JAVA device. In SAS Studio, use the ODS HTML destination instead.

- If you download the HTML file from SAS Studio to your local machine, you might need to change JAVA_CODEBASE in the HTML file to specify the path to the SAS/GRAPH Java applet JAR files before you open it in your browser.
The initial graph displayed by this example is shown in the following figure. In this graph, REGION is the independent variable, and SALES is the dependent variable. YEAR is the group variable, and STATE is the subgroup variable.

**Output 16.6** Graph Applet: Level 1

**Company Sales, Mid Year**

Clicking the bar segment labeled East generates the graph shown in **Output 16.7** on page 223. 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, and YEAR is the independent variable.
Output 16.7  Graph Applet: Level 2

Company Sales, Mid Year

Clicking the bar segment labeled 1998 generates the graph shown in Output 16.8 on page 224. 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, and STATE is the independent variable.
Program

```sas
/* Create a file references for the ODS output */
filename odsout ".";
/* 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;
```

Chapter 16 • Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality

Output 16.8  Graph Applet: Level 3
/* Specify the JAVA device for generating the chart. */
goptions reset=all border device=java;

/* Close the currently open ODS destinations. */
ods _all_ close;

/* Open the HTML destination. Specify vbarweb.htm as */
/* the output filename and LISTING as the style. */
ods html path=odsout file="vbarweb.htm" style=list;

/* Generate the bar chart. Group by YEAR and subgroup */
/* by STATE. */
title "Company Sales, Mid Year";
proc gchart data=sales;
  vbar3d region / sumvar=sales
    group=year subgroup=state;
run;
quit;

/* ODS html to close the output file, and then reopen ODS HTML. */
ods html close;
ods html; /* Not required in SAS Studio */

Example 4: Enhancing an SVG Drill-Down Presentation Using HTML Attributes

Features: VBAR statement
          VBAR options:
            HTML=
            NAME=
            PATTERNID=
            WIDTH=
          GREPLAY DELETE statement

Other features: BY-Group processing
                GOPTIONS statement options:
                  BORDER
                  DEVICE=
                  XPIXELS=
                  YPIXELS=
                ODS HTML statement options:
                  ANCHOR=
                  BODY=
                  PARAMETERS=
                  PATH=
                  STYLE=
                OPTIONS statement NOBYLINE option

This example uses the SVG graphics device with the ONMOUSEOVER= HTML attribute in the HTML= string to add special effects to a drill-down graph. The drill-down graph is a simple bar chart of regional sales data. Each bar in the main chart is linked to a drill-down graph that displays detailed sales data for that region. The special
effects are activated when the mouse pointer is positioned on a bar. The drill-down link for each bar contains the ONMOUSEOVER= HTML attribute. The ONMOUSEOVER= HTML attribute value includes the showImage and changeOpacity functions. The showImage function displays a pop-up preview image of a bar’s drill-down chart in the upper right corner of the graph when the mouse pointer is positioned on that bar. The changeOpacity function changes the fill opacity of a bar to 50% when the mouse pointer is positioned on that bar. When a bar is clicked, an enlarged version of the drill-down graph for that bar opens in the browser window.

The main graph is stored in file SALESREPORT.HTML. Here is the graph as it is initially displayed when this file is opened.

For example, when the mouse pointer is positioned on the Central bar, the fill opacity of the bar changes to 50% as shown in the following figure. A pop-up image of the Central Region Sales drill-down chart is displayed in the top right corner of the graphics output area.
If the Central bar is clicked, an enlarged version of the Central Region Sales drill-down chart is opened in the browser as shown in the following figure.
Program

filename gout ".";

data regsales(drop=drillurl imgurl);
  length Region State $8 drillurl imgurl $50 htmlstr $256;
  format Sales dollar8.;
  input Region State Sales drillurl imgurl;

  /* Create and add the HTML variable for each observation. */
  htmlstr="href="||quote(trim(drillurl))||
  " title="||quote(trim(Region)||" Region") ||
  " ONMOUSEOVER=ShowImage("||quote(trim(imgurl))||
  ",400,25,200,200); changeOpacity(.5)";

datalines;
  Central IL 18038 ./regsales.html#rpt ./sales.png
  Central IN 13611 ./regsales.html#rpt ./sales.png
  Central OH 11084 ./regsales.html#rpt ./sales.png
  Central MI 19660 ./regsales.html#rpt ./sales.png
  South FL 14541 ./regsales.html#rpt1 ./sales1.png
  South GA 19022 ./regsales.html#rpt1 ./sales1.png
  West CA 13636 ./regsales.html#rpt2 ./sales2.png
  West OR 18988 ./regsales.html#rpt2 ./sales2.png
  West WA 14523 ./regsales.html#rpt2 ./sales2.png
;

goptions reset=all border device=svg xpixels=600 ypixels=450;
ods _all_ close;
ods html body="salesreport.html" path=gout style=listing
  parameters=("drilldownmode"="html");
title1 "Company Sales";
footnote1 j=c "Click bar to view an enlarged region graph.";
proc gchart data=regsales;
  vbar region / sumvar=sales width=8
    patternid=midpoint
    html=htmlstr; /* Set the HTML variable to htmlstr. */
run;
quit;
ods html close;
proc greplay nofs igout=work.gseg;
  delete _all_
run;
quit;

options reset=all border device=pngt;
ods html body="regsales.html" path=gout anchor="rpt"
  style=listing;
title "#byval(region) Region Sales";
proc gchart data=regsales;
  vbar state / sumvar=sales width=10 name="sales" patternid=midpoint;
    by region;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

Program Description
Here is a detailed description of the SAS program for this example.

Set the file output path. By default, ODS sends its output to the SAS Work directory. You can create a fileref to specify a different location for your ODS output, which is done later in this program.

filename gout ".";

Create the sales data set. The data set includes the region name, state codes, and the regional sales figures. It also includes the drill-down URL for each of the drill-down graphs (DRILLURL) and the path to the pop-up image for each drill-down graph (IMGURL). Notice that the base anchor RPT is used in the drill-down URLs. This base anchor is specified later in this program. The HTMLSTR variable is added to the data for each observation to store the HTML drill-down string. It is formed from the DRILLURL and IMGURL variables, and includes the TITLE= and ONMOUSEOVER= HTML attributes. The ONMOUSEOVER= HTML attribute value contains the ShowImage and changeOpacity functions to achieve the desired effects. After the HTMLSTR variable is formed, the DRILLURL and IMGURL variables are no longer needed and are dropped from the data set.

data regsales(drop=drillurl imgurl);
  length Region State $8 drillurl imgurl $50 htmlstr $256;
  format Sales dollar8.;
  input Region State Sales drillurl imgurl;
  /* Create and add the HTML variable for each observation. */
  htmlstr="href="||quote(trim(drillurl))||
  " title="||quote(trim(Region)||" Region") ||
  " ONMOUSEOVER=ShowImage("||quote(trim(imgurl))||
  ",400,25,200,200); changeOpacity(.5)"
  ;
datalines;
  Central IL 18038 ./regsales.html#rpt ./sales.png
  Central IN 13611 ./regsales.html#rpt ./sales.png
  Central OH 11084 ./regsales.html#rpt ./sales.png
  Central MI 19660 ./regsales.html#rpt ./sales.png
  South FL 14541 ./regsales.html#rpt1 ./sales1.png
  South GA 19022 ./regsales.html#rpt1 ./sales1.png
  West CA 13636 ./regsales.html#rpt2 ./sales2.png
  West OR 18988 ./regsales.html#rpt2 ./sales2.png
  West WA 14523 ./regsales.html#rpt2 ./sales2.png
;

Set the graphics options for the top-level graph. The SVG graphics device is used for the top-level graph.

goptions reset=all border device=svg xpixels=600 ypixels=450;

Close the currently open ODS destinations.

ods _all_ close;
Specify the ODS HTML settings for the top-level graph. The top-level graph output is sent to file SALESREPORT.HTML in the directory that was defined earlier by fileref GOUT (the current directory in this example). The LISTING style is used for this graph. The drill-down mode is set to HTML.

```
ods html body="salesreport.html" path=gout style=listing
    parameters=("drilldownmode"="html");
```

Generate the top-level drill-down graph.

```
title1 "Company Sales";
footnote1 j=c "Click bar to view an enlarged region graph.";
proc gchart data=regsales;
    vbar region / sumvar=sales width=8
        patternid=midpoint
        html=htmlstr; /* Set the HTML variable to htmlstr. */
    run;
quit;
```

Close the ODS HTML destination.

```
ods html close;
```

Clear the GRSEG catalog. Because the image files must be SALES.SVG, SALES1.SVG, and SALES2.SVG, as defined in variable IMGURL in the data set, the existing SALES entries must be cleared from the GRSEG catalog. Otherwise, different filenames might be used. In that case, the pop-up images will not be displayed.

```
proc greplay nofs igout=work.gseg;
    delete _all_;
run;
quit;
```

Specify the graphics options for the drill-down graphs. Because the drill-down graph images are overlaid onto the main graph, the PNGT device is used for the drill-down graphs. As a result, the background of the main graph shows through the pop-up graph images.

```
goptions reset=all border device=pngt;
```

Suppress the BY line. BY-group processing is used to generate the drill-down graphs. To suppress the BY line in the drill-down graphs, set the NOBYLINE system option.

```
options nobyline;
```

Specify the ODS HTML settings for the drill-down graphs. The drill-down graphs are stored in file REGSALES.HTML in the directory that is defined by fileref GOUT. The ANCHOR=RPT sets the base anchor to RPT, which is specified in the DRILLURL variable in the data set.

```
ods html body="regsales.html" path=gout anchor="rpt" style=listing;
```

Generate the drill-down graphs.

```
title "#byval(region) Region Sales";
proc gchart data=regsales;
    vbar state / sumvar=sales width=10 name="sales" patternid=midpoint;
    by region;
```
run;
quit;

Close the ODS HTML destination to close the output file, and then reopen ODS HTML (not required in SAS Studio).

ods html close;
ods html; /* Not required in SAS Studio */
Chapter 17
Using SAS/GRAPH Output with Microsoft Office Products

What to Consider When Choosing an Output Format

Considerations

- whether you need output in a graphics format or a document format
- the resolution and size of your graphs
- the color depth required for your graphs
- whether you need multiple graphs per page
- whether you need to edit your graphs using Microsoft products or using other third-party software

Graphics Formats versus Document Formats

The SAS/GRAPH software supports output in both graphics format and document format. The graphics format includes graphics information and some text, such as titles, footnotes, and legends.
The document format can include both text and graphics in a single document. These documents store graphics in one of the following ways:

- in the format of the document
- in a graphics format embedded in the document
- in an external file that the document links to

To include images in a document, the images must be compatible with the document. Here is a summary of the compatibility between the SAS/GRAPH document and graphics formats:

**Table 17.1  Document Format and Graphics Format Compatibility**

<table>
<thead>
<tr>
<th>Document Format</th>
<th>Compatible Graphics Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>SVG, PNG, GIF, and ActiveX</td>
</tr>
<tr>
<td>RTF</td>
<td>EMF, PNG, and ActiveX</td>
</tr>
</tbody>
</table>

**Image Resolution and Size**

Each of the SAS/GRAPH graphics output devices has a default size and resolution setting for the graphics that they generate. For information about the default settings for each device, see “Overview: Using Graphics Devices” on page 79. 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 quality of your raster image, when you create your graph in SAS, set the size to the size that you need in the Microsoft application. Doing so avoids the need to resize after it is imported. See “Setting the Size of Your Graph” on page 113. 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 114.

**Color Depth**

Another consideration when choosing a graphics format is color depth. This is the number of bits that are used to represent each color in an image. Color depth can affect the smoothness, clarity, and color trueness of the elements in a rasterized image. A greater color depth means that more distinct colors are available to represent elements such as gradient shading and antialiasing in text.

Most of the graphics file formats support Truecolor, which provides a 24-bit color depth. The GIF format provides only an 8-bit color depth, which can represent up to 256 distinct colors in a single image. For many graphics, 8-bit color depth is sufficient. However, if your output includes background images, color gradients, or other color-intensive elements, consider using a format that supports Truecolor. The formats that support Truecolor include the following:

BMP  EPS  SVG
EMF  PNG
Multiple-Image Graphics Files

If you need to store more than one graph in a file, you can use one of the following methods:

- Use the GREPLAY procedure to replay multiple graphs to a file of the same format that was used to generate the original graphs.
- Use the ODS DOCUMENT destination and the DOCUMENT procedure to replay multiple graphs to a file of any supported format.
- Use the ODS PRINTER destination with a Universal Printer device that supports multiple-page documents.
- Use the GIF graphics device to insert multiple graphs into an animated GIF image. See Chapter 15, “Generating Animations,” on page 167.

If you intend to use animated GIFs, be aware that animated GIFs in Microsoft Word documents, PowerPoint presentations, and Excel worksheets do not play by default. Only the first image in the animation sequence is displayed. To play animated GIFs:

- in a Microsoft Word document, you can save the document as an HTML file, and then open the saved HTML file in a web browser.
- in a Microsoft PowerPoint presentation, you can view the presentation in the Slide Show mode.
- in a Microsoft Excel workbook, you can add ActiveX controls to your workbook forms to play each animated GIF.

For more information, refer to the Microsoft Office documentation.

Ability to Edit: Vector versus Raster Formats

You might need the ability to edit your graphs using Microsoft or other third-party software. Make sure that you choose a graphics format that enables you to perform the type of editing that you need to do. For vector formats, such as EMF and SVG, you can edit individual text and graphics elements using graphics editing software. For raster images, some programs such as Microsoft Paint enable you to edit the image. However, in Microsoft Office products, editing is limited to changing only the global attributes of the image, such the size, contrast, brightness, and so on.

Comparison of the Graphics Output

SAS/GRAPH Images That You Can Import into Microsoft Products

The SAS/GRAPH software can generate the following types of graphics output that can be imported into Microsoft products:
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 ODS RTF destination, when used with a device such as PNG or GIF, generates a single RTF file with an embedded image. When used with the ACTIVEX device, the ODS RTF destination generates a single page file with an embedded SAS/GRAPH ActiveX Control. When used with the JAVA device, the ODS RTF destination generates a single page file with an embedded JAVAIMG image.

- The ACTIVEX and JAVA devices are not supported in the Microsoft PowerPoint destination. If you specify either of these devices, then SAS/GRAPH substitutes the ACTXIMG device for ACTIVEX, and the JAVAIMG device for JAVA.

- The ACTXIMG and JAVAIMG devices generate a PNG file that contains a graph. This PNG image file is identical to the image generated by the ACTIVEX and JAVA devices.

- When you specify the ACTIVEX or JAVA device in a procedure that does not support that device, the ACTXIMG or JAVAIMG device is used instead.

Table 17.2 on page 236 provides a brief comparison of these graphics output formats and lists some of the graphics output devices that generate each output type. For detailed information about all of the graphics output devices, see “Overview: Using Graphics Devices” on page 79.

Table 17.2  Comparison of the Graphics and Document Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Advantages and Limitations</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMF</td>
<td>Advantages:</td>
<td>SASEMF</td>
</tr>
<tr>
<td></td>
<td>• Most Windows applications recognize the EMF format.</td>
<td>EMF</td>
</tr>
<tr>
<td></td>
<td>• Graphs stored in EMF can usually be edited after they are imported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Graphs are imported at full size into Office, and can be resized without a loss of quality.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Limitations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The EMF format does not support transparency.  See “Working around the EMF Transparency Limitation” on page 238.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Only one graph per file is supported.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Advantages and Limitations</td>
<td>Devices</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>PNG</td>
<td><strong>Advantages:</strong></td>
<td>PNG (no transparency)</td>
</tr>
<tr>
<td></td>
<td>• Designed to display images on the web.</td>
<td>PNG300 (no transparency)</td>
</tr>
<tr>
<td></td>
<td>• Uses lossless data compression.</td>
<td>PNGT (transparency)</td>
</tr>
<tr>
<td></td>
<td>• Supports transparency (with the PNGT device).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Can store high-resolution images.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Supports truecolor images.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Limitation:</strong> cannot be resized without a loss of quality.</td>
<td></td>
</tr>
<tr>
<td>TIFF</td>
<td><strong>Advantages:</strong></td>
<td>TIFF (RGBA color)</td>
</tr>
<tr>
<td></td>
<td>• TIFF can store high-resolution graphics.</td>
<td>TIFFB (monochrome)</td>
</tr>
<tr>
<td></td>
<td><strong>Limitations:</strong></td>
<td>TIFFK (CMYK color)</td>
</tr>
<tr>
<td></td>
<td>• TIFF is not a web graphics format.</td>
<td>TIFFP (256 colors)</td>
</tr>
<tr>
<td>GIF and BMP</td>
<td><strong>Advantages:</strong></td>
<td>BMP</td>
</tr>
<tr>
<td></td>
<td>• GIF supports transparent backgrounds.</td>
<td>GIF</td>
</tr>
<tr>
<td></td>
<td>• GIF supports alpha-transparency (RGBA), which enables the blending of overlapping colors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• GIF supports anti-aliasing, which improves the appearance of text and angled lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• GIF can store multiple images per file when it is formatted as an animated GIF.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Both formats support the IBACK option and the IMAGE annotation function for including logos and other images in the background of the graph.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Limitations:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Both formats have a fixed resolution of 96 DPI.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The GIF standard is limited to 256 colors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Both formats lose quality when resized.</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td><strong>Advantage:</strong> Can store text and graphics.</td>
<td>GIF</td>
</tr>
<tr>
<td></td>
<td><strong>Limitation:</strong> The images are not loaded into a PowerPoint or Excel document when the HTML is imported. Only the text and tables are imported.</td>
<td>ACTIVEX <strong>&quot;</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACTXIMG and JAVAIMG, which create PNG files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PNG and PNGT</td>
</tr>
<tr>
<td>Type</td>
<td>Advantages and Limitations</td>
<td>Devices</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>RTF</td>
<td>Advantages:</td>
<td>ACTIVEX ** **</td>
</tr>
<tr>
<td></td>
<td>• Designed specifically for sharing documents between word processors.</td>
<td>ACTXIMG and JAVAIMG, which create PNG files</td>
</tr>
<tr>
<td></td>
<td>• Can store both text and graphics.</td>
<td>PNG and PNGT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SASEMF and EMF</td>
</tr>
</tbody>
</table>

* The default output environment in SAS Studio does not support the ACTIVEX device in the HTML format. To use the ACTIVEX device in HTML output in SAS Studio, you must open your own ODS HTML destination. See “About Using the SAS/GRAPH ACTIVEX and JAVA Devices in SAS Studio” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.

** See SAS Usage Note 57554 on support.sas.com.

*** The SAS/GRAPH ActiveX Control is not compatible with the RTF output format when UTF-8 character encoding is used. See “The ACTIVEX Device and RTF with UTF-8 Character Encoding” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.

** Working around the EMF Transparency Limitation

To work around the EMF transparency limitation, use the CBACK= or IBACK= graphics options to assign the matching color or image for the graph background. You could instead edit the EMF file after it is imported to remove the default background.

** Enhancing Your Graphs

You can use various features in SAS/GRAPH that enable you to enhance your graphs. The following table lists some of these features.

** Table 17.3 Features That Can Enhance Your Graph

<table>
<thead>
<tr>
<th>Feature in SAS/GRAPH</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing the style of the graphic</td>
<td>Chapter 20, “Using ODS Styles, Device Parameters, and Options,” (p. 273)</td>
</tr>
<tr>
<td>Adding annotations to the graph</td>
<td>Chapter 27, “Using Annotate Data Sets,” (p. 635)</td>
</tr>
<tr>
<td>Making the graph interactive</td>
<td>“Generating Interactive ActiveX Output” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide</td>
</tr>
<tr>
<td>Adding drill-down links and data tips to the graph</td>
<td>Chapter 16, “Enhancing Web Presentations with Chart Descriptions, Data Tips, and Drill-Down Functionality,” (p. 187)</td>
</tr>
<tr>
<td>Animating the graph</td>
<td>Chapter 15, “Generating Animations,” (p. 167)</td>
</tr>
</tbody>
</table>
You can import SAS/GRAPH output that you generated in the HTML or RTF format into Microsoft Office documents such as Microsoft Word, Microsoft Excel, and Microsoft PowerPoint. For information about how to import HTML or RTF files into Microsoft Office documents, refer to the Microsoft web site. You can also use the ODS destination for PowerPoint to generate SAS/GRAPH output in a Microsoft PowerPoint format. If your SAS/GRAPH output uses the SAS/GRAPH ActiveX Control, ActiveX controls must be enabled in the Microsoft product in order for the import to be successful. Refer to the Microsoft product documentation for information about how to enable ActiveX controls. See “Generating Interactive ActiveX Output” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide for information about how to generate interactive output using the SAS/GRAPH ActiveX Control.
Overview of Writing Graphs to a PDF File

You can use one of the following methods to write your graph output to a PDF file:

- use the ODS PDF destination to write to a standard PDF file or use the ODS PDF destination with the PRINTER=PDFA ODS option to write to a PDF file that you can archive
- use the ODS PRINTER destination with the PRINTER=PDF ODS option to write to a standard PDF file or with the PRINTER=PDFA ODS option to write to a PDF file that you can archive
- use the ODS LISTING destination with the DEVICE=PDF graphics option to write to a standard PDF file
- use the ODS LISTING destination with the DEVICE=PDFA graphics option to write to a PDF file that you can archive
Using any of these methods, you can do the following:

• create a PDF document that contains one or more pages of graphs with one or more graphs per page.
• use the system fonts or the resident PDF fonts in your document. See “Specifying Fonts” on page 242.
• use system options to control the page layout in your document. See “Changing the Page Layout” on page 244.

The ODS PDF destination and the PDF devices provide additional features and capabilities of their own. To help you determine which method to use for generating your PDF file, Table 18.1 on page 242 lists the additional features and capabilities that the ODS PDF destination and the PDF devices provide.

Table 18.1 Features and Capabilities Supported by ODS PDF and the PDF Devices

<table>
<thead>
<tr>
<th>Method Used</th>
<th>PDF File Types</th>
<th>Document Metadata</th>
<th>Bookmarks</th>
<th>Compression</th>
<th>Drill-down Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS PDF</td>
<td>PDF Version 1.4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ODS PDF PRINTER=PDFA</td>
<td>PDF/A-1b</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ODS PRINTER PRINTER=PDF</td>
<td>PDF Version 1.4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ODS PRINTER PRINTER=PDFA</td>
<td>PDF/A-1b</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ODS LISTING with DEVICE=PDF</td>
<td>PDF Version 1.4</td>
<td>No</td>
<td>No</td>
<td>Yes*</td>
<td>Yes</td>
</tr>
<tr>
<td>graphics option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODS LISTING with DEVICE=PDF</td>
<td>PDF/A-1b</td>
<td>No</td>
<td>No</td>
<td>Yes*</td>
<td>Yes</td>
</tr>
<tr>
<td>graphics option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The DEFLATION= system option must be specified in the OPTIONS statement.

See also “Creating an Archivable PDF File” on page 243 and “Adding Drill-Down Graphs in Your PDF File” on page 245.

Specifying Fonts

The ODS PDF destination and the PDF devices support the SAS/GRAPH fonts, the TrueType fonts that are installed with the Base SAS product, the TrueType fonts that are installed on the system and are registered in the SAS registry, and the resident PDF fonts. The resident PDF fonts are the Base 14 fonts that are installed by default with the Adobe Acrobat Reader. These fonts include:

Courier
Courier/oblique
Courier/bold

Helvética/oblique
Helvética/bold
Helvética/bold/oblique

Times/bold
Times/bold/italic
Symbol
Creating an Archivable PDF File

You can use the ODS PDF destination, or the ODS LISTING destination with the DEVICE=PDFA graphics option, to write your graph output to an archivable PDF file. By default, the ODS PDF destination writes your output to a PDF Version 1.4 file. To write your graphs to a PDF file that can be archived, add the PRINTER=PDFA option to your ODS statement. The PDFA Universal Printer shortcut device creates a PDF file that is compliant with PDF/A-1b standards and can be archived. See Chapter 9, “Using Graphics Devices,” on page 79 for information about the PDFA Universal Printer shortcut device.

See “Example: Writing Multiple Graphs to a One-Page, PDF/A-1b-Compliant File” on page 250 for an example of how to create an archivable PDF file using the ODS PDF destination.

You can also use the ODS LISTING destination with the DEVICE=PDFA graphics option to create an archivable PDF file. The following example writes a simple bar chart of electrical power revenue and generation sources to an archivable PDF file named useprev.pdf.

```sas
/* Create the output file reference */
filename pdfout "useprev.pdf";

/* Specify the PDFA device and the output filename */
goptions reset=all device=pdfa border gsfname=pdfout;

/* Close the currently open ODS destinations */
ods _all_ close;

/* Open the LISTING destination and specify the style */
ods listing style=statistical;

/* Generate the graph */
title "US Electric Power - Revenue and Generation Sources";
proc gchart data=sashelp.electric (where=(year >= 2000)) ;
  vbar year / discrete sumvar=Revenue subgroup=Customer;
run;
quit;

/* Close ODS LISTING and open ODS HTML */
ods listing close;
ods html; /* Not required in SAS Studio */
```
Changing the Page Layout

The following system options enable you to change the page layout for your PDF document when you use the ODS PDF destination or one of the PDF devices:

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

The `PAPERSIZE=` and margin options must be set before the ODS PDF statement is executed. These settings apply to the entire file and cannot be changed as the output is being generated. In contrast, the `ORIENTATION=` system option can be used to set the initial page orientation and to change the page orientation at any point as the output is being generated. See SAS System Options: Reference for information about these system options. See “Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata” on page 246 for an example of how to use these system options to change the page layout of a PDF file.

Adding Metadata to Your PDF File

The following ODS PDF options enable you to add document metadata to your PDF file:

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

See “Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata” on page 246 for an example of how to add metadata to a PDF file.

Adding Bookmarks for Your Graphs

The ODS `PROCLABEL=label` statement enables you to add bookmarks for your graphs when you use the ODS PDF destination. The `PROCLABEL=` ODS option specifies the name of the top-level bookmark. The description for each procedure that you run after your ODS `PROCLABEL=` statement is added as a subtopic under the top-level bookmark that the `PROCLABEL=` option defines. You can use the `DESCRIPTION=` option to set the text of the subtopic bookmark for each graph procedure. If you do not specify a description, the default graph description is used. See “Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata” on page 246 for an example.
Adding Drill-Down Graphs in Your PDF File

The SAS/GRA PH PDF device enables you to add drill-down graphs to your PDF file when you generate your PDF file using the ODS PDF or ODS PRINTER destination. The drill-down links include links that are created with the URL=, HTML=, and HTML_LEGEND= options for graph statements and annotation data, and the LINK= option for TITLE, FOOTNOTE, and NOTE statements. The links can be to an external file or URL. Creating drill-down graphs for a PDF document is similar to creating drill-down graphs for a web presentation. See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192.

**Note:** In the PDF format, when the mouse pointer is positioned on a drill-down link, a data tip displays the destination URL for that link. Custom data tips are not supported in the PDF format.

In order to write a drill-down graph to a PDF file, you must do all of the following:

- use one of the following ODS statements to open the ODS PDF or ODS PRINTER destination:
  
  `ODS PDF`  
  
  `ODS PRINTER PRINTER=PDF`

  If you use the ODS PRINTER destination, you must specify the PDF printer. For more information about the ODS PDF and ODS PRINTER destinations, see *SAS Output Delivery System: User’s Guide.*

  **Note:** Because the drill-down graphs are external to the PDF file, do not use the PDFA printer to generate an archivable PDF file. An archivable PDF file must be a totally self-contained file.

- use one or more of the following options to specify the URL for your drill-down links:
  
  - the URL= or HTML= option in your drill-down chart statement  
  - the HTML_LEGEND= option in your drill-down chart statement  
  - the HTML= option in your annotate data  
  - the LINK= option on your TITLE, NOTE, or FOOTNOTE statement

If all of these conditions are not met, your graph is written to the PDF file, but the drill-down links do not work as expected.

See the following examples:

- “Example: Writing a Drill-Down Graph to a PDF File” on page 251
- “Example: Writing Multiple Drill-Down Graphs to a PDF File Using BY-Group Processing” on page 255
- “Example: Writing Multiple Graphs to a PDF File Using the GREPLAY Procedure” on page 258
Changing the Default Compression Level for Your PDF File

The COMPRESS= ODS PDF option enables you to change the default compression level for your PDF file. The COMPRESS= option can be set to an integer value between 0 and 9, which specifies the level of compression. A value of 0 means no compression. The default level is 6. For information about the ODS PDF statement COMPRESS= option, see “ODS HTML Statement” in SAS Output Delivery System: User’s Guide.

You can also use the DEFLATION= system option to compress your PDF files. Like the COMPRESS= ODS PDF option, the COMPRESS= system option can be set to an integer value between 0 and 9, where 0 indicates no compression and 9 indicates maximum compression. For information about the DEFLATION= system option, see “DEFLATION= System Option” in SAS System Options: Reference.

Examples: Writing Graphs to a PDF File

About These Examples

The examples in this section show you how to export graphs to a PDF file. The examples include:

- “Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata” on page 246
- “Example: Writing Multiple Graphs to a One-Page, PDF/A-1b-Compliant File” on page 250
- “Example: Writing a Drill-Down Graph to a PDF File” on page 251
- “Example: Writing Multiple Drill-Down Graphs to a PDF File Using BY-Group Processing” on page 255
- “Example: Writing Multiple Graphs to a PDF File Using the GREPLAY Procedure” on page 258

Example: Writing Graphs to a PDF File That Contains Bookmarks and Metadata

Here is an example that writes a multipage PDF document to file EuropeanCars.pdf. The document contains two sets of graphs that show information about European cars and car makers. The first set of graphs shows the number of models that each car maker produces that fall within the following ranges of fuel efficiency in highway miles per gallon (MPG):

- 19 MPG or less
- between 20 and 29 MPG (inclusive)
- 30 MPG or higher

The second set of graphs shows the total number of models of the following vehicle types: sedans, SUVs, wagons, and sports cars. RUN-GROUP processing is used to
generate all of the graphs. Each page displays a single bar chart and is set up for A4 paper with a 1 cm right, left, and bottom margin, and a 2 cm top margin. For the first set of graphs, the page orientation for each graph is set as shown in the following table.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Page Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 MPG or less</td>
<td>Portrait</td>
</tr>
<tr>
<td>between 20 and 29 MPG</td>
<td>Landscape</td>
</tr>
<tr>
<td>30 MPG or higher</td>
<td>Portrait</td>
</tr>
</tbody>
</table>

For the second set, the page orientation for each graph is set as shown in the following table:

<table>
<thead>
<tr>
<th>Graph(s)</th>
<th>Page Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sedans</td>
<td>Landscape</td>
</tr>
<tr>
<td>SUVs, wagons, and sports cars</td>
<td>Portrait</td>
</tr>
</tbody>
</table>

The document also contains bookmarks and metadata. The PROCLABEL= ODS option is used to set the text of the top-level bookmark heading for each graph set. The DESCRIPTION= option is used with each procedure to set the text of each subheading bookmark. The document metadata includes the following:

- the document’s title: European Cars by MPG and Type
- the document’s author: J. L. Cho
- the subject of the document: European Cars
- a list of search keywords: automobiles, European, cars, MPG, sedans, wagons, and SUVs

Here is the SAS program for this example.

```sas
/* Close the currently open ODS destinations */
ods _all_ close;

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

/* Modify the PDF page properties. These options must
be set before the ODS PDF statement is executed. */
options papersize=A4
      leftmargin=1cm
      rightmargin=1cm
      bottommargin=1cm
      topmargin=2cm;

/* Open the PDF destination */
ods pdf style=seaside
    file="EuropeanCars.pdf" /* Output filename */
    compress=0 /* No compression */
```
/* Add metadata */
author="J. L. Cho"
subject="European Cars"
title="European Cars by MPG and Type"
keywords="automobiles European cars MPG sedans wagons SUVs";

/* Create a data set of European cars from SASHELP.CARS */
proc sort data=sashelp.cars(where=(origin="Europe")) out=europeancars;
   by make type mpg_highway;
run;

/* Create the axes */
axis1 label=(angle=90 "Number of Models Rated at Specified MPG Highway");
axis2 label=(angle=90 "Number of Models of Specified Type");
run;

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

/* Create the first set of graphs */
proc gchart data=europeancars;
   vbar Make / name="HighMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="30 MPG or Higher"; /* Set subheading text */
      title1 "30 MPG or Higher";
      where MPG_Highway >= 30;
run;
options orientation=LANDSCAPE; /* Display LANDSCAPE */
   vbar Make / name="MedMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="20 to 29 MPG"; /* Set subheading text */
      title1 "20 to 29 MPG";
      where MPG_Highway < 30 and MPG_Highway >= 20;
run;
options orientation=PORTRAIT; /* Display PORTRAIT */
   vbar Make / name="LowMPG"
      raxis=axis1 autoref clipref cref=lightgray lautoref=33
      description="19 MPG or Lower"; /* Set subheading text */
      title1 "19 MPG or Lower";
      where MPG_Highway < 20;
run;
quit;

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

/* Create the second set of graphs */
proc gchart data=europeancars;
   options orientation=LANDSCAPE; /* Display LANDSCAPE */
   vbar Make / name="Sedans"
      raxis=axis2 autoref clipref cref=lightgray lautoref=33
      description="Sedans"; /* Set subheading text */
      title1 "Sedans";
      where Type = "Sedan";
run;
options orientation=PORTRAIT; /* Display PORTRAIT */
vbar Make / name="SUVs"
   raxis=axis2 autoref clipref cref=lightgray lautoref=33
   description="SUVs"; /* Set subheading text */
   title1 "SUVs";
   where Type="SUV";
run;

vbar Make / name="Wagons"
   raxis=axis2 autoref clipref cref=lightgray lautoref=33
   description="Wagons"; /* Set subheading text */
   title1 "Wagons";
   where type="Wagon";
run;

vbar Make / name="Sports"
   raxis=axis2 autoref clipref cref=lightgray lautoref=33
   description="Sports Cars"; /* Set subheading text */
   title1 "Sports Cars";
   where type="Sports";
run;
quit;

/* Close the PDF destination, and then open the HTML destination
(not required in SAS Studio) */
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset options, titles, and footnotes */
goptions reset=all;
title;
footnote;

The bookmarks in the PDF document are shown in the following figure:

The document metadata is displayed on the Description tab of the Document Properties dialog box. To open the Document Properties dialog box, type CTRL-D anywhere in the PDF viewer window or right-click in the PDF viewer window, and then select Document Properties from the pop-up menu. The following figure shows the document metadata that is displayed for this example.
Example: Writing Multiple Graphs to a One-Page, PDF/A-1b-Compliant File

Here is an example that creates the PDF/A-1b-compliant file FourVbars.pdf, which contains four vertical bar charts on one page. The graphs show the average sales of sofas, chairs, desks, and beds in Canada by province. The PRINTER=PDFA ODS option is used to create a PDF file that is compliant with PDF/A-1b standards. To create a standard Version 1.4 PDF file, remove the PRINTER=PDFA option from the ODS statement. The STARTPAGE=NEVER ODS option is used to prevent a page break from being inserted between each graph. The HORIGIN= and VORIGIN= graphics options are used to position each graph on the page. Here is the SAS code for this example.

```sas
/* Close the currently open ODS destinations */
ods _all_ close;

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

/* Create the axes */
axis1 label=(angle=90 "Average Sales"); /* Left-side graph Y axis */
axis2 label=none;         /* Right-side graph Y axis */
axis3 label=("Province"); /* Midpoint axis */

/* Open PDF */
ods pdf style=printer notoc

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

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

/* Generate the graphs */
proc gchart data=sashelp.prdsal3;
    /* Format the sales values */
    format Actual comma5.0;

    /* Create the Sofas graph in the top-left quadrant */
```
Example: Writing a Drill-Down Graph to a PDF File

Here is an example that exports a drill-down graph to a PDF file. This example writes a top-level drill-down graph to a PDF file, and then writes the drill-down target graphs to HTML files. The top-level drill-down graph in the PDF file shows the total sales in three regions in which a company operates. A reference line indicates a sales target of $62,000 for each region. Each region bar in the graph is linked to a graph that breaks down the sales data for that region. When the mouse pointer is positioned on a bar, a data tip shows the URL to the drill-down graph. An example is shown in the following figure.
The drill-down graphs consist of one HTML file for each region: Central.htm, South.htm, and West.htm. Each HTML file contains a graph that breaks down the total sales data by state for that region. When you click a bar in the top-level graph, the HTML file to which that bar is linked appears in your web browser. In this example, if you click the Central bar, the Central.htm file is opened, which displays the following graph in your web browser.

Here is the SAS code for this example.

```sas
/* Define the ODS output path. */
```
%let outpath=.;
filename odsout "&outpath";

/* Define the PDF output filename. */
filename pdfout "&outpath\salesrpt_single_page.pdf";

/* Define the base URL for the links. */
%let baseurl=.;

/* Create the data set REGSALES. */
data regsales;
  length Region State $ 8 Location $ 15;
  format Sales dollar8.;
  input State Region Location Sales;
datalines;
IL Central EVANSTON 18038
IL Central CHICAGO 14322
IL Central AURORA 10768
OH Central COLUMBUS 13611
OH Central DAYTON 11084
OH Central CINCINNATI 19534
FL South MIAMI 14541
FL South TAMPA 16733
NC South RALEIGH 19022
NC South WILMINGTON 12876
NC South CHARLOTTE 13498
CA West SANTA-CRUZ 13636
CA West LONG-BEACH 15687
WA West SEATTLE 18988
WA West TACOMA 14523;

/* Add the link information to the data. */
data regsales;
  set regsales;
  length RPTR $ 80;
  if (Region="Central") then RPTR="&baseurl./Central.htm";
  else if (Region="South") then RPTR="&baseurl./South.htm";
  else if (Region="West") then RPTR="&baseurl./West.htm";
  else RPTR=.;
r un;

/* Close the currently open ODS destinations, and then set the
graphics options. */
ods _all_ close;
goptions reset=all device=png border xpixels=520 ypixels=450;

/* Clear titles and footnotes */
title;
footnote;

/* Create a macro to use for generating the region charts. */
%macro do_region(region);
  /* Open ODS HTML destination. Use the region name as the filename */
  ods html file="&region..htm" path=odsout style=listing;

/* Set the axis label and title */
axis1 label=("Total Sales");
title1 "Total Sales in &region Region";

/* Create the chart */
proc gchart data=regsales;
  vbar3d state / sumvar=sales outside=sum name="region"
    raxis=axis1 patternid=midpoint
    description="Total sales for &region region"
    shape=cylinder width=15;
    where region=&region;
run; quit;

/* Close the ODS HTML destination */
ods html close;
%mend do_region;

/* Call the %DO_REGION macro for Central, South, and West. */
%do_region(Central);
%do_region(South);
%do_region(West);

/* Open the PDF destination and set the style. */
ods pdf file=pdfout style=statistical;

/* Set the graphics options. You must use the PDF or PDFA device. */
goptions reset=all gsfname=outp
device=pdf border
  vorigin=2.5in horigin=0.6in vsize=550pt hsize=520pt;

/* Generate the Company Sales report PDF document. */
title1 "Company Sales Report";
title2 "Sales by Region";
footnote1 "(Click a bar for details.)";
axis1 label=("Total Sales") reflabel=(j=r "Target ");
proc gchart data=regsales;
  vbar3d region /
    cref=red lref=3 ref=62000 /* Draw a reference line at the */
    /* sales target. */
    sumvar=sales outside=sum raxis=axis1
    shape=cylinder width=15 patternid=midpoint
    description="Company sales report"
    url=RPTR; /* Specify RPT as the URL variable in the data. */
run; quit;

/* Close the PDF destination, and then open HTML (not required in SAS Studio)*/
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset graphics options, titles, and footnotes */
goptions reset=all;
title;
footnote;

In the SAS program, notice that column RPTR is added to the data. For each observation, RTPR is set to the URL of the appropriate report based on the value of
column REGION. Because column RPTR contains a URL string only, it is specified in the URL= option on the drill-down chart GCHART statement. The HTML= option cannot be used in this case.

**Example: Writing Multiple Drill-Down Graphs to a PDF File Using BY-Group Processing**

Here is an example that creates a PDF file that contains three drill-down graphs of sales data. This example uses the ODS LISTING destination with the PDF device and BY-group processing to write the top-level graphs to a PDF file. It uses the ODS HTML destination with BY-group processing to generate the drill-down graphs. In the PDF file, each graph shows the total sales data by state in each of three regions: Central, South, and West. In each of the graphs, each bar is linked to a drill-down graph that breaks down the total sales data for that state by store location. A data tip displays the URL to the drill-down graph as shown in the following figure.

The drill-down graphs consist of one graph for each of the following states: IL, OH, FL, NC, CA, and WA. Each drill-down graph shows the total sales data for a state by store location. All six of the drill-down graphs are written to the HTML file state.htm. In the state.htm file, a named anchor is placed at the top of each graph. The ANCHOR= option is used in the ODS HTML statement to set the base anchor name to ST. When BY-group processing occurs, the anchor names are assigned to each STATE value as they occur in the data. The first name assigned is the base name, which is ST in this example. For the subsequent names, a number is appended to the base name to create a unique name for each STATE value. Because the data is sorted by STATE, the anchor names are assigned as shown in the following table.
The named anchors are included in the URLs that link the bars in the top-level graph to the drill-down graph in the state.htm file. The data tip shown in the previous figure provides an example. Clicking the IL bar in this example opens the state.htm file and positions the cursor at the IL graph anchor (ST2) as shown in the following figure.

Here is the SAS code for this example.

```sas
/* Define the ODS output path. */
%let outpath=.;
filename odsout "&outpath";

/* Define the PDF output filename. */
filename pdfout "&outpath\salesrpt_single_page.pdf";

/* Define the base URL for the links. */
```
%let baseurl=./state.htm;

/* Create the data set REGSALES. */
data sales;
  length Region State $ 8 Location $ 15;
  format Sales dollar8.;
  input State Region Location Sales;
datalines;
CA West    SANTA-CRUZ 13636
CA West    LONG-BEACH 15687
FL South   MIAMI      14541
FL South   TAMPA      16733
IL Central EVANSTON 18038
IL Central CHICAGO 14322
IL Central AURORA   10768
NC South   RALEIGH    19022
NC South   WILMINGTON 12876
NC South   CHARLOTTE  13498
OH Central COLUMBUS 13611
OH Central DAYTON   11084
OH Central CINCINNATI 19534
WA West    SEATTLE    18988
WA West    TACOMA     14523
;
run;

/* Add the link information to the data */
data sales;
  set sales;
  length RPTS $ 80;
  if (State="CA") then RPTS="&baseurl.#ST";
  else if (State="FL") then RPTS="&baseurl.#ST1";
  else if (State="IL") then RPTS="&baseurl.#ST2";
  else if (State="NC") then RPTS="&baseurl.#ST3";
  else if (State="OH") then RPTS="&baseurl.#ST4";
  else if (State="WA") then RPTS="&baseurl.#ST5";
  else RPTS=.;
run;

/* Close the currently open ODS destinations, and then set the */
/* graphics options. */
ods _all_ close;
goptions reset=all device=png border hsize=5in vsize=4in;

/* Generate the state sales charts. Use ST as the base HTML anchor */
/* and state.htm as the output filename. The drill-down chart links */
/* to this file. */
ods html file="state.htm" path=odsout anchor="ST" style=listing;
title1 "Total Sales by Store In #byval(State)";
title2;
proc gchart data=sales;
pie3d Location / sumvar=Sales
  noheader
  slice=outside value=inside
  description="Total Sales by Store In #byval(State)"
  by State;
run; quit;
ods html close; /* Close ODS HTML. */

/* Sort the data by region. */
proc sort data=sales out=regsort;
  by Region;
run; quit;

/* Set the graphics options. You must use the PDF or PDFA device. */
goptions reset=all gsfname=outp device=pdf border
  vorigin=2.5in horigin=0.5in hsize=450pt vsize=380pt;

/* Open the PDF destination and specify the STATISTICAL style */
ods pdf file=pdfout style=statistical;

/* Generate the drill-down chart. */
axis1 label=("Total Sales");
title1 "Total Sales By State In the #byval(Region) Region";
footnote "(Click a bar for details.)";
proc gchart data=regsort;
  vbar3d state / sumvar=sales outside=sum name="region"
    raxis=axis1 patternid=midpoint
    shape=cylinder width=15
    url=RPTS; /* Specify RTPS as the URL variable. */
  by Region;
run; quit;

/* Close the PDF destination, and then open ODS HTML (not required in SAS Studio) */
ods pdf close;
ods html; /* Not required in SAS Studio */

/* Reset graphics options, titles, and footnotes */
goptions reset=all;
title;
footnote;

---

**Example: Writing Multiple Graphs to a PDF File Using the GREPLAY Procedure**

Here is an example that creates a sales report in both the HTML and PDF format. It first generates the sales report in the HTML format. It then uses the GREPLAY procedure to replay the graphs to a PDF file. The second graph in the report is a drill-down graph in both the HTML report and the PDF report.

/* Define the ODS output path. */
%let outpath=.;
filename odsout "&outpath";

/* Define the output filename. */
%let outfilename=salesrpt_single_page;
filename pdfout "&outpath\&outfilename.pdf";

/* Define the base URL for the links. */
%let baseurl=./state.htm;
/* Create the data set REGSALES. */
data sales;
  length Region State $ 20 Location $ 20;
  format Sales dollar8.;
  input State Region Location Sales;
datalines;
CA West    SANTA-CRUZ 13636
CA West    LONG-BEACH 15687
FL South   MIAMI      14541
FL South   TAMPA      16733
IL Central EVANSTON 18038
IL Central CHICAGO   14322
IL Central AURORA     10768
NC South   RALEIGH    19022
NC South   WILMINGTON 12876
NC South   CHARLOTTE  13498
OH Central COLUMBUS   13611
OH Central DAYTON     11084
OH Central CINCINNATI 19534
WA West    SEATTLE    18988
WA West    TACOMA     14523
;
/* Add the drill-down link information to the data. */
data sales;
  set sales;
  length RPTS $ 80;
  select(State);
    when("CA") RPTS="&baseurl.#ST";
    when("FL") RPTS="&baseurl.#ST1";
    when("IL") RPTS="&baseurl.#ST2";
    when("NC") RPTS="&baseurl.#ST3";
    when("OH") RPTS="&baseurl.#ST4";
    when("WA") RPTS="&baseurl.#ST5";
  otherwise RPTS=.;
end;
run;

/* Close the currently open ODS destinations, and then set the graphics options. */
ods _all_ close;
goptions reset=all device=png border xpixels=450 ypixels=450;

/* Generate the state sales charts. Use ST as the base HTML anchor, and state.htm as the output filename. The drill-down graph will link to this file. */
ods html path=odsout file="state.htm" anchor="ST" style=listing;
title1 "Total Sales by Store Location";
title2;
proc gchart data=sales;
  pie3d Location / sumvar=Sales noheader;
  by State;
run; quit;
ods html close; /* Close ODS HTML. */

/* Delete the old chart GRSEGs. */
/* Set the graph size. */
goptions reset=all xpixels=550 ypixels=550 gunit=pt;

/* Open the HTML destination to generate the HTML report. */
ods html path=odsout file="&outfilename..html" style=statistical;

/* Generate the region, state, and local charts. Use Region, State, and Location as the GRSEG names. */
axis1 label=('Total Sales');
proc gchart data=sales;
   /* Generate the Regional sales chart. */
   title1 'Total Sales by Region';
   vbar3d region / sumvar=sales outside=sum name="Region"
      raxis=axis1 patternid=midpoint shape=cylinder width=15
      description='Total Sales by Region';
run;

   /* Generate the drill-down State sales chart. */
   title1 'Total Sales by State';
   footnote '(Click a bar for details.)';
   hbar3d state / sumvar=sales sum sumlabel='Total Sales' name="State"
      patternid=midpoint raxis=axis1 shape=cylinder width=5
      description='Total Sales by State'
      url=RPTS; /* Specify the URL variable in the data. */
run;

   /* Generate the Location sales chart. */
   title1 'Total Sales by Location';
   footnote; /* Clear the footnote */
   pie location / name='Location' sumvar=sales noheading
      plabel=(height=9pt)
      description='Total Sales by Location';
run; quit;

/* Close the HTML destination. */
ods html close;

/* Set the graphics options for write to a PDF file. */
goptions device=pdf noborder;

/* Suppress the creation date and time in the PDF output. */
option nodate;

/* Create a replay template that centers that graph on the page and draws a navy border around each graph. */
proc greplay tc=work.tempcat nofs;
tdef newtemp
   1/llx=10 lly=20
   ulx=10 uly=80
   urx=90 ury=80
   lrx=90 lry=20
   color=navy;
   template newtemp;
quit;
/* Set the report title and footnote. */
title1 "Sales Report";
footnote1 "Sales data is based on first-quarter results.";

/* Open the ODS PDF destination to generate the PDF report. 
Specify NOGTITLE and NOGFOOTNOTE so that the report title 
and footnote appear in the printer output. */
ods pdf file=pdfout nogtitle nogfootnote;

/* Replay the three sales charts to the PDF file. */
proc greplay tc=work.tempcat igout=work.gseg nofs
  template=newtemp;
  treplay 1:Region des="Total Sales by Region";
  treplay 1:State des="Total Sales by State";
  treplay 1:Location des="Total Sales by Location";
run; quit;

/* Close ODS PDF, and then open ODS HTML (not required 
in SAS Studio. */
ods pdf close;
ods html; /* Not required in SAS Studio */

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

In order to add a title and footnote to the PDF report, TITLE1 and FOOTNOTE1
statements are used to define the title and footnote text. The NOGTITLE and
NOGFOOTNOTE options are then specified in the ODS PDF statement, which add the
title and footnote to the PDF output. For more information about the ODS PDF
statement NOGTITLE and NOGFOOTNOTE options, see SAS Output Delivery System:
User’s Guide.
Chapter 19
Troubleshooting Web Output

Overview
This chapter contains information that you can use to resolve rendering problems on client workstations. Included is a table listing solutions to common problems as well as various tips ranging from resolving web server access issues to modifying HTML character entities.

Solutions to Common Rendering Problems
If you or a member of your audience cannot display your presentation, then refer to the following table for solutions.

Note: To view your web presentation, your audience must view the presentation through one of the supported browsers. For a list of supported browsers, refer to the SAS web site Install Center. Select the System Requirements link for the appropriate operating system environment and search for the section about viewing HTML pages created for Java and ActiveX.

Note: For information about software required to open HTML output that is generated by either the JAVA device or the ActiveX device, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.
### Table 19.1 Web Troubleshooting

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Cause</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot access the HTML file.</td>
<td>Incorrect URL.</td>
<td>Check the URL in the browser.</td>
</tr>
<tr>
<td></td>
<td>Network access denied.</td>
<td>Check operating environment permissions for the HTML file.</td>
</tr>
<tr>
<td></td>
<td>Check firewall access permissions for Internet clients.</td>
<td>Check firewall access permissions for Internet clients.</td>
</tr>
<tr>
<td>Browser cannot display the file.</td>
<td>Browser or Java plug-in might not meet requirements.</td>
<td>Check the requirements. To view your web presentation, your audience must view the presentation through one of the supported browsers. For a list of supported browsers, refer to the SAS web site Install Center. Select the System Requirements link for the appropriate operating system environment and search for the section about viewing HTML pages created for Java and ActiveX. For information about software required to open HTML output that is generated by either the JAVA device or the ActiveX device, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.</td>
</tr>
<tr>
<td>ActiveX installation or running issues.</td>
<td>For ActiveX control installation or update instructions, or for the Internet Explorer web browser version required to run ActiveX control, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.</td>
<td>For ActiveX control installation or update instructions, or for the Internet Explorer web browser version required to run ActiveX control, see SAS/GRAPH: Java Applets and ActiveX Control User’s Guide.</td>
</tr>
<tr>
<td>User has not been authenticated for that browser and that web page.</td>
<td>Check to see whether authentication is needed, and then authenticate. See “Connecting to Web Servers That Require Authentication” on page 267.</td>
<td>Check to see whether authentication is needed, and then authenticate. See “Connecting to Web Servers That Require Authentication” on page 267.</td>
</tr>
<tr>
<td>Symptom</td>
<td>Cause</td>
<td>Remedy</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Browser does not recognize the file as HTML.</td>
<td>Ensure that the type of the HTML file is correctly specified.</td>
<td>Ensure that the DOCTYPE and MIME tags are correctly formatted.</td>
</tr>
<tr>
<td>Browser permissions too restrictive.</td>
<td>Check browser permissions. See “Checking Browser Permissions” on page 267.</td>
<td></td>
</tr>
<tr>
<td>Browser displays blank page.</td>
<td>Browser cannot access the referenced image file.</td>
<td>If not running an applet or control, check the image file at the location specified in the HTML file.</td>
</tr>
<tr>
<td>Browser cannot run the applet or control.</td>
<td>You can specify the location of control and applet files, or check browser permissions for running Java scripts. Or you can open the browser’s Java Console and trace the source of the error. For information detailing all of these tasks, see <em>SAS/GRAPH: Java Applets and ActiveX Control User’s Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Browser displays pop-up message Error: Not enough virtual memory to produce plot.</td>
<td>Client RAM is insufficient for rendering.</td>
<td>Generate a new graph using a smaller data set or a simpler graph. If using PROC GMAP, consider using PROC GREDUCE.</td>
</tr>
<tr>
<td>Graph is not rendering as specified by the ODS graph style.</td>
<td>A style attribute might not be enabled for your ODS destination.</td>
<td>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 <em>SAS Output Delivery System: User’s Guide</em>.</td>
</tr>
<tr>
<td>Symptom</td>
<td>Cause</td>
<td>Remedy</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A style attribute might be overridden by a global option,</td>
<td>Specify the minimum options needed for your graph, for example:</td>
<td></td>
</tr>
<tr>
<td>global statement option, procedure option, or statement option.</td>
<td><strong>goptions reset=all device=svg;</strong></td>
<td></td>
</tr>
<tr>
<td>In ActiveX, the user gets the message There is a pending reboot for</td>
<td>See SAS/GRAPH: Java Applets and ActiveX Control User's Guide.</td>
<td></td>
</tr>
<tr>
<td>this machine...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text font is incorrect.</td>
<td>Java font is defined differently.</td>
<td>See SAS/GRAPH: Java Applets and ActiveX Control User's Guide.</td>
</tr>
<tr>
<td>Text in browser shows incorrect characters.</td>
<td>Browser misinterpreting special characters.</td>
<td>Replace special characters with character entities. See “Using HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Character Entities” on page 267.</td>
</tr>
<tr>
<td>Graph in browser differs from graph in SAS.</td>
<td>A graphics option or global statement might be unsupported or partially</td>
<td>Refer to the descriptions for the options that you are using and to</td>
</tr>
<tr>
<td></td>
<td>supported for that applet or control. See also “Resolving Differences</td>
<td>Appendix 1, “Summary of ActiveX and Java Support,” on page 1441 for</td>
</tr>
<tr>
<td></td>
<td>between Graphs Generated with Different Technologies” on page 268.</td>
<td>information about whether a statement or option is supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A default value in the applet or control is overriding a default</td>
<td>Specify a value for the option rather than relying on the default.</td>
</tr>
<tr>
<td></td>
<td>option value.</td>
<td>See “Resolving Differences between Graphs Generated with Different</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technologies” on page 268.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPLOT lines drawn in reverse order on the client.</td>
<td>This change was made intentionally to maintain the integrity of plots</td>
</tr>
<tr>
<td></td>
<td></td>
<td>drawn with the AREAS= option.</td>
</tr>
<tr>
<td>In ActiveX, black-and-white image is not displayed</td>
<td></td>
<td>See SAS/GRAPH: Java Applets and ActiveX Control User's Guide.</td>
</tr>
<tr>
<td>Graph loses attributes after graph type is changed in the web browser.</td>
<td>Some attribute loss is inherent in graph type changes.</td>
<td>Select the Refresh button in the web browser to restore the original</td>
</tr>
<tr>
<td>Changes made through the Data Options dialog box cause the graph to</td>
<td>The graph discards subsetting information if you make changes through</td>
<td>Make any changes needed through the Data Options dialog box before</td>
</tr>
<tr>
<td>revert to its original view.</td>
<td>the Data Options dialog box.</td>
<td>subsetting the graph.</td>
</tr>
</tbody>
</table>
Checking Browser Permissions

Access permissions vary from browser to browser, but some form of access control is enforced in most browsers. To check your permissions, open the browser's Preferences or Internet Options window. Then look for the advanced options. Use your browser's Help system and contact your system support representative as needed to ensure that the browser permissions allow the following:

- Style sheets
- 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

A special character in your web presentation might not resolve in the browser. The character might need to be changed to a character entity in the source file or in the SAS program. A character entity is a standardized string of characters that represents a special character. The browser recognizes the string and replaces it with the special character when it is formatting the display. One common character entity is &gt; . This entity represents the greater-than symbol (>).

Lists of standard character entities are provided in HTML reference books and in HTML references on the Worldwide web.

Connecting to Web Servers That Require Authentication

If you are unable to run a Java applet or install the ActiveX control, then you might be trying to access a web server that requires authentication. To resolve this problem, access a different file on that server and enter your user ID and password. Redisplaying your web presentation should now enable you to access that web server.

Removing CLASSPATH Environment Variables

You might run across a case where, in the UNIX operating environment, the Java applet does not run. After you have verified that your Java archive is correctly specified, you should next remove any CLASSPATH environment variables that have been set. The Java archive files contain all the required classes to run the applets. Your CLASSPATH might 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 enable you to
set a CLASSPATH at start-up, by using a start-up 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 might need to define a special environment variable, SAS_ALT_DISPLAY, because
some server features require a valid X Windows System graphics display. This
environment variable is used to locate a graphic display when the value of the
environment variable commonly used by the X Window System, DISPLAY, has not been
set. The value of SAS_ALT_DISPLAY must refer to a display that is always available
during the operation of a SAS server. For example, if the server machine on which SAS
servers are running also runs an X server, then set the value of SAS_ALT_DISPLAY to
the name of the server machine. To set the SAS_ALT_DISPLAY environment variable,
edit the file /SASROOT/bin/sasenv and substitute your display name for value:0.0 in the
line,

SAS_ALT_DISPLAY=value:0.0

If an X server is not available on the server machine, an alternative is to use the X virtual
frame buffer (Xvfb) as supplied by the operating system vendor. Refer to your vendor-
supplied documentation for information about the use of Xvfb.

Correcting Text Fonts

Your presentation might display an incorrect text font on a given client computer. The
cause might be that the client computer maps a logical font name such as Courier to a
different physical font set. If the logical font is not mapped to any physical font, Java
uses a default font.

It is recommended that you specify system fonts whenever possible. See “Determining
What Fonts Are Available” on page 300 and “TrueType Fonts Supplied by SAS” on
page 300 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, sans-serif, 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. The font used is
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. Graphics output that is rendered with other devices
such as PNG, GIF, or SVG is rendered with SAS technology.
Because of technological differences between SAS, Java, and ActiveX, output generated with these different technologies might differ from each other. This is true even if the output is generated with the same SAS procedure code. The graphs might differ in appearance, in the default values used for certain options, or in the availability of certain features. See *SAS/GRAPH: Java Applets and ActiveX Control User’s Guide* for more information.

For example, differences might 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 1441.

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 4

Controlling the Appearance of Your Output

Chapter 20
Using ODS Styles, Device Parameters, and Options ............ 273

Chapter 21
Specifying Fonts in SAS/GRAPH Programs ..................... 299

Chapter 22
Using Colors in SAS/GRAPH Programs ......................... 313

Chapter 23
Adding Images to SAS/GRAPH Output .......................... 331
Chapter 20

Using ODS Styles, Device Parameters, and Options

Overview .......................................................... 274
Style Attributes versus Device Entry Parameters ..................... 274
About Style Templates ........................................... 275
  ODS Destinations and Default Styles .................................. 275
  Recommended Styles .................................................. 276
  Examples of Output Using Different Styles .......................... 277
Specifying a Style .................................................. 279
  Changing the Current Style By Using the STYLE= Option in ODS Destination Statements .................................. 279
  Changing the Default Style in the SAS Registry ...................... 279
Overriding Style Attributes with SAS/GRAPH Statement Options .... 280
Precedence of Appearance Option Specifications ..................... 281
Output from Different Devices and the GSTYLE/NOGSTYLE System Options 281
Additional Considerations with ODS Styles .......................... 283
Viewing the List of Styles Provided by SAS .......................... 284
  Using the TEMPLATE Procedure .................................... 284
  Using the Templates Window ....................................... 284
Modifying a Style .................................................. 284
  Using the TEMPLATE Procedure .................................... 284
  Example: Modifying a Style Element ................................ 285
  Ways to Modify Graph Fonts or Colors Specified by Styles ........ 285
  Modifying the GraphFonts and GraphColors Style Elements .......... 286
Graphical Style Element Reference for Device-Based Graphics ........ 287
  The GraphColors Style Element .................................... 287
  The GraphFonts Style Element ..................................... 288
  Font Specifications in the GraphFonts Style Element ............... 289
  Style Elements for Use with SAS/GRAPH Output .................. 289
Turning Off Styles .................................................. 298
Changing the Appearance of Output to Match That of Earlier SAS Releases 298
Related Topics ...................................................... 298
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 20.1  Controlling Graph Appearance

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Level of Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a different style template.</td>
<td>Specify a style template with the STYLE= option to change the appearance of the entire graph. Requires no further modification.</td>
<td>Low</td>
<td>“Changing the Current Style By Using the STYLE= Option in ODS Destination Statements” (p. 279)</td>
</tr>
<tr>
<td>Use appearance options.</td>
<td>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.</td>
<td>Medium</td>
<td>“Overriding Style Attributes with SAS/GRAPH Statement Options” (p. 280)</td>
</tr>
<tr>
<td>Modify individual style elements.</td>
<td>Specify or change style attributes in order to modify a style element. This requires the use of PROC TEMPLATE style statements.</td>
<td>High</td>
<td>“Modifying a Style” (p. 284)</td>
</tr>
</tbody>
</table>

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 274 and “Turning Off Styles” on page 298 for more information.

Style Attributes versus Device Entry Parameters

The default appearance of SAS/GRAPH output is determined by either style attributes or device entry parameters. Which one depends on the setting of the GSTYLE system option and on the device that is being used.
By default, the GSTYLE system option is in effect, and the appearance of all SAS/GRAPH output is determined by style attributes. If the NOGSTYLE system option is in effect, then the device entry parameters govern the appearance of SAS/GRAPH output for all devices except the 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 20.2  The GSTYLE System Option and Default Appearance

<table>
<thead>
<tr>
<th>Current Device</th>
<th>GSTYLE</th>
<th>NOGSTYLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java or ActiveX device</td>
<td>style</td>
<td>style</td>
</tr>
<tr>
<td>All other devices</td>
<td>style</td>
<td>device entry parameters</td>
</tr>
</tbody>
</table>

For information about device entries, see “What Is a SAS/GRAPH Device?” on page 80 and “Viewing and Modifying Device (DEV) Entries” on page 91. See also “Output from Different Devices and the GSTYLE/NOGSTYLE System Options” on page 281, “Changing the Appearance of Output to Match That of Earlier SAS Releases” on page 298, and “Turning Off Styles” on page 298.

About Style Templates

An ODS style is a collection of named style elements that provides specific visual attributes for your graphical and tabular SAS output. Each style element is a named collection of style attributes such as background color, text color, marker symbol, line style, font face, font size, as well as many others. Each graphical element of a plot, such as a marker, a bar, a line or a title, derives its visual attributes from a specific style element from the current style. Note that the style that a destination uses is applied to tabular output as well as graphical output.

ODS Destinations and Default Styles

Every ODS output destination, except the Document and Output destinations, is associated with a default style. These styles are customized for each destination. Therefore, your output might look different depending on which destination you use. If your program does not specify a style, SAS uses the styles listed in Table 20.3 on page 275.

Table 20.3  Default Style Templates

<table>
<thead>
<tr>
<th>ODS Destination</th>
<th>Default Style Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>LISTING</td>
<td>Listing</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>Not applicable</td>
</tr>
<tr>
<td>HTML</td>
<td>Default (Styles.Default)</td>
</tr>
</tbody>
</table>
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 279.

Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 49 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 277 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 20.4 on page 276 describes a subset of the styles provided by SAS that are particularly well-suited to displaying graphics.

<table>
<thead>
<tr>
<th>Desired Output</th>
<th>Recommended Styles</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Color</td>
<td>Default (Styles.Default)</td>
<td>Gray background, optimized for HTML output</td>
</tr>
<tr>
<td>Analysis</td>
<td>Yellow background</td>
<td></td>
</tr>
<tr>
<td>Statistical</td>
<td>White background, colored fills</td>
<td></td>
</tr>
<tr>
<td>Listing</td>
<td>White background, optimized for color format on white paper</td>
<td></td>
</tr>
<tr>
<td>Printer</td>
<td>White background; serif fonts; optimized for PS and PDF output</td>
<td></td>
</tr>
<tr>
<td>Rtf</td>
<td>Similar to Printer; optimized for RTF output</td>
<td></td>
</tr>
<tr>
<td>Black and White</td>
<td>MonochromePrinter</td>
<td>Black and white output; patterned fills; optimized for PCL output</td>
</tr>
<tr>
<td>Journal2</td>
<td>Interior filled areas have no color</td>
<td></td>
</tr>
</tbody>
</table>
Recommended Styles

<table>
<thead>
<tr>
<th>Desired Output</th>
<th>Recommended Styles</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gray Scale</td>
<td>Journal</td>
<td>Interior filled areas are gray scale</td>
</tr>
</tbody>
</table>

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 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 49 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 277 shows examples of graphs and tables using the Printer, Rtf, Analysis, and Journal styles.

Examples of Output Using Different Styles

Each of the following sets of output was created using a different style. Additional examples of output in Chapter 6, “Overview of Devices, ODS Destinations, and ODS Styles,” on page 49.

Figure 20.1  Output Using the Printer Style
Figure 20.2 Output Using the RTF Style

Figure 20.3 Output Using the Analysis Style
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 20.3 on page 275. To permanently change the default style associated with a destination, you can change the setting of Selected Style in the SAS registry.

CAUTION: The table in Figure 20.4 on page 279 was sent to the PDF destination.
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 in SAS Language Reference: Concepts.

Note: You may 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 in SAS Language Reference: Concepts.


Overriding Style Attributes with SAS/GRAPH Statement Options

By default, the attributes of various elements of the graph are derived from specific style elements (or from device entry parameters if the NOGSTYLE system option is in effect). Exceptions occur when the attributes are explicitly overridden with procedure or global statement options. For example, you can use the CTITLE= and CTEXT= options in the GOPTIONS global statement to change the color of the text in all of your graphs. You can use the SYMBOL statement to specify colors for markers. The settings remain in effect until you change them or end your SAS session. For information about GOPTIONS, see “GOPTIONS Statement” on page 375 and “Specifying Colors in a GOPTIONS Statement” on page 314. See the examples in Chapter 24, “SAS/GRAPH Statements,” on page 343.

Instead of specifying global options, which affect all of your SAS/GRAPH output, you can specify options on specific action statements that affect only the output produced by that statement. Values that you specify on procedure action statements override default style attributes (or device entry parameters) and global options. For an example, see “Example 5: Modifying the Appearance of Radar Charts” on page 1271.

The documentation for each option that overrides a style element includes the name of the style element and attribute. For example, the documentation for the “CAXIS=grid-color” on page 891 GCHART procedure includes the following style reference information:

\[ CAXIS= \]

\[ \text{Style reference } \text{Color attribute of the GraphAxisLines element} \]

You might want to change the color of the same graphical elements that are affected by the CAXIS= option by modifying a style. In this case, you need to modify the Color attribute of the GraphAxisLines element. See “Modifying a Style” on page 284 for more information.

Attributes that are used repeatedly might be best specified in an ODS style. However, if you have created a customized style, be aware that you might need to make this style available to anyone to whom you send your SAS code.

Attributes that are used only once or occasionally are best specified using SAS/GRAPH statements.
Precedence of Appearance Option Specifications

When you specify options that override style attributes or device parameters, the general order of precedence that SAS/GRAPH uses is as follows:

1. options in a SAS/GRAPH procedure action statement
2. options in AXIS, FOOTNOTE, LEGEND, NOTE, PATTERN, SYMBOL, or TITLE statements
3. graphics options in a GOPTIONS statement
   a. color options in the GOPTIONS statement that control specific graph elements such as the background color or title text color
   b. the color list specified with the COLORS= option in the GOPTIONS statement
4. attributes specified in the current style or, if the NOGSTYLE option is in effect, device parameters in a device entry for the current device
5. default hardware settings for a device

SAS/GRAPH uses the first specification that it finds in this list. Any exceptions to this rule are noted in the documentation for the specific option as described in “Overriding Style Attributes with SAS/GRAPH Statement Options” on page 280.

Output from Different Devices and the GSTYLE/NOGSTYLE System Options

The GSTYLE and NOGSTYLE system options might or might not have an effect on the output from different devices. The non-interactive devices, namely SVG, PNG, GIF, and TIFF all support ODS styles. However, the ACTXIMG and JAVA IMG devices are not affected by the GSTYLE|NOGSTYLE system option. An ODS style is applied to an ACTXIMG or JAVA IMG image regardless of the setting of this system option. Let us demonstrate the impact of the GSTYLE and NOGSTYLE system options on the device output. Here is an example that uses the GSTYLE system option with the default ODS destination and graphics output device.

```sas
options gstyle;
proc gchart data=sashelp.cars;
  vbar Make;
    where MPG_Highway >= 37;
  run;
quit;
```
The output is similar for the other destinations and devices.

Here is the previous example with the NOGSTYLE system option on:

```plaintext
options nogstyle;
proc gchart data=sashelp.cars;
  vbar make;
    where MPG_Highway >= 37;
run;
quit;
```
Figure 20.6  A Bar Chart Using the GIF Device with the Statistical Style and the NOGESTYLE System Option

Notice that the default style is overridden by the NOGESTYLE system option and that no style is applied to the graph. The NOGESTYLE system option is valid only for the SVG, PNG, GIF, and TIFF devices. For the ACTXIMG and JAVAIMG devices, the NOGESTYLE system option has no effect. In this example, if the DEVICE=JAVAIMG or DEVICE=ACTXIMG graphics option is used, the default style is applied even though the NOGESTYLE system option is on.

Additional Considerations with ODS Styles

- Some ODS styles, such as Dtree, Netdraw, and BlockPrint, include background images. Not all devices support these images. You can suppress the display of these images by specifying the NOIMAGEPRINT graphics option. See “Disabling and Enabling Image Output” on page 340.

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

- Not all the styles work with all the chart types. For example, the default colors for axis labels might blend into the background color of the style. Specify a different style.
Viewing the List of Styles Provided by SAS

You can view the styles that SAS provides using the TEMPLATE procedure or through the Templates window.

**Using the TEMPLATE Procedure**

To view the list of all styles available, submit the following code:

```
proc template;
   list styles;
run;
```

SAS writes the list of available styles in the Output window.

**Using the Templates Window**

To view the list of all styles available, follow these steps:

1. Open the Templates window. You can open the Templates window in two ways:
   - Enter the `odstemplates` command on the SAS command line.
   - In the Results window, select the Results folder. Right-click and select Templates to open the Templates window.

   The Templates window contains the item stores `Sasuser.Templat` and `Sashelp.Tmplmst`.

2. Double-click an item store, such as `Sashelp.Tmplmst`, to expand the list of directories where ODS templates are stored. The templates that SAS provides are in the item store `Sashelp.Tmplmst`.

3. Double-click Styles to view the list of styles defined in the selected item store.

4. Double-click the style that you want to view. For example, the HTMLBlue style is the default style for HTML output. Similarly, the Rtf style is the default style for RTF output.

To view the actual style template, double-click a style name. The style template is displayed in the Template Browser window.

Modifying a Style

**Using the TEMPLATE Procedure**

Within the TEMPLATE procedure, you can use the DEFINE STYLE statement to create a completely new style or you can start from an existing style. When you create styles from existing styles, you can modify the individual style elements.

Example: Modifying a Style Element

The style element GraphData1 is defined in the Default style as follows:

```sas
proc template;
  define style Styles.Default;
    ...more style elements...
  class GraphData1 /
    markersymbol = "circle"
    linestyle = 1
    contrastcolor = GraphColors('gcdata1')
    color = GraphColors('gdata1');
end;
run;
```

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:

```sas
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:

```sas
ods html style=MyStyleDefault;
```

Ways to Modify Graph Fonts or Colors Specified by Styles

There are different ways to change the fonts or colors used by a style. Which method you choose depends on how extensively you want to change the font or color specifications used in your output. You can do any of the following:

- Modify a specific style element that controls a specific graphical element. For example, the GraphValueText element specifies the font and color for tick mark values and legend value descriptions. You could change the font or color specified by the GraphValueText element for the Analysis style. Changes to specific style elements affect only the graphical elements that they control and affect them in only the styles where you change them. See “Style Elements for Use with SAS/GRAPH Output” on page 289 for information about the specific style elements that you can modify.

- Modify the font or color specifications in the GraphFonts or GraphColors style elements for a specific style. The settings specified in GraphFonts and GraphColors are referenced by specific style elements elsewhere in the style. Other style elements that reference the GraphFonts or GraphColors style elements use the modified settings. See “The GraphFonts Style Element” on page 288 and “The GraphColors
A single change in the specifications in the GraphFonts or GraphColors style elements can potentially change the appearance of several graphical elements. And it can affect output of any style that refers to GraphFonts or GraphColors.

• Modify the font settings for one or more subkeys in the SAS registry. Many styles refer to the font settings in the SAS registry to determine the fonts to use for various graphical elements. Modifying the SAS registry settings changes the fonts used for all styles that refer to the subkeys that you change. For more information, see “Changing SAS Registry Settings for ODS” in SAS Output Delivery System: User’s Guide. (Colors used by the styles supplied by the company are not controlled through the SAS registry.)

Modifying the GraphFonts and GraphColors Style Elements

The attributes in the GraphFonts and GraphColors style elements are used as the values for specific style elements elsewhere in the style. In other words, the GraphFonts and GraphColors elements are abstract elements. They are used to assign values to other elements.

For example, the GraphFonts element could be defined as 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 another font attribute. (For information about the syntax used in the GraphFonts style element, see “Font Specifications in the GraphFonts Style Element” on page 289.)

For example, the GraphValueText element specifies the font and color for tick mark values and legend value descriptions. Suppose the GraphValueText element is defined as follows:

```
class GraphValueText /
  font = GraphFonts('GraphValueFont')
  color = GraphColors('gtext');
```

The font and color for GraphValueText are specified by elements in the GraphFonts and GraphColors style elements.

```
GraphFonts('GraphValueFont')
```
tells SAS/GRAPH to use the font specified by the GraphValueFont attribute in the GraphFonts style element.

```
GraphColors('gtext')
```
tells SAS/GRAPH to use the color specified by the gtext attribute in the GraphColors style element.

To change the font and color for tick mark values and legend value descriptions, you could modify either of the following:

• the FONT= and COLOR= attributes in the GraphValueText element
• the GraphValueFont attribute in the GraphFonts style element and the gtext attribute in the GraphColors style element.

However, because elements in GraphFonts and GraphColors are referred to by other elements in the style, changing the values in GraphFonts and GraphColors result in more extensive changes. A targeted change involves modifying a specific style element such as GraphValueText directly. If you modify the GraphValueText element directly, your modifications affect only the items controlled by GraphValueText. If you modify the GraphValueFont or gtext attributes, then your modifications might affect other portions of the graph in addition to tick mark values and legend value descriptions. This list includes pie labels, regression equations, data point labels, bar labels, and graph titles.

The styles supplied with SAS/GRAPH are designed to provide a consistent visual appearance for all graphical elements in your output. Modifying attributes in the GraphFonts or GraphColors elements instead of modifying several specific style elements makes it easier to maintain the consistent appearance in your output.

The tables listed in “Graphical Style Element Reference for Device-Based Graphics” on page 287 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 20.5 on page 287 lists the style attributes that are defined in the GraphColors style element and the graphical elements that they affect by default.

#### Table 20.5  GraphColors Attributes for Device-Based Output

<table>
<thead>
<tr>
<th>GraphColors Attribute</th>
<th>Portion of Graph Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>gaxis</td>
<td>Axis lines and tick marks</td>
</tr>
<tr>
<td>gborderlines</td>
<td>Border around the graph wall, legend border, and borders to complete axis frame</td>
</tr>
<tr>
<td>gconnectLine</td>
<td>Line for connecting boxes</td>
</tr>
<tr>
<td>gfloor</td>
<td>Graph floor</td>
</tr>
<tr>
<td>ggrid</td>
<td>Grid lines</td>
</tr>
<tr>
<td>glabel</td>
<td>Axis labels and legend titles</td>
</tr>
<tr>
<td>glegend</td>
<td>Background of the legend</td>
</tr>
<tr>
<td>goutline</td>
<td>Outlines for data primitives such as bars, pie slices, and boxes</td>
</tr>
</tbody>
</table>
GraphColors Attribute | Portion of Graph Affected
--- | ---
gshadow | Drop shadows used with text
gtext** | 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

* Elements in the GraphColors style element that are not included in this table are used with template-based (ODS Graphics) output only.

** The gtext attribute does not affect text that is not rendered as part of the graph. See also “Controlling Titles and Footnotes” on page 101.

The GraphFonts Style Element

The GraphFonts style element specifies the fonts that are used for different categories of graphical elements. Table 20.6 on page 288 lists the style attributes that are defined in the GraphFonts style element and the graphical elements that they affect by default.

Table 20.6 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 \(\rightarrow\) FONTS in the SAS registry. For more information, see “Changing SAS Registry Settings for ODS” in SAS Output Delivery System: User’s Guide in SAS Output Delivery System: User’s Guide.

Arial
specifies the Arial font family. If SAS cannot find the first font listed, it tries to find the second font listed.

sans-serif
specifies the sans-serif font family. If SAS cannot find the specific fonts listed, then it looks for a font in the sans-serif font family.

10pt,bold
specifies the weight and style that should be used.

In this example, if the SAS registry entry for the MTsans-serif subkey specifies Albany AMT, then SAS/GRAPH first tries to use the Albany AMT 10 point bold font. If it cannot find this font, then it tries to use Arial 10 point bold, and so on.

Note: SAS might not be able to find a specific font unless it is registered with the FONTREG procedure. The fonts provided by SAS are already registered. If you want to add additional fonts, see SAS Language Reference: Concepts for information about registering TrueType fonts. See Base SAS Procedures Guide for information about the FONTREG procedure.

Style Elements for Use with SAS/GRAPH Output

The style elements listed in the following tables affect SAS/GRAPH output and can be used in styles. These tables list each style element, the portion of the graph that it affects or was created to use with, and its attribute values. Attribute values can be changed with PROC TEMPLATE, as described in “Using the TEMPLATE Procedure” on page 284 and “Example: Modifying a Style Element” on page 285. For complete documentation about the style attributes that can be specified in each style element, see “Style Attributes Overview” in SAS Output Delivery System: Procedures Guide.

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropShadowStyle</td>
<td>Used with text types</td>
<td>Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Graph</td>
<td>Graph size and outer border appearance</td>
<td>OutputWidth, OutputHeight, BorderColor, BorderWidth, CellPadding, CellSpacing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>X, Y, and Z-axis lines</td>
<td>Color, LineStyle, LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBackground</td>
<td>Background of the graph</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image, VerticalAlign, TextAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Border around graph wall, legend border, borders to complete axis frame</td>
<td>Color, LineThickness, LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphCharts</td>
<td>All charts within the graph</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image, VerticalAlign, TextAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphDataText</td>
<td>Text font and color for point and line labels</td>
<td>Font or font-attributes, Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphFloor</td>
<td>3-D floor</td>
<td>BackgroundColor, Transparency, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image, VerticalAlign, TextAlign</td>
</tr>
<tr>
<td>GraphFootnoteText</td>
<td>Text font and color for footnotes</td>
<td>Font or font-attributes*, Color</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color, LineStyle, LineThickness, Transparency, displayopts</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color, LineStyle, LineThickness, Transparency, displayopts</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Background color of the legend</td>
<td>Color, FrameBorder, Transparency</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Outline properties for fill areas such as bars, pie slices, and box plots.</td>
<td>Color, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphTitle1Text</td>
<td>Text font and color for the first title</td>
<td>Font or font-attributes*, Color</td>
</tr>
<tr>
<td>GraphTitleText</td>
<td>Text font and color for titles subsequent to the first title</td>
<td>Font or font-attributes*, Color</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Text font and color for axis tick values and legend values</td>
<td>Font or font-attributes*, Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>GraphWalls</td>
<td>Vertical walls bounded by axes</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image</td>
</tr>
</tbody>
</table>

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

Table 20.8  Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphCutLine</td>
<td>Cutline attributes for a dendogram</td>
<td>Color, LineStyle</td>
</tr>
<tr>
<td>GraphFinal</td>
<td>Final data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphInitial</td>
<td>Initial data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphOther</td>
<td>Other data for the graph. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>GraphOverflow</td>
<td>Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphUnderflow</td>
<td>Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>ThreeColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>ThreeColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>TwoColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor, EndColor</td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor, EndColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>GraphData1</td>
<td>Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData2</td>
<td>Primitives related to 2nd grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData3</td>
<td>Primitives related to 3rd grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>GraphData4</td>
<td>Primitives related to 4th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData5</td>
<td>Primitives related to 5th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData6</td>
<td>Primitives related to 6th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>GraphData7</td>
<td>Primitives related to 7th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData8</td>
<td>Primitives related to 8th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData9</td>
<td>Primitives related to 9th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphData10</td>
<td>Primitives related to 10th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData11</td>
<td>Primitives related to 11th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData12</td>
<td>Primitives related to 12th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
</tbody>
</table>
Turning Off Styles

To turn off styles, specify the SAS system option NOGSTYLE. To change the setting of the SAS system option from GSTYLE to NOGSTYLE, you can do either of the following:

- Submit the following OPTIONS statement:
  ```
  OPTIONS NOGSTYLE;
  ```
- Enter OPTIONS on the SAS command line, or select Tools ⇒ Options ⇒ System to open the SAS System Options window. Expand Graphics, and select Driver settings. Right-click Gstyle, select Modify value, and select 0=False as the new value.

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

Starting with SAS/GRAPH 9.2, the introduction of many new features significantly change the default appearance of your SAS/GRAPH output. To produce output that looks as if it were produced with previous versions of SAS/GRAPH, do the following:

- Specify the NOGSTYLE system option. This option turns off the use of ODS styles. See “Turning Off Styles” on page 298.
- Specify the FONTRENDERING=HOST_PIXELS system option. This option specifies whether devices that are based on the SASGDGIF, SASGDTIF, and SASGDIMG modules render fonts by using the operating system or by using the FreeType engine. This option applies to certain traditional SAS/GRAPH devices. (See “Device Categories and Modifying Default Output Attributes” on page 87.) For example, this option works for GIF, TIFFP, and JPEG devices, but it is not applicable to PNG, SVG, SASPRT* devices.
- Specify DEVICE=ZGIF or DEVICE=ZPNG in the GOPTIONS statement when you are sending output to the HTML destination.

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

See “Compatibility Device Drivers (Z Drivers)” on page 93 for more information.

Related Topics

- “Setting the Size of Your Graph” on page 113
- “Setting the Resolution of Your Graph” on page 114
- Chapter 13, “Using SVG Graphics,” on page 141
Chapter 21

Specifying Fonts in SAS/GRAPH Programs

Introduction: Specifying Fonts in SAS/GRAPH Programs

SAS/GRAPH provides access to a variety of fonts, or typefaces, to display text and special characters for your graphics output. SAS provides a number of TrueType fonts that you can use in your applications. By default, ODS styles use system fonts, including the TrueType fonts shipped with SAS, for the various titles, labels, and other text in SAS/GRAPH output. You can modify the default fonts by modifying the styles, by specifying graphics options, or by using font options in procedure action statements. You can specify special characters using character codes or hexadecimal codes.

For additional information about fonts, see “Using Fonts with Universal Printers and SAS/GRAPH Devices” in SAS Language Reference: Concepts.
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 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. For information about these fonts, see Appendix 2, “Using SAS/GRAPH Fonts,” on page 1493.

system fonts
fonts that can be used by any SAS procedure and by other software, such as Microsoft Word. These fonts include TrueType and Type1 fonts. Examples of system fonts include Albany AMT, Monotype Sorts, and Arial. Some system fonts, such as Helvetica, can also be present as device-resident fonts. System fonts are installed on the operating system, and then registered with SAS using the FONTREG procedure. System fonts generally provide the highest quality output. SAS/GRAPH installs and registers a set of TrueType fonts, and it is recommended that you use these fonts whenever possible. See “TrueType Fonts Supplied by SAS” on page 300 for more information.

device-resident fonts
fonts that are burned into the chips in a device's hardware. These fonts are specific to the device being used and are not portable between devices. Some device-resident fonts, such as Helvetica, can also be present as system fonts.

TrueType Fonts Supplied by SAS

When you install SAS, a number of TrueType fonts are available based on choices that were made during the installation. These fonts are referred to collectively as system fonts. You can use these fonts in your SAS programs by assigning the font name to font options, enclosed in quotation marks. For example, you can specify the following:

goptions ftext="Thorndale AMT";

For information about the TrueType fonts that are provided by SAS, see “TrueType Fonts Supplied by SAS” in SAS Language Reference: Concepts.

Determining What Fonts Are Available

The fonts listed in “TrueType Fonts Supplied by SAS” on page 300 are available on all systems where SAS is installed. It is recommended that you use these fonts when possible. Additional system fonts that are available to your application and the methods for determining those fonts depend on the following:

• the operating environment that you are working in
• the device or universal printer that you are using

For more information about determining what fonts are available, see *SAS 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 1493 for more information.

When you are deciding what font to use, consider all operating environments in which your SAS code will be run. For example, if you specify a font, such as Arial, that is available only on Windows systems, then your output will appear different on other systems. If you specify one of the fonts that are installed with SAS (see “TrueType Fonts Supplied by SAS” on page 300), 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 302. 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 286 and “The GraphFonts Style Element” on page 288 for more information about the font specifications in the styles. See “Style Attributes versus Device Entry Parameters” on page 274 for more information about the GSTYLE system option.

Table 21.1 on page 301 shows the fonts used when styles are not active (the NOGSTYLE system option is in effect).

<table>
<thead>
<tr>
<th>Device</th>
<th>TITLE1</th>
<th>All other text</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVG, PNG, GIF, TIFF, SASPRTx, PCL, PS, PDF</td>
<td>Swiss</td>
<td>Font specified by <code>&lt;MTmonospace&gt;</code> subkey in the SAS registry</td>
</tr>
</tbody>
</table>
### 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.
Specifying a Font

Specifying the FONT= Option

To specify a font in your SAS program, include a font name, enclosed in quotation marks, anywhere fonts are supported. For example, you can specify Thorndale AMT as the font for legend labels as follows:

```sas
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 $E=mc^2$.

```sas
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:

```sas
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 304 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:

```sas
title font="<MTsans-serif>" "My Title";
```

The font specified by the <MTsans-serif> registry subkey is used for the title.

The SAS registry is localized. If you specify a font using a registry subkey, the actual font that is used is the localized value specified in your registry.
Specifying International Characters (Unicode Encoding)

You can use the `/unicode` modifier with a hexadecimal code to print any character in a font that supports Unicode encoding. Most of the TrueType fonts listed in “TrueType Fonts Supplied by SAS” on page 300 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 304) to display the symbol for the Euro sign.

```title "Euro Symbol" font="Albany AMT/unicode" "20ac"x;```


The Java and ActiveX devices do not support the `/unicode` modifier.

Specifying Special Characters Using Character and Hexadecimal Codes

Some fonts contain characters that are not mapped to the keyboard and cannot be entered directly into a text string. To display these special characters, substitute a character code or a hexadecimal value in the text string. Hexadecimal values are recommended over character codes.

Note: You can also display special characters using Unicode code points. Unicode code points are specified with the `/unicode` font modifier followed by a hexadecimal value. See “Specifying International Characters (Unicode Encoding)” on page 304 for more information.

Character codes include the letters, numbers, punctuation marks, and symbols that are commonly found on a keyboard. They are usually associated with symbols or national alphabets. These codes enable you to display the character by specifying the font and using the keyboard character in the text string. For example, on Windows operating environments, to produce the character ζ, you can specify the Symbol MT font and the character code z in the text string.

```title font="Symbol MT" "z";```

Hexadecimal values are any two-digit hexadecimal numbers enclosed in quotation marks, followed by the letter x (for example, “3D”x). In double-byte character sets, the hexadecimal values contain four digits (for example, “4E60”x). Unicode characters also contain four digits.

You display characters with hexadecimal values the same way you display them with character codes. You specify the font that contains the special character and place the hexadecimal value in the text string. For example, this TITLE statement uses hexadecimal A9 to produce © in the Albany AMT font.

```title font="Albany AMT" "a9"x;```

Note: The character code or hexadecimal value associated with characters in a font might be dependent on the key map that is currently being used. Keymaps are not used if the /unicode modifier is specified, a symbol font is specified, or NOKEYMAP is specified in the font header. Contact Technical Support if you need assistance with creating or modifying key maps.
To determine the hexadecimal codes that you need to specify for a specific character, you can use the program shown in Example Code 21.1 on page 305. 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 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. */
```sas
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 21.1 on page 307 and Figure 21.2 on page 308 show the output of the program above for the TrueType fonts Symbol MT and Monotype Sorts.
```
Figure 21.1 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   | ;   | <   | =   | >   | ?   |     |
| ≈   | A   | B   | X   | Δ   | E   | Φ   | Γ   | H   | I   | θ   | Κ   | Λ   | Μ   | Ν   | O   |     |
| 50  | 51  | 52  | 53  | 54  | 55  | 56  | 57  | 58  | 59  | 5a  | 5b  | 5c  | 5d  | 5e  | 5f  |
| Π   | Θ   | Ρ   | Σ   | T   | Υ   | ζ   | Ω   | Ξ   | Ψ   | Z   | [   | ]   | \   | |    |     |
| 60  | 61  | 62  | 63  | 64  | 65  | 66  | 67  | 68  | 69  | 6a  | 6b  | 6c  | 6d  | 6e  | 6f  |
| α   | β   | γ   | δ   | ε   | ζ   | η   | θ   | ι   | ξ   | ψ   | ψ   | ρ   | θ   | υ   | υ   |     |
| 70  | 71  | 72  | 73  | 74  | 75  | 76  | 77  | 78  | 79  | 7a  | 7b  | 7c  | 7d  | 7e  | 7f  |
| η   | θ   | ρ   | σ   | τ   | υ   | ω   | ω   | ξ   | ζ   | ζ   | ψ   | ψ   | {   | |   |     |
| 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  |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 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  |
| 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  |

Specifying a Font 307
Different Ways to Specify Fonts

In general, there are four ways to specify fonts. The method that you choose depends on how extensively you want to change font specifications used in your program.

- Many procedures support font options that enable you to specify the fonts for certain graph elements. For example, with the GCHART procedure, you can use the FONT= suboption with the PLABEL= option to control the font for the pie slice labels. With the GKPI procedure, you can use the BFONT= option to specify the font for boundary labels. Changes specified using procedure options affect the output of the current invocation of the procedure only.

For information about the font options that are available for a specific procedure, see the documentation for the procedure.

- You can specify fonts in the AXIS, LEGEND, or SYMBOL global statements. Fonts specified with these statements affect the output of any procedure that references
those statements. See “Using SAS/GRAPH Global Statement Options to Specify Fonts” on page 309.

- 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 309.

- 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 310.

Using SAS/GRAPH Global Statement Options to Specify Fonts

Font options on SAS/GRAPH AXIS, LEGEND, and SYMBOL global statements enable you to specify fonts for the following:

- axis labels, reference line labels, and tick mark values
- legend labels and legend value descriptions
- contour line labels and plot point labels

For example, the following statement could be used to label contour lines:
symbol value="Deep" font="CUMBERLAND AMT/bold/italic";

See “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033 for an example that uses SYMBOL statements to label contour lines.

As with the options specified in the GOPTIONS statement, options specified with these global statements remain in effect until you change them or until you start a new SAS session.

For specific information about each of the global statements, see Chapter 24, “SAS/GRAPH Statements,” on page 343.

Using GOPTIONS to Specify Fonts

The GOPTIONS statement has several options that can be used to specify fonts for your graphs.

- FBY= sets the BY line font in your graphs.
- FTEXT= sets the font for all the text in your graphs.
- FTITLE= sets the font for the first title in your graphs.
For example, to specify Cumberland AMT for all of the text in your graphs, use

goptions ftext="Cumberland AMT";

Settings specified in the GOPTIONS statement remain in effect until you change them, until you specify `reset=all`, or until you close the SAS session.

If you want most or all of the text in your output to use a single font, specifying this font with the FTEXT= graphics option is frequently the best alternative. Using the FTEXT= option in the GOPTIONS statement instead of adding font specifications to several procedure action statements in addition to other global statements makes your code easier to maintain.

Note: The FBY= option is not supported by the Java or ActiveX devices. For specific information about the GOPTIONS statement, see “GOPTIONS Statement” on page 375. Information for specific graphics options is in Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

Note: When you are sending SAS/GRAPH output to the HTML or RTF destinations (MARKUP destinations), titles and footnotes can be rendered as part of your graph image or as part of the HTML or RTF files. Where your titles and footnotes are rendered determines the fonts that are used for them. See “Controlling Titles and Footnotes” on page 101 for information about the GTITLE and GFOOTNOTE destination options and the ODS USEGOPT statement.

**Changing the Font Specifications Used by a Style**

There are three ways to change the font specifications used by a style. Which method you choose depends on how extensively you want to change the fonts used in your output.

- You can modify the style element that controls a specific graph element such as graph titles or contour line labels.
- You can modify the abstract font specifications in the GraphFonts class. These font specifications can be referenced in multiple places in a style and affect several graph elements.
- You can modify the font settings in the SAS registry that the styles use to determine the default fonts. Changes to the SAS registry affect the fonts used by all styles that reference the SAS registry entry.

Modifying an existing style to use different fonts might be the best alternative if you need to create a style for all of your company's output. If you want to change only the fonts used in a few applications, then using the GOPTIONS statement is a better alternative.

For information about changing the font specifications used by the styles, see “Ways to Modify Graph Fonts or Colors Specified by Styles” on page 285.

**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 301
Chapter 22
Using Colors in SAS/GRAPH Programs

Colors, Styles, and Device Parameters .............................................. 313

Specifying Colors in SAS/GRAPH Programs ...................................... 314
  How to Specify Colors ................................................................. 314
  Specifying Colors in a GOPTIONS Statement .................................. 314
  Defining and Using a Color List .................................................... 315

Using Transparency ........................................................................... 316

Color-Naming Schemes ...................................................................... 317
  Overview of Color-Naming Schemes ............................................... 317
  RGB Color Codes ........................................................................... 318
  RGBA Color Codes ......................................................................... 319
  CMYK Color Codes ......................................................................... 320
  HLS Color Codes ............................................................................ 321
  HSV (or HSB) Color Codes ............................................................. 322
  Gray-Scale Color Codes ................................................................. 322
  Color Naming System (CNS) Values ............................................... 323
  SAS Color Names and RGB Values in the SAS Registry .................... 324
  Converting Color Values between Color-Naming Schemes ............... 325
  Converting Numeric Color Component Values to Color Names .......... 325
  Converting Color Values from Other Applications .......................... 325

Using the Color Utility Macros .......................................................... 327

Processing Limitations for Colors .................................................... 328
  Maximum Number of Colors Displayed on a Device ......................... 328
  Replaying Graphs on a Device That Displays Fewer Colors ............... 329

Colors, Styles, and Device Parameters

The appearance of SAS/GRAPH output is determined by the current ODS style by default. Styles set the overall appearance of your output, including the colors and fonts that are used. Some styles also add an image to the background of your graphs.

You can turn off the use of styles if needed. In this case, the default appearance of your output is controlled by device entry parameters. See “Style Attributes versus Device Entry Parameters” on page 274 and “Turning Off Styles” on page 298 for more information.
Whether you are using ODS styles or using device parameters, you can override the default colors and change them as described in “Specifying Colors in SAS/GRAPH Programs” on page 314. You can add images to your output as described in Chapter 23, “Adding Images to SAS/GRAPH Output,” on page 331.

For additional information about colors, see “Color Support for Universal Printers” in SAS Language Reference: Concepts.

Specifying Colors in SAS/GRAPH Programs

How to Specify Colors

SAS/GRAPH enables you to set colors in several ways. You can do any of the following:

- specify colors in procedure action statements for any procedures that create graphics output. For example, the CAXIS= option in the HBAR statement specifies a color for the response and midpoint axis lines. These options are described in the documentation for the individual procedures.

- specify colors in global statements that enhance procedure output: AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, and TITLE. You can also specify colors in the NOTE statement, which is a local statement, not a global statement. See Chapter 24, “SAS/GRAPH Statements,” on page 343.

- use options in the GOPTIONS statement that define colors for specific graphics elements. See “Specifying Colors in a GOPTIONS Statement” on page 314.

- define a color list with the GOPTIONS COLORS= option. See “COLORS” on page 532.

- specify a different style, modify an existing style, or create a custom style. See Chapter 20, “Using ODS Styles, Device Parameters, and Options,” on page 273 for more information about styles.

- modify the color list in the device entry for the device that you want to use. However, the colors listed in the device entry are not used unless styles are turned off. See “Using a Device's Color List” on page 315 and Chapter 37, “GDEVICE Procedure,” on page 1046 for more information.

See “Precedence of Appearance Option Specifications” on page 281 for information about which settings take precedence when colors are set in more than one way.

Specifying Colors in a GOPTIONS Statement

The GOPTIONS statement has several graphics options that set colors for specific graphical elements. These colors are used unless they are overridden by more specific options specified on other global statements or on procedure statements.

<table>
<thead>
<tr>
<th>Option</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBACK=</td>
<td>Specifies the color of background for graphics output</td>
</tr>
<tr>
<td>CBY=</td>
<td>Specifies the color of BY lines in graphics output</td>
</tr>
<tr>
<td>CPATTERN=</td>
<td>Specifies the color of fill patterns</td>
</tr>
</tbody>
</table>
**Option** | **Purpose**
---|---
**CSYMBOL=** | Specifies the color of SYMBOL definitions
**CTEXT=** | Specifies the color of all text and the border in graphics output
**CTITLE=** | Specifies the color of border, plus all titles, footnotes, and notes

You can also use the COLORS= option in a GOPTIONS statement to specify a list of colors rather than specific colors for individual graphical elements. Refer to Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515 for complete information about each of these graphics options.

### Defining and Using a Color List

#### Introduction to the Device Color Lists

Each device is associated with a list of colors that it can use. This list is defined in the device entry for the device. You can modify this list as needed. However, this device-specific list of colors is not used unless you turn off styles by specifying the NOGSTYLE system option. See “Using a Device's Color List” on page 315.

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 316 for more information.

The color selected from a color list varies depending on the procedure using the color and graphical element that it is drawing. Usually, the first color in the list is used. However, certain procedures can select other colors. For example, if the CAXIS= option is not specified in the GCONTOUR procedure's PLOT statement, the procedure selects the second color from the color list to draw the axes. See the documentation for individual procedures for more information.

#### Using a Device's Color List

If you specify the NOGSTYLE system option and you do not define a color list with the COLORS= graphics option, then SAS/GRAPH uses the color list from the current device. This color list is found in the device entry of the specified device. The color list might change if you select a different device during a SAS session.

When SAS/GRAPH assigns colors from the current device's color list, this assignment uses some of the colors that you can specify for a graph. The limit on the number of colors that can be used in your output is set by the current device. For example, the PNG device is a true color device and can use up to 16 million different colors. However, the GIF device is limited to 256 colors.

To view, create, or modify a device's color list, use the GDEVICE procedure. See Chapter 37, “GDEVICE Procedure,” on page 1046.

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 317. Each value specified in a color list must be one of the following:

- a valid color name, not to exceed 64 characters
- a valid color code, not exceed eight characters

Note: The COLORS= graphics option provides only a default lookup table. Anytime you explicitly select any other colors in your SAS/GRAPH program, those colors are used to draw the graphical elements for which you have specified them.

See “COLORS” on page 532 for more information.

Using Transparency

There are two primary uses for transparency.

- The first use is to produce output with a transparent background. For example, you might want to add your output to a web page that already has a specific background color or image. Output that has a transparent background allows the background on the web page to show through. The background has a low color opacity. The following devices produce a transparent background by default.

  - PNGT
  - SVGT

In addition, the following devices produce output with a transparent background when you also specify the TRANSPARENCY option in the GOPTION statement:

  - EMF and EMFDUAL
  - PNG and PNG300
  - SVG, SVGVIEW, and SVGZ
  - ACTIVEX and ACTXIMG when the output is used in a Microsoft PowerPoint presentation

- The second use is to produce output with semi-transparent colors. Semi-transparent colors are particularly useful when you want to add annotations to your output while allowing the underlying graph or map to show through the annotations. Semi-transparent colors, when annotated on top of each other, blend together (alpha blending). In the following images, the red, green, and blue circles are all semi-transparent (in this case, 50% opaque). The yellow background in the first image is totally opaque. You can specify semi-transparent colors with the RGBA color model. See “RGBA Color Codes” on page 319 for information.
The following devices use semi-transparent colors when the transparency is specified with RGBA color codes.

- GIF
- EMF and EMFDUAL
- PCL5c
- PNG, PNGT, and PNG300
- SVG, SVGT, SVGVIEW, and SVGZ
- TIFF

### Color-Naming Schemes

#### Overview of Color-Naming Schemes

The valid color-naming schemes in SAS are as follows:

- RGB (red green blue)
- RGBA (red green blue transparency)
- CMYK (cyan magenta yellow black)
- HLS (hue lightness saturation)
- HSV (hue saturation brightness), also called HSB
- Gray scale
- SAS color names (from the SAS Registry)
- SAS Color Naming System (CNS)

Table 22.1 on page 317 shows examples of each color-naming scheme.

<table>
<thead>
<tr>
<th>Color-Naming Scheme</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>COLORS=(CX98FB98 CXDDA0DD CXFFDAB9 CXDB7093 CXB0E0E6)</td>
</tr>
<tr>
<td>RGBA</td>
<td>COLORS=(A98FB9833 ADDA0DD66 AFFDAB999 ADB7093CC AB0E0E6FF)</td>
</tr>
</tbody>
</table>

*Note:* RGBA color values are not supported by Java devices. RGBA color values are supported by ActiveX devices when the output is used in Microsoft applications.
Color-Naming Scheme | Example
---|---
CMYK | COLORS=('FF00FF00' '00FFFF00' 'FFFFFF00')
HLS | COLORS=(H14055FF H0F060FF H0B485FF H07880FF)
HSV | COLORS=(V0F055FF V010FFFF V038FFFF V12C55E8)
Gray Scale | COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)
SAS Registry Colors | COLORS=(palegreen plum peachpuff palevioletred powderblue)
CNS Color Names | COLOR=('very light purplish blue')
| COLOR=(VeryLightPurplishBlue)
| COLOR=(Very_Light_Purplish_Blue)

You can also mix color-naming schemes in the same statement. For example:

goptions colors=(CXEE0044 "vivid blue" darkgreen);

Note: The hardware characteristics of your output device might cause some colors with different color definitions to appear the same. The same color is likely to appear differently on different devices and might not appear correctly on some devices. To determine whether your device supports a specific color-naming scheme, refer to your graphics device documentation.

Each of the color-naming schemes has its advantages and disadvantages based on how the output is used. For example, if you are creating a report that will be viewed only online, then specifying colors using the RGB naming scheme or the SAS color names defined in the registry might produce better results. If you are creating a report for publishing in printed form, you might want to use the CMYK color-naming scheme.

Note: Invalid color names, such as a misspelled color name, are mapped to gray, and a NOTE is issued to the SAS log. A valid color name that is not supported by the current device is mapped to the closest color that is supported by the device.


**RGB Color Codes**

The RGB color-naming scheme is usually used to define colors for a display screen. This color-naming scheme is based on the properties of light. An RGB color code defines a color by combining red, green, and blue colors in different ratios. All the colors combined together create white. The absence of all color creates black.

Color names are in the form CXrrggbb, where the following is true:

- CX indicates to SAS that this is an RGB color specification.
- rr is the red component.
- gg is the green component.
- bb is the blue component.
The components are hexadecimal numbers in the range 00–FF (0% to 100%). Each hexadecimal number indicates how much of the red, green, or blue is included in the color. Lower percentage values are darker and higher values are lighter. This scheme allows for up to 256 levels of each color component (more than 16 million different colors).

**Table 22.2 Examples of RGB Color Values**

<table>
<thead>
<tr>
<th>Color</th>
<th>RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>CF0FF00</td>
</tr>
<tr>
<td>Green</td>
<td>CX00FF00</td>
</tr>
<tr>
<td>Blue</td>
<td>CX0000FF</td>
</tr>
<tr>
<td>White</td>
<td>CFFFFFF</td>
</tr>
<tr>
<td>Black</td>
<td>CX000000</td>
</tr>
</tbody>
</table>

Any combination of the color components is valid. Some combinations match the colors produced by predefined SAS color names. See “Predefined Colors” on page 1511.

**RGBA Color Codes**

The RGBA color-naming scheme is the same as the RGB color scheme, except that RGBA includes an additional hexadecimal number for the alpha channel. This number specifies a percentage of opacity (indicating the transparency of the color). For a description of the primary uses of transparency, see “Using Transparency” on page 316.

Color names are in the form RGBA or A, where the following is true:

- RGBA or A indicates to SAS that this is an RGBA color specification.
- r is the red component.
- g is the green component.
- b is the blue component.
- a is the alpha channel specification, which represents a percentage of opacity.

The components are hexadecimal numbers in the range 00–FF. The alpha channel specification defines the intensity (opacity) of the color. The value 00 is transparent (0% opacity), and FF is opaque (100% opacity).

**Table 22.3 Examples of RGBA Color Values**

<table>
<thead>
<tr>
<th>Color</th>
<th>RGBA Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red, very transparent</td>
<td>RGBAFF000015</td>
</tr>
<tr>
<td>Red, 20% opaque</td>
<td>AFF00033</td>
</tr>
<tr>
<td>Green, 40% opaque</td>
<td>A0FF0066</td>
</tr>
</tbody>
</table>
Any combination of the color components is valid. Some combinations match the colors produced by predefined SAS color names. See “Predefined Colors” on page 1511.

**CMYK Color Codes**

The CMYK color-naming scheme is used in four-color printing. CMYK is based on the principles of objects reflecting light. Combining equal values of cyan, magenta, and yellow produces process black, which might not appear as pure black. The black component (K) of CMYK can be used to specify the level of blackness in the output. A lack of all colors produces white when the output is printed on white paper.

To specify the colors from a printer's Pantone Color Lookup Table, you can use the CMYK color-naming scheme. Specify colors in terms of their cyan, magenta, yellow, and black components. Color names are in the form CMYKcmmmyykk or Kcmmmyykk, where the following is true:

- CMYK or K is an optional prefix that indicates to SAS that this is a CMYK color specification.
- cc is the cyan component.
- mm is the magenta component.
- yy is the yellow component.
- kk is the black component.

The color-value components are hexadecimal numbers in the range 00–FF, where higher values are darker and lower values are brighter. This scheme allows for up to 256 levels of each color component. When the color value begins with a letter (A–F), you can omit the CMYK or K prefix. When the color value begins with a number (0–9), you must use the CMYK or K prefix, or enclose the value in quotation marks.

<table>
<thead>
<tr>
<th>Color</th>
<th>RGBA Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue, 60% opaque</td>
<td>A000FF99</td>
</tr>
<tr>
<td>Blue, 80% opaque</td>
<td>A000FFCC</td>
</tr>
<tr>
<td>Black, completely opaque</td>
<td>A00000FF</td>
</tr>
</tbody>
</table>

Any combination of the color components is valid. Some combinations match the colors produced by predefined SAS color names. See “Predefined Colors” on page 1511.

**CMYK Color Codes**

The CMYK color-naming scheme is used in four-color printing. CMYK is based on the principles of objects reflecting light. Combining equal values of cyan, magenta, and yellow produces process black, which might not appear as pure black. The black component (K) of CMYK can be used to specify the level of blackness in the output. A lack of all colors produces white when the output is printed on white paper.

To specify the colors from a printer's Pantone Color Lookup Table, you can use the CMYK color-naming scheme. Specify colors in terms of their cyan, magenta, yellow, and black components. Color names are in the form CMYKcmmmyykk or Kcmmmyykk, where the following is true:

- CMYK or K is an optional prefix that indicates to SAS that this is a CMYK color specification.
- cc is the cyan component.
- mm is the magenta component.
- yy is the yellow component.
- kk is the black component.

The color-value components are hexadecimal numbers in the range 00–FF, where higher values are darker and lower values are brighter. This scheme allows for up to 256 levels of each color component. When the color value begins with a letter (A–F), you can omit the CMYK or K prefix. When the color value begins with a number (0–9), you must use the CMYK or K prefix, or enclose the value in quotation marks.

**Table 22.4 Examples of CMYK Color Values**

<table>
<thead>
<tr>
<th>Color</th>
<th>CMYK Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>&quot;00FFFF00&quot;</td>
</tr>
<tr>
<td>Green</td>
<td>FF00FF00</td>
</tr>
<tr>
<td>Blue</td>
<td>CMYKFFFF0000</td>
</tr>
<tr>
<td>White</td>
<td>&quot;00000000&quot;</td>
</tr>
<tr>
<td>Process black (using cyan, magenta, and yellow ink)</td>
<td>KFFFFF00</td>
</tr>
<tr>
<td>Pure black (using only black ink)</td>
<td>&quot;000000FF&quot;</td>
</tr>
</tbody>
</table>
Note: You can specify a CMY value by specifying zero (00) for \( kk \), the color's black component.

CMYK color specifications are for output devices that support four colors. If you specify CMYK colors on an output device that supports three colors such as RGB, the CMYK colors are converted to colors in the three-color space. Because the four-color space supports many more colors than a three-color space, the CMYK colors might map to different colors in the three-color space. To preserve your CMYK colors in that case, specify a device that supports the CMYK color space. Some of the SAS universal printers support CMYK colors. For more information, see “Color Support for Universal Printers” in SAS Language Reference: Concepts.

**HLS Color Codes**

The HLS color scheme specifies colors in terms of hue, lightness, and saturation levels. It is based on the Tektronix Color Standard. HLS color names are of the form \( Hhhhlss \), where the following is true:

- \( H \) indicates to SAS that this is an HLS color specification.
- \( hhh \) is the hue component, which is expressed as an angle.
- \( ll \) is the lightness component.
- \( ss \) is the saturation component.

The components are hexadecimal numbers. The hue component is in the range 000–168 hexadecimal, which represents an angular value in the range 0–360 decimal. Hue starts with the primary color blue at 0 degrees, and then progresses to red at 120 degrees, to green at 240 degrees, and back to blue at 360 degrees. Both the lightness and saturation components are in the range 00–FF hexadecimal (0–255 decimal), which provides 256 levels that represent 0% to 100% for each component.

**Table 22.5 Examples of HLS Color Codes**

<table>
<thead>
<tr>
<th>Color</th>
<th>HLS Color Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>H07880FF</td>
</tr>
<tr>
<td>Green</td>
<td>H0F080FF</td>
</tr>
<tr>
<td>Blue</td>
<td>H00080FF</td>
</tr>
<tr>
<td>Light gray</td>
<td>H000BB00</td>
</tr>
<tr>
<td>White*</td>
<td>H( xxx )FF00, such as H000FF00</td>
</tr>
<tr>
<td>Black*</td>
<td>H( xxx )0000 such as H00000000</td>
</tr>
</tbody>
</table>

* When the saturation is set to 00, the color is a shade of gray that is determined by the lightness value. Therefore, white is defined as H\( xxx \)FF00 and black as H\( xxx \)0000, where \( xxx \) can be any hue.
HSV (or HSB) Color Codes

The HSV color-naming scheme specifies colors in terms of hue, saturation, and value (or brightness) components. HSV color names are of the form \texttt{Vhhhssvv}, where the following is true:

- \texttt{V} indicates to SAS that this is an HSV color specification.
- \texttt{hhh} is the hue component, which is expressed as an angle.
- \texttt{ss} is the saturation component.
- \texttt{vv} is the value or brightness component.

The components are hexadecimal numbers. The hue component is in the range \texttt{000–168} hexadecimal, which represents an angular value in the range \texttt{0–360} decimal. Hue starts with the primary color red at 0 degrees, and then progresses to green at 120 degrees, to blue at 240 degrees, and back to red at 360 degrees. Both the saturation and value (brightness) components are in the range \texttt{00–FF} hexadecimal (\texttt{0–255} decimal), which provides 256 levels for each component.

Table 22.6  Examples of HSV (or HSB) Color Codes

<table>
<thead>
<tr>
<th>Color</th>
<th>HSV Color Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>\texttt{V000FFFF}</td>
</tr>
<tr>
<td>Green</td>
<td>\texttt{V078FFFF}</td>
</tr>
<tr>
<td>Blue</td>
<td>\texttt{V0F0FFFF}</td>
</tr>
<tr>
<td>Light gray*</td>
<td>\texttt{Vxxx00BB} such as \texttt{V07900BB}</td>
</tr>
<tr>
<td>White*</td>
<td>\texttt{Vxxx00FF} such as \texttt{V07900FF}</td>
</tr>
<tr>
<td>Black*</td>
<td>\texttt{Vxxx0000} such as \texttt{V0790000}</td>
</tr>
</tbody>
</table>

* When the saturation is set to \texttt{00}, the color is a shade of gray. The value component determines the intensity of gray level. The \texttt{xxx} value can be any hue.

Gray-Scale Color Codes

Gray-scale colors are specified using the word GRAY and a lightness value in the form \texttt{GRAYhh}. The value \texttt{hh} is the lightness of the gray and is a hexadecimal number in the range \texttt{00–FF}, which provides 256 levels on the gray scale.

\textit{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 324.) Its value is \texttt{CX808080}. Invalid color specifications are mapped to GRAY.

The following figure shows the gray-scale color for each hexadecimal value.
Color Naming System (CNS) Values

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

Table 22.7  Color Naming System Values

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>Gray</td>
<td>Blue</td>
</tr>
<tr>
<td>Very Dark</td>
<td>Grayish</td>
<td>Purple</td>
</tr>
<tr>
<td>Dark</td>
<td>Moderate</td>
<td>Red</td>
</tr>
<tr>
<td>Medium</td>
<td>Strong</td>
<td>Orange/Brown</td>
</tr>
<tr>
<td>Light</td>
<td>Vivid</td>
<td>Yellow</td>
</tr>
<tr>
<td>Very Light</td>
<td></td>
<td>Green</td>
</tr>
<tr>
<td>White</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Follow these rules when you are determining the CNS color name:

- You should not use the lightness values Black and White with saturation or hue values.
- If you do not specify default values, medium is the default lightness value and vivid is the default saturation value.
• Gray is the only saturation value that can be used without a hue.
• Unless the color that you want is black, white, or some form of gray, you must specify at least one hue.

You can use one or two hue values in the CNS color name. When you use two hue values, the hues must be adjacent to each other in the following list: blue, purple, red, orange/brown, yellow, green, and then returning to blue. Two hue values result in a color that is a combination of both colors. Use the suffix “ish” to reduce the effect of a hue when two hues are combined. Reddish purple is less red than red purple. The color specified with the “ish” suffix must precede the color without the “ish” suffix.

You can write color names in the following ways:
• without space separators between words
• with an underscore to separate words
• with a space to separate words, enclosed in quotation marks

For example, the following color specifications are valid:
• verylightmoderatepurplishblue
• very_light_moderate_purplish_blue
• "very light moderate purplish blue"

Note: If a CNS color name is also a color name in the SAS Registry, the SAS Registry color value takes precedence. Some CNS color names and color names in the SAS Registry have different color values. Dark blue is an example. To use a CNS color value when the color name is also in the SAS Registry, do one of the following:
• Include a space to separate the words and enclose the entire color name in quotation marks. Here is an example:
  color="dark blue"
• Include an underscore to separate the words. Here is an example:
  color=dark_blue

**SAS Color Names and RGB Values in the SAS Registry**

The SAS Registry provides a set of color names and RGB values that you can use to specify colors. These color names and RGB values are common to most web browsers. You can specify the name itself or the RGB value associated with that color name. To view the color names as associated RGB values that are defined in the registry, submit the following code:

```sas
proc registry list
  startat="COLORNAMES";
run;
```

SAS prints the output in the SAS log. Here is a partial listing.

```
NOTE: Contents of SASHELP REGISTRY starting at subkey [COLORNAMES]
[ COLORNAMES]
  Active="HTML"
[ HTML]
  AliceBlue=hex: F0,F8,FF
  AntiqueWhite=hex: FA,EB,D7
  Aqua=hex: 00,FF,FF
  Aquamarine=hex: 7F,FD,D4
```
For a list of the color names that are defined in the SAS Registry, see “Predefined Colors” on page 1511.

You can also create your own color values by adding them to the SAS registry. For information about viewing and modifying the list of color names, see *SAS Language Reference: Concepts*.

---

**Converting Color Values between Color-Naming Schemes**

The SAS/GRAPH software provides several macros that enable you to perform selected conversions between color-naming schemes. The following table shows the conversions that are possible using these macros.

<table>
<thead>
<tr>
<th>From Color Value</th>
<th>To Color Value</th>
<th>Conversion Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMY</td>
<td>RGB</td>
<td>%CMY</td>
</tr>
<tr>
<td>CNS</td>
<td>HLS</td>
<td>%CNS</td>
</tr>
<tr>
<td>RGB</td>
<td>HLS</td>
<td>%RGB2HLS</td>
</tr>
<tr>
<td>HLS</td>
<td>RGB</td>
<td>%HLS2RGB</td>
</tr>
</tbody>
</table>

For information about these macros, see Appendix 5, “Color Utility Macro Dictionary,” on page 1525.

---

**Converting Numeric Color Component Values to Color Names**

The SAS/GRAPH software provides macros that enable you to perform selected conversions between color-naming schemes. The following table shows the macros that you can use for this purpose.

<table>
<thead>
<tr>
<th>From Numeric Color Component Values</th>
<th>To Color Name</th>
<th>Conversion Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>cyan, magenta, yellow, black</td>
<td>CMYK</td>
<td>%CMYK</td>
</tr>
<tr>
<td>hue, lightness, saturation</td>
<td>HLS</td>
<td>%HLS</td>
</tr>
<tr>
<td>hue, saturation, value</td>
<td>HSV</td>
<td>%HSV</td>
</tr>
<tr>
<td>red, green, blue</td>
<td>RGB</td>
<td>%RGB</td>
</tr>
</tbody>
</table>

---

**Converting Color Values from Other Applications**

Many software programs enable you to change or customize various colors. A dialog box typically provides a means of selecting a different color or modifying the attributes...
of an existing color. In the SAS ODS Graphics Editor, for example, the More Colors dialog box shown in the following figure serves this purpose.

*Figure 22.1  More Colors Dialog Box*

![More Colors Dialog Box](image)

*Figure 22.1 on page 326* shows the HSB and RGB numeric color component values for the currently selected color. It also shows the RGB values in hexadecimal. To use the RGB specification for this color in a SAS program, add the CX prefix to the Hex value E7B3B4. If the application provides only the numeric component values, you must convert the decimal component values to hexadecimal. In *Figure 22.1 on page 326*, the H value is in the range 0–360 degrees, and the S and B values are each in the range 0–100 percent. The R, G, and B values are in the range 0–255 each. You can convert the component values manually. (See “Understanding Hexadecimal Values” on page 1547.) Be aware that you must first convert the S and B values from percentages to 255-based values by rounding the result of the following computation to the nearest integer:

\[
\left(\frac{\text{Value}}{100}\right) \times 255
\]

You can also use the SAS/GRAPH color utility macros to convert the values. The %HSV color utility macro converts HSV (HSB) numeric color component values to an HSV color name. Likewise, the %RGB macro converts RGB numeric color component values to an RGB color name. The following example shows how to use the %HSV color utility macro to convert the H, S, and B color component values in *Figure 22.1 on page 326* to an HSV color name.

```sas
%COLORMAC;
data _null_;
  put "*HSV(357,22,90)";
run;
```

*Note:* The %COLORMAC macro compiles all of the SAS/GRAPH color utility macros. You need to run it only once during a SAS session.

Because the %HSV macro accepts values in the range 0–100 as a percentage of 255 for saturation and value, use the S and B values as shown. The result is V16538E6.
The following example shows how to use the %RGB color utility macro to convert the R, G, and B numeric color component values in Figure 22.1 on page 326 to an RGB color name.

```sas
/* Compute the RGB percentages */
data _null_;    
r = 231;    
g = 179;    
b = 180;    
call symputx("r", round((r/255)*100));    
call symputx("g", round((g/255)*100));    
call symputx("b", round((b/255)*100));    
run;

/* Convert to RGB color name */
%COLORMAC;
data _null_;    
put "%RGB(&r,&g,&b)";
run;
```

Because the %RGB color utility macro accepts integer values in the range 0–100 as a percentage of 255, you must convert the 255-based values to integer percentages in order to use them in the %RGB macro call. The result is CXE8B3B5. The result from the %RGB macro is not exact due to rounding. If you want more exact results, use the following program.

```sas
data _null_;    
r = 231;    
g = 179;    
b = 180;    
rgb="CX" || put(r,hex2.) || put(g,hex2.) || put(b,hex2.);
put rgb;
run;
```

The result is CXE7B3B4.

To convert the RGB color name to an HLS color name, use the %RGB2HLS macro as shown in the following program.

```sas%
COLORMAC;
data _null_;    
put "%RGB2HLS(CXE7B3B4)";
run;
```

The result is H077CD84.

For more information about the color utility macros, see Appendix 5, “Color Utility Macro Dictionary,” on page 1525.

---

**Using the Color Utility Macros**

The color utility macros enable you to define colors for a specific color-naming scheme and convert color values between color-naming schemes. These macros are useful when specifying colors in your SAS/GRAPH programs. Table 22.8 on page 328 provides a summary of the color utility macros that are available. A complete description is in Appendix 5, “Color Utility Macro Dictionary,” on page 1525.
### Table 22.8 Color Utility Macros

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%COLORMAC</td>
<td>Creates the color utility macros in the current SAS session. You must run this macro in your current SAS session before you can use any of the macros that are listed in this table.</td>
</tr>
<tr>
<td>%HELPCLR</td>
<td>Writes Help information for the color utility macros to the SAS log.</td>
</tr>
<tr>
<td>%CMY</td>
<td>Creates an RGB color name from numeric CMY component values.</td>
</tr>
<tr>
<td>%CMYK</td>
<td>Creates a CMYK color name from numeric CMYK component values.</td>
</tr>
<tr>
<td>%CNS</td>
<td>Creates a SAS/GRAPH HLS color name from a color-naming scheme (CNS) color name.</td>
</tr>
<tr>
<td>%HLS</td>
<td>Creates an HLS color name from numeric HLS component values.</td>
</tr>
<tr>
<td>%HSV</td>
<td>Creates an HLS color name from numeric HSV component values.</td>
</tr>
<tr>
<td>%RGB</td>
<td>Creates an RGB color name from numeric RGB component values.</td>
</tr>
<tr>
<td>%HLS2RGB</td>
<td>Creates an RGB color name from an HLS color name.</td>
</tr>
<tr>
<td>%RGB2HLS</td>
<td>Creates an HLS color name from an RGB color name.</td>
</tr>
</tbody>
</table>

### 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 that it can display simultaneously. MAXCOLORS is the number of foreground colors plus the background color. If you use more than the number of colors set by the MAXCOLORS device parameter, the excess colors are remapped. This can occur when a photograph is used as the background image for a graph or when gradient shading is used in a 3-D graph. In that case, use the MAXCOLOR device parameter to increase the maximum number of colors. See “MAXCOLORS” on page 582.

**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 parameter.
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 531) to control some of the remapping.

When you replay graphs on devices that display fewer colors than are in the graph, two situations can cause problems:

- Colors are specified that the device does not support.
- More colors are specified than the device can display at one time.

If you specify colors on a device that does not support the colors requested, the colors are remapped to gray. A note is issued to the SAS log telling you when a color is mapped gray.

The number of colors that your device can display affects the actual colors displayed. If your graphics output device can create a maximum of 64 distinct colors, and your graph contains 256 colors, then the 65th through the 256th color specifications are remapped to the colors specified in the current style. If the NOGSTYLE system option is in effect, the colors are remapped to the device's available colors and might not be displayed as the color that you specify.

You can use the TARGETDEVICE= graphics option to preview how a graph is going to look on a different device. Set the device entry name of the device driver to this graphics option. The graph is displayed as close as possible to the display when the other device is used.

*Note:* When you use the TARGETDEVICE= graphics option, SAS/GRAPH uses the color list of the target device as the default color list; any color that you explicitly use is displayed when you preview the graph, although the color might be mapped by the target device. Refer to “TARGETDEVICE” on page 616 for complete information about the TARGETDEVICE= graphics option.
Chapter 23
Adding Images to SAS/GRAPH Output

Introduction

By default, the appearance of SAS/GRAPH output is determined by the current ODS style. Some styles add an image to your graph, but most do not. You can use SAS/GRAPH statements and options to add images to your graphs as needed. You can place an image in the background area of a graph, in the backplane of graphs that support frames, on the bars of two-dimensional bar charts, or on the bars of three-dimensional bar charts when using the ACTIVEX device or the ACTXIMG device. You can also apply images at specified graph-coordinate positions using the Annotate facility. The images that you add to your graphs can be SAS files or external files, in a range of image formats.

Image File Types Supported by SAS/GRAPH

For displaying images in your graphs, SAS/GRAPH supports the image file types shown in the following table.
### Table 23.1  Image File Types Supported by SAS/GRAPH

<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP (Microsoft Windows Device Independent Bitmap)</td>
<td>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.</td>
</tr>
<tr>
<td>DIB (Microsoft Windows Device Independent Bitmap)</td>
<td>See the description of BMP.</td>
</tr>
<tr>
<td>GIF (Graphics Interchange Format)</td>
<td>Supports only color-mapped images. GIF is owned by CompuServe, Inc.</td>
</tr>
<tr>
<td>JPEG (Joint Photographic Experts Group)</td>
<td>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.</td>
</tr>
<tr>
<td>PBM (Portable Bitmap Utilities)</td>
<td>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.</td>
</tr>
<tr>
<td>PCD (Photo CD)</td>
<td>Kodak Photo CD format, which supports multiple image resolutions.</td>
</tr>
<tr>
<td>PCX (PC Paintbrush)</td>
<td>Supports bitmap, color-mapped, and true color images. PCX and PC Paintbrush are owned by Zsoft Corporation.</td>
</tr>
<tr>
<td>PNG (Portable Network Graphic)</td>
<td>Supports truecolor, gray-scale, and 8-bit images.</td>
</tr>
<tr>
<td>SVG (Scalable Vector Graphics)</td>
<td>Is an XML language for describing two-dimensional vector graphics. SVG supports true color images and transparency.</td>
</tr>
<tr>
<td>TGA (Targa)</td>
<td>Supports true color images. Targa is owned by Truevision, Inc.</td>
</tr>
<tr>
<td>TIFF (Tagged Image File Format)</td>
<td>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.</td>
</tr>
<tr>
<td>XBM (X Window Bitmaps)</td>
<td>Supports bitmap images only. XBM is owned by MIT X Consortium.</td>
</tr>
<tr>
<td>XWD (X Window Dump)</td>
<td>Supports all X visual types (bitmap, color-mapped, and true color.) XWD is owned by MIT X Consortium.</td>
</tr>
</tbody>
</table>
To place an image on the graph background, use the IBACK= option in a GOPTIONS statement. Specify either the path to the image file in quotation marks or a fileref that has been defined to point to the image file as follows:

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

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

```
goptions reset=all
   htitle=1.25
   colors=(cx7c95ca cxde7d6f  cx66ada0
cxb689cd cxa9865b cxbabc5c)
   iback="external-image-file";
title "Projected Automobile Sales";
data sales;
   input Month Amount;
   informat month monyy. ;
datalines;
   jan08  200
   feb08  145
   mar08  220
   apr08  180
   may08  155
   jun08  250
;
proc sort;
   by month;
proc gchart;
   format month monname8. ;
   pie month / discrete freq=amount value=inside
          noheading coutline=black;
run;
quit;
```

Because the default value for the IMAGESTYLE= graphics option is TILE, the image is copied as many times as needed to fill the background area.
You can specify IMAGESTYLE=FIT in the GOPTIONS statement to stretch the image so that a single image fits within the entire background area.

Displaying an Image in Graph Frame

Procedure action statements that support the IFRAME= support frames, which are the backplanes behind the graphs. The backplane is the area within the graph axes. To place an image on the backplane of a graph, specify the IFRAME= option in the procedure action statement that generates the graph. On the IFRAME= option, specify either the
path to the image file in quotation marks or a fileref that has been defined to point to the image file:

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

**Note:** If you specify the NOFRAME option in the procedure action statement or if you specify STYLE=0 in the AXIS statement, the IFRAME= option is ignored.

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

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

Because the default value for the GPLOT procedure’s IMAGESTYLE= option is TILE, the image is copied as many times as needed to fill the frame area.
You can specify IMAGESTYLE=FIT in the GOPTIONS statement to stretch the image so that a single image fits within the entire frame area.

![Projected Automobile Sales](image)

### Displaying Images on Data Elements

You can place images on the bars in two-dimensional bar charts generated by the GCHART HBAR or VBAR statements. You can also place images on the bars of three-dimensional bar charts if you are using the ACTIVEX device or the ACTXIMG device.

In the IMAGE= option of the PATTERN statement, specify either the path to the image file in quotation marks or a fileref that has been defined to point to the image file.

```sas
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.

```sas
pattern image="external-image-file"
imagestyle=fit;
```

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

**Note:** Images are supported on bar charts generated by the HBAR and VBAR statements. If an image is specified in a PATTERN statement that is used with another type of chart, then the PATTERN statement is ignored and default pattern rotation is affected. If you submit a PIE statement when an image has been specified in the PATTERN= option, the default fill pattern is used for the pie slices. Each pie slice displays the same fill pattern.

The following example places an image on the bars of a vertical bar chart:

```sas
goptions reset=all htitle=1.25 colors=(yellow cxd7d6f);
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
coutine=black;
run;
quit;
The image is tiled to fill each bar.

**Projected Automobile Sales**

![Projected Automobile Sales Chart]

If the PATTERN IMAGESTYLE=FIT option is used, the image is stretched to fill each bar.

`pattern=fileref | "external-image-file" imagestyle=fit;`
Displaying Images Using Annotate

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

- Specify the image file in quotation marks on the IMGPATH variable.
- Set the image coordinates with the X and Y variables.
- Specify the IMAGE function.

One corner of the image is located by the current X and Y position. The opposite corner is located by the X and Y variables associated with the IMGPATH variable. Here is an example that uses the Annotate facility to place the SAS logo on a map of the United States. You must specify the path to the image file before running the code.

*Note:* The following program uses the GMAP procedure, which is described in *SAS/GRAPH and Base SAS: Mapping Reference*.

```sas
options reset=all border htitle=1.75cells hsize=5.5in vsize=4.2in;
data my_anno;
length function $8;
xsys="3"; ysys="3"; when="a";
function="move"; x=55; y=5; output;
function="image"; style="fit"; imgpath="external-image-file";
x=x+25; y=y+10; output;
run;
title1 "GMAP with Annotated Image";
proc gmap data=mapsgfk.us map=mapsgfk.us anno=my_anno;
id state;
choro state/levels=1
nolegend
statistic=freq;
```

![Projected Automobile Sales](image-url)
run;
quit;

The **style="fit"** variable on the IMAGE function stretches the image as needed to fill the area.

To tile the image to fill the area where the logo is positioned, set the **STYLE** variable equal to **"tile"**. Starting with SAS 9.4M5, you can specify **STYLE="single"** to center a single instance of the image on the specified coordinates.

---

**Displaying Pop-Up Images Using the SVG Graphics Devices**

When you use an SVG graphics device, the **HTML=** option enables you to display an image in a specific area of a graph when the mouse pointer is positioned on a linked shape. The pop-up image is displayed only as long as the mouse pointer is positioned on the linked shape. The following figure shows an example of a drill-down bar chart that displays a pop-up preview image of the Central Region Sales drill-down chart when the mouse pointer is positioned on the Central Region bar.
The pop-up images appear in the top right corner of the graphics output area. A pop-up image is displayed only as long as the mouse pointer is positioned on a bar. It disappears when the mouse pointer is moved off of the bar. The pop-up image can be placed anywhere in the graphics output area and can be sized as needed.

For more information about creating pop-up images using the SVG graphics devices, see “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198.

Disabling and Enabling Image Output

The NOIMAGEPRINT graphics option disables image output without removing code from your SAS/GRAPH program. Also, some ODS styles (such as Dtree, Netdraw, and BlockPrint) include background images, but not all devices support these images. You can suppress the images with the NOIMAGEPRINT option.

goptions noimageprint;

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

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

goptions imageprint;
Part 5

Statements and Options Used by SAS/GRAPH

Chapter 24
SAS/GRAPH Statements ........................................ 343

Chapter 25
Graphics Options and Device Parameters Dictionary .......... 515

Chapter 26
SAS System Options Used by SAS/GRAPH .................... 629
Overview of Global Statements

SAS/GRAPH programs can use some of the SAS language statements that you typically use with the Base SAS procedures or with the DATA step, such as LABEL, WHERE, and FORMAT. These statements are described in the SAS DATA Step Statements: Reference.

In addition, SAS/GRAPH has its own set of statements that affect only graphics output generated by the SAS/GRAPH procedures and the SAS/GRAPH Annotate facility. Most of these statements are global statements. That is, they can be specified anywhere in your program and remain in effect until explicitly changed, overridden, or canceled. These are the SAS/GRAPH global statements:
AXIS
modifies the appearance, position, and range of values of axes in charts and plots.

FOOTNOTE
adds footnotes to graphics output. This statement is like the TITLE statement and is described in that section.

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

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

NOTE
adds text to the graphics output. This statement is an exception because it is not global but local, meaning that it must be submitted within a procedure. Otherwise, the NOTE statement is like the TITLE statement and is described in that section. When including text in a NOTE in output from the GMAP procedure, it is displayed behind the polygons of a map.

PATTERN
controls the color and fill of patterns assigned to areas in charts, maps, and plots.

SYMBOL
specifies the shape and color of plot symbols as well the interpolation method for plot data. It also controls the appearance of lines in contour plots.

TITLE
adds titles to graphics output. The section describing the TITLE statement includes the FOOTNOTE and NOTE statements.

The above statements are described in this chapter, as well as the following two Base language statements that have a special effect when used with SAS/GRAPH procedures:

BY
processes data according to the values of a classification (BY) variable and produces a separate graph for each BY-group value. This statement is not a global statement. It must be specified within a DATA step or a PROC step.

ODS HTML
generates one or more files written in Hypertext Markup Language (HTML). If you use it with SAS/GRAPH procedures or Base SAS mapping procedures, you can specify one of the device drivers, such as GIF, PNG, or SVG. With the GIF device driver, the graphics output is stored in GIF files. The HTML files that are generated reference the graphics output. When viewed with a web browser, the HTML files can display graphics and non-graphics output together on the same web page.

For more information about the BY, LABEL, and WHERE statements in Base SAS software, see SAS DATA Step Statements: Reference. For more information about the OPTIONS statement, see SAS Global Statements: Reference.

---

Specifying Units of Measurement

When the syntax of an option includes units, use one of these:
CELLS
  character cells

CM
  centimeters

IN
  inches

PCT
  percentage of the graphics output area

PT
  points

Note: Java applets does not support CM, IN, or PT.

If you omit units, a unit specification is derived in this order:
1. The GUNIT= option in a GOPTIONS statement.
2. The default unit, CELLS.

Dictionary

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

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

**Type:** Global

**Restrictions:** For the G3D procedure, the AXIS statement is supported by the JAVA and ActiveX devices only.
For the GCHART procedure, the AXIS statement and its options are ignored when used in conjunction with the BLOCK statement.

**Video:** Watch the video: “Control the Graph Axis Using the Axis Statement”.

**Syntax**

```
AXIS<1 ...99> <option(s)>;
```

**Summary of Optional Arguments**

**Appearance options**

- **COLOR=axis-color**
  specifies the color for all axis components (the axis line, all tick marks, the frame outline, and all text) unless you include a more explicit AXIS statement color specification.
- **LENGTH=axis length <units >**
  specifies the length of the axis in number of units.
- **NOBRACKETS**
suppresses the printing of group brackets drawn around the values on the group axis in a bar chart.

NOPLANE
removes either the horizontal or vertical three-dimensional axis plane in bar charts produced by the HBAR3D and VBAR3D statements.

OFFSET=(<n1><,n2>)<units > | (<n1><units >>,<n2><units >>)
specifies the distance from the first and last major tick marks or bars to the ends of the axis line.

ORIGIN=(<x><,y>)<units > | (<x><units >>,<y><units >>)
specifies the x coordinate and the y coordinate of the origin of the axis.

STAGGER
offsets the axis values on a horizontal axis.

STYLE=linetype
specifies a line type for the axis line.

WIDTH=thickness-factor
specifies the thickness of the axis line.

Axis scale options

INTERVAL=EVEN | UNEVEN | PARTIAL
The INTERVAL option affects the LOGBASE option in the AXIS statement.

LOGBASE=base | E | PI
scales the axis values logarithmically according to the value specified.

LOGSTYLE=EXPAND | POWER
specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power.

ORDER=(value-list)
specifies the order in which data values appear on the axis.

Text options

LABEL=(text-argument(s)) | NONE
modifies an axis label.

REFLABEL=(text-argument(s)) | NONE
creates and defines the appearance of a reference-line label.

SPLIT="split-char(s)"
specifies one or more characters that the AXIS statement uses to break axis values into multiple lines.

VALUE=(text-argument(s)) | NONE
modifies the major tick mark values.

Tick mark options

MAJOR=(tick-mark-suboption(s)) | NONE
modifies the major tick marks.

MINOR=(tick-mark-suboption(s)) | NONE
modifies the minor tick marks that appear between major tick marks.

Optional Arguments

COLOR=axis-color
specifies the color for all axis components (the axis line, all tick marks, the frame outline, and all text) unless you include a more explicit AXIS statement color specification.
The following table lists the SAS/GRAPH statement options that can be used to override the COLOR= specification. The table also lists the name of the style reference associated with each of the options.

Table 24.1 SAS/GRAPH Statement Options per Axis Component

<table>
<thead>
<tr>
<th>Option</th>
<th>Graph Element</th>
<th>Style Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIS statement:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LABEL=</td>
<td>Axis label</td>
<td>GraphLabelText</td>
</tr>
<tr>
<td>(COLOR=color)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFLABEL=</td>
<td>Reference-line labels</td>
<td></td>
</tr>
<tr>
<td>(COLOR=color)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALUE=</td>
<td>Major tick mark values</td>
<td>GraphValueText</td>
</tr>
<tr>
<td>(COLOR=color)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>calling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>procedure:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTEXT=</td>
<td>All axis text</td>
<td>GraphLabelText</td>
</tr>
<tr>
<td>(AXIS label and</td>
<td>(AXIS label and major tick mark</td>
<td></td>
</tr>
<tr>
<td>major tick mark</td>
<td>value descriptions)</td>
<td></td>
</tr>
<tr>
<td>CAXIS=</td>
<td>Axis line, axis frame outline,</td>
<td>GraphAxisLines</td>
</tr>
<tr>
<td>outline, and</td>
<td>major minor tick marks</td>
<td></td>
</tr>
<tr>
<td>minor tick marks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you omit all color options, the AXIS statement looks for a color specification in this order:

1. the CTEXT= graphics option in a GOPTIONS statement
2. the color of all axis components is the color of the default style

Alias          C=

Example        “Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

INTERVAL=EVEN | UNEVEN | PARTIAL
The INTERVAL option affects the LOGBASE option in the AXIS statement. Specifying the options INTERVAL=UNEVEN and LOGBASE=10 permits non-base-10 values to be specified for the ORDER option, while retaining a logarithmic scale for the axis.

Restriction    Not supported by Java and ActiveX

Note           PARTIAL is an alias for UNEVEN. They have the same effect.
LABEL=(text-argument(s)) | NONE
modifies an axis label. Text-argument(s) defines the appearance or the text of an axis label, or both. NONE suppresses the axis label. Text-argument(s) can be one or more of these:

"text-string"
provides up to 256 characters of label text. By default, the text of the axis label is either the variable name or a previously assigned variable label. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVALn | #BYVAL(BY-variable-name)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. Variable-name is not case sensitive.

Examples
“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488
“Example 8: Combining Graphs and Reports in a Web Page” on page 495

#BYVARn | #BYVAR(BY-variable-name)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. Variable-name is not case sensitive.

Note
A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See
“Substituting BY Line Values in a Text String” on page 369

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. Text-description-suboption can be any of the following:

• ANGLE=degrees
• COLOR=text-color
• FONT=font | NONE
• HEIGHT=text-height <units >
• JUSTIFY=LEFT | CENTER | RIGHT
• ROTATE=degrees

See “Text Description Suboptions” on page 361 for a complete description of these suboptions.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

### Style reference
Color attribute of the GraphLabelText style element

### Restrictions
Partially supported by Java and ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. This is because BY lines are not created in a DATA step.

### Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

“Example 2: Specifying Logarithmic Axes” on page 469

“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

### LENGTH=axis length <units >

specifies the length of the axis in number of units. If you request a length that cannot fit the display, a warning message is written to the log and your graph might produce unexpected results.

### Style reference
Color attribute of the GraphLabelText graph element

### Restrictions
Not supported by the GRADAR Procedure

Not supported by Java

### Examples
“Example 2: Specifying Logarithmic Axes” on page 469

“Example 8: Combining Graphs and Reports in a Web Page” on page 495

### 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 24.1 on page 350.
Figure 24.1  Axis Generated with LOGBASE=10 and LOGSTYLE=EXPAND

Restrictions

Not supported by the GRADAR Procedure

Not supported by Java

When the device specified is ActiveX and LOGBASE=10, formats such as percentage are not applied to the axis values

Example

“Example 2: Specifying Logarithmic Axes” on page 469

LOGSTYLE=EXPAND | POWER

specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power. The LOGSTYLE= option is meaningful only when you use the LOGBASE= option.

LOGSTYLE=EXPAND specifies that the values displayed are the values of the base raised to successive powers and that the minor tick marks are logarithmically placed. For example, if the base is 10, the values displayed are 10, 100, 1000, 10000, and so on. The default is LOGSTYLE=EXPAND. This statement generates an axis like the one in part (a) of Figure 24.2 on page 350:

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 24.2 on page 350:

axis logbase=10 logstyle=power;

Figure 24.2  Axes Generated with the LOGSTYLE=option

If you use the ORDER= option with a logarithmic axis, the values specified by the ORDER= option must match the style specified by the LOGSTYLE= option. For example, if you specify a logarithmic axis with a base of 2 and you want to display the first five expanded values, use this statement:

axis logbase=2 logstyle=expand
order=(2 4 8 16 32);

If you use LOGSTYLE=POWER, the values in the ORDER= option must represent the powers to which the base is raised, as in this example:

axis logbase=2 logstyle=power order=(1 2 3 4 5);

If the values that are specified by ORDER= do not match the type of values specified by LOGSTYLE=, the request for a logarithmic axis is ignored.

Restrictions
Not supported by the GRADAR Procedure
Not supported by Java

Example
“Example 2: Specifying Logarithmic Axes” on page 469

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. List all suboptions and their values within one set of parentheses. **Tick-mark-suboption** can be the following:

• COLOR=tick-color
• HEIGHT=tick-height <units>
• NUMBER=number-of-ticks
• WIDTH=thickness-factor

Restrictions
Partially supported by Java and ActiveX. HEIGHT is not supported by Java or ActiveX. WIDTH is not supported by Java.

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MAJOR= because the axis does not use tick marks.

AXIS definitions assigned to the group axis of a block chart ignore MAJOR= because the grid does not use either axes or tick marks.

Note
By default, tick marks are placed at three intervals on the spokes of a GRADAR chart. They are placed at the minimum value, maximum value, and at one value in between. The tick marks on the 12 o’clock spoke are also labeled by default.

See
“Tick Mark Description Suboptions” on page 366 for complete descriptions

Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

“Example 2: Specifying Logarithmic Axes” on page 469

“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

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. List all suboptions and their values within one set of parentheses. **Tick-mark-suboption** can be the following:
COLOR=tick-color
HEIGHT=tick-height <units >
NUMBER=number-of-ticks
WIDTH=thickness-factor

Restrictions
Partially supported by Java and ActiveX. HEIGHT is not supported by Java or ActiveX.
Not supported by the GRADAR Procedure

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MINOR= because the axis does not use tick marks.

AXIS definitions assigned to the group axis of a block chart ignore MINOR= because the grid does not use either axes or tick marks.

See
“Tick Mark Description Suboptions” on page 366 for complete descriptions

Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465
“Example 2: Specifying Logarithmic Axes” on page 469
“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

NOBRACKETS
suppresses the printing of group brackets drawn around the values on the group axis in a bar chart. NOBRACKETS applies only to the group axis of bar charts.

Restrictions
Not supported by Java and ActiveX
Not supported by the GRADAR Procedure

See
“GROUP=group-variable” on page 915
“GAXIS=AXIS<1 ...99>” on page 915

NOPLANE
removes either the horizontal or vertical three-dimensional axis plane in bar charts produced by the HBAR3D and VBAR3D statements. NOPLANE affects only the axis to which the AXIS statement applies.

To remove selected axis elements such as lines, values or labels, use specific AXIS statement options. To remove all axis elements except the three-dimensional planes use the NOAXIS option in the procedure. To remove the three-dimensional back wall, use the NOFRAME option in the procedure.

Restriction
Not supported by the GRADAR Procedure

Example
“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488
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.

Restrictions
Not supported by Java
Not supported by the GRADAR procedure

Example
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

ORDER=(value-list)
specifies the order in which data values appear on the axis. The values specified by the ORDER= option are the major tick mark values. You can modify the appearance of these values with the VALUE= option.

The way you specify value-list depends on the type of variable:

value-list
For numeric variables, value-list is either an explicit list of values or a starting value and an ending value with an interval increment, or a combination of both forms:

- n < ...n>
- n TO n <BY increment>
- n< ...n> TO n <BY increment > <n < ...n > >

If a numeric variable has an associated format, the specified values must be the unformatted values.

Values must be listed in either ascending or descending order. By default the increment value is 1. You can use a negative integer for increment to specify a value list in descending order. In all forms, multiple n values can be separated by blanks or commas. Here are some examples:

order=(2 4 6)
order=(6,4,2)
order=(2 to 10 by 2)
order=(50 to 10 by -5)
order=(0 to 4 by 1, 5 to 19 by 2)

There are some instances where the specified range is not evenly divisible by the increment value. In those cases the highest value displayed on the axis is the last incremental value below the ending value for the range. For example, this value list produces a maximum axis value of 9:
order=(0 to 10 by 3)

value-list
For character variables, value-list is a list of unique character values enclosed in quotation marks and separated by blanks:
"value-1" <" value-2" ..." value-n">

If a character variable has an associated format, the specified values must be the formatted values for PROC GCHART and the unformatted values for PROC GPLOT.

You are able to specify up to 256 characters for each variable value.

Character values can be specified in any order, but the character strings must match exactly the variable values in case and spelling. For example:
order=("Paris" *London* "Tokyo")

Observations can be inadvertently excluded if entries in the value-list are misspelled or if the case does not match exactly.

value-list
For date and time values, value-list can have the following forms:

• "SAS-value"i < ..."SAS-value"i>
• "SAS-value"i TO "SAS-value"i <BY interval>

"SAS-value"i
is any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX. Enclose the value in quotation marks and specify one of the following for i:
D
date
T
time
DT
datetime

interval
is one of the valid arguments for the INTCK or INTNX functions. These are the default intervals:
DAY
default interval for date
SECOND
default interval for time
DTSECOND
default interval for datetime
These value lists use SAS date and time values:

\[
\text{order=}"25\text{MAY}16\text{d} \ 04\text{JUL}16\text{d} \ 07\text{SEP}16\text{d}\"
\]
\[
\text{order=}"01\text{JUL}16\text{d} \text{to} \ 01\text{AUG}16\text{d}\"
\]
\[
\text{order=}"01\text{JUL}15\text{d} \text{to} \ 01\text{JAN}16\text{d} \text{by week}\"
\]
\[
\text{order=}"9:25\text{t} \text{to} \ 11:25\text{t} \text{by minute}\"
\]
\[
\text{order=}"04\text{JUN}16\text{d} \ 12:00:00\text{dt to} \ 10\text{JUN}16\text{d} \ 12:00:00\text{dt by dtday}\"
\]
\[
\text{order=}"18\text{JAN}2015\text{d} \text{to} \ 17\text{DEC}2015\text{d} \text{by semimonth}\"
\]
\[
\text{order=}"18\text{JAN}2015\text{d} \ 01\text{FEB}2015\text{d} \text{to} \ 17\text{DEC}2015\text{d} \text{by semimonth}\"
\]

With SAS date and time values, use a FORMAT statement so that the tick mark values have an understandable form. For more information about SAS date and time values, see the SAS Formats and Informats: Reference For more information about SAS date and time values, see the “Formats by Category” in SAS Formats and Informats: Reference.

Note: The last two value lists result in different axis values. In the first order by semi-month statement, the first axis tick mark value is the starting date of 18JAN2015. The remaining tick marks are generated in 15-day intervals from the 18th. The tick marks are 18JAN2015, 03FEB2015, 18FEB2015, ... 16DEC2015. In the second order by semi-month statement, the starting tick mark is specified at 18JAN2015. This is followed by the start and end dates specified for the SEMIMONTH interval. In this instance the tick marks are 01FEB2015, 15FEB2015, 01MAR2015 ... 15DEC2015.

With any type of value-list, specifying values that are not distributed uniformly or are not in ascending or descending order, generates a warning message in the SAS log. The specified values are spaced evenly along the axis even if the values are not distributed uniformly.

Using the ORDER= option to restrict the values displayed on the axis can result in clipping. For example, the data range is 1 to 10 and you specify ORDER=(3 TO 5). Only the data values from 3 to 5 appear on the plot or chart. For charts, the omitted values are still included in the statistic calculation.

Note: Values out of range do not always produce a warning message in the SAS log.

Using the ORDER= option to specify non-uniform intervals for major tick marks can result in no minor tick marks being drawn. It also generates a warning message in the SAS log.

Using the ORDER= option to specify only starting and ending tick mark values without a BY increment can result in a recalculation of the requested order. Unless specified, the BY increment is 1 by default. The recalculation occurs when more than 20 tick marks are requested. A warning message in the SAS log indicates that the original order request is ignored in favor of using the default order algorithm. This recalculation avoids displaying output with overlapping tick marks.

**CAUTION:**

The ORDER= option does not calculate midpoint values. As a result, it is not interchangeable with the MIDPOINTS= option in the GCHART procedure.

You can use the ORDER= option to specify the order in which the midpoints are displayed on a chart, but do not use it to calculate midpoint values. Make sure that the values that you specify match the midpoint values that are calculated either by default by the GCHART procedure or by the MIDPOINTS= option. For details, see the description of the MIDPOINTS= option for the appropriate statement in Chapter 35, “GCHART Procedure,” on page 872.
Restrictions

The Java applet supports the ORDER= option for uniform numeric axes, but does not support the ORDER= option for categorical, character, midpoint, or group axes. The Java applet does not support non-uniform interval values. If you specify such values, the Java applet displays only the minimum and maximum values along with an even spacing of the number of tick marks specified between those values. For example, this non-uniform value list produces a minimum value of 0 and a maximum value of 50. It also produces a total of 4 evenly spaced tick marks with values 0, 16.67, 33.34, and 50:

ORDER=(0 20 30 50)

Not valid with the ASCENDING, DESCENDING, and NOZEROS options used with the bar chart statements in the GCHART procedure.

Interaction

The ORDER= option overrides the suboption NUMBER= described in “Tick Mark Description Suboptions” on page 366.

Note

The ActiveX control supports simple order lists and non-uniform interval values for numerical data and dates.

Examples

“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

“Example 5: Filling the Area between Plot Lines” on page 481

“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

ORIGIN=(<x><,y><units> | (<x><units><,y><units>))

specifies the x coordinate and the y coordinate of the origin of the axis. The origin of the horizontal axis is the left end of the axis, and the origin of the vertical axis is the bottom of the axis. The ORIGIN= option explicitly positions the axis anywhere on the graphics output area.

If you specify only one value, with or without a comma following it, only the x coordinate is set to that value. For example, this specification sets x to 4 centimeters:

origin=(4 cm,)

If you specify two values, with or without a comma separating them, the first value sets the x coordinate and the second value sets the y coordinate:

origin=(2 pct, 4 pct)

If you specify one value preceded by a comma, only the y coordinate is set to that value, as shown here:

origin=(,3 pct)

You can specify units for the x,y pair or for the individual coordinates.

Restrictions

Not supported by 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 quotation marks. Separate multiple strings with blank spaces. The strings are applied to the reference lines specified by the VREF or HREF option.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVALn | #BYVAL(BY-variable-name)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. Variable-name is not case sensitive.

Examples
“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

“Example 8: Combining Graphs and Reports in a Web Page” on page 495

#BYVARn | #BYVAR(BY-variable-name)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. Variable-name is not case sensitive.

Note
A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See
“Substituting BY Line Values in a Text String” on page 369

text-description-suboption modifies a characteristic such as the font, color, or size of the text string(s) that follows it. Text-description-suboption can be the following:

• ANGLE=degrees
• AUTOREF
• COLOR=text-color
• FONT=font | NONE
• FORMAT="ValidFormat’
• HEIGHT=text-height <units >
• JUSTIFY=LEFT | CENTER | RIGHT
• POSITION=TOP | MIDDLE | BOTTOM
• ROTATE=degrees
• T=n

See “Text Description Suboptions” on page 361 for a complete description of these suboptions.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference  Font and Color attributes of the GraphLabelText element

Restrictions  Not supported by Java and ActiveX
Not supported by the GRADAR procedure

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

SPLIT="split-char(s)"
specifies one or more characters that the AXIS statement uses to break axis values into multiple lines. Split-char(s) can be any character value that can be specified in a SAS character variable. Do not delimit when specifying multiple split characters. Each split character must be embedded in the variable values in the data set or sets, or in an associated format. When the AXIS statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the data set contains the value Berlin, Germany/Europe, and you specify SPLIT=",/". The value would appear on the axis as follows: If the AXIS statement does not encounter a specified split character, no break in the variable value occurs, and no warning or error is issued.

Berlin
Germany
Europe

Axis values specified with VALUE= do not use the split character. For example, suppose you specify this statement:

axis1 split="", value=(tick=1 "December, 1999");

The value appears on the axis on one line as December, 1999. However, any other axis values containing a comma honors the split character.

Restrictions  Not supported by Java and ActiveX
Not supported by the GRADAR procedure

Up to 31 split characters are supported. If you specify more than 31 characters, only the first 31 are honored.

Note  The split characters themselves are not displayed.

Example  “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204
**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.

**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.

<table>
<thead>
<tr>
<th>Style reference</th>
<th>Line style attribute of the GraphAxisLine element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>In order for the axis line to be altered by the STYLE= option, the NOFRAME option must also be set. If only the STYLE= option is set, the line framing the axis area is modified.</td>
</tr>
<tr>
<td></td>
<td>When specifying both the STYLE= and WIDTH= suboptions, only the WIDTH= specification is used, resulting in a modification to the tick mark thickness but not the axis line type</td>
</tr>
<tr>
<td></td>
<td>Using the PNG device with STYLE= results in the specified line type being applied to the line framing the axis area</td>
</tr>
</tbody>
</table>

**VALUE=(text-argument(s)) | NONE**
modifies the major tick mark values. That is, this option modifies the text that labels the major tick marks on the axis. *Text-argument(s)* defines the appearance or the text of a major tick mark value, or both. NONE suppresses the major tick mark values, although the major tick marks are still displayed. *Text-argument(s)* can be one or more of these:

"text-string"
provides up to 256 characters of text for the major tick mark value. By default, the value is either the variable value or an associated format value. Enclose each string in quotation marks and separate multiple strings with blanks.

Specified text strings are assigned to major tick marks in order. If you specify only one text string, only the first tick mark value changes, and all the other tick mark values display the default. If you specify multiple strings, the first string is the value of the first major tick mark. The second string is the value of the second major tick mark, and so on. For example, to change default tick mark values 1, 2, and 3 to *First, Second, and Third*, use this option:

```
value="First" "Second" "Third"
```

**Note:** Although the VALUE= option changes the text displayed at a major tick mark, it does not affect the actual value represented by the tick mark. To change the tick mark values, use the ORDER= option. Also note that with the Java or ActiveX devices, it is necessary to use the ORDER= option. This ensures that the same number of tick marks are displayed as are with graphics rendered with the other device drivers. For example, specify ORDER=(1 to 12) to ensure that tick marks for all twelve months are displayed.

**Note:** To change the value of midpoints in bar charts produced with the GCHART procedure, use the MIDPOINTS= option in the procedure.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:
#BYVALn | #BYVAL(BY-variable-name)
substitutes the current value of the specified BY variable for #BYVAL in the
text string and displays the value produced by the statement. Specify the
variable with one of these:

\( n \)

specifies which variable in the BY statement #BYVAL should use. The
value of \( n \) indicates the position of the variable in the BY statement. For
example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY
variable, YEAR. Variable-name is not case sensitive.

Examples

“Example 7: Using BY-group Processing to Generate a Series of
Charts” on page 488

“Example 8: Combining Graphs and Reports in a Web Page” on
page 495

#BYVARn | #BYVAR(BY-variable-name)
substitutes the name of the BY variable or label associated with the variable
(whatever the BY line would normally display) for #BYVAR in the text
string and displays the name or label produced by the statement. Specify the
variable with one of these:

\( n \)

specifies which variable in the BY statement #BYVAR should use. The
value of \( n \) indicates the position of the variable in the BY statement. For
example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAR(SITES) specifies the BY
variable, SITES. Variable-name is not case sensitive.

Note

A BY variable name displayed is always in uppercase. If a label is
used, it appears as specified in the LABEL statement.

See

“Substituting BY Line Values in a Text String” on page 369

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that
follows it. Text-description-suboption can be the following:

- \( \text{ANGLE} = \text{degrees} \)
- \( \text{COLOR} = \text{text-color} \)
- \( \text{FONT} = \text{font} | \text{NONE} \)
- \( \text{HEIGHT} = \text{text-height <units>} \)
- \( \text{JUSTIFY} = \text{LEFT} | \text{CENTER} | \text{RIGHT} \)
- \( \text{ROTATE} = \text{degrees} \)
- \( \text{TICK} = n \)

Specify as many text strings and text description suboptions as you want, but
enclose them all in one set of parentheses.

For a complete description of these suboptions, see “Text Description
Suboptions” on page 361.
### AXIS Statement

**Style reference**
- Color attribute of the GraphLabelText graph element

**Restrictions**
- Partially supported by Java

Place text description suboptions before the text strings that they modify. Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

**Interaction**
- This option, with any of its specified suboptions, overrides the RANGE= option when creating vertical bar charts, as with the GBARLINE and GCHART procedures. This is noted in the log as follows:

```
NOTE: Axis value is specified. RANGE option is ignored.
```

**Note**
- You might view a graph in the Java applet or ActiveX control and zoom in on a particular part of a graph for which the VALUE= option is specified. The values are not readjusted in coordination with the zooming.

**Examples**
- “Example 2: Specifying Logarithmic Axes” on page 469
- “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488
- “Example 8: Combining Graphs and Reports in a Web Page” on page 495

**WIDTH=thickness-factor**
- Specifies the thickness of the axis line. Thickness increases directly with the value of `thickness-factor`. By default, WIDTH=1.

**Style reference**
- LineThickness attribute of the GraphAxisLines element

**Restriction**
- Not supported by Java

**Interactions**
- In order for the axis line to be altered by the WIDTH= option, the NOFRAME option must also be set. If only the WIDTH= option is set, the line framing the axis area is modified.
- When specifying both the STYLE= and WIDTH= suboptions, only the WIDTH= specification is used, resulting in a modification to the tick mark thickness but not the axis line type.

**Example**
- “Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

---

**Text Description Suboptions**

**ANGLE=degrees**
- Specifies the angle of the baseline with respect to the horizontal. A positive value for `degrees` moves the baseline counterclockwise; a negative value moves it clockwise.
By default, ANGLE=0 (horizontal) unless the text is automatically angled or rotated to avoid overlapping.

*Note:* Changing the angle of a vertical axis-label can result in the label being positioned above the graph when using the Java or ActiveX device drivers.

<table>
<thead>
<tr>
<th>Alias</th>
<th>A=</th>
</tr>
</thead>
</table>

**Restrictions**

You cannot justify angled text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored when the ANGLE= value is anything other than zero.

<table>
<thead>
<tr>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partially supported by Java</td>
</tr>
</tbody>
</table>

**Example**

“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**AUTOREF**

automatically labels each reference line on an axis with the response value at the reference line's position. The AUTOREF suboption is used only with the REFLABEL= option, and can be used in conjunction with the FORMAT= suboption to refine the labels. When the FORMAT= suboption applies a valid format to a response variable value, AUTOREF displays those formatted values as the reference line labels. When the FORMAT= suboption is omitted, AUTOREF displays (as reference line labels) the response variable values as is.

The automatic labels are applied only to reference lines without specific labels assigned to them. For example, the following option uses the response-axis value as the label for every reference line except the second reference line, which is assigned the label *two*:

```
relabel=(autoref t=2 "two")
```

*Note:* You might simultaneously request automatic labeling with a PLOT or BUBBLE statement (using the AUTOHREF or AUTOVREF option). In this case the automatic labeling can write on top of the custom label that you specified using the AXIS statement. You must ensure that your custom labels specified using the AXIS statement are not at the same position as automatic labels requested with a different statement.

**Restriction**

Not supported by Java and ActiveX

<table>
<thead>
<tr>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>“FORMAT=’ValidFormat’” on page 363</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Alias</th>
<th>C=</th>
</tr>
</thead>
</table>
**FONT=**<font> | NONE

specifies the font for the text. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about specifying font. If you omit the FONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default style font, NONE

You might use a device-resident font for your midpoint axis labels and find that one or more of the axis labels overlap the legend in your graph. In this case the device-resident font for each label that overlaps the legend is replaced with a SAS/GRAPH font. Because device-resident fonts do not support clipping, a SAS/GRAPH font must be substituted in that case. This includes the axis label and the tick mark labels. An overlap can occur when you use the MODE=PROTECT option on your LEGEND statement with labels on your midpoint axis. To correct an overlap condition, you can use the NOLEGEND option on your plot statement to suppress the legend. Or you can use the positioning options on your LEGEND statement to reposition the legend. See “ POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)” on page 384.

**Alias**

F=

**Restriction**

Partially supported by Java

**FORMAT=’ValidFormat’**

specifies a valid format to be applied to the response-axis value shown with the AUTOREF label on each reference line.

This suboption allows for the character or numeric formatting of variable values for the X axis and Y axis. The FORMAT= suboption of the REFLABEL option controls the display of different values at the reference lines relating to conversion rates, scales, and the like. If an invalid format is specified, then a warning message is issued and the resulting labels have response-axis values with no formatting. When FORMAT= is omitted, the default reverts to the AUTOREF suboption behavior of producing labels automatically with unformatted response-axis values.

**Restrictions**

Must be used with the AUTOREF suboption of the REFLABEL option.

Formatting might accommodate response-axis variable values that are up to 256 characters in length.

Not supported by Java and ActiveX

**Note**

Although the AUTOREF suboption ignores any format assigned to an x- or y-axis variable with the FORMAT statement, the FORMAT= suboption provides a way to format each response-axis variable value.

**See**

“ AUTOREF” on page 362

**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
JUSTIFY=LEFT | CENTER | RIGHT

specifies the alignment of the text. The default depends on the option with which it is used and the text that it applies to.

- With the LABEL= option, the following is true:
  - For a left vertical axis label, the default is JUSTIFY=RIGHT.
  - For a right vertical axis label, the default is JUSTIFY=LEFT.
  - For a horizontal axis label, the default is JUSTIFY=CENTER.

- With the REFLABEL= option, the following is true:
  - For a reference line that intersects a vertical axis, the default is JUSTIFY=CENTER. RIGHT places the text string on the right end of the line. CENTER places the text string in the middle of the line. LEFT places the text string to the left of the line.
  - For a reference line that intersects a horizontal axis, the default is JUSTIFY=RIGHT for all procedures except the BAR statement in GBARLINE. For the BAR statement in GBARLINE the default is JUSTIFY=LEFT. RIGHT places the text string just to the right of the line. CENTER centers the text string on top of the line. LEFT places the text string just to the left of the line.

- With the VALUE= option, the following is true:
  - For numeric variables on a vertical axis, the default is JUSTIFY=LEFT, starting with SAS/GRAPH release 9.2. Specify JUSTIFY=RIGHT to align the variable values according to release 9.1 and earlier behavior.
  - For character variables on a vertical axis, the default is JUSTIFY=LEFT.
  - For all variables on a horizontal axis, the default is JUSTIFY=CENTER.

You can use the JUSTIFY= option to print multiple lines of text by repeating the JUSTIFY= option before the text string for each line. You can also use JUSTIFY= to specify multi-line text at specified major tick marks. For example, this statement produces an axis label and major tick mark values like those shown in Figure 24.3 on page 364.

```sas
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 24.3 The JUSTIFY= suboption

Place text description suboptions before the text strings that they modify.
Restrictions
You cannot justify angled or rotated text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored.

Not supported by Java

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.

See
“TICK=n” on page 365

POSITION=TOP | MIDDLE | BOTTOM
specifies the position of a reference-line label relative to the reference line. The default is TOP for both vertical and horizontal reference lines. The POSITION= option is available only on the REFLABEL= option.

• For horizontal reference lines, TOP places the label just above the reference line. MIDDLE places the label on the reference line. BOTTOM places the label just below the reference line.

• For vertical reference lines, TOP places the label at the top end of the reference line. MIDDLE places the label in the middle of the line. BOTTOM places the label at the bottom end of the line.

Restriction
Not supported by 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=

Restrictions
You cannot justify angled text that is used to describe tick marks on a horizontal axis; the JUSTIFY= suboption is not honored when the ROTATE= value is anything other than zero.

Partially supported by Java

See
“ANGLE=degrees” on page 361

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<sup>th</sup> 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:
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:

```sas
color=green "one" "two" t=3 j=left "three"
```

In order for the options to be applied to a text string, they must precede the quoted string. In the following option, the `j=left` is ignored because it follows the string:

```sas
color=green "one" "two" t=3 "three" j=left
```

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:

```sas
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:

```sas
height=4 color=blue tick=3 color=red
```

**Tick Mark Description Suboptions**

- **COLOR=tick-mark-color**
  
  colors the tick marks. If you omit the COLOR= suboption, a color specification is searched for in this order:
  
  1. the COLOR= option in the AXIS statement
  2. the CAXIS= option for the procedure
  3. the color of the default style

- **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.
NUMBER=number-of-ticks
specifies the number of tick marks to be drawn. With the MAJOR= option, number-of-ticks must be greater than 1. With the MINOR= option, number-of-ticks must be greater than 0.

With the MAJOR= option, the NUMBER= suboption can be overridden by a major tick mark specification in the procedure, which in turn can be overridden by the ORDER= option.

With the MINOR= option, the NUMBER= suboption can be overridden by a minor tick mark specification in the procedure.

The NUMBER= option is not valid with logarithmic axes.

Alias N=

WIDTH=thickness-factor
specifies the thickness of the tick mark, where thickness-factor is a number. Thickness increases directly with thickness-factor. By default, WIDTH=1.

Alias W=

Style reference LineThickness attribute of the GraphAxisLines element

Restriction Partially supported by Java

Details

Description: AXIS Statement
AXIS statements specify the following characteristics of an axis:
• how the axis is scaled
• how the data values are ordered
• the location and appearance of the axis line and the tick marks
• the text and appearance of the axis label and major tick mark values

AXIS definitions are used only when they are explicitly assigned by an option in a procedure that produces graphs with axes.

Figure 24.4 on page 368 illustrates the terms associated with the various parts of axes.
Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```plaintext
label=(font=swiss height=4 "Weight"
       justify=right height=3 "(in tons)"
)
```

FONT=SWISS applies to both Weight and (in tons). HEIGHT=4 affects Weight, but is respecified as HEIGHT=3 for (in tons). JUSTIFY=RIGHT affects only (in tons).

Using the AXIS Statement

AXIS statements can be defined anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session. AXIS statements are not applied automatically, and must be explicitly assigned by an option in the procedure that uses them.

You can define up to 99 different AXIS statements. If you define two AXIS statements of the same number, the most recently defined statement replaces the previously defined statement of the same number. An AXIS statement without a number is treated as an AXIS1 statement.

Cancel individual AXIS statements by defining an AXIS statement of the same number without options (a null statement):

```plaintext
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:

```plaintext
gooptions reset=axis;
```
Specifying RESET=GLOBAL or RESET=ALL cancels all current AXIS definitions as well as other settings.

To display a list of current AXIS definitions in the LOG window, use the GOPTIONS procedure with the AXIS option:

```
proc goptions axis nolist;
run;
```

### Assigning AXIS Definitions

AXIS definitions must always be explicitly assigned by the appropriate option in the statement that generates the graph. The following table lists the procedures and statements that generate axes, the type of axis, and the statement option that assigns an AXIS definitions to that axis:

#### Table 24.2 Procedures and Statements That Generate Axes

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Statement that generates an axis</th>
<th>Type of axis</th>
<th>Option that assigns an AXIS definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBARLINE</td>
<td>BAR</td>
<td>PLOT</td>
<td>Midpoint axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response axis</td>
<td></td>
</tr>
<tr>
<td>GCHART</td>
<td>HBAR</td>
<td>VBAR</td>
<td>Group axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Midpoint axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response axis</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCONTOUR</td>
<td>PLOT</td>
<td>Horizontal axis</td>
<td>HAXIS=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vertical axis</td>
<td></td>
</tr>
<tr>
<td>GPLOT</td>
<td>PLOT</td>
<td>Horizontal axis</td>
<td>HAXIS=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vertical axis</td>
<td></td>
</tr>
<tr>
<td>GRADAR</td>
<td>CHART</td>
<td>Star axis</td>
<td>STARAXIS=</td>
</tr>
</tbody>
</table>

Some types of axes cannot use certain AXIS statement options:

- Group and midpoint axes ignore the LOGBASE=, MAJOR=, and MINOR= options.
- Midpoint, horizontal and vertical axes ignore the NOBRACKETS option.

### Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.
To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the “BY Statement” on page 370.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

---

**BY Statement**

Processes data and orders output according to the BY group.

**Used by:** GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, GRADAR, GTILE, G3D, G3GRID

**Type:** DATA step statement

**Syntax**

```
BY <DESCENDING> variable-1 <DESCENDING> variable-2 ...<NOTSORTED>;
```

**Required Argument**

`variable`

specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. By default, the procedure expects observations in the data set to be sorted in ascending order by all the variables that you specify or to be indexed appropriately.

**Optional Arguments**

DESCENDING

indicates that the data set is sorted in descending order by the specified variable. The option affects only the variable that immediately follows the option name, and must be repeated before every variable that is not sorted in ascending order. For example, this BY statement indicates that observations in the input data set are arranged in descending order of VAR1 values and ascending order of VAR2 values:

```
by descending var1 var2;
```
This BY statement indicates that the input data set is sorted in descending order of both VAR1 and VAR2 values:

```
by descending var1 descending var2;
```

**NOTSORTED**

specifies that observations with the same BY value are grouped together, but are not necessarily sorted in alphabetical or numeric order. The observations can be grouped in another way (for example, in chronological order).

NOTSORTED can appear anywhere in the BY statement and affects all variables specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

The requirement for ordering or indexing observations according to the values of BY variables is suspended when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. For NOTSORTED, the procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. Observations with the same value for the BY variables might not be contiguous. The procedure treats each new value that it encounters as the first observation in a new BY group. The procedure creates a graph for that value, even if it is only one observation.

**Details**

**Description: BY Statement**

The BY statement divides the observations from an input data set into groups for processing. Each set of contiguous observations with the same value for a specified variable is called a **BY group**. A variable that defines BY groups is called a **BY variable** and is the variable that is specified in the BY statement. When you use a BY statement, the graphics procedure performs the following operations:

- processes each group of observations independently
- generates a separate graph or output for each BY group
- automatically adds a heading called a **BY line** to each graph identifying the BY group represented in the graph
- adds BY statement information below the **Description** field of the catalog entry

By default, the procedure expects the observations in the input data set to be sorted in ascending order of the BY variable values.

**Note:** The BY statement in SAS/GRAPH is essentially the same as the BY statement in Base SAS. However, the effect on the output is different when it is used with SAS/GRAPH procedures.

**Preparing Data for BY-Group Processing**

Unless you specify the NOTSORTED option, observations in the input data set must be in ascending numeric or alphabetic order. To prepare the data set, you can sort it with the SORT procedure using the same BY statement that you plan to use in the target SAS/GRAPH procedure. Or you can create an appropriate index on the BY variables.

If the procedure encounters an observation that is out of the proper order, it issues an error message.

If you need to group data in some other order, you can still use BY-group processing. To do so, process the data so that observations are arranged in contiguous groups that have the same BY-variable values and specify the NOTSORTED option in the BY statement.
For an example of sorting the input data set, see “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488.

### Controlling BY Lines

#### Understanding Default Behavior
By default, the BY statement prints a BY line above each graph that contains the variable name followed by an equal sign and the variable value. For example, if you specify BY SITE in the procedure, the default heading when the value of SITE is London would be SITE=London.

#### Suppressing the BY Line
To suppress the entire BY line, use the NOBYLINE option in an OPTION statement or specify HBY=0 in the GOPTIONS statement. See “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488.

#### Suppressing the Name of the BY Variable
You can suppress the variable name and the equal sign in the heading and leave only the BY value. Use the LABEL statement to assign a null label ("00"X) to the BY variable. For example, this statement assigns a null label to the SITE variable:

```plaintext
label site="00"x;
```

#### Controlling the Appearance of the BY Line
To control the color, font, and height of the BY lines, use the following graphics options in a GOPTIONS statement:

- **CBY=BY-line-color**: specifies the color for BY lines.
- **FBY=font**: specifies the font for BY lines.
- **HBY=n<units>**: specifies the height for BY lines.

For a description of each option, see the Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

### Naming the Catalog Entries
The catalog entries generated with BY-group processing always use incremental naming. This means that the first entry created by the procedure uses the base name and subsequent entries increment that name. The base name is either the default entry name for the procedure (for example, GPlot) or the name specified with the NAME= option in the action statement. Incrementing the base name automatically appends a number to each subsequent entry (for example, GPlot1, GPlot2, and so on). See “Specifying the Catalog Name and Entry Name for Your GRSEGs ” on page 120. For an example of incremented catalog names, see “Example 8: Combining Graphs and Reports in a Web Page” on page 495.

### Using the BY Statement

#### Overview
This section describes the following:

- the effect of BY-group processing on the GCHART and GPLOT procedures
- the interaction between BY-group and RUN-group processing
• the requirements for using BY-group processing with the Annotate facility
• how to include BY information in axis labels, reference labels and tick mark values, donut labels, legend labels and values, and titles, notes, and footnotes
• how patterns and symbols are assigned to BY-groups
• the effect of using BY-group processing with the ODS HTML statement

For additional information about any of these topics, refer to the appropriate chapter.

Using the BY Statement with the GCHART Procedure

When you use BY-group processing with the GCHART procedure, you can do the following tasks:

• With the BLOCK, HBAR, and VBAR statements, you can use the PATTERNID=BY option to assign patterns according to BY groups. With PATTERNID=BY, each BY group uses a different PATTERN definition, but all bars or blocks within a BY group use the same pattern. For further information, see “Example: PATTERN and SYMBOL Definitions with BY Groups in the GCHART Procedure” on page 374.

• With the BLOCK statement, you can use the BLOCKMAX= option to produce the same block-height scaling in all block charts in a BY group.

• With the HBAR or VBAR statement, you can use the RAXIS= option. This option produces the same response axis scaling in all horizontal or vertical bar charts in a BY group.

• With the DONUT statement, you can use the LABEL= option to substitute a BY variable value. Or you can substitute a name for #BYVAL or #BYVAR in a text string to display in the donut hole label.

With the PIE and STAR statements, the effect of a BY statement is similar to that of the GROUP= option. The exception is that the GROUP= option enables you to put more than one graph on a single page while the BY statement does not. Do not use a BY variable as the group variable in STAR or PIE statements.

Using the BY Statement with the GPLOT Procedure

You can use the UNIFORM option in the PROC GPLOT statement to produce the same axis scaling for all graphs in a BY group. By default, the range of the axes can vary from graph to graph, but UNIFORM forces the scaling to be the same for all graphs generated by the procedure.

The UNIFORM option applies colors and heights uniformly across all BY-groups.

Using the BY Statement with the RUN Groups

You can use the BY statement with a GCHART or GPLOT procedure that processes data and supports RUN-group processing. In this case, each time you submit an action statement or a RUN statement, you get a separate graph for each value of the BY variable. For example, each of these two RUN-groups produces a separate plot for every value of the BY variable YEAR:

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

The BY statement stays in effect for every subsequent RUN group until you submit another BY statement or exit the procedure. Variables in subsequent BY statements replace any previous BY variables.

You can also turn off BY-group processing by submitting a null BY statement (BY;) in a RUN group. Do this with care however, because the null BY statement turns off BY-group processing and the RUN group generates a graph.

For more information, see Chapter 7, “Using Run-Group Processing,” on page 67.

Using the BY Statement with the Annotate Facility
If a procedure that is using BY-group processing also specifies annotation with the ANNOTATE= option in the PROC statement, the same annotation is applied to every graph generated by the procedure.

If you specify annotation with the ANNOTATE= option in the action statements for a procedure, the BY-group processing is applied to the Annotate data set. In this way, you can customize the annotation for the output from each BY group. Include the BY variable in the Annotate data set and use each BY-variable value as a condition for the annotation to be applied to the output for that value.

Using the BY Statement with AXIS, LEGEND, TITLE, FOOTNOTE, and NOTE Statements
AXIS and LEGEND statements can automatically include the BY variable name or BY variable value in the text that they produce for labels, reference labels, values for major tick marks, and legend labels and values. In addition, TITLE, FOOTNOTE, and NOTE statements can automatically include the BY lines in the text that they produce. To insert BY line information into the text strings used by these statements, use the appropriate #BYVAR, #BYVAL, and #BYLINE substitution options. For an example, see “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488.

Using the BY Statement with PATTERN and SYMBOL Definitions
By default, when using a BY statement, the graph for each BY group uses the same patterns or symbols in their defined order. For example, the BY variable contains four values and there are two response levels for each BY value. In this case the PATTERN1 and PATTERN2 or SYMBOL1 and SYMBOL2 statements are used for each graph. Each BY-group starts over with PATTERN1 or SYMBOL1.

Example: PATTERN and SYMBOL Definitions with BY Groups in the GCHART Procedure
The GCHART procedure, when used with SYMBOL or PATTERN definitions, assigns the symbols or patterns in order to each BY group. For example, if the BY variable REGION has four values—East, North, South, and West—the patterns are assigned to the BY-groups in this order:

1. PATTERN1 is assigned to East.
2. PATTERN2 is assigned to North.
3. PATTERN3 is assigned to South.
4. PATTERN4 is assigned to West.

You might create sets of graphs from several data sets containing the variable REGION. If you want the same pattern assigned to the same region each time, you must make sure that REGION always has the same four values. Otherwise, the patterns might not be the
same across graphs. For example, if the value **North** is missing from the data, the patterns are assigned as follows:

1. PATTERN1 is assigned to *East*.
2. PATTERN2 is assigned to *South*.
3. PATTERN3 is assigned to *West*.

In this case, *South* is assigned pattern 2 instead of pattern 3 and *West* is assigned pattern 3 instead of pattern 4. To avoid this, include the value **North** for the variable REGION, but assign it a missing value for all other variables.

---

**FOOTNOTE Statement**

Writes up to 10 lines of text at the bottom of the graph.

**Type:** Global  
**See:** “TITLE, FOOTNOTE, and NOTE Statements” on page 447

**Syntax**

FOOTNOTE<1 \ldots 10> <text-argument(s)>;

---

**GOPTIONS Statement**

Temporarily sets default values for many graphics attributes and device parameters used by SAS/GRAPH procedures.

**Used by:** all statements and procedures in a SAS session  
**Type:** Global

**Syntax**

GOPTIONS <options-list>;

**Optional Argument**

**options-list**

can be one or more options as listed and described in the Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

**Details**

**Description: GOPTIONS Statement**

The GOPTIONS statement specifies values for *graphics options*. Graphics options control characteristics of the graph, such as size, colors, type fonts, fill patterns, and symbols. If GOPTIONS are specified, they override the default style. In addition, they affect the settings of device parameters, which are defined in the device entry. Device parameters control such characteristics as the appearance of the display, the type of output produced, and the destination of the output.
The GOPTIONS statement enables you to change these settings temporarily, either for a single graph or for the duration of your SAS session. You can use the GOPTIONS statement to do the following tasks:

- override default values for graphics options that control either graphics attributes or device parameters for a single graph or for an entire SAS session
- reset individual graphics options or all graphics options to their default values
- cancel definitions for AXIS, FOOTNOTE, PATTERN, SYMBOL, and TITLE statements

To change device parameters permanently, you must use the GDEVICE procedure to modify the appropriate device entry or to create a new one. See the Chapter 37, “GDEVICE Procedure,” on page 1045 for details.

To review the current settings of all graphics options, use the GOPTIONS procedure. See the Chapter 40, “GOPTIONS Procedure,” on page 1125 for details.

**Using the GOPTIONS Statement**

GOPTIONS statements are global and can be located anywhere in your SAS program. However, for the graphics options to affect the output from a procedure, the GOPTIONS statement must execute before the procedure.

With the exception of the RESET= option, graphics options can be listed in any order in a GOPTIONS statement. The RESET= option should be the first option in the GOPTIONS statement.

A graphics option remains in effect until you specify the option in another GOPTIONS statement, use the RESET= option to reset the values, or end the SAS session. When a session ends, the values of the graphics options return to their default values.

Graphics options are additive. That is, the value of a graphics option remains in effect until the graphics option is explicitly changed or reset or until your SAS session ends. Graphics options remain in effect even after you submit additional GOPTIONS statements specifying different options.

To reset an individual option to its default value, submit the option without a value (a null graphics option.) You can use a comma (but it is not required) to separate a null graphics option from the next one. For example, this GOPTIONS statement sets the values for the background color, the text height, and the text font:

```
goptions cback=blue htext=6 pct ftext="Albany AMT";
```

*Note:* When there is a space in the font name, surround the name in quotation marks. The quotation marks also indicate use of a system font rather than a software graph font.

To reset only the background color specification to the default and keep the remaining values, use this GOPTIONS statement:

```
goptions cback=;
```

To reset all graphic options to their default values, specify RESET=GOPTIONS:

```
goptions reset=goptions;
```

Alternatively, you can use RESET=ALL, but it also cancels any global statement definitions in addition to resetting all graphics options to default values.
Graphics Option Processing
You can control many graphics attributes through statement options, graphics options, device parameters, or a combination of these. SAS/GRAPH searches these places to determine the value to use, stopping at the first place that gives it an explicit value:

1. statement options
2. the value of the corresponding graphics option
3. the value of a device parameter found in the catalog entry for your device driver

Note: Not every graphics attribute can be set in all three places. See the statement and procedure chapters for the options that can be used with each.

Some graphics options are supported for specific devices or operating environments only. See the SAS Help facility for SAS/GRAPH or the SAS companion for your operating environment for more information.

---

LEGEND Statement
Controls the location and appearance of legends on two-dimensional plots, contour plots, and charts.

**Used by:** GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT

**Type:** Global

**Syntax**

LEGEND<1 ...99><option(s)>;

**Summary of Optional Arguments**

**Appearance options**

ACROSS=number-of-columns
specifies the number of columns to use for legend entries.

CBLOCK=block-color
generates and colors a three-dimensional block effect behind the legend.

CBORDER=frame-color
draws a colored frame around the legend.

CFRAME=background-color
specifies the background color of the legend.

CSHADOW=shadow-color
generates and colors a drop shadow behind the legend.

DOWN=number-of-rows
specifies the number of rows to use for legend entries.

FRAME | NOFRAME
draws a frame around the legend.

FWIDTH=thickness-factor
specifies the thickness of the frame, where thickness-factor is a number.

REPEAT=1 | 2 | 3
specifies how many times the plot symbol is repeated in the legend.

ROWMAJOR | COLMAJOR
specifies the arrangement of legend entries when there are multiple rows and multiple columns.
specifies the size and shape of the legend values displayed in each legend entry.

\section*{Position options}

MODE=PROTECT | RESERVE | SHARE
specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements.

OFFSET=(<x><,y>)<units> | (<<x><units>>><,y<units>>)
specifies the distance to move the entire legend.

ORIGIN=(<x><,y>)<units> | (<<x><units>>><,y<units>>)
specifies the x and y coordinates of the lower left corner of the legend box.

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)
positions the legend on the graph.

\section*{Text options}

LABEL=(text-argument(s)) | NONE
modifies a legend label.

ORDER=(value-list) | DESCENDING
selects or orders the legend values that appear in the legend.

SPLIT="split-char(s)"
specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines.

VALUE=(text-argument(s)) | NONE
modifies the legend value descriptions.

\section*{Optional Arguments}

ACROSS=number-of-columns
specifies the number of columns to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

See "ROWMAJOR | COLMAJOR" on page 385

Example "Creating a Simple Web Page with the ODS HTML Statement" in SAS/GRAPH and Base SAS: Mapping Reference

CBLOCK=block-color
generates and colors a three-dimensional block effect behind the legend. The size and position of the block are controlled by the graphics option OFFSHADOW=(x,y).

The Java applet treats the CBLOCK option like the CSHADOW option.

Restrictions Not supported by Java.
The CBLOCK= and CSHADOW= options are mutually exclusive. If both are present, SAS/GRAPH software uses the last one specified.

**Interaction**
The CBLOCK= option is usually used in conjunction with the FRAME, CFRAME=, or CBORDER= options.

**See**
“OFFSHADOW” on page 584
“Creating Drop Shadows and Block Effects” on page 395

**CBORDER=**
frame-color
draws a colored frame around the legend.

**Style reference**
Color attribute of the GraphBorderLines graph element

**Restriction**
This option overrides the FRAME option.

**Interaction**
CBORDER= can be used in conjunction with the CFRAME= option.

**CFRAME=**
background-color
specifies the background color of the legend. This option overrides the FRAME option.

**Style reference**
Color attribute of the GraphLegendBackground graph element

**Restriction**
If both the CFRAME= and FRAME= options are specified, only the solid background produced by the CFRAME= option is displayed.

**Interaction**
The CFRAME= option can be used in conjunction with the CBORDER= option.

**CSHADOW=**
shadow-color
generates and colors a drop shadow behind the legend. The size and position of the shadow is controlled by the graphics option OFFSHADOW=(x,y).

**Restriction**
The CSHADOW= and CBLOCK= options are mutually exclusive. If both are present, SAS/GRAPH uses the last one specified.

**Interaction**
The CSHADOW= option is usually specified in conjunction with the FRAME, CFRAME=, or CBORDER= options.

**See**
“OFFSHADOW” on page 584
“Creating Drop Shadows and Block Effects” on page 395

**DOWN=**
number-of-rows
specifies the number of rows to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

**Default**
When there are multiple rows and columns in a legend, the ROWMAJOR option is the default
See “ROWMAJOR | COLMAJOR” on page 385

FRAME | NOFRAME
     draws a frame around the legend. The color of the frame is the first color in the color list. NOFRAME suppresses the drawing of a frame, and is the default.

FWIDTH=thickness-factor
     specifies the thickness of the frame, where thickness-factor is a number. The thickness of the line increases directly with thickness-factor. By default, FWIDTH=1.

Restriction Not supported by 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 in most instances. By default, the text of the legend label is either the variable name or a previously assigned variable label (except in the case of GPLOT with OVERLAY. In that case, the default label is “PLOT”). You can use an overlay variable and suppress the legend label that would display that variable name. Specify a SAS software font to generate the unprintable hexadecimal character of ‘00’x, as shown in this example:

legend1 label=(font=swiss '00'x);

Text-argument(s) can be one or more of these:

"text-string"
     provides up to 256 characters of label text. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVALn | #BYVAL(BY-variable-name)
     substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
     specifies which variable in the BY statement #BYVAL should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
     names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. Variable-name is not case sensitive.

Examples

“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

“Example 8: Combining Graphs and Reports in a Web Page” on page 495

#BYVARn | #BYVAR(BY-variable-name)
     substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:
specifies which variable in the BY statement \#BYVAR should use. The value of \( n \) indicates the position of the variable in the BY statement. For example, \#BYVAR2 specifies the second variable in the BY statement.

**BY-variable-name**

names the BY variable. For example, \#BYVAR(SITES) specifies the BY variable, SITES. Variable-name is not case sensitive.

**Note**  A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See  “Substituting BY Line Values in a Text String” on page 395

**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:

- \( \text{ANGLE=degrees} \)
- \( \text{COLOR=\text{text-color}} \)
- \( \text{FONT=\text{font | NONE}} \)
- \( \text{HEIGHT=\text{text-height \ <units>}} \)
- \( \text{JUSTIFY=LEFT | CENTER | RIGHT} \)
- \( \text{POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)} \)
- \( \text{ROTATE=\text{degrees}} \)

For a complete description of these suboptions, see “Text Description Suboptions” on page 389.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

**Style reference**  Color attribute of the GraphLabelText graph element

**Restrictions**  Partially supported by Java and ActiveX

\#BYVAL or \#BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

Starting with SAS 9.4M5, when there is not enough room in the output to display a long label text, the variable name is displayed instead. However, when POSITION=INSIDE and the legend label is too long, the legend is suppressed. In either case, the SAS log displays a warning message.

**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).

**Examples**  “Example 3: Rotating Plot Symbols through the Color List” on page 473
MODE=PROTECT | RESERVE | SHARE

specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements. If the space required to display the legend exceeds that of the space required to display the graph, a warning is issued and the legend is suppressed. The MODE= option can take one of these values:

PROTECT
draws the legend in the procedure output area, but a blanking area surrounds the legend, preventing other graphics elements from being displayed in the legend. (A blanking area is a protected area in which no other graphics elements are displayed.)

RESERVE
takes space for the legend from the procedure output area, thereby reducing the amount of space available for the graph. If MODE=RESERVE is specified in conjunction with OFFSET=, the legend can push the graph off the graphics output area. RESERVE is valid only when POSITION=OUTSIDE. If POSITION=INSIDE is specified, a warning is issued and MODE= value is changed to PROTECT.

SHARE
draws the legend in the procedure output area. If the legend is positioned over elements of the graph itself, both graphics elements and legend elements are displayed.

By default, MODE=RESERVE unless POSITION=INSIDE. In this case, the default changes to MODE=PROTECT.

If the MODE=PROTECT option is used with labels on the midpoint axis, the axis labels might overlap the legend. In that case, if a device-resident font is used as the label font, the device-resident font for each label that overlaps the legend is replaced with a SAS/GRAPH font. This includes the axis label and the tick mark labels. Because device-resident fonts do not support clipping, a SAS/GRAPH font must be substituted in that case. To correct an overlap condition, you can use the LABEL=None and VALUE=None options on your midpoint axis statement to suppress the axis labels. Another choice is to use the positioning options on your LEGEND statement to reposition the legend.

Restriction  Not supported by Java and ActiveX

See  “Positioning the Legend” on page 393

OFFSET=<<x><,y><units>> | (<<x><units>>>><,y><units>>>)
specifies the distance to move the entire legend. x is the number of units to move the legend right (positive numbers) or left (negative numbers). y is the number of units to move the legend up (positive numbers) or down (negative numbers).

To set only the x offset, specify one value, with or without a following comma:

```
offset=(4 cm,)
```

To set both the x and y offset, specify two values, with or without a comma separating them:

```
offset=(2 pct, 4 pct)
```

To set only the y offset, specify one value preceded by a comma:

```
```
The `OFFSET=` option is usually used in conjunction with the `POSITION=` option to adjust the position of the legend. Moves are relative to the location specified by the `POSITION=` option, with `OFFSET=(0,0)` representing the initial position. You can also apply the `OFFSET=` option to the default legend position.

The `OFFSET=` option is unnecessary with the `ORIGIN=` option because the `ORIGIN=` option explicitly positions the legend and requires no further adjustment. However, if you specify both options, the `OFFSET=` values are added to the `ORIGIN=` values, and the LEGEND is positioned accordingly.

**Restriction**
Not supported by Java and ActiveX

**See**
“POSITIONing the Legend” on page 393

“ POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)” on page 384

**ORDER=(value-list) | DESCENDING**
selects or orders the legend values that appear in the legend. DESCENDING specifies that the legend values appear in the legend in reverse of the default order, sorted from maximum to minimum value. Alternatively, use `value-list` to select only those legend values that you want to display and by their placement in the list, order where they appear in the legend. The way you specify `value-list` depends on the type of variable that generates the legend:

- For numeric variables, `value-list` is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - \( n < ...n > \)
  - \( n \text{ TO } n < \text{BY increment}> \)
  - \( n < ...n > \text{ TO } n < \text{BY increment}> <n < ...n> > \)

  If a numeric variable has an associated format, the specified values must be the unformatted values.

- For character variables, `value-list` is a list of unique character values enclosed in quotation marks and separated by blanks:
  - "value-1"<" value-2" ... " value-n”>

  If a character variable has an associated format, the specified values must be the formatted values.

For a complete description of `value-list`, see the ORDER= option in the “AXIS Statement” on page 345.

Even though the ORDER= option controls whether a legend value is displayed and where it appears, the VALUE= option controls the text that the legend value displays.

**Restrictions**
Not supported by Java and ActiveX

Up to 256 characters in a variable value might be displayed.

**ORIGIN=(<x><y><units> | (<x <units >>><y <units>>>)**
specifies the \( x \) and \( y \) coordinates of the lower left corner of the legend box. The `ORIGIN=` option explicitly positions the legend anywhere on the graphics output area. It is possible to run a legend off the page or overlay the graph.
To set only the \( x \) coordinate, specify one value, with or without a following comma:

\[
\text{origin}=(4 \text{ cm},)
\]

To set both the \( x \) and \( y \) coordinates, specify two values, with or without a comma separating them:

\[
\text{origin}=(2 \text{ pct}, 4 \text{ pct})
\]

To set only the \( y \) coordinate, specify one value preceded by a comma:

\[
\text{origin}=(,3 \text{ pct})
\]

The \texttt{ORIGIN=} option overrides the \texttt{POSITION=} option if both are used. Although using the \texttt{OFFSET=} option with the \texttt{ORIGIN=} option is unnecessary, if the \texttt{OFFSET=} option is also specified, it is applied after the \texttt{ORIGIN=} request has been processed.

**Restriction** Not supported by Java and ActiveX

**See** “Positioning the Legend” on page 393

\[
\texttt{POSITION=}(<\text{BOTTOM | MIDDLE | TOP}<\text{ LEFT | CENTER | RIGHT}<\text{ OUTSIDE | INSIDE}>)
\]

positions the legend on the graph. Values for \texttt{POSITION=} are as follows:

- \texttt{OUTSIDE} or \texttt{INSIDE}
  - specifies the location of the legend in relation to the axis area.
- \texttt{BOTTOM} or \texttt{MIDDLE} or \texttt{TOP}
  - specifies the vertical position.
- \texttt{LEFT} or \texttt{CENTER} or \texttt{RIGHT}
  - specifies the horizontal position.

By default, \texttt{POSITION}=(\texttt{BOTTOM CENTER OUTSIDE}). You can change one or more settings. If you supply only one value, then the parentheses are not required. If you specify two or three values and omit the parentheses, \texttt{SAS/GRAPH} accepts the first value and ignores the others.

Once you assign the initial legend position, you can adjust it with the \texttt{OFFSET=} option.

The \texttt{ORIGIN=} option overrides the \texttt{POSITION=} option. The value of the \texttt{MODE=} option can affect the behavior of the \texttt{POSITION=} option.

**Restrictions** Partially supported by Java

Starting with \texttt{SAS 9.4M5}, when there is not enough room in the output to display a long label text, the variable name is displayed instead. However, when \texttt{POSITION=}\texttt{INSIDE} and the legend label is too long, the legend is suppressed. In either case, the SAS log displays a warning message.

**Note** The Java applet defaults to \texttt{BOTTOM-CENTER} and supports all possible combinations of \texttt{BOTTOM | MIDDLE | TOP} with \texttt{LEFT | CENTER | RIGHT} except for \texttt{MIDDLE-CENTER} (which would overwrite the map.) The Java applet does not support \texttt{INSIDE}.

**See** “\texttt{OFFSET=}(<\texttt{x}<,<\texttt{y}>)<\texttt{units}> | (<\texttt{x}<\texttt{units}>>,<\texttt{y}<\texttt{units}>>)” on page 382

“\texttt{MODE=}\texttt{PROTECT | RESERVE | SHARE}” on page 382
REPEAT=1 | 2 | 3

specifies how many times the plot symbol is repeated in the legend. Valid values are 1 to 3. The default value is 3.

Restriction Not supported by Java or ActiveX

ROWMAJOR | COLMAJOR

specifies the arrangement of legend entries when there are multiple rows and multiple columns. Specify the ROWMAJOR option (the default) to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange the entries starting from top to bottom, and then left to right.

See “ACROSS=number-of-columns” on page 378

“DOWN=number-of-rows” on page 379

SHAPE=BAR(width<units>,height<units>) <units> | LINE(length)<units> | SYMBOL(width<units>,height<units>) <units>

specifies the size and shape of the legend values displayed in each legend entry. The SHAPE= value that you specify depends on which procedure generates the legend.

BAR(width,height)<units> | (width<units>,height<units>) <units>

is used with the GCHART procedure, with the GPLOT procedure if you use the AREAS= option, and with the GCONTOUR procedure if you use the PATTERN option. Each legend value is a bar of the specified width and height.

Default width is 5, height is 0.8, and units are CELLS.

Note You can specify the width and height in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

Tip You can specify units for the width,height pair or for either or both of the individual coordinates.

LINE(length)<units>

is used with the GPLOT and GCONTOUR procedures. Each legend value is a line of the length that you specify. Plotting symbols are omitted from the legend values. You can specify units for length.

Default length is 5 and units are CELLS.

Note You can specify the length in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

SYMBOL (width,height)<units> | (width<units>,height<units>) <units>

is used with the GPLOT procedure. Each legend value (not each symbol), is the width and height that you specify. For example, this specification produces legend values like the ones in Figure 24.5 on page 386 (a):

shape=symbol(.5,.5)

This specification produces legend values like the ones in Figure 24.5 on page 386 (b):

shape=symbol(2,.5)
Figure 24.5  Legend Values Produced with SHAPE= SYMBOL

Default  width is 5, height is 1, and units are CELLS.

Restriction  Not supported by Java and ActiveX

Note  You can specify the width and height in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

Tips  You can specify units for the width, height pair or for either or both of the individual coordinates.

Use the SHAPE=SYMBOL option when using the SYMBOL statement option INTERPOL=NEEDLE to modify the width or height values of the legend entry separately from the size of the GPLOT procedure’s plot needles.

**SPACE=**value-spacing

specifies the amount of space between individual legend values along a horizontal baseline. Value-spacing can be any nonnegative number, including decimal values. Units are only character cells. You cannot specify a unit of measure, such as inches or percent. The LEGEND statement calculates spacing based on the available display area, and the number and width of legend entries. If the spacing requested exceeds the display area, the legend values and their descriptions are rearranged to fit.

The SPACE= option is ignored if the following is true:

- SPACE=0 is specified.
- The specified spacing is requested in conjunction with vertically stacked columns.

Restriction  Not supported by Java or ActiveX

**SPLIT=**“split-char(s)"

specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines. Split-char(s) can be any character value that can be specified in a SAS character variable. Do not delimit when specifying multiple split characters. When the LEGEND statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the legend text description contains the string Berlin, Germany/Europe, and you specify SPLIT=“,”/”. The legend breaks the text string into top-aligned lines as follows:

Berlin
Germany
Europe
If the LEGEND statement does not encounter a specified split character, no break in
the text occurs, and no warning or error is issued.

Restrictions

Not supported by Java and ActiveX

Not supported by the GRADAR procedure

Up to 31 split characters are supported. If you specify more than 31
characters, only the first 31 are honored.

Note

The split characters themselves are not displayed.

Example

“Example 2: Creating a Drill-Down HTML Presentation for the Web”
on page 204

VALUE=(text-argument(s)) | NONE
 modifies the legend value descriptions. Text-argument(s) defines the appearance or
the text of the value descriptions. By default, value descriptions are the values of the
variable that generates the legend or an associated format value. Numeric values are
right-justified and character values are left-justified.

NONE suppresses the value descriptions although the legend values (bars, lines, and
so on) are still displayed. Text-argument(s) can be one or more of these:

"text-string"
 provides up to 256 characters of text for the value description. Enclose each
string in quotation marks. Separate multiple strings with blanks. Value
description text that is too long to fit in a legend area can result in the entire value
not being displayed.

Specified text strings are assigned to the legend values in order. If you submit
only one string, only the first legend entry uses the value of that string. If you
specify multiple strings, the first string is the text for the first entry; the second
string is the text for the second entry; and so on. For example, this specification
produces legend entries like those shown in Figure 24.6 on page 387:

value=("1986" "1987"
 *1988")

Figure 24.6  Specifying Value Descriptions with the VALUE= Option

In addition, if you have a BY statement and you specify the variable that it
names, you can embed one or both of the following in the string:

#BYVAL.n | #BYVAL(BY-variable-name)
 substitutes the current value of the specified BY variable for #BYVAL in the
text string and displays the value produced by the statement. Specify the
variable with one of these:

n
 specifies which variable in the BY statement #BYVAL should use. The
value of n indicates the position of the variable in the BY statement. For
example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
 names the BY variable. For example, #BYVAL(YEAR) specifies the BY
variable, YEAR. Variable-name is not case sensitive.
Examples

“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

“Example 8: Combining Graphs and Reports in a Web Page” on page 495

#BYVARn | #BYVAR(BY-variable-name)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n specifies which variable in the BY statement #BYVAR should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. Variable-name is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See “Substituting BY Line Values in a Text String” on page 395

text-description-suboption modifies a characteristic such as the font, color, or size of the text string(s) that follows it. Text-description-suboption can be as follows:

• ANGLE=degrees
• COLOR=text-color
• FONT=font | NONE
• HEIGHT=text-height <units >
• JUSTIFY=LEFT | CENTER | RIGHT
• POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
• ROTATE=degrees
• TICK=n

Place text description suboptions before the text strings that they modify. Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

To order or select legend entries, use the ORDER= option.

Restrictions Partially supported by Java and ActiveX

ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

NONE is not supported by Java or ActiveX
BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

See “Text Description Suboptions” on page 389

“ORDER=(value-list) | DESCENDING” on page 383

Text Description Suboptions

ANGLE=degrees
specifies the angle of the legend label or legend value description text with respect to the horizontal. A positive value for degrees moves the text counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal).

Note: Some settings of ANGLE= in the LEGEND statement might result in undesirable text positioning.

Alias A=

Restriction Not supported by Java and ACTIVEX

See “ROTATE=degrees” on page 391

COLOR=text-color
specifies the color of the text. If you omit the COLOR= suboption, a color specification is searched for in this order:

1. the CTEXT= option for the procedure
2. the CTEXT= option in a GOPTIONS statement
3. the color of the default style

Alias C=

FONT=font | NONE
specifies the font for the text. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for information about specifying fonts. If you omit the FONT= suboption, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default style font, NONE

Alias F=

Restriction ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

HEIGHT=text-height <units>
specifies the height of the text characters in the number of units. By default, HEIGHT=1 CELL. If you omit the HEIGHT= suboption, a text height specification is searched for in this order:

1. the HTEXT= option in a GOPTIONS statement
2. the height specified by the default style

Alias H=
JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text. The default for character variables is
JUSTIFY=LEFT. The default for numeric variables is JUSTIFY=RIGHT.
Associating a character format with a numeric variable does not change the default
justification of the variable.

You can use the JUSTIFY= suboption to print multiple lines of text by repeating the
suboption before the text string for each line. For example, this statement produces a
legend label and value descriptions like those shown in Figure 24.7 on page 390:

```sas
legend label=(justify=c "Distribution"
            justify=c "Centers")
value=(tick=1 justify=c "Portland,"
      justify=c "Main"
      tick=2 justify=c "Paris,"
      justify=c "France"
      tick=3 justify=c "Sydney,"
      justify=c "Australia");
```

Figure 24.7 Specifying Multiple Lines of Text with the JUSTIFY= Suboption

Place text description suboptions before the text strings that they modify.

Alias  J=L | C | R

See  “TICK=n” on page 392

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 24.8 on page 391 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:

```sas
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 24.8 Using the POSITION= Suboption with Multiple-line Legend Labels

In addition, specifying POSITION=RIGHT mirrors the effect of POSITION=LEFT, and specifying POSITION=BOTTOM mirrors the effect of POSITION=TOP.

Restriction Not supported by Java. Partially supported by ActiveX

**ROTATE=degrees**

specifies the angle at which each character of text is rotated with respect to the baseline of the text string. A positive value for degree rotates the character counterclockwise; a negative value moves it clockwise. By default, ROTATE=0 (parallel to the baseline).

*Note:* Some settings of ROTATE= in the LEGEND statement might result in undesirable text positioning.

**Alias** R=

**Restriction** Not supported by Java and ACTIVEX

**See** “ANGLE=degrees” on page 389
**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:

```plaintext
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:

```plaintext
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:

```plaintext
value=(color=red font=swiss tick=2 color=blue)
```

The text of all three entries would use the Swiss font; the first and third entries would be red and only the second entry would be blue.

**Alias** T=

### Details

**Description: LEGEND Statement**

LEGEND statements specify the characteristics of a legend but do not create legends. The characteristics are as follows:

- the position and appearance of the legend box
- the text and appearance of the legend label
- the appearance of the legend entries, including the size and shape of the legend values
- the text of the labels for the legend values

LEGEND definitions are not automatically applied when a procedure generates a legend. Instead, they must be explicitly assigned with a LEGEND= option in the appropriate procedure statement.

The following figure illustrates the terms associated with the various parts of a legend.

**Figure 24.9  Parts of a Legend**
Using Text Description Suboptions
Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=albany amt height=4 "Weight"
      justify=right height=3 "(in tons)")
```

FONT=ALBANY applies to both Weight and (in tons). HEIGHT=4 affects Weight, but is respesecified as HEIGHT=3 for (in tons). JUSTIFY=RIGHT affects only (in tons).

Using the LEGEND Statement
LEGEND statements can be located anywhere in your SAS program. They are global and remain in effect until canceled or until you end your SAS session. LEGEND statements are not applied automatically, and must be explicitly assigned by an option in the procedure that uses them.

You can define up to 99 different LEGEND statements. If you define two LEGEND statements of the same number, the most recently defined statement replaces the previously defined statement of the same number. A LEGEND statement without a number is treated as a LEGEND1 statement.

Cancel individual LEGEND statements by defining a LEGEND statement of the same number without options (a null statement):

```
legend4;
```

Canceling one LEGEND statement does not affect any other LEGEND definitions. To cancel all current LEGEND statements, use RESET= in a GOPTIONS statement:

```
goptions reset=legend;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current LEGEND definitions as well as other settings.

To display a list of current LEGEND definitions in the SAS LOG window, use the GOPTIONS procedure with the LEGEND option:

```
proc goptions legend nolist;
run;
```

Positioning the Legend

How to Position a Legend
By default, the legend shares the procedure output area with the procedure output, such as a map or bar chart. See “How Graphics Elements Are Placed in the Graphics Output Area” on page 76. However, several LEGEND statement options enable you to position a legend anywhere on the graphics output area and even to overlay the procedure output. This section describes these options and their effect on each other.

Positioning the Legend on the Graphics Output Area
There are two ways that you can position the legend on the graphics output area:

- Describe the general location of the legend with the POSITION= option. If necessary, fine-tune the position with the OFFSET= option.
- Position the legend explicitly with the ORIGIN=option.

Using POSITION= and OFFSET=
The values of the POSITION= option affect the legend in two ways:
• OUTSIDE and INSIDE determine whether the legend is located outside or inside the axis area.

• BOTTOM or MIDDLE or TOP (vertical position) and LEFT or CENTER or RIGHT (horizontal position) determine where the legend is located in relation to its OUTSIDE or INSIDE position.

Figure 24.10 on page 394 shows the legend positions inside the axis area.

**Figure 24.10  Legend Positions inside the Axis Area**

Figure 24.11 on page 394 shows legend positions outside the axis area.

**Figure 24.11  Legend Positions outside the Axis Area**

The default combination is POSITION=(BOTTOM CENTER OUTSIDE). The combination (OUTSIDE MIDDLE CENTER) is not valid.

Use OFFSET=(x,y) to adjust the position of the legend specified by the POSITION= option. The x value shifts the legend either left or right and the y value shifts the legend either up or down.

The offset values are always applied after the POSITION= request. For example, if POSITION=(TOP RIGHT OUTSIDE), the legend is located in the upper right corner of the graphics output area. If OFFSET=(0,0) is specified, the legend does not move. If OFFSET=(−5,−8)CM, the legend moves 5 centimeters to the left and 8 centimeters down.

**Using ORIGIN=**

Use ORIGIN=(x,y) to specify the coordinates of the exact location of the lower left corner of the legend box. Because ORIGIN=(0,0) is the lower left corner of the graphics output area, the values of x and y must be positive. If you specify negative values, a warning is issued and the default value is used.

**Relating Legends to Other Graphics Elements**

By default, the legend is inside the procedure output area and the space that it occupies reduces the size of the graph itself. If a choice must be made between displaying the legend or the graph, the legend is suppressed. To control how the legend relates to the other elements of the graph, use the MODE= option. These are values for the MODE= option:
• RESERVE reserves space for the legend outside the axis area and moves the graph to make room for the legend. This is the default setting and is valid only when POSITION=OUTSIDE.

• PROTECT prevents the legend from being overwritten by the procedure output. PROTECT blanks out graphics elements, allowing only legend elements to be displayed in the legend's space.

• SHARE displays both graphics elements and legend elements in the same space. This setting is usually used when the legend is positioned inside the axis area. SHARE is useful when the graph has a space that the legend can fit into.

Interactions between POSITION= and MODE=
You cannot specify both POSITION=INSIDE and MODE=RESERVE because MODE=RESERVE assumes that the legend is outside the axis area, and POSITION=INSIDE positions the legend inside the axis area. Therefore, when you specify POSITION=INSIDE, change the value of the MODE= option to SHARE or PROTECT. Otherwise, SAS/GRAPH issues a warning and automatically changes the MODE= value to PROTECT.

Creating Drop Shadows and Block Effects
You can produce a drop shadow or a three-dimensional block effect behind the legend. Use the CSHADOW= or CBLOCK= option in the LEGEND statement in conjunction with the graphics option OFFSHADOW=(x,y).

The value of x determines how far the shadow or block extends to the right (positive numbers) or to the left (negative numbers) of the legend. The value of y determines how far the shadow or block extends above (positive numbers) or below (negative numbers) the legend. If OFFSHADOW=(0,0) is specified, the shadow or block is not visible.

By default, OFFSHADOW=(0.0625, -0.0625) IN. That is, the shadow or block extends 1/16th of an inch to the right and 1/16th of an inch below the legend.

Substituting BY Line Values in a Text String
The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

• The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.

• There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY
NOTE Statement

Writes lines of text in the output.

**Type:** Local

**Restriction:** Not supported by Java and ActiveX

**Tip:** By drawing a white box in back of a note, that boxed note can appear on top of the graph instead of behind it. Here is an example of code that outputs a note on top of a map and a note on top of a chart.

```sas
filename odsout '.';

goptions device=png;
goptions cback=white;

ODS LISTING CLOSE;
ODS HTML path=odsout body="map01.htm" style=htmlblue;

title "Note Text On a Map";
pattern1 v=s color=tan;

proc gmap data=mapsgfk.us map=mapsgfk.us;
note move=(2,32)pct font="albany amt/bold" height=4pct
  color=white box=1 blank=yes
  color=black "Note with 'box' behind it displays on top of map";
id state;
choro state / levels=1 nolegend;
run;

title "Note Text On a Chart";
proc gchart data=maps.us;
note move=(2,50)pct font="albany amt/bold" height=4pct
  color=white box=1 blank=yes
  color=black "Note with 'box' behind it displays on top of chart";
hbar state / type=freq levels=7;
run;
quit;
ODS HTML CLOSE;
ODS LISTING;
```

**See:** “TITLE, FOOTNOTE, and NOTE Statements” on page 447

**Syntax**

```
NOTE <text-arguments(s)>;
```
ODS HTML Statement

Opens or closes the HTML destination.

**Used by:**  
GANNO, GAREABAR, GBARLINE, GCART, GCONTOUR, GPLOT, GRADAR, GREPLAY, GSLIDE, G3D, G3GRID

**Requirement:**  
On mainframes, either GPATH= or PATH= is required.

**Syntax**

```
ODS HTML (<ID=> identifier>) <action | option(s)>;
```

**Details**

**Description: ODS HTML Statement**

This section describes the ODS HTML statement as it relates to SAS/GRAPH procedures. For complete information about the ODS HTML statement, see the *SAS Output Delivery System: User’s Guide*.

The ODS HTML statement opens or closes the HTML destination. This destination is the default output destination for the Windows and UNIX operating systems. When the destination is open, the procedure produces output that is written in Hypertext Markup Language in the form of an HTML file. If no device is specified, SAS/GRAPH by default, creates a PNG file containing the graph. The HTML file references the PNG file in order to display the graph in a web page.

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 about DEVICE=JAVAMETA see “Developing Web Presentations for the Metaview Applet” in *SAS/GRAPH: Java Applets and ActiveX Control User’s Guide*.

You can also use the DEVICE=JAVA and DEVICE=ACTIVEX options to create interactive graphics presentations for the web.

SAS/GRAPH adds data tip text to some graphs depending on the device specified. These data tips are generated by default using the values of fields in a SAS data set. You can specify the DESCRIPTION= option on the SAS/GRAPH procedure to change or remove the data tip text. For more information about using data tips see “Data Tips for Web Presentations” on page 191.

The FILE= option identifies the file that contains the HTML version of the procedure output. With SAS/GRAPH, the body file contains references to the graphs. If DEVICE=PNG, the graphs are stored in separate PNG files. When you view the body file in a browser, the graphs are automatically displayed. By default with ODS processing, both the HTML and PNG files are stored in the current WORK library. To specify a destination for all the HTML and PNG files, use the PATH= option. You can store the PNG files in a different location than the HTML files. Use the GPATH= option to specify a location for the PNG files, and the PATH= option to specify the location of the HTML files. In both cases, the destination must be an aggregate storage location.
**Anchors**

ODS HTML automatically creates an *anchor* for every piece of output generated by the SAS procedures. An anchor specifies a particular location within an HTML file. In SAS/GRAPH, an anchor usually defines a link target such as a graph whose location is defined in an IMG element.

In order for the links from the contents, page, or frame file to work, each piece of output in the body files must have a unique anchor to link to. The anchor for the first piece of output in a body file acts as the anchor for that file. These anchors are used by the frame and contents files, if they are created, to identify the targets for the links that ODS HTML automatically generates. For more information about using anchors with the ODS HTML statement, see the *SAS Output Delivery System: User’s Guide*.

You can specify a name for an HTML anchor with the following code:

```plaintext
ods html anchor="string";
```

This enables you to link directly to that identifying name.

---

**PATTERN Statement**

Defines the characteristics of patterns used in graphs.

**Used by:**
- GAREABAR, GBARLINE, GCHART, GCONTOUR, GPLOT, SYMBOL statement,
- Annotate facility

**Type:**
- Global

**Syntax**

```
PATTERN<1 ...255>
<COLOR=pattern-color | _style_>
<IMAGE=fileref | "external-file">
<IMAGESTYLE=TILE | FIT>
<REPEAT=number-of-times>
<V ALUE=bar/block-pattern | plot-pattern | pie/star-pattern>;
```

**Optional Arguments**

**COLOR=pattern-color | _style_**

specifies the color of the fill. *Pattern-color* is any SAS/GRAPH color name. The _STYLE_ value specifies the appropriate color based on the current style. See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313.

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 958.

No color can be specified for a PATTERN statement, that is, neither the COLOR= nor the CPATTERN= option is used. In this case the PATTERN statement rotates the specified fill through each color in the color list before the next PATTERN statement is used.
Alias    C=

Restriction   Partially supported by Java and ActiveX

Note   ActiveX assigns colors in a different order from Java, so the same data can appear differently with those two drivers.

See   “Working with PATTERN Statements” on page 408

Example   “Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488

CAUTION   Omitting the COLOR= option in a PATTERN statement can cause the PATTERN statement to generate multiple PATTERN definitions.

IMAGE=fileref | "external-file"

specifies an image file that is used to fill one or more bars of a bar chart. The image file can be generated by the HBAR and VBAR statements of the GCHART procedure. And, in some cases, the image file can be generated by the HBAR3D and VBAR3D statements of the GCHART procedure. The format of the external file specification varies across operating environments. See the companion reference for your specific operating environment.

Restriction   Partially supported by Java and ActiveX

Notes   When you specify an image file to fill a bar, the bar is not outlined. Also, the COLOR= and VALUE= options are ignored.

If an image is specified in a PATTERN statement that is used with another type of chart, then the PATTERN statement is ignored and default pattern rotation is affected. For example, you can submit a PIE statement when an image has been specified in a PATTERN statement. In this case the default fill pattern is used for the pie slices, with each slice in the pie displaying the fill pattern in the same color.

For DEVICE=ACTIVEX and DEVICE=ACTXIMG, if you do not specify a pathname to the image, then the ActiveX control searches a predefined list of locations to try to find the image. If all else fails, the ActiveX control looks for the image on the web. It is recommended that you specify the pathname to the image.

For DEVICE=ACTIVEX and DEVICE=ACTXIMG, the IMAGE= option works for the VBAR and HBAR statements as well as the VBAR3D and HBAR3D statements. For all other devices, the IMAGE= option works only for the VBAR and HBAR statements.

See   “Displaying Images on Data Elements” on page 336 for related information

“IMAGESTYLE=TILE | FIT” on page 399

IMAGESTYLE=TILE | FIT

specifies how the image specified in the IMAGE= option is to be applied to fill a bar in a bar chart. The TILE value repeats the image as needed to fill the bar. The FIT value stretches a single instance of the image to fill the bar.
Default  TILE
Restriction  Partially supported by 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.

The behavior of the REPEAT= option depends on the color specification:

- If you use both the COLOR= and REPEAT= options in a PATTERN statement, the pattern is repeated the specified number of times in the specified color. The fill can be either the default solid or a fill specified with the VALUE= option.

- You can use the CPATTERN= option in a GOPTIONS statement to specify a single pattern color. Use the REPEAT= option either alone or with the VALUE= option in a PATTERN statement. The resulting hatch pattern is repeated the specified number of times.

- You can omit both the COLOR= and CPATTERN= options and use the REPEAT= option. If you use the REPEAT= option alone, it generates default solids. Or you can use the REPEAT= option in conjunction with the VALUE= option in a PATTERN statement. The resulting pattern is rotated through each color in the color list. The entire group generated by this cycle is then repeated the number of times specified in the REPEAT= option. Thus, the total number of patterns produced depends on the number of colors in the current color list.

Using REPEAT= with a null value cancels the repetition specified in a previous PATTERN statement of the same number without affecting the values of other options.

**Alias** \( R \)

| Default | REPEAT=1 |
|------------------------|

**Restriction** Partially supported by Java and ActiveX

**See** “Understanding Pattern Sequences” on page 410

**VALUE=bar/block-pattern**

Patterns are specified for the following:

- bar charts produced by the HBAR, HBAR3D, VBAR, and VBAR3D statements in the GCHART procedure including two-dimensional and three-dimensional bar shapes.

- the front surface of blocks in block charts produced by the BLOCK statement in the GCHART procedure.

Values for **bar/block-pattern** are as follows:

**EMPTY**

an empty pattern. Neither the Java applet nor the ActiveX control supports EMPTY.

**Alias** \( E \)

**SOLID**

a solid pattern (the only valid value for three-dimensional charts).

**Alias** \( S \)
**style <density>**
a shaded pattern. *Style* specifies the direction of the lines:

- **L** specifies left-slanting lines.
- **R** specifies right-slanting lines.
- **X** specifies crosshatched lines.

*Density* specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

**Note:** *style<density>* is not supported by the Java or ActiveX device drivers.

Figure 24.12 on page 401 shows all of the patterns available for bars and blocks.

**Figure 24.12  Bar and Block Patterns**

If no valid patterns are available, default bar and block fill patterns are selected in this order:

1. SOLID
2. X1–X5
3. L1–L5
4. R1–R5

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

**Alias**  
V=

**Restriction**  
Partially supported by Java and ActiveX

**VALUE=plot-pattern**
specifies patterns for the following:

- contour levels in contour plots produced by the GCONTOUR procedure
• areas under curves in plots produced by the AREAS= option in the PLOT statement in the GPLOT procedure

Values for plot-pattern are as follows:

**MEMPTY**
specifies an empty pattern.

**Aliases**

**ME**

EMPTY is a valid alias for the GCONTOUR and GPLOT procedures.

**E** is a valid alias for the GCONTOUR and GPLOT procedures.

**MSOLID**
specifies a solid pattern.

**Aliases**

**MS**

SOLID is also a valid alias.

S is also a valid alias.

**Mdensity <style <angle> >**
specifies a shaded pattern.

*Density* specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

*Style* specifies the type of the pattern lines:

**N** specifies parallel lines (the default).

**X** specifies crosshatched lines.

*Angle* specifies the angle of the pattern lines. Specify a number from 0 to 360. The number specifies the degrees at which the parallel lines are drawn, measured from the horizontal. By default, *angle* is 0 (lines are horizontal).

**Note** *Mdensity=style<angle>* is not supported by the Java or ActiveX device drivers.
Figure 24.13 on page 403 shows some typical plot patterns.

*Figure 24.13  Plot Patterns*

If no valid patterns are available, default plot fill patterns are selected in this order:

1. MSOLID
2. M2N0
3. M2N90
4. M2X45
5. M4N0
6. M4N90
7. M4X90

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

**Alias**

**V=**

**Restriction**

Partially supported by 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 as follows:

**PEMPTY**

specifies an empty pattern. EMPTY or E are also valid aliases.

**Alias**

**PE**

**PSOLID**

specifies a solid pattern. SOLID or S are also valid aliases.
Alias  PS

P density <style <angle>>
specifies a shaded pattern.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N  specifies parallel lines (the default).

X  specifies crosshatched lines.

Angle specifies the angle of the pattern lines. Specify a number from 0 to 360. The number specifies the angle of the lines, measured in degrees from perpendicular to the radius of the slice. By default, angle is 0.

Note  P density<style<angle>> is not supported by the Java or ActiveX device drivers.

The FILL= option in the PIE and STAR statements in the GCHART procedure overrides VALUE=.

Figure 24.14 on page 404 shows some typical pie and star patterns.

Figure 24.14  Pie and Star Patterns
If no valid patterns are available, default pie and star fill patterns are selected in this order:

1. PSOLID
2. P2N0
3. P2N90
4. P2X45
5. P4N0
6. P4N90
7. P4X90

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias   V=

Restriction Partially supported by Java and ActiveX

Note If you use hatch patterns and request a legend instead of slice labels, the patterns in the slices are oriented to be visually equivalent to the legend.

Details

Description: PATTERN Statement

PATTERN statements create PATTERN definitions that define the color and type of area fill for patterns used in graphs. These are the procedures and the graphics areas that they create that use PATTERN definitions:

GCHART  
- color, fill pattern, or image for the bars in two-dimensional bar charts; color and fill pattern for the segments of three-dimensional bar charts, pie charts, and star charts.

GCONTOUR  
- contour levels in contour plots

GPLOT  
- areas beneath or between plotted lines

In addition, the SYMBOL statement and certain Annotate facility functions and macros can use pattern specifications. For details, see the “SYMBOL Statement” on page 412 and Chapter 27, “Using Annotate Data Sets,” on page 635.

You can use the PATTERN statement to control the fill and color of a pattern, and whether the pattern is repeated. There are various types of patterns:

- bar and block patterns
- plot patterns
- pie and star patterns

Pattern fills can be solid or empty, or composed of parallel or crosshatched lines. For two-dimensional bar charts, the PATTERN statement can specify images to fill horizontal or vertical bars. In addition, you can specify device-dependent hardware patterns for polygon, rectangle, and pie fills on devices that support hardware patterns.
If you do not create PATTERN definitions, SAS/GRAPH software generates them as needed and assigns them to your graphs by default. Generally, the default behavior is to rotate a solid pattern through the current color list. For details, see “About Default Patterns” on page 407.

Using the PATTERN Statement

How PATTERN Definitions Are Generated
PATTERN statements can be located anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session.

You can define up to 255 different PATTERN statements. A PATTERN statement without a number is treated as a PATTERN1 statement.

PATTERN statements generate one or more PATTERN definitions, depending on how the COLOR=, VALUE=, and IMAGE= options are used. For information about PATTERN definitions, see “Working with PATTERN Statements” on page 408, as well as the descriptions of “COLOR=pattern-color | _style_” on page 398, “IMAGE=fileref | external-file” on page 399, and “VALUE=plot-pattern” on page 401.

PATTERN definitions are generated in the order in which the statements are numbered, regardless of gaps in the numbering or the statement's position in the program. Although it is common practice, you do not have to start with PATTERN1, and you do not have to use sequential statement numbers.

PATTERN definitions are applied automatically to all areas of the graphics output that require patterns. When assigning PATTERN definitions, SAS/GRAPH starts with the lowest-numbered definition with an appropriate fill specification or with no fill specification. It continues to use the specified patterns until all valid PATTERN definitions have been used. Then, if more patterns are required, SAS/GRAPH returns to the default pattern rotation, but continues to outline the areas in the same color as the fill.

Altering or Canceling PATTERN Statements
PATTERN statements are additive. You can define a PATTERN statement and later submit another PATTERN statement with the same number. The new PATTERN statement redefines or cancels only the options that are included in the new statement. Options not included in the new statement are not changed and remain in effect. For example, assume you define PATTERN4 as follows:

```
pattern4 value=x3 color=red repeat=2;
```

This statement cancels only REPEAT= without affecting the rest of the definition:

```
pattern4 repeat=;
```

Add or change options in the same way. This statement changes the color of the pattern from red to blue:

```
pattern4 color=blue;
```

After all these modifications, PATTERN4 has these characteristics:

```
pattern4 value=x3 color=blue;
```

Cancel individual PATTERN statements by defining a PATTERN statement of the same number without options (a null statement):

```
pattern4;
```

Canceling one PATTERN statement does not affect any other PATTERN definitions. To cancel all current PATTERN statements, use the RESET= option in a GOPTIONS statement:
goptions reset=pattern;

Specifying RESET=GLOBAL or RESET=ALL cancels all current PATTERN definitions as well as other settings.

To display a list of current PATTERN definitions in the LOG window, use the GOPTIONS procedure with the PATTERN option:

proc goptions pattern nolist;
run;

About Default Patterns

How SAS/GRAPH Generates and Assigns PATTERN Definitions

When a procedure produces a graph that needs one or more patterns, SAS/GRAPH does one of the following actions:

- automatically generates the appropriate default patterns and outlines to fill the areas
- uses patterns, colors, and outlines that are defined by PATTERN statements, graphics options, and procedure options.

In order to understand how SAS/GRAPH generates and assigns patterns defined with PATTERN statements, it is helpful to understand how it generates and assigns default patterns. The following sections describe the default pattern behavior for all procedures. See “Working with PATTERN Statements” on page 408 for details about defining patterns.

How Default Patterns and Outlines Are Generated

In general, the default pattern that SAS/GRAPH uses is a solid fill. The default colors are determined by the current style and the device.

SAS/GRAPH uses default patterns when no PATTERN statements are defined. The default colors are determined by the current style and the device.

Because the system option-GSTYLE-is in effect by default, the procedure uses the style's default bar and block fill colors, plot line colors, widths, symbols, patterns, and outline colors when producing output. Specifically, SAS/GRAPH uses the default values when you do not specify any of the following:

- any PATTERN statements
- the CPATTERN= graphics option
- the COLORS= graphics options (that is, you use the device's default color list and it has more than one color)
- the COUTLINE= option in the action statement

If all of these conditions are true, then SAS/GRAPH performs the following operations:

- selects the first default fill for the appropriate pattern, which is always solid. It rotates once through the list of colors available in the current style, generating one solid pattern for each color. If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, then the procedure uses all the colors in the list to generate the patterns.
- uses the style's outline color to outline every patterned area.

If a procedure needs additional patterns, SAS/GRAPH selects the next default pattern fill appropriate to the graph and rotates it through the color list, skipping the foreground
color as before. SAS/GRAPH continues in this fashion until it has generated enough patterns for the chart.

**Things That Affect Default Patterns**
Changing any of these conditions can change or override the default behavior:

- If you specify a color list with the COLORS= option in a GOPTIONS statement and the list contains more than one color, SAS/GRAPH rotates the default fills, beginning with SOLID, through that list. In this case, it uses every color, even if the foreground color is black (or white). The default outline color remains the foreground color.

- If you specify either COLORS=(one-color) or the CPATTERN= graphics option, the default fill changes from SOLID to the appropriate list of hatch patterns. SAS/GRAPH uses the specified color to generate one pattern definition for each hatch pattern in the list.

For a description of these options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

**Working with PATTERN Statements**

**What You Can Specify with PATTERN Statements**
With PATTERN statements, you can specify the following:

- the type of fill (VALUE=)
- the color of the fill (COLOR=)
- the images used to fill the bars in a two-dimensional chart (IMAGE=)
- how many times to apply the statement before using the next one (REPEAT=).

See “Displaying Images on Data Elements” on page 336 for information about filling the bars of two-dimensional bar charts with images using the PATTERN statement.

You can also use procedure options to specify the pattern outline color and the CPATTERN= graphics option to specify a default color for all patterns.

Whether you use PATTERN statement options alone or with each other affects the number and type of patterns your PATTERN statements generate. Depending on which options you use, you can explicitly specify every pattern used by your graphs. Or you can let the PATTERN statement generate a series of pattern definitions using either the color list or the list of default fills.

**Explicitly Specifying Patterns**
To explicitly specify all the patterns in your graph, you need to do one of the following for every pattern your graph requires:

- Provide a PATTERN statement that uses the COLOR= option to specify the pattern color, for example:
  ```
  pattern1 color=red;
  ```
  By default, the fill type is SOLID.

- Provide a PATTERN statement that uses both the COLOR= option and the VALUE= option to specify the fill, for example:
  ```
  pattern1 color=blue value=r3;
  ```
  Including the COLOR= option in the PATTERN statement is the simplest way to ensure that you get exactly the patterns that you want. When you use the COLOR= option, the
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:

```plaintext
pattern1 value=r3;
```

In this case, the PATTERN statement rotates the R3 fill through all the colors in the color list. For more information about pattern rotation, see “Understanding Pattern Sequences” on page 410.

**Selecting an Appropriate Pattern**

The type of fill you specify depends on the type of graph that you are producing:

<table>
<thead>
<tr>
<th>Type of graph</th>
<th>Appropriate type of fill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar and block charts (PROC GCHART)</td>
<td>“VALUE=bar/block-pattern” on page 400</td>
</tr>
<tr>
<td>Contour plots (PROC GCONTOUR)</td>
<td>“VALUE=plot-pattern” on page 401</td>
</tr>
<tr>
<td>Pie and star charts (PROC GCHART)</td>
<td>“VALUE=pie/star-pattern” on page 403</td>
</tr>
</tbody>
</table>

*Note:* You might specify a fill that is inappropriate for the type of graph that you are generating. In this case, SAS/GRAPH ignores the PATTERN statement and continues searching for a valid pattern. If it does not find a definition with a valid fill specification, it uses default patterns instead.

**Controlling Outline Colors**

Whenever you use PATTERN statements, the default outline color uses the style's outline color to outline every patterned area.

You can change the outline color of any pattern, whether the pattern is default or user-defined. Use the COUTLINE= option or the CEMPTY= option, or both, in the action statement that generates the chart.

**The Effect of the CPATTERN= Graphics Option**

Although the CPATTERN= graphics option is used most often with default patterns, it does affect the PATTERN statement. With default patterns (no PATTERN statements specified), it does the following:

- specifies the color for all patterns
- causes default patterns to use hatched fills instead of the default SOLID

In conjunction with the PATTERN statement, it does the following:

- With a PATTERN statement that specifies only a fill (VALUE=), the CPATTERN= option determines the color of that fill. For example, these statements produce two green, hatched patterns:

```plaintext
goptions cpattern=green;
pattern1 value=x3;
```
With a PATTERN statement that specifies only a color (COLOR=), the COLOR= option overrides the CPATTERN= color, but CPATTERN= causes the fill to be hatched, not the default SOLID. For example, these statements produce one red, hatched pattern:

```sas
goptions cpattern=green;
pattern1 color=red;
```

See also the description of “CPATTERN” on page 534.

**Understanding Pattern Sequences**

### About Pattern Sequences

Pattern sequences are sets of PATTERN definitions that SAS/GRAPH automatically generates when a PATTERN statement specifies a fill but not a color. In this case, the specified fill is used once with every color in the color list. If the REPEAT= option is also used, the resulting PATTERN definitions are repeated the specified number of times.

### Generating Pattern Sequences

SAS/GRAPH generates pattern sequences when a PATTERN statement uses VALUE= to specify a fill and all of the following conditions are also true:

- The COLOR= option is not used in the PATTERN statement.
- The CPATTERN= graphics option is not used.
- The color list, either default or user-specified, contains more than one color.

In this case, the PATTERN statement rotates the fill specified by the VALUE= option through every color in the color list. One PATTERN definition is generated for every color in the list. After every color has been used once, SAS/GRAPH goes to the next PATTERN statement. For example, suppose you specified the following color list and PATTERN statements for bar or block patterns:

```sas
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:

**Table 24.4  Color and Fill PATTERN Definitions for the First Nine Patterns**

<table>
<thead>
<tr>
<th>Definition Number</th>
<th>Source</th>
<th>Color</th>
<th>Fill</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PATTERN1</td>
<td>Red</td>
<td>x3</td>
</tr>
<tr>
<td>2</td>
<td>PATTERN2</td>
<td>Blue</td>
<td>r3</td>
</tr>
<tr>
<td>3</td>
<td>PATTERN2</td>
<td>Red</td>
<td>r3</td>
</tr>
</tbody>
</table>
The following table shows the cyclical use of PATTERN statements:

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Source</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PATTERN1</td>
<td>Gold Solid (first default)</td>
</tr>
<tr>
<td>2</td>
<td>PATTERN1</td>
<td>Gold Solid (first default)</td>
</tr>
<tr>
<td>3</td>
<td>PATTERN2</td>
<td>Red x1</td>
</tr>
<tr>
<td>4</td>
<td>PATTERN2</td>
<td>Blue x1</td>
</tr>
<tr>
<td>5</td>
<td>PATTERN2</td>
<td>Green x1</td>
</tr>
<tr>
<td>6</td>
<td>PATTERN2</td>
<td>Red x1</td>
</tr>
<tr>
<td>7</td>
<td>PATTERN2</td>
<td>Blue x1</td>
</tr>
</tbody>
</table>
SYMBOL Statement

Defines the characteristics of symbols that display the data plotted by a PLOT statement used by the GBARLINE, GCONTOUR, and GPLOT procedures.

**Used by:** GBARLINE, GCONTOUR, GPLOT

**Type:** Global

**Syntax**

SYMBOL<1 …255>

(COLOR=symbol-color | _style >

(MODE=EXCLUDE | INCLUDE>

(REPEAT=number-of-times>)

(STEP=distance<units>)

(appearance-option(s))

(interpolation-option>

(SINGULAR=n>);

**Summary of Optional Arguments**

**Appearance options**

BWIDTH=box-width

specifies the width of the box generated by either the INTERPOL=BOX or INTERPOL=HILOB option.

CI=line-color | _style_

specifies a color for an interpolation line (GPLOT and GBARLINE) or a contour line (GCONTOUR).

CO=color

specifies a color for outlines of filled areas, confidence limit lines, and staffs, boxes, and bars.

COLOR=symbol-color | _style_

specifies a color for the entire definition, unless it is followed by a more explicit specification.

CV=value-color | _style_

specifies a color for plot symbols, filled areas, and contour labels.

FONT="font"

specifies the font for the plot symbol (GPLOT, GBARLINE) or contour labels (GCONTOUR) specified by the VALUE= option.

HEIGHT=symbol-height<units>

specifies the height in number of units of plot symbols (GPLOT, GBARLINE) or contour labels (GCONTOUR).

LINE= line-type
specifies the line type of plot or contour lines.

**MODE=EXCLUDE | INCLUDE**
specifies that any interpolation method exclude or include data values that are outside the range of plot axes.

**POINTLABEL<=(label-description(s)) | NONE>**
labels plot points.

**STEP=distance<units>**
specifies the minimum distance between labels on contour lines

**VALUE=**
special-symbol | text-string | SPECIAL | NONE
specifies a plot symbol for the data points (GPLOT and GBARLINE).

**WIDTH=**
thickness-factor
specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR).

**General interpolation options**

INTERPOL=JOIN
connects data points with straight lines.

INTERPOL=map/plot-pattern
specifies that a pattern fill the polygon that has been defined by the data points.

INTERPOL=NEEDLE
draws a vertical line from each data point to a horizontal line at the 0 value on the vertical axis or the minimum value on the vertical axis.

INTERPOL=NONE
suppresses any interpolation and, if the VALUE= option is not specified, also suppresses plot points.

INTERPOL=STEP<placement><J><S>
specifies that the data are plotted with a step function.

**High-Low interpolation options**

INTERPOL=BOX<option(s)><00 …25>
produces box-and-whisker plots.

INTERPOL=HILO<option>
specifies that a solid vertical line connect the minimum and maximum Y values for each X value.

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.

**Regression interpolation option**

INTERPOL=R<type><0><CLM | CLI<50 …99>
specifies that a plot is a regression analysis.

**Specification options**

REPEAT=number-of-times
specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used.

SINGULAR=n
tunes the algorithm used to check for singularities.

**Spline interpolation options**

INTERPOL=L<degree><P><S>
specifies a Lagrange interpolation to smooth the plot line.

\texttt{INTERPOL=SM}<\texttt{nn}><P><S>

specifies that a smooth line is fit to data using a spline routine.

\texttt{INTERPOL=SPLINE}<\texttt{P}><S>

specifies that the interpolation for the plot line use a spline routine.

\textbf{Optional Arguments}

\texttt{BWIDTH=box-width}

specifies the width of the box generated by either the \texttt{INTERPOL=BOX} or \texttt{INTERPOL=HILOB} option. \textit{Box-width} can be any number greater than 0. By default, the value of \textit{box-width} is the same as the value of the \texttt{WIDTH=} option, whose default value is 1. Therefore, if you specify a \texttt{WIDTH=} value for the line thickness of interpolated lines, and omit the \texttt{BWIDTH=} option, the width of the box changes according to the \texttt{WIDTH=} value.

\texttt{Example}  \hspace{1em} “Example 4: Creating and Modifying Box Plots” on page 476

\texttt{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 \texttt{CI=} option but specify the \texttt{CV=} option, the \texttt{CI=} option assumes the value of the \texttt{CV=} option. In this case, the \texttt{CI=} and \texttt{CV=} options specify the same color, which is the same as specifying the \texttt{COLOR=} option alone.

If you omit the \texttt{CI=} option, the color specification is searched for in this order:

1. the \texttt{COLOR=} option
2. the \texttt{CV=} option
3. the \texttt{CSYMBOL=} option in a \texttt{GOPTIONS} statement
4. each color in the color list sequentially before the next \texttt{SYMBOL} definition is used

\texttt{See}  \hspace{1em} “Using Color” on page 441

\texttt{Example}  \hspace{1em} “Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

\texttt{CO=color}

specifies a color for the following:

- outlines of filled areas generated by the \texttt{INTERPOL=}map/plot-pattern\ option
- confidence limit lines generated by the \texttt{INTERPOL=}R\ series\ option
- staffs, boxes, and bars generated by the high-low interpolation methods: \texttt{INTERPOL=HILO}, \texttt{INTERPOL=BOX}, and \texttt{INTERPOL=STD}

If you omit the \texttt{CO=} option, the search order for a color specification depends on the interpolation method being used.

\texttt{See}  \hspace{1em} “Using Color” on page 441

\texttt{Examples}  \hspace{1em} “Example 5: Filling the Area between Plot Lines” on page 481

“Example 4: Creating and Modifying Box Plots” on page 476
COLOR=\textit{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.

\textbf{Alias}\n
\textit{C=}  

\textbf{Style reference}\n
Color attribute of the GraphLabelText style element

\textbf{Restriction}\n
Partially supported by Java and ActiveX

\textbf{Notes}\n
Neither the Java applet nor the ActiveX control supports using COLOR= with PROC GCONTOUR.

You might not use a SYMBOL statement to specify a color for each symbol. Instead, you specify a color list in a GOPTIONS statement. then Java and ActiveX assign symbols and symbol colors differently than other devices. To ensure consistency on all devices, you should use SYMBOL statements to explicitly specify the symbols and symbol colors that you want to use in your plot.

\textbf{See}\n
“Using Color” on page 441

CV=value-color | \_style_

specifies a color for the following:

- plot symbols in the GPLOT procedure
- the filled areas generated by the INTERPOL=\textit{map/plot-pattern} option
- contour labels in the GCONTOUR procedure

The \_STYLE\_ value specifies the appropriate color based on the current style. If you omit the CV= option but specify the CI=, the CV= option assumes the value of the CI= option. In this case, the CV= and CI= options specify the same color, which is the same as specifying the COLOR= option alone.

If you omit the CV= option, the color specification is searched for in this order:

1. the COLOR= option
2. the CI= option
3. the CSymbol= option in a GOPTIONS statement
4. each color in the color list sequentially before the next SYMBOL definition is used
Restriction
Partially supported by Java and ActiveX

Note
Neither the Java applet nor the ActiveX control supports using the CV= option with PROC GCONTOUR.

See
“Using Color” on page 441

Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465
“Example 4: Creating and Modifying Box Plots” on page 476
“Example 5: Filling the Area between Plot Lines” on page 481

**FONT=”font”**

specifies the font for the plot symbol (GPLOT, GBARLINE) or contour labels (GCONTOUR) specified by the VALUE= option. The font specification must be enclosed in quotation marks and can include the /bold and /italic font modifiers.

By default, the symbol specified by the VALUE= option is taken from the special symbol table shown in Table 24.6 on page 433. To use symbols from the special symbol table, you must omit the FONT= option.

You might want to use a symbol that is not in that special symbol table. Specify the font containing the symbol and the character code or hexadecimal code of the symbol that you want to use. You can also specify text instead of special symbols. For example:

```sas
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:

```sas
symbol font= value=dot;
```

Alias
**F=**

Restrictions
Not supported by Java and ActiveX

Note
If the font is specified with the Unicode attribute, then the font symbol values are processed as double-byte characters. Double-byte characters resolve to question marks.

See
“ VALUE=special-symbol | text-string | SPECIAL | NONE” on page 432

“Specifying Plot Symbols” on page 439

“Specifying Special Characters Using Character and Hexadecimal Codes” on page 304

Example
“Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033
**HEIGHT=**`<symbol-height><units>`
specifies the height in number of units of plot symbols (GPLOT, GBARLINE) or contour labels (GCONTOUR).

**Alias**
H=

**Restriction**
Partially supported by Java and ActiveX

**Notes**
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.

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 ODSs. For more information about ODS styles, see the *SAS Output Delivery System: User’s Guide.*

**See**
the option “ **SHAPE=BAR(</width><units>,height<units>)<units> | LINE(length)<units> | SYMBOL(</width><units>,height<units>)<units>” on page 385 in the LEGEND statement

**Examples**
“Example 4: Creating and Modifying Box Plots” on page 476

“Example 3: Rotating Plot Symbols through the Color List” on page 473

**INTERPOL=BOX<](option(s))<00 …25>**
produces box-and-whisker plots.

In all instances the bottom and top edges of the box are located at the sample 25th and 75th percentiles. The center horizontal line is drawn at the 50th percentile (median). Specifying options controls the appearance and the drawing of reference lines. Percentile specifications control the whiskers.

By default, **INTERPOL=BOX**, with no option or percentile specified. In this case the vertical lines, or whiskers, are drawn from the box to the most extreme point less than or equal to 1.5 interquartile ranges. (An interquartile range is the distance between the 25th and 75th sample percentiles.) Any value more extreme than this is marked with a plot symbol.

Values for **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:
specifies high or low extremes. INTERPOL=BOX00 is not the same as the default, INTERPOL=BOX.

specifies 1st percentile low, 99th high.

specifies 5th percentile low, 95th high.

specifies 10th percentile low, 90th high.

specifies 25th percentile low, 75th high. Because the box extends from the 25th to the 75th percentile, no whiskers are produced.

Figure 24.15 on page 418 shows the type of plot INTERPOL=BOX produces.

To increase the thickness of all box plot lines, including the box, whiskers, join line, and top and bottom ticks, use the WIDTH= option.

To increase the width of the box itself, use the BWIDTH= option. By default the value of the BWIDTH= option is the same as the value of the WIDTH= option. Therefore, if you specify a value for the WIDTH= option and omit the BWIDTH= option, the width of the box changes.

For a scatter effect with the box, use a multiple plot request, as in this example:

```sas
symbol1 i=none v=star color=green;
symbol2 i=box v=none color=blue;
proc gplot data=test;
  plot (y y)*x / overlay;
```

### Alias

I=

### Default

INTERPOL=BOX

### Restriction

You cannot use the GPLOT procedure PLOT statement option AREAS= with INTERPOL=BOX.

### Notes

You can use the HAXIS= or VAXIS= options in the PLOT statement. Or you can use the ORDER= option in an AXIS definition. Use any of these options to restrict the range of axis values. By default any observations that fall outside the axis range are excluded from the
interpolation calculation. See the “MODE=EXCLUDE | INCLUDE” on page 428.

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.

Example

“Example 4: Creating and Modifying Box Plots” on page 476

**INTERPOL=HILO</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 24.16 on page 420.

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 24.16 on page 420 shows the type of plot INTERPOL=HILO produces. Plot symbols in the form of dots have been added to this figure.
When using DEVICE=JAVA and DEVICE=JAVA IMG 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=

**Restriction**

Partially supported by Java

**Note**

You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. See the option “MODE=EXCLUDE | INCLUDE” on page 428.

**Tip**

To increase the thickness of all lines generated by the INTERPOL=HILO option, use the WIDTH= option.

**Example**

“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

**INTERPOL=JOIN**

connects data points with straight lines. Points are connected in the order in which they occur in the input data set. Therefore, the data should be sorted by the independent (horizontal axis) variable.

If the data contain missing values, the observations are omitted. However, the plot line is not broken at missing values unless the SKIPMISS option is used.

**Alias**

I=

**See**

“SKIPMISS” on page 1185

“Missing Values” on page 1140

**INTERPOL=L<degree><P><S>**

specifies a Lagrange interpolation to smooth the plot line.

Specify one of these values for `degree`:
specifies the degree of the Lagrange interpolation polynomial. By default, \textit{degree} is 1.

In addition, you can specify one or both of these:

\textbf{P}

specifies a parametric interpolation.

\textbf{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 \texttt{INTERPOL=L1P}, \texttt{INTERPOL=L3P}, or \texttt{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.

\textbf{Alias I=}\texttt{INTERPOL=map/plot-pattern}

specifies that a pattern fill the polygon that has been defined by the data points. Values for \texttt{map/plot-pattern} are as follows:

\textbf{MEMPTY}

specifies an empty pattern. EMPTY and E are valid aliases.

The Java applet does not support this option.

\textbf{Alias ME}

\textbf{MSOLID}

specifies a solid pattern. SOLID and S are valid aliases

\textbf{Alias MS}

\textbf{Mdensity<style<angle>}}

specifies a shaded pattern. (The Java applet does not support this option.)

\textit{Density} specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest.

\textit{Style} specifies the direction of pattern lines:

\textbf{N}

specifies parallel lines (the default)

\textbf{X}

specifies crosshatched lines.

\textit{Angle} specifies the starting angle for parallel or crosshatched lines. Specify a number from 0 to 360. The number specifies the degree at which the parallel lines are drawn. By default, \textit{angle} is 0 (lines are parallel to the horizontal axis).
Restrictions Partially supported by Java

The INTERPOL=map/plot-pattern option works only if the data are structured so that the data points and, consequently, the plot lines form an enclosed area. The plot lines should not cross each other.

See “PATTERN Statement” on page 398

Example “Example 5: Filling the Area between Plot Lines” on page 481

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 24.17 on page 422 shows the type of plot INTERPOL=NEEDLE produces. Plot symbols are not displayed in this figure.

Figure 24.17 Needle Plot

Alias I=

Restriction You cannot use the PLOT statement option AREAS= with INTERPOL=NEEDLE

Tip Use the WIDTH= option or the HEIGHT= option in the SYMBOL statement to vary the width or the height of the plot needles, respectively. The specified width or height is also reflected in the legend entry. You can vary the width or height of the legend separately from the size of the plot needles. In this case, use the LEGEND statement option SHAPE=" SYMBOL (width,height)<units> | (width<units>,height<units>) <units>" on page 385 when using the SYMBOL statement option INTERPOL=NEEDLE to modify the width or height values of the legend entry.

INTERPOL=NONE

suppresses any interpolation and, if the VALUE= option is not specified, also suppresses plot points. If no interpolation method is specified in a SYMBOL statement and if the graphics option INTERPOL= is not used, INTERPOL=NONE is the default.

Alias I=
Restriction: You cannot use the PLOT statement option AREAS= with INTERPOL=NONE.

**INTERPOL=R<type><0><CLM | CLI<50 ...99>>**

specifies that a plot is a regression analysis. By default, the regression type is L, regression lines are not forced through plot origins, and confidence limits are not displayed.

The regression line is drawn in the line type specified in the LINE= option. By default, the line type of the regression line is 1.

**Type** indicates the type of regression. Specify one of these values for **type**:

- **L**
  - requests linear regression representing the regression equation
    \[ Y = \beta_0 + \beta_1 X. \]

- **Q**
  - requests quadratic regression representing the regression equation
    \[ Y = \beta_0 + \beta_1 X + \beta_2 X^2. \]

- **C**
  - requests cubic regression representing the regression equation
    \[ Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3. \]

*Note:* When least-square solutions for the parameters are not unique, most devices use a quadratic equation by default for the interpolation. However, the Java and ActiveX device drivers might use a cubic solution.

**CAUTION:**
- You must specify **type** if you use 0, or CLI, or CLM, in order to achieve the expected results.

For example, the following interpolation request is correct:

\[ I=RL0CLM95 \]

while the following interpolation request is incorrect because **type** is missing although 0 is specified:

\[ I=R0CLM95 \]

and the following interpolation request is incorrect because **type** is missing although CLM is indicated:

\[ I=RCLM95 \]

To force the regression line through a (0,0) origin, specify the following:

0

eliminates the \( \beta_0 \) parameter, or intercept, from the regression equation. If the origin is at (0,0), also forces the regression line through the origin. For example, if you specify 0 for a linear regression, the plot line represents the equation

\[ Y = \beta_1 X. \]

*Note:* You can force the regression line through the origin (0,0) when the data ranges do not place the origin at (0,0). Use the G PLOT 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 24.18 on page 424 shows the type of plot INTERPOL=RCCLM95 produces (cubic regression analysis with 95% confidence limits).

Figure 24.18  Plot of Regression Analysis and Confidence Limits

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. The greater the nn value, the smoother the fitted curve. By default, the value of nn is 0.

In addition, specify one or both of these:

P
  specifies a parametric cubic spline.

---

S
sorts data by the independent variable before plotting.

Alias I=

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	ontrivial spline interpolation methods.
Spline interpolation smooths a plot line using a cubic spline method with continuous
second derivatives1 This method uses a piecewise third-degree polynomial for each
set of two adjacent points. The polynomial passes through the plotted points and
matches the first and second derivatives of neighboring segments at the points.
Specify one or both of these:
P
specifies a parametric spline interpolation method. This interpolation uses a
parametric spline method with continuous second derivatives. Using the method
described earlier for the spline interpolation, a parametric spline is fitted to both
the horizontal and vertical values. The parameter used is the distance between
points
\[
t = \sqrt{x^2 + y^2}
\]
If two points are so close together that the computations overflow, the second
point is not used.
S
sorts a data set by the independent variable before plotting its data.

Alias I=

Notes 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.
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.

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. 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 \( s_y \).
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 24.19 on page 426 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 24.19  Plot of Standard Deviations

You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. See the “ MODE=EXCLUDE | INCLUDE” on page 428.

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=JAVA IMG 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=

Restriction  Partially supported by Java

Notes  By default, two standard deviations are used.

By default, the vertical axis ranges from the minimum to the maximum Y value in the data. The requested number of standard deviations from the mean covers a range of values that might exceed the maximum or is less than the minimum. In this case the STD lines are cut off at the minimum and maximum Y values. When this cutoff occurs, rescale the
axis using VAXIS= in the PLOT statement or ORDER= in an AXIS definition so that the STD lines are shown.

**INTERPOL=STEP<placement><J><S>**

specifies that the data are plotted with a step function. By default, the data point is on the left of the step, the steps are not joined with a vertical line, and the data are not sorted before processing.

Specify one of these values for placement:

- **L**
  - displays the data point on the left of the step.
- **R**
  - displays the data point on the right of the step.
- **C**
  - displays the data point in the center of the step.

In addition, specify one or both of these:

- **J**
  - produces steps joined with a vertical line.
- **S**
  - sorts unordered data by the independent variable before plotting.

*Figure 24.20 on page 427* shows the type of plot that INTERPOL=STEPJR produces. Plot symbols in the form of dots have been added to this figure.

*Figure 24.20  Step Plot*

**Alias**

I=

**Notes**

Sometimes a step is retraced in order to locate its center point. In this case the GIF, JPEG, PNG, ACTXIMG, 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, which results in a somewhat different graph.

The ActiveX and Java devices do not support the S option. For these devices, use the SORT procedure or the ORDER= option in an AXIS statement to sort your data by the independent variable before calling the GPLOT or GBARLINE procedure.

**LINE=**<line-type>

specifies the line type of the plot line in the GPLOT procedure, or the contour line in the GCONTOUR procedure:
1 specifies a solid line.

2 ...46 specifies a dashed line.

Line types are shown in Figure 24.21 on page 443. By default, LINE=1.

**Alias**

\[ \text{L=} \]

**Restriction**

Partially supported by Java and ActiveX

**Notes**

This option overrides the LineStyle attribute in graph styles.

Neither the Java applet nor ActiveX control supports GCONTOUR.

---

**MODE=EXCLUDE | INCLUDE**

specifies that any interpolation method exclude or include data values that are outside the range of plot axes. By default, MODE=EXCLUDE prevents values outside the axis range from being displayed.

You can restrict the range of axis values by using the HAXIS= or VAXIS= option in a PLOT statement. Or use the ORDER= option in an AXIS definition. By default any observations that fall outside the axis range are excluded from the interpolation calculation. Using these options to control value ranges has a particularly noticeable effect on the high-low interpolation methods, which include INTERPOL=HILO, INTERPOL=BOX, and INTERPOL=STD. Regression analysis also represents only part of the original data.

**Restrictions**

Not supported by Java and partially supported by ActiveX

When plotting multiple plot lines on the same output graph, MODE=INCLUDE is not supported by the ActiveX device driver. Specifying a PNG or GIF device driver generates the graph with values that are interpolated but not plotted when those values fall outside of the axis range.

**See**

“Values Out of Range” on page 1140

**POINTLABEL<=\((\text{label-description(s)})\) | NONE>**

labels plot points. The labels always use the format that is assigned to the variable or variables whose values are used for the labels. POINTLABEL without any specified description labels results in no displayed labels for the points. NONE suppresses the point labels. Label-description(s) can be used to change the variable whose values are used to label points. This value can also change features of the label text, such as the color, font, or size of the text.

Label-description(s) can be one or more of these:

**COLOR=text-color**

specifies the color of the label text.

The POINTLABEL COLOR= specification searches for a color in this order:

1. the color specified by the COLOR= option of the POINTLABEL= option of the SYMBOL statement
2. the color specified by the CTEXT= option in the PLOT statement of the GGRAPH procedure
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the first color in the color list of the GOPTIONS statement

**Alias**  
C=

**Default**  
The default is the color of the axis label.

**Note**  
If you do not specify a color in a SYMBOL statement, the symbol definition is rotated through the color list before the next SYMBOL statement is used. Take the case where your plot contains multiple plot lines and you want to limit your POINTLABEL specification to a single line. You must specify a color in the SYMBOL statement that contains the POINTLABEL description.

**DROPCollisions | NODropCollisions**  
specify DROPCollisions to drop new labels if they collide with a label already in use. Specify NODropCollisions to retain all labels. The default is DROPCollisions.

The algorithm for the placement of markers tries to avoid placing labels such that they collide. If the algorithm is unable to avoid a collision, then the default DROPCollisions is to drop the new label, whereas NODropCollisions retains even colliding labels.

**FONT=font | NONE**  
specifies the font for the text. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about specifying font. If you omit FONT=, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default hardware font, NONE

**Alias**  
F=

**Restriction**  
Only single-byte symbol characters are supported.

**Note**  
If the font is specified with the Unicode attribute, then the font symbol values are processed as double-byte characters. Double-byte characters resolve to question marks. POINTLABEL specifications that mix double- and single-byte characters are processed as if the entire string consists of double-byte characters.

**HEIGHT=text-height <units>**  
specifies the height of the text characters in number of units. By default, HEIGHT=1 CELL. If you omit HEIGHT=, a text height specification is searched for in this order:

1. the HTEXT= option in a GOPTIONS statement
2. the default value, 1

**Alias**  
H=

**JUSTIFY=CENTER | LEFT | RIGHT**  
specifies the horizontal alignment of the label text. The default is CENTER. The location of the point label is relative to the location of the corresponding data point.

**Alias**  
J=C | L | R
POSITION=TOP | MIDDLE | BOTTOM

specifies the vertical placement of the label text. The default is TOP. The location
of the point label is relative to the location of the corresponding data point.

"#var1:<#var2 <$char>>"

specifies the variable or variables whose values label the plot points, and the
delimiter between more than one variable. The variable specification must be
enclosed in either single or double quotation marks. The first specified variable
must be prefixed with a number sign (#). If a second variable is specified, it must
be prefixed with a colon and a number sign (:#). When you specify two variables,
you can also specify the character to display as the delimiter between variable
values in the plot label.

By default, if the POINTLABEL= option is specified without naming a label
variable, the Y values label the plot points. You can change the default by using
"#var" to specify a different variable whose values should label the points. For
example, you might specify the name of the X variable. The following option
specifies the variable SALES as the variable whose values label plot points:

POINTLABEL=("#sales")

Alternatively, you can label the plot points with the values of two variables, in
either order. The order in which you specify the variables determines the order
that the values are displayed in the label. The following option specifies variables
HEIGHT and WEIGHT; which in the label displays the value for HEIGHT and
then the value for WEIGHT:

POINTLABEL=("#height:#weight")

By default, when you specify two variables, a colon (:) is displayed in the label to
separate the variable values. To change the character that is displayed as the
delimiter, use the $ syntax to specify an alternative character. The following
option specifies a vertical bar (|) as the delimiter in the label:

POINTLABEL=("#height:#weight $|")

The $ syntax must be within the same quotation marks as the variable
specification. The $ specification can precede or follow the variable
specification, but it must be separated from the variable specification by at least
one space.

Specify as many label-description suboptions as you want. Enclose them all within a
single set of parentheses, and separate each suboption from the others by at least one
space.

Restrictions When creating output using the JAVA or ACTIVEX devices, the
variables that you specify must be for the plot's X and Y variables. Specifying any other variables causes unexpected labeling.

There is a 16-character length limit for each variable. A maximum
character length limit of thirty-three characters is possible. This can be
composed of X and Y variables, any other valid data set variable, and
a separator as required.

Notes Specifying a delimiting character with the $ changes only the
character that is displayed in the label. It does not change the syntax
of the variable specification, which requires a colon and number sign
(:#) to precede the second variable.
Creating multiple plots that share the same or close-ranging data points, along with specifying appearance suboptions, can result in multiple data point labels.

Depending on the appearance suboptions that you use for point labels, the labels can overlap other elements on the graph. The HEIGHT= suboption is an example. When this occurs, a warning appears in the SAS log.

The algorithm for placing markers on the graph repositions point labels that are overwriting a graphics output area boundary. When this occurs, a note appears in the SAS log.

**REPEAT=number-of-times**

specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used. The behavior of REPEAT= depends on whether any of the SYMBOL color options (CI=, CV=, CO=, and COLOR=) or the CSYMBOL= graphics option also is used:

- If any SYMBOL color option also is used in the SYMBOL definition, that SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL color option is used but the CSYMBOL= graphics option is currently in effect, the SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL statement color options are used and the CSYMBOL= graphics option is not used, the SYMBOL definition is cycled through each color in the color list. The entire group generated by this cycle then repeats the number of times specified by the REPEAT= option. Thus, the total number of iterations of the SYMBOL definition depends on the number of colors in the current color list.

**Alias**

R=

**Default**

REPEAT=1.

**Restrictions**

Partially supported by Java and ActiveX

Neither the Java applet nor ActiveX control supports GCONTOUR.

**See**

“Using the SYMBOL Statement” on page 437

**SINGULAR=n**

 tunes the algorithm used to check for singularities. The default value is machine dependent but is approximately 1E-7 on most machines. This option is rarely needed.

**STEP=distance<units>**

specifies the minimum distance between labels on contour lines. The value of distance must be greater than zero.

When you use the STEP= option, specify the minimum distance that you want between labels. The option then calculates how many labels it can fit on the contour line, taking into account the length of the labels and the minimum distance that you specified. Once it has calculated how many labels it can fit while retaining the minimum distance between them, it places the labels, evenly spaced, along the line. Consequently, the space between labels can be greater than what you specify, although it is never less.
In general, to increase the number of labels from the default, reduce the value of distance.

There are times when the procedure cannot write the label at a particular location on the contour. For example, when the contour line makes a sharp turn, the label might be placed farther along the line or omitted. If labels are omitted, a note appears in the log. Specifying a low value for the GCONTOUR procedure’s TOLANGLE= option can also cause labels to be omitted. This specification forces the procedure to select smoother labeling locations, which might not be available on some contours.

Default

STEP=65PCT

Restriction

Not supported by Java and ActiveX

Note

If you specify units of PCT or CELLS, the STEP= option calculates the distance between the labels based on the width of the graphics output area, not the height. For example, if you specify STEP=50PCT and if the graphics output area is 9 inches wide, the distance specified is 4.5 inches. A value less than 10% is ignored and 10% is used instead.

Example

“Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033

VALUE=\text{\textit{special-symbol} | text-string | SPECIAL | NONE}

specifies a plot symbol for the data points (GPLOT and GBARLINE).

- If you omit the SYMBOL statement, plot points are generated using the default plot symbols. If you specify a SYMBOL statement, but do not specify the VALUE= option, the plot symbols are suppressed on the graph and in the legend.
- specifies contour-label text in a contour plot (GCONTOUR). By default with the AUTOLABEL option, GCONTOUR labels contour lines with the contour variable's value at that contour level.
- VALUE=SPECIAL enables you to define unique special symbols for up to 12 plots (GPLOT) in one SYMBOL statement. Some of the symbols include the following: CIRCLE, DIAMOND, DIAMONDFILLED, DOT, HASH, PAW, POINT, PLUS, SQUARE, SQUAREFILLED, STAR, TRIANGLE, TRIANGLEFILLED, X, Y, and Z. See Table 24.6 on page 433 for the complete list. This option is useful when the number of plots in your graph is variable. Rather than writing a SYMBOL statement for each possible plot, you can use VALUE=SPECIAL in one SYMBOL statement to define symbols for up to 12 individual plots.
- VALUE=NONE suppresses plot symbols at the data points, or labels on the contour lines. You can set the VALUE=NONE option independent of the INTERPOL= option.

Values for special-symbol are the names and characters shown in Table 24.6 on page 433. The special symbol table can be used only if the FONT= option is not used or a null value is specified:

\texttt{font=,}

This means that you cannot use VALUE=\textit{special-symbol} or VALUE=SPECIAL if you are using the FONT= option.
### Table 24.6  Supported Plot Symbols

<table>
<thead>
<tr>
<th>Value</th>
<th>Symbol</th>
<th>Value</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLUS</td>
<td>$+$</td>
<td>$$</td>
<td>$&amp;$</td>
</tr>
<tr>
<td>X</td>
<td>$\times$</td>
<td>$%$</td>
<td>$&amp;$</td>
</tr>
<tr>
<td>STAR</td>
<td>$\star$</td>
<td>$&amp;$</td>
<td>$&amp;$</td>
</tr>
<tr>
<td>SQUARE</td>
<td>$\square$</td>
<td>$'$</td>
<td>$\prime$</td>
</tr>
<tr>
<td>SQUAREFILLED</td>
<td>$\square$</td>
<td>$=$</td>
<td>$\star$</td>
</tr>
<tr>
<td>DIAMOND</td>
<td>$\diamond$</td>
<td>$-$ (hyphen)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>DIAMONDFILLED</td>
<td>$\diamond$</td>
<td>$@$</td>
<td>$\bullet$</td>
</tr>
<tr>
<td>TRIANGLE</td>
<td>$\triangle$</td>
<td>$*$ (asterisk)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>TRIANGLEFILLED</td>
<td>$\triangle$</td>
<td>$+$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>HASH</td>
<td>$#$</td>
<td>$&gt;$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>Y</td>
<td>$\gamma$</td>
<td>. (period)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>Z</td>
<td>$\zeta$</td>
<td>$&lt;$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>PAW</td>
<td>$\pi$</td>
<td>$,$ (comma)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>POINT</td>
<td>$.$</td>
<td>$/$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>DOT</td>
<td>$.$</td>
<td>$?$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>$\circ$</td>
<td>$($</td>
<td>$\circ$</td>
</tr>
<tr>
<td>_ (underscore)</td>
<td>$\square$</td>
<td>$)$</td>
<td>$\circ$</td>
</tr>
<tr>
<td>&quot; (double quote)</td>
<td>$\prime$</td>
<td>$:$</td>
<td>$\star$</td>
</tr>
<tr>
<td>#</td>
<td>$\prime$</td>
<td></td>
<td>$\prime$</td>
</tr>
</tbody>
</table>
Note: The words or special characters in the VALUE= column are entered exactly as shown.

To specify a single quotation mark, you must enclose it in double quotation marks:

```
value="'
```

To specify a double quotation mark, you must enclose it in single quotation marks:

```
value='"'
```

In some operating environments, punctuation characters might require single quotation marks.

If you use VALUE=text-string to specify a plot symbol, you must also use the FONT= option to specify a symbol font or a text font. If you specify a symbol font, the characters in the string are character codes for the symbols in the font. If you specify a text font, the characters in the string are displayed. If you specify a text string containing quotation marks or blanks, enclose the string in single quotation marks.

For example, if you specify this statement, the plot symbol is the word "plus" instead of the symbol +:

```
symbol font=swiss value=\"plus\";
```

Java and ActiveX support the following characters from the Figure A2.5 on page 1501 for special-symbol and SPECIAL:

**Table 24.7 Marker-font Symbols Supported by Java and ActiveX**

<table>
<thead>
<tr>
<th>Character</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marker</td>
<td>Cone, Pyramid, Default</td>
</tr>
<tr>
<td>Square</td>
<td>Cube</td>
</tr>
<tr>
<td>Star</td>
<td></td>
</tr>
<tr>
<td>Circle</td>
<td>Sphere, Dot, Balloon</td>
</tr>
<tr>
<td>Plus</td>
<td>Cross</td>
</tr>
<tr>
<td>Flag</td>
<td>Y</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Prism</td>
<td>Z</td>
</tr>
<tr>
<td>Spade</td>
<td>&quot;</td>
</tr>
<tr>
<td>Heart</td>
<td>#</td>
</tr>
<tr>
<td>Diamond</td>
<td>$</td>
</tr>
<tr>
<td>Club</td>
<td>%</td>
</tr>
<tr>
<td>Hexagon</td>
<td>Paw</td>
</tr>
</tbody>
</table>
Note: You might not use a SYMBOL statement to specify a color for each symbol, but you do specify a color list in a GOPTIONS statement. In this case, Java and ActiveX assign colors to symbols differently than do the other device drivers. To ensure consistency on all devices, for VALUE=special-symbol or VALUE=text-string, you should specify the desired color of each symbol. If you do not specify a symbol color, SAS/GRAPH uses the first default color and the first symbol. It uses each color in the list of default colors until the list is exhausted. SAS/GRAPH then selects the next symbol and begins again with the first default color. It rotates the new symbol through the list of default colors before selecting another symbol. It continues selecting new symbols and colors until no more symbols are needed.

For VALUE=SPECIAL, the way in which symbols and their colors are assigned is different from VALUE=special-symbol and VALUE=text-string. For example, you specify VALUE=SPECIAL when generating symbols for a plot. Rather than rotating each symbol through the list of colors, SAS/GRAPH rotates through the 12 special symbols and assigns each symbol a color from the color list. The symbols and the symbol colors that are used in a plot depend on the following:

- whether a symbol color is specified in the SYMBOL statement or in the GOPTIONS statement
- the number of colors in the color list when a symbol color is not specified

Restriction

Partially supported by Java and ActiveX

Notes

For ActiveX output, the VALUE= option is not supported when INTERPOL=HILO or INTERPOL=STD. You can use the OVERLAY option with GPLOT to get symbols to appear on the data points.

The VALUE option overrides the MarkerSymbol attribute in graph styles.

See

“Symbol Sequences Generated from SYMBOL Statements” on page 444

“FONT=font | NONE ” on page 429

“Specifying Plot Symbols” on page 439

“About Special Fonts and Symbols in JAVA” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide for a description of the special symbols that the Java devices support and the value that you must specify in the SYMBOL statement VALUE= option for each.

“About Special Fonts and Symbols in ActiveX” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide for a description of the special symbols that the ActiveX devices support and the value that you must specify in the SYMBOL statement VALUE= option for each.
WIDTH=thickness-factor
specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR). thickness-factor is a number. The thickness of the line increases directly with thickness-factor. By default, WIDTH=1.

WIDTH= also affects all the lines in box plots (INTERPOL=BOX), high-low plots with bars (INTERPOL=HILOB), and standard deviation plots (INTERPOL=STD). It also affects the outlines of the area generated by the AREAS= option in the PLOT statement of the GPLOT procedure.

Alias W=
Style reference LineThickness attribute of the GraphAxisLines element
Restriction Partially supported by Java and ActiveX
Notes By default, the value specified by WIDTH= is used as the default value for the BWIDTH= option. For example, specifying WIDTH=6 also sets BWIDTH= to 6 unless you explicitly assign a value to the BWIDTH= option.

Java and ActiveX do not provide the same measure of control for width as the other SAS/GRAPH device drivers. Measurements are translated to pixels rather than a percentage. For the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG devices, the maximum width is 6.

Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465
“Example 4: Creating and Modifying Box Plots” on page 476

Details

Description: SYMBOL Statement
SYMBOL statements create SYMBOL definitions. These definitions are used by the GPLOT, GBARLINE, and GCONTOUR procedures. For GPLOT and GBARLINE procedures, SYMBOL definitions control the following:

• the appearance of plot symbols and plot lines, including bars, boxes, confidence limit lines, and area fills
• interpolation methods
• how plots handle data out of range

For the GCONTOUR procedure, SYMBOL definitions control the following:

• the appearance and text of contour labels
• 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.

**Using the SYMBOL Statement**

**How SYMBOL Definitions Are Generated**

A SYMBOL statement specifies one or more options that indicate the color and other attributes used by the GPLOT, GBARLINE, and GCONTOUR procedures. For GPLOT and GBARLINE, the main attributes include the plot symbol, interpolation method, and type of plot line. For GCONTOUR, the main attributes include the type of contour lines used and the text used to label those lines.

*Note:* SYMBOL statements can be applied only to contour plots when the AUTOLABEL option is specified on GCONTOUR.

You can define up to 255 different SYMBOL statements. A SYMBOL statement without a number is treated as a SYMBOL1 statement.

SYMBOL definitions can be defined anywhere in your SAS program. They are global and remain in effect until canceled or until you end your SAS session. Once defined, SYMBOL definitions can be used as follows:

- assigned by default by GPLOT or explicitly selected with the plot request
- used by GCONTOUR to control the labels and attributes of contour lines

SYMBOL statements generate one or more symbol definitions, depending on how color is used and whether a plot symbol or type of contour line is specified. For more information, see “Controlling Consecutive SYMBOL Statements” on page 438 and “Using Generated Symbol Sequences” on page 443.

Although it is common practice, you do not have to start with SYMBOL1, and you do not have to use sequential statement numbers. When assigning SYMBOL definitions, SAS/GRAPH software starts with the lowest-numbered definition and works upward, ignoring gaps in the numbering.

**Altering or Canceling SYMBOL Statements**

SYMBOL statements are additive. You can define a SYMBOL statement and later submit another SYMBOL statement with the same number. The new SYMBOL statement defines or cancels only the options that are included in the new statement. Options that are not included in the new statement are not changed and remain in effect.

*Note:* An exception to this rule is presented by POINTLABEL= suboptions that are not carried over to subsequent SYMBOL statements.

Assume you define SYMBOL4 as follows:

```
symbol4 value=star cv=red height=4;
```

The following statement cancels only HEIGHT= without affecting the rest of the definition:

```
symbol4 height=;
```

Add or change options in the same way. This statement adds an interpolation method to SYMBOL4:

```
symbol4 interpol=join;
```

This statement changes the color of the plot symbol from red to blue:
symbol4 cv=blue;

After all these modifications, SYMBOL4 has these characteristics:

symbol4 value=star cv=blue interpol=join;

Cancel individual SYMBOL statements by defining a SYMBOL statement of the same number without options (a null statement):

symbol4;

Canceling one SYMBOL statement does not affect any other SYMBOL definitions. To cancel all current SYMBOL statements, use the RESET= option in a GOPTIONS statement:

goptions reset=symbol;

Specifying RESET=GLOBAL or RESET=ALL cancels all current SYMBOL definitions as well as other settings.

To display current SYMBOL definitions in the LOG window, use the GOPTIONS procedure with the SYMBOL option:

proc goptions symbol nolist;
run;

Controlling Consecutive SYMBOL Statements

You can specify consecutively numbered SYMBOL statements and you might want SAS/GRAPH to use each definition only once. Use color specifications to ensure that each SYMBOL statement generates only one symbol definition. You can do the following actions:

- specify colors in each SYMBOL statement, using the COLOR=, CI=, CV=, or CO= options. This method lets you explicitly assign colors for each definition. For example, these statements generate four definitions:

  symbol1 value=star color=green;
  symbol2 value=square color=yellow;
  symbol3 value=special color=cyan;
  symbol4 value=special color=orange;

- specify a default color for all SYMBOL statements using the CSYMBOL= option in the GOPTIONS statement. This method makes it easy to specify the same color for each definition when you do not need more explicit color specifications.

- limit the color list to a single color using the COLORS= option in the GOPTIONS statement. This method makes it easy to specify the same color for each definition when you want the color to apply to other definitions also, such as PATTERN definitions.

For more information about specifying colors for symbol definitions, see “Using Color” on page 441.

You do not have to use a color to limit a SYMBOL statement to a single symbol definition. In this case, SAS/GRAPH generates multiple symbol definitions from that statement by rotating the current definition through the color. For more details, see “Using Generated Symbol Sequences” on page 443. Note that SAS/GRAPH uses symbol definitions in the order in which they are generated. This can result in the $n^{th}$ symbol definition applied to a graph to not necessarily correspond to the SYMBOL$n$ statement.

For example, assuming that no color is specified on the CSYMBOL= graphics option, these statements generate seven definitions:
Because no color is specified on SYMBOL1, SAS/GRAPH rotates the symbol definition through the color list, which has three colors. Thus, SYMBOL1 defines the first three applied symbol definitions. The SYMBOL2 definition specifies a color for the symbol, which means that it generates only one symbol. Thus, SYMBOL2 defines the fourth. Finally, SYMBOL3 specifies SPECIAL as its value, which means that it rotates through the list of 12 special symbols using the colors in the color list. Thus, SYMBOL3 defines the last three symbol definitions. Here is a summary of the symbols that are generated by these SYMBOL statements.

**Table 24.8 Example Output Using 3 Symbol Definitions**

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Source</th>
<th>Characteristics: Color</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYMBOL1</td>
<td>Red</td>
<td>Star</td>
</tr>
<tr>
<td>2</td>
<td>SYMBOL1</td>
<td>Blue</td>
<td>Star</td>
</tr>
<tr>
<td>3</td>
<td>SYMBOL1</td>
<td>Green</td>
<td>Star</td>
</tr>
<tr>
<td>4</td>
<td>SYMBOL2</td>
<td>Yellow</td>
<td>Square</td>
</tr>
<tr>
<td>5</td>
<td>SYMBOL3</td>
<td>Red</td>
<td>Dot</td>
</tr>
<tr>
<td>6</td>
<td>SYMBOL3</td>
<td>Blue</td>
<td>Plus</td>
</tr>
<tr>
<td>7</td>
<td>SYMBOL3</td>
<td>Green</td>
<td>X</td>
</tr>
</tbody>
</table>

In this case, if a graph needs only three symbols, the SYMBOL2 and SYMBOL3 definitions are not used.

To make the $n^{th}$ applied symbol definition correspond to the SYMBOL$n$ statement, limit each SYMBOL statement to a single color. You can use one of the techniques listed at the beginning of this section.

**Setting Definitions for PROC GPLOT and PROC GBARLINE**

**Specifying Plot and Interpolation Methods**

The following topics apply only for SYMBOL statements used with PROC GPLOT and PROC GBARLINE:

- specifying plot symbols
- specifying default interpolation methods
- sorting data with spline interpolation

**Specifying Plot Symbols**

The VALUE= option specifies the plot symbols that PROC GPLOT and PROC GBARLINE uses to mark the data points on a plot. Plot symbols can be in the following forms:
• special symbols as shown in Table 24.6 on page 433
• characters from symbol fonts
• text strings

By default, the plot symbol is the DOT symbol for the Java and ActiveX devices or the + symbol for all of the other devices. To specify a special symbol, use the VALUE= option to specify a name or a character from Table 24.6 on page 433:

```
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 443.

To use plot symbols other than those in Table 24.6 on page 433, use the FONT= option to specify a font for the plot symbol. If the font is a symbol font, such as Marker, the string specified with the VALUE= option is the character code for the symbol to be displayed. If the font is a text font, the string specified with the VALUE= option is displayed as the plot symbol. (See “VALUE= special-symbol | text-string | SPECIAL | NONE” on page 432 and “FONT="font"” on page 416.)

This table illustrates some of the ways that you can define a plot symbol:

<table>
<thead>
<tr>
<th>Definition</th>
<th>Plot Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol1 value=plus;</td>
<td>+</td>
</tr>
<tr>
<td>symbol2 value=+;</td>
<td>☐</td>
</tr>
<tr>
<td>symbol3 font=swiss value=plus;</td>
<td>plus</td>
</tr>
<tr>
<td>symbol4 font=marker value=U;</td>
<td>□</td>
</tr>
<tr>
<td>symbol5 value=&quot;'&quot;;</td>
<td>☕</td>
</tr>
</tbody>
</table>

**Specifying a Default Interpolation Method**

The INTERPOL= option in a GOPTIONS statement specifies a default interpolation method to be used with all SYMBOL definitions. This default interpolation method is in effect unless you specify a different interpolation in a SYMBOL statement. If the GOPTIONS statement does not specify an interpolation method, the default for each SYMBOL statement is NONE.

**Sorting Data with Spline Interpolation**

To use the GPLOT procedure to sort by the horizontal axis variable before plotting, add the letter S to the end of any of the spline interpolation methods (INTERPOL=L, INTERPOL=SM, and INTERPOL=SPLINE). For example, suppose you want to overlay three plots (Y1*X1, Y2*X2, and Y3*X3). For each plot, you want the X variable sorted in ascending order. Use these statements:

```
symbol1 i=splines c=red;
symbol2 i=splines c=blue;
```
symbol3 i=splines c=green;
proc gplot;
  plot y1*x1 y2*x2 y3*x3 / overlay;
run;

Using Color

How to Specify Color for Symbols
Generally, there are two ways to explicitly specify color with SYMBOL statements:

• specify color in the SYMBOL statement
• specify color on the CSYMBOL= graphics option

You can also let SAS/GRAPH rotate symbol definitions through the color list. For details, see “Using Generated Symbol Sequences” on page 443.

Specifying Colors with SYMBOL Statements
The SYMBOL statement has these options for specifying color:

• The CV= option specifies color for plot symbols in GPLOT and GBARLINE, or for contour labels in GCONTOUR.
• The CO= option specifies color for confidence limit lines and area outlines in GPLOT and GBARLINE.
• The CI= option specifies color for plot lines in GPLOT and GBARLINE, or contour lines in GCONTOUR.
• The COLOR= option specifies color for the entire symbol. For GPLOT and GBARLINE, this includes plot symbols, plot lines, and outlines. For GCONTOUR, this includes contour lines and labels.

The CV= and CI= options have the same effect as using the COLOR= option when they are used in these ways:

• Only CV= or CI= option is used. (The option that is not used is assigned the value of the option used.)
• Both the CV= and CI= options specify the same color.

In general, the CI=, CV=, and CO= options color specific areas of the symbol. Use these options to produce symbols and plot lines of different colors without having to overlay multiple plot pairs. For example, if you request regression analysis with confidence limits, use the following statement. It assigns red to the plot symbol, blue to the regression lines, and green to the confidence limit lines:
symbol cv=red ci=blue co=green;

The COLOR= option colors the entire symbol or those portions of it not colored by one of the other color options. If the COLOR= option precedes the CI= or CV= options, the CI= or CV= specification is used instead. If none of the SYMBOL color options is used, color specifications are searched for in this order:

1. the CSYMBOL= option in a GOPTIONS statement
2. each color in the color list sequentially before the next SYMBOL definition is used

CAUTION:
If no color options are used, the SYMBOL definition cycles through each color in the color list.
If the SYMBOL color options and the CSYMBOL= graphics option are not used, the SYMBOL definition cycles through each color in the color list before the next definition is used. For details, see “Using Generated Symbol Sequences” on page 443.

**Specifying Color with CSYMBOL=**
The CSYMBOL= option in the GOPTIONS statement specifies the default color to be used by all SYMBOL definitions:

```sas
goptions csymbol=green;
symbol1 value=star;
symbol2 value=square;
symbol3 value=special;
```

In this example, all three of the SYMBOL statements use green.

*Note:* Although the SYMBOL3 definition specifies VALUE=SPECIAL, because the CSYMBOL= graphics option is used, it generates only the first symbol in the 12–symbol list (DOT) using the color green.

CSYMBOL= is overridden by any of the SYMBOL statement color options. See “Using Color” on page 441 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 24.21 on page 443 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.
Note: These line types are also used by other statements and procedures. Some options accept a line type of 0, which produces no line.

Using Generated Symbol Sequences

How SYMBOL Definitions Are Generated
Symbol sequences are sets of SYMBOL definitions that are automatically generated by SAS/GRAPH software if any of these conditions is true:

- no valid SYMBOL definition is available. In this case, default symbol sequences are generated by rotating symbol definitions through the color specified in the GOPTIONS statement's CSYMBOL= option. If a CSYMBOL= color is not in effect, the definitions are rotated through the color list.

- a SYMBOL statement specifies color but not a plot symbol for the GPLOT procedure. Or a SYMBOL statement specifies a line type for the GCONTOUR procedure (assuming that GCONTOUR does not specify the needed line types). Either a default plot symbol or a line type is used with the specified color and only one definition is generated.

- a SYMBOL statement specifies a plot symbol for GPLOT or a line type for GCONTOUR, but no color options. In this case, the specified plot symbol or line type is used once with the color specified by the CSYMBOL= graphics option. If a
CSYMBOL= color is not in effect, the specified plot symbol or line type is rotated through the color list.

- a SYMBOL statement specifies VALUE=SPECIAL for GLOT, but no color options. In this case, the first symbol in the 12–symbol list (DOT) is used once with the color specified by the CSYMBOL= graphics option. If CSYMBOL= color is not in effect, the SAS/GRAPH software rotates through the 12 symbols and assigns each symbol a color from the color list.

If the REPEAT= option is also used, the resulting SYMBOL definition is repeated the specified number of times.

**Default Symbol Sequences**

Default symbol sequences are generated by rotating symbol definitions through the current color list.

- Definitions used for GLOT rotate plot symbols through the color list; the first default plot symbol is a plus sign (+).
- Definitions used for GCONTOUR rotate line types; the first default line type is a solid line (line type 1).

Each time a default definition is required, SAS/GRAPH takes the first default plot symbol or line type and uses it with the first color in the color list. If more than one definition is required, it uses the same plot symbol or line type with the next color in the color list. It continues until all the colors have been used once. If more definitions are needed, SAS/GRAPH selects the second default plot symbol or line type and rotates it through the color list. It continues in this fashion, selecting default plot symbols or line types and cycling them through the color list until all the required definitions are generated.

A color can be specified with the CSYMBOL= option in the GOPTIONS statement. In this case, each default plot symbol or line type is used once with the specified color, and the colors in the color list are ignored.

**Symbol Sequences Generated from SYMBOL Statements**

You can use SYMBOL statements to generate a custom sequence of symbols and symbol colors for a plot. This is true if you do not specify the symbol color in the SYMBOL statement or with the CSYMBOL= graphics option. For symbols, you can use the default symbols or you can use the VALUE= option in the SYMBOL statement to specify a symbol by name (VALUE=symbol-name). Or instead use VALUE=text-string. Another possibility is to specify a predefined list of 12 special symbols (VALUE=SPECIAL). The predefined list of 12 symbols includes DOT, PLUS, X, TRIANGLE, SQUARE, STAR, DIAMOND, CIRCLE, HASH, Y, Z, and PAW. For the symbol colors, you can use the default color list or you can use the COLORS= graphics option to specify a custom list of colors.

Sometimes the symbol color is not specified and the VALUE=symbol-name or VALUE=text-string is used. In order to generate symbols for a plot in this case, SAS/GRAPH rotates each symbol through every color in the color list before it proceeds to the next SYMBOL definition. Sometimes the VALUE=SPECIAL option is used to generate symbols for a plot. In this case SAS/GRAPH rotates through each symbol in the 12–symbol list and assigns each symbol a color from the color list. Before proceeding to the next SYMBOL definition, the SAS/GRAPH software continues rotating in this manner. It rotates until either all of the 12 symbols are used or all of the colors in the color list are used.

Here is an example of SYMBOL statements that use a custom color list with both default symbols and symbols that is identified by name:
In this example, the SYMBOL1 statement generates the first SYMBOL definition. The SYMBOL2 statement does not include color, so the first default plot symbol is rotated through all colors in the color list before the SYMBOL3 statement is used. This table shows the colors and symbols that would be used if nine symbol definitions were required for PROC GPLOT:

**Table 24.10 Example Output Using 9 Symbol Definitions in PROC GPLOT**

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Source</th>
<th>Characteristics: Color</th>
<th>Symbol</th>
<th>Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYMBOL1</td>
<td>cv=red</td>
<td>First default</td>
<td>Join</td>
</tr>
<tr>
<td>2</td>
<td>SYMBOL2</td>
<td>color=blue</td>
<td>Dot</td>
<td>Spline</td>
</tr>
<tr>
<td>3</td>
<td>SYMBOL2</td>
<td>color=red</td>
<td>Dot</td>
<td>Spline</td>
</tr>
<tr>
<td>4</td>
<td>SYMBOL2</td>
<td>color=green</td>
<td>Dot</td>
<td>Spline</td>
</tr>
<tr>
<td>5</td>
<td>SYMBOL3</td>
<td>cv=green</td>
<td>Star</td>
<td>NONE</td>
</tr>
<tr>
<td>6</td>
<td>First default</td>
<td>color=blue</td>
<td>First default</td>
<td>Default</td>
</tr>
<tr>
<td>7</td>
<td>First default</td>
<td>color=red</td>
<td>First default</td>
<td>Default</td>
</tr>
<tr>
<td>8</td>
<td>First default</td>
<td>color=green</td>
<td>First default</td>
<td>Default</td>
</tr>
<tr>
<td>9</td>
<td>Second default</td>
<td>color=blue</td>
<td>Second default</td>
<td>Default</td>
</tr>
</tbody>
</table>

Notice that after the SYMBOL statements are exhausted, the procedure begins using the default definitions (sequences 6 through 9). Each plot symbol from the default list is rotated through all colors in the color list before the next plot symbol is used. Also, SYMBOL1 does not specify a plot symbol, so the default sequencing provides the first default symbol (a + sign). When sequencing resumes in sequence number 6, it starts at the beginning again, selecting the first default plot symbol and rotating it through the color list.

If you use the REPEAT= option but no color, the sequence generated by cycling the definition through the color list is repeated the number of times specified by the REPEAT= option. Here is an example of statements that define a color list and illustrate the effect of the REPEAT= option on SYMBOL statements both with and without explicit color specifications:

```plaintext
goptions colors=(blue red green);
symbol1 cv=red i=join;
symbol2 i=spline v=dot;
symbol3 cv=green v=star;
```

In this example, SYMBOL1 is used twice, SYMBOL2 is used once, and SYMBOL3 rotates through the list of three colors and then repeats this cycle a second time:
Table 24.11  Example Output Using 3 SYMBOL Statements

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Source</th>
<th>Characteristics: Color</th>
<th>Symbol</th>
<th>Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYMBOL1</td>
<td>Gold</td>
<td>First default</td>
<td>Default</td>
</tr>
<tr>
<td>2</td>
<td>SYMBOL1</td>
<td>Gold</td>
<td>First default</td>
<td>Default</td>
</tr>
<tr>
<td>3</td>
<td>SYMBOL2</td>
<td>Cyan</td>
<td>Star</td>
<td>Default</td>
</tr>
<tr>
<td>4</td>
<td>SYMBOL3</td>
<td>Blue</td>
<td>Square</td>
<td>Default</td>
</tr>
<tr>
<td>5</td>
<td>SYMBOL3</td>
<td>Red</td>
<td>Square</td>
<td>Default</td>
</tr>
<tr>
<td>6</td>
<td>SYMBOL3</td>
<td>Green</td>
<td>Square</td>
<td>Default</td>
</tr>
<tr>
<td>7</td>
<td>SYMBOL3</td>
<td>Blue</td>
<td>Square</td>
<td>Default</td>
</tr>
<tr>
<td>8</td>
<td>SYMBOL3</td>
<td>Red</td>
<td>Square</td>
<td>Default</td>
</tr>
<tr>
<td>9</td>
<td>SYMBOL3</td>
<td>Green</td>
<td>Square</td>
<td>Default</td>
</tr>
</tbody>
</table>

Here is an example that uses a custom list of colors with the VALUE=SPECIAL option in the SYMBOL statement:

```plaintext
goptions reset=all colors(red green blue cyan);
symbol1 v=special;
```

In this example, assuming that four symbols are needed for a plot, the SYMBOL1 statement iterates through the 12–symbol list and assigns each symbol a color from the color list. Because four colors are specified in the color list, the SYMBOL1 statement generates only four symbols. Here is a summary of the symbols that are generated.

Table 24.12  Example Output of 4 Symbols Generated with 1 SYMBOL Statement

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Source</th>
<th>Characteristics: Color</th>
<th>Symbol</th>
<th>Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYMBOL1</td>
<td>Red</td>
<td>Dot</td>
<td>Default</td>
</tr>
<tr>
<td>2</td>
<td>SYMBOL1</td>
<td>Green</td>
<td>Plus</td>
<td>Default</td>
</tr>
<tr>
<td>3</td>
<td>SYMBOL1</td>
<td>Blue</td>
<td>X</td>
<td>Default</td>
</tr>
<tr>
<td>4</td>
<td>SYMBOL1</td>
<td>Cyan</td>
<td>Triangle</td>
<td>Default</td>
</tr>
</tbody>
</table>

If more than four symbols are needed, SAS/GRAPH proceeds to the next SYMBOL definition. If no more SYMBOL definitions are provided, SAS/GRAPH returns to generating default symbol sequences for the remaining symbols.

If you want to specify the same color for all of the 12 symbols, use the CSYMBOL= graphics option. Or use a color option in the SYMBOL statement to specify the symbol color. You would then use the REPEAT= option to repeat the SYMBOL statement. Here is an example that specifies the color blue for all 12 of the symbols:

```plaintext
symbol1 value=special color=blue repeat=12;
```
The SYMBOL statement in this example repeats 12 times to generate the 12 special symbols. The symbol color is specified in the SYMBOL statement. So each time that the SYMBOL statement is repeated, the next symbol in the 12–symbol list is generated and is assigned the color blue. When a subsequent PLOT statement is executed, the symbol sequence begins again with the first symbol in the 12–symbol list (DOT).

### TITLE, FOOTNOTE, and NOTE Statements

**Control the content, appearance, and placement of text.**

**Valid in:** Anywhere  
**Used by:** GAREABAR, GBARLINE, GCHART, GCONTOUR, GFONT, GKPI, GPLOT, GRADAR, GREPLAY, GSLIDE, GTILE, G3D  
**Type:** TITLE and FOOTNOTE are global statements. NOTE is a local statement.  
**Restriction:** The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

#### Syntax

TITLE<1 …10> <text-argument(s)>;

FOOTNOTE<1 …10> <text-argument(s)>;

NOTE<text-arguments(s)>;

#### Summary of Optional Arguments

**Appearance options**

- **COLOR=** *color*  
  specifies the color for the following text, box, or line.

- **FONT=** *font*  
  specifies the font for the subsequent text.

- **HEIGHT=** *text-height*<units>  
  specifies the height of text characters in number of units.

**Baseline angling and character rotation options**

- **ANGLE=** *degrees*  
  specifies the angle of the baseline of the entire text string with respect to the horizontal.

- **LANGLE=** *degrees*  
  specifies the angle of the entire text-string(s).

- **ROTATE=** *degrees*  
  specifies the angle at which each character of text is rotated.

**Boxing, underlining, and line drawing options**

- **BCOLOR=** *background-color*  
  protects the box and its contents from being overwritten by any subsequent graphics elements.

- **BOX=1 …4**  
  draws a box around one line of text.
BSPACE=box-space<units>
specifies the amount of space between the boxed text and the box.

DRAW=(x,y ….x-n,y-n)<units>
draws lines anywhere on the graphics output area.

UNDERLIN=0 …3
underlines subsequent text.

Linking option

ALT="text-string"
specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself.

LINK="URL"
specifies a uniform resource locator (URL) to which a title or footnote links.

Placement and spacing options

JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text string.

LSPACE=line-space <units>
specifies the amount of spacing above and below lines of text.

MOVE=(x,y) <units>
positions a text string.

WRAP
wraps the text to a second line if the text does not fit on one line.

Text provision option

text-string(s)
is an element of text-argument(s) and specifies one or more strings up to 512 characters. Any of the following text-options that are used to modify a text-string must precede text-string.

Optional Arguments

text-arguments(s) can be one or more of "text-string" and text-options. Text options must precede the "text-string" that they modify. text-options can be one or more of the following optional arguments, in any order:

ALT="text-string"
specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself. The "text-string" that you use to describe a title or a footnote, or the URL specified by the LINK= option, can contain occurrences of the variables #BYVAL, #BYVAR, and #BYLINE, as described in “text-string(s)” on page 460.

Supports
The ALT= option can be used in conjunction with the LINK= option.

Note
The title or footnote can be displayed by using an ODS markup destination (such as HTML). Also, the corresponding ODS option NOGTITLE or NOGFOOTNOTE is specified. In this case, the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. And the ALT= text is not associated with the title or footnote.

See
“Controlling Where Titles and Footnotes Are Rendered” on page 101

“ LINK="URL"” on page 458
ANGLE=degrees

specifies the angle of the baseline of the entire text string with respect to the horizontal. A positive degrees value angles the baseline counterclockwise; a negative value angles it clockwise. By default, ANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using the BOX= option with angled text does not produce angled boxes; the box is sized to accommodate the angled note.

Using the ANGLE= option after one text string and before another can reset some options to their default values. See “Using Options That Can Reset Other Options” on page 464.

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 24.22 on page 449 shows how ANGLE=90-degrees or ANGLE=–90-degrees positions and rotates the title text.

  ANGLE=90 positions the title at the left edge of the graphics output area, angled 90 -degrees (counterclockwise) and centered vertically.

  ANGLE=–90 positions the title at the right edge of the graphics output area, angled –90-degrees (clockwise) and centered vertically.

  Figure 24.22  Positioning Titles with the ANGLE= Option

- With the FOOTNOTE statement:
  Figure 24.23 on page 450 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.
• With the NOTE statement:

Figure 24.24 on page 450 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.

**Alias**  
A=

**Restriction**  
Not supported by Java and ActiveX

**See**  
“ LANGLE=degrees” on page 456

“ ROTATE=degrees” on page 459

**Example**  
“Example 6: Enhancing Titles” on page 485

**BCOLOR=background-color**  
specifies the background color of a box produced by the BOX= option. By default, the background color of the box is the same as the background color for the entire
The color of the frame of the box is determined by the color specification used in BOX=.

**Alias**

BC=

**Restriction**

If you omit BOX=, BCOLOR= is ignored.

**Note**

The BCOLOR= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

**See**

“Using Options That Can Reset Other Options” on page 464 for details

“ BOX=1 …4 ” on page 451

**Example**

“Example 6: Enhancing Titles” on page 485

---

**BLANK=**

Protects the box and its contents from being overwritten by any subsequent graphics elements. It does this by blanking out the area where the box is displayed. The BLANK= option enables you to overlay graphics elements with boxed text. It is ignored if you omit the BOX= option. Because titles and footnotes are written from the highest numbered to the lowest numbered, the BLANK= option only blanks out titles and footnotes of a lower number.

**Alias**

BL=

**Restriction**

Not supported by Java and ActiveX

**Note**

The BLANK= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

**See**

“Using Options That Can Reset Other Options” on page 464 for details

“ BOX=1 …4 ” on page 451

**Example**

“Example 6: Enhancing Titles” on page 485

---

**BOX=1 ...4**

draws a box around one line of text. A value of 1 produces the thinnest box lines; 4 produces the thickest. Boxing angled text does not produce an angled box; the box is sized to include the angled text.

The color of the box is one of the following:

- the color specified by the COLOR= option in the statement
- the default text color

The COLOR= option, preceding the BOX= option, controls the box frame color. To color the background of the box, use the BCOLOR= option. Specify the same color with both the COLOR= option and the BCOLOR= option to effect a box without a frame.

You can include more than one text string in the box as long as no text break occurs between the strings. That is, you cannot use the JUSTIFY= option to create multiple lines of text within a box.

To draw a box around multiple lines of text, you can do either of the following:

- Use the MOVE= option with relative coordinates to position an additional line of text relative to the preceding line and enclose them within the box drawn by the
BOX= option. For example, this statement produces the boxed note shown in Figure 24.25 on page 452:

```
note font=swiss justify=center box=1
  "Office Hours"  move={-7,-2}pct "9-5";
```

The MOVE= option in this NOTE statement overrides the JUSTIFY= option that center-positioned the first line of text. The MOVE= option specifies the x and y coordinates in percentage units. The text string ‘9–5’ is positioned relative to the first text string ‘Office Hours’.

Using the MOVE= option in a NOTE statement enables you to position a note on top of a graph. See the “NOTE Statement” for a code sample.

- Use the DRAW= option to draw the box and do not use the BOX= option.

**Figure 24.25**  Using the BOX= Option and the MOVE= Option to Box Multiple Lines of Text

```
Office Hours
9 – 5
```

**Alias**  
BO=

**Restriction**  
Not supported by Java and ActiveX

**Note**  
The BOX= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= options with absolute coordinates.

**See**  
“Using Options That Can Reset Other Options” on page 464 for details

- “BCOLOR=background-color” on page 450
- “BLANK=YES” on page 451
- “BSPACE=box-space<units>” on page 452
- “Example 6: Enhancing Titles” on page 485

**BSPACE=box-space<units>**

specifies the amount of space between the boxed text and the box. The space above the text is measured from the font maximum, and the space below the text is measured from the font minimum. By default, BSPACE=1.

The spacing is uniform around the box. Notice that BSPACE=.5IN leaves one-half inch of space between the text and the top, bottom, and sides of the box.

**Alias**  
BS=
Restriction | Not supported by Java and ActiveX  
Interaction | If the BOX= option is not used, the BSPACE= option is ignored  
Note | The BSPACE= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.  
See | “Using Options That Can Reset Other Options” on page 464 for details  

**COLOR=**<i>color</i>  
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:  
```plaintext
title color=red "Total Sales" color=blue  
   box=3 bcolor=cream;
```

Although the BCOLOR= option controls the background color of the box, the frame color is controlled with the COLOR= option that precedes the BOX= option.  

If you omit the COLOR= option, a color specification is searched for in this order:  
1. the CTITLE= option in a GOPTIONS statement  
2. the CTEXT= option in a GOPTIONS statement  
3. the default, the first color in the color list.  

**Alias**  
C=  

**Style reference**  
Color attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements  

**See**  
“BCOLOR=background-color” on page 450  
“Controlling Titles and Footnotes” on page 101  

**DRAW=(x,y …x-n,y-n)<units>**  
draws lines anywhere on the graphics output area. Lines are drawn using x and y as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:  

<table>
<thead>
<tr>
<th>Absolute Coordinates</th>
<th>Relative Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>x&lt;units&gt;</td>
<td>±x&lt;units&gt;</td>
</tr>
<tr>
<td>y&lt;units&gt;</td>
<td>±y&lt;units&gt;</td>
</tr>
</tbody>
</table>

The coordinate position (0,0) is the lower left corner of the graphics output area. Specify at least two coordinate pairs. Commas between coordinates are optional; blanks can be used instead. The DRAW= option does not affect the positioning of text.
The starting point for lines specified with relative coordinates begins at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued. The relative draw is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates. For example, DRAW=(+0,+0,+0,1IN) draws a vertical line from the end of the text to one inch from the bottom of the graphics output area.

**Alias**

D=

**Restriction**

Not supported by Java and ActiveX

**Example**

“Example 6: Enhancing Titles” on page 485

**FONT=font**

specifies the font for the subsequent text. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about specifying SAS/GRAPH fonts. If you omit this option, a font specification is searched for in this order:

- for a TITLE1 statement:
  1. the FTITLE= option in a GOPTIONS statement
  2. the FTEXT= option in a GOPTIONS statement
  3. the default font, SWISS (COMPLEX in Release 6.06 and earlier)
- for all other TITLE statements and the FOOTNOTE and NOTE statements:
  1. the FTEXT= option in a GOPTIONS statement
  2. the default hardware font, NONE

**Alias**

F=

**Style reference**

Font attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements

**Notes**

Font names greater than eight characters in length must be enclosed in quotation marks.

If the TITLE or FOOTNOTE is displayed using an ODS markup destination and the corresponding NOGTITLE or NOGFOOTNOTE option is specified, then the **bold** and **italic** FONT attributes are on by default. However, if you specify different attributes with the FONT= option, the **bold** and **italic** attributes are turned off.

**See**

“Controlling Titles and Footnotes” on page 101

**Example**

“Example 6: Enhancing Titles” on page 485

**HEIGHT=text-height<units>**

specifies the height of text characters in number of units.

Height is measured from the font minimum to the capline. Ascenders can extend above the capline, depending on the font.

If your text line is too long to be displayed in the height specified in the HEIGHT= option, the height specification is reduced so that the text can be displayed. A note in the SAS log tells you what percentage of the specified size was used.
If you omit the HEIGHT= option, a text height specification is searched for in this order:

- for a TITLE or TITLE1 statement:
  1. the HTITLE= option in a GOPTIONS statement
  2. the HTEXT= option in a GOPTIONS statement
  3. the default value specified by the current ODS style.
- for all other TITLE statements and the FOOTNOTE and NOTE statements:
  1. the HTEXT= option in a GOPTIONS statement
  2. the default value, 1

### Alias
H=

### Default
For TITLE or TITLE1 statements, HEIGHT is determined by the current ODS style. HEIGHT=1 for subsequent TITLE statements.

### Style reference
Height attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements

### Restriction
Partially supported by Java and ActiveX

### Notes
The Java applet and ActiveX control enable you to control the relative height of text with the HEIGHT= option, but not the absolute height in terms of specific units.

A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement.

### See
“Controlling Titles and Footnotes” on page 101

### Examples
“Example 1: Ordering Axis Tick Marks with SAS Date Values” on page 465

“Example 6: Enhancing Titles” on page 485

**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:

```plaintext
note "June 28, 1997" justify=right "Page 3";
```

In addition, you can use the JUSTIFY= option to produce multiple lines of text by repeating the JUSTIFY= option with the same value before the text string for each line. Multiple lines of text with the same justification are blocked together. For
example, this TITLE statement produces a three-line title with each line right-justified:

```
TITLE justify=right *First Line*
     justify=right *Second Line*
     justify=right *Third Line*;
```

You can get the same effect with three TITLE statements, each specifying JUSTIFY=RIGHT. You can produce a block of text by specifying the same justification for multiple text strings. You can then change the justification for an additional text string. The result is that the text is placed on the same line as the first string specified in the statement.

**Alias**

J=L | C | R

**Note**

Using the JUSTIFY= option after one text string and before another can reset some options to their default values.

**See**

“Using Options That Can Reset Other Options” on page 464 for details

**Example**

“Example 3: Rotating Plot Symbols through the Color List” on page 473

**LANGLE=degrees**

specifies the angle of the baseline of the entire text string(s) with respect to the horizontal. A positive value for degrees moves the baseline counterclockwise; a negative value moves it clockwise. By default, LANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase in 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 24.26 on page 457** 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.
With the FOOTNOTE statement:

*Figure 24.27 on page 457* 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.

With the NOTE statement:

*Figure 24.28 on page 457* 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.

Alias LA=

Restriction Not supported by Java and ActiveX
LINK="URL"
specifies a uniform resource locator (URL) to which a title or footnote links. The
text-string that you use to specify the URL can contain occurrences of the variables
#BYVAL, #BYVAR, and #BYLINE, as described in “text-string(s)” on page 460.

Supports
The LINK= option can be used in conjunction with the ALT= option.

Note
The title or footnote can display using an ODS markup destination (such as HTML). And the corresponding ODS option NOGTITLE or
NOGFOOTNOTE can be specified. Then the title or footnote is rendered
in the body of the HTML file rather than in the graphic itself. Specifying
the NOGTITLE or NOGFOOTNOTE options results in increasing the
amount of space allowed for the procedure output area, which can result
in increasing the size of the graph. Space that would have been used for
the title or footnote is devoted instead to the graph. You might need to be
aware of this possible difference if you are using annotate or map
coordinates.

Example
“Using Traditional Map Data to Produce a Drilldown Choropleth Map”
in SAS/GRAPH and Base SAS: Mapping Reference

LSPACE=\text-line-space <units>
specifies the amount of spacing above lines of note and title text and the amount of
spacing below lines of footnote text. For notes and titles, the spacing is measured
from the capline of the current line to the font minimum of the line above. For
footnotes, the spacing is measured from the font minimum of the current line to the
capline of the line below.

Alias
LS=

Default
LSPACE=0

Restriction
Not supported by Java and ActiveX

Note
The LSPACE= option can be reset by the ANGLE= or JUSTIFY=
option, or by the MOVE= option with absolute coordinates.

See
“Using Options That Can Reset Other Options” on page 464 for details

MOVE=(x,y) <units>
positions an initial or subsequent text string, or lines of text anywhere on the
graphics output area. The positioning uses x and y as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

<table>
<thead>
<tr>
<th>Absolute Coordinates</th>
<th>Relative Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>x&lt;units&gt;</td>
<td>±x&lt;units&gt;</td>
</tr>
<tr>
<td>y&lt;units&gt;</td>
<td>±y&lt;units&gt;</td>
</tr>
</tbody>
</table>
Commas between coordinates are optional; you can use blanks instead.

The starting point for lines specified with absolute coordinates is at the bottom left corner of the graphics output area. For example, the following code fragment uses the NOTE statement to add a simple line of text to the upper right quadrant of the graphics output area:

```
note move=(65,70)pct color=purple "My Note";
```

The starting point for lines specified with relative coordinates is at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued in the SAS log. The relative move is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates.

**Alias**

M=

**Restriction**

Not supported by Java and ActiveX

**Interactions**

The MOVE= option overrides a JUSTIFY= option specified for the same text string.

A NOTE, FOOTNOTE, or TITLE statement can use the MOVE= option to position the text so that the statement does not use its default position. In this case, the text of the next NOTE, FOOTNOTE, or TITLE statement occupies the unused position and no blank lines are displayed.

**Notes**

You can specify the MOVE= option with at least one absolute coordinate. If the option follows one text string and precedes another, some options can be reset to their default values.

If you specify the GUNIT graphics option, then that unit is the default unit. If you do not specify the GUNIT= graphics option, then the default unit is CELLS.

**See**

“Using Options That Can Reset Other Options” on page 464 for details

“BOX=1 …4 ” on page 451 for illustrative code that draws a box around lines of text positioned with relative coordinates

**Examples**

“Example 2: Specifying Logarithmic Axes” on page 469

“Example 6: Enhancing Titles” on page 485

**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 24.29 on page 460 shows how characters are positioned when ROTATE=90 is used with the default (horizontal) baseline.
Figure 24.29  Tilting Characters with the ROTATE= Option

<table>
<thead>
<tr>
<th>Alias</th>
<th>R=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>Not supported by Java and ActiveX</td>
</tr>
<tr>
<td>See</td>
<td>“ANGLE=degrees” on page 449</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 6: Enhancing Titles” on page 485</td>
</tr>
</tbody>
</table>

text-string(s)
is an element of text-argument(s) and specifies one or more strings up to 512 characters. Any of the following text-options that are used to modify a text-string must precede text-string.

You must enclose text strings in single or double quotation marks. The text appears exactly as you enter it in the statement, including uppercase and lowercase characters and blanks.

To use single quotation marks or apostrophes within the title, you can do either of the following:
- use a pair of single quotation marks together:
  footnote 'All''s Well That Ends Well';
- enclose the text in double quotation marks:
  footnote "All's Well That Ends Well";

You can specify multiple text strings in FOOTNOTE and TITLE statements. If you want each string to be separated by a blank, add a blank to the beginning of the second and subsequent text strings rather than at the end the preceding string, as in this example:

```
note color=red "Sales:" color=blue " 2000";
```

With fonts that support Unicode, you can specify special characters by specifying a hexadecimal constant in quotation marks. A trailing x identifies a string as a hexadecimal constant. You must designate the hexadecimal constant as a Unicode specification and escape it: (*ESC*){Unicode 'hexadecimal-value'x}. For example:

```
title "Regression with Confidence Limits ((*ESC*)\{unicode '03B1'x\}=.05)";
```

This statement produces the title, "Regression with Confidence Limits (α = .05)" because '03B1'x is the hexadecimal value for the lowercase Greek letter alpha in all Unicode fonts.

**TIP** You can instead specify lowercase Greek letters by name, and uppercase Greek letters by name_u: (*ESC*){Unicode alpha}=.05 and (*ESC*){Unicode alpha_u}=.05.

For more information see "Specifying Special Characters Using Character and Hexadecimal Codes" on page 304.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:
#BYLINE
substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string. It also displays the BY line in the footnote, note, or title produced by the statement.

#BYVAL\text{\textbackslash n} | #BYVAL(\text{BY-variable-name})
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

- \text{\textbackslash n}
specifies which variable in the BY statement #BYVAL should use. The value of \text{\textbackslash n} indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

- \text{BY-variable-name}
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. \text{Variable-name} is not case sensitive.

Examples

"Example 7: Using BY-group Processing to Generate a Series of Charts" on page 488

"Example 8: Combining Graphs and Reports in a Web Page" on page 495

#BYVAR\text{\textbackslash n} | #BYVAR(\text{BY-variable-name})
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

- \text{\textbackslash n}
specifies which variable in the BY statement #BYVAR should use. The value of \text{\textbackslash n} indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

- \text{BY-variable-name}
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. \text{Variable-name} is not case sensitive.

Note
A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See
"Substituting BY Line Values in a Text String" on page 465

Restriction
#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

UNDERLIN=0 …3
underlines subsequent text. Values of 1, 2 and 3 underline with an increasingly thicker line. UNDERLIN=0 halts underlining for subsequent text.

Underlines follow the text baseline. If you use an LANGLE= or ANGLE= option for the line of text, the underline is drawn at the same angle as the text. Underlines do not break up to follow rotated characters.

To make the text and the underline the same color, specify a COLOR= option before the UNDERLIN= option that precedes the text string. To make the text a different color, specify the COLOR= option after the UNDERLIN= option.
Aliases

<table>
<thead>
<tr>
<th>Aliases</th>
<th>U=</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDERLINE=</td>
<td></td>
</tr>
</tbody>
</table>

Restriction

| Restriction | Partially supported by Java and ActiveX |

Notes

<table>
<thead>
<tr>
<th>Notes</th>
<th>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 464 for details.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td>The Java applet and ActiveX control underline text when the UNDERLIN= option is specified, but they do not vary the thickness of the line.</td>
</tr>
</tbody>
</table>

See

<table>
<thead>
<tr>
<th>See</th>
<th>“ROTATE=degrees” on page 459</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td>“Example 6: Enhancing Titles” on page 485</td>
</tr>
</tbody>
</table>

Example

| Example | “Example 6: Enhancing Titles” on page 485 |

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

| Restriction | The WRAP option does not work with the BOX, BLANK, UNDERLINE, and MOVE options. |

Details

Using TITLE and FOOTNOTE Statements

You can define TITLE and FOOTNOTE statements anywhere in your SAS program. They are global and remain in effect until you cancel them or until you end your SAS session. All currently defined FOOTNOTE and TITLE statements are automatically displayed.

You can define up to ten TITLE statements and ten FOOTNOTE statements in your SAS session. A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement. You do not have to start with TITLE1 and you do not have to use sequential statement numbers. Skipping a number in the sequence leaves a blank line.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See “Using Multiple Options” on page 463.

The most recently specified TITLE or FOOTNOTE statement of any number completely replaces any other TITLE or FOOTNOTE statement of that number. In addition, it cancels all TITLE or FOOTNOTE statements of a higher number. For example, if you define TITLE1, TITLE2, and TITLE3, then resubmitting the TITLE2 statement cancels TITLE3.

To cancel individual TITLE or FOOTNOTE statements, define a TITLE or FOOTNOTE statement of the same number without options (a null statement):

title4;

But remember that this cancels all other existing statements of a higher number.

To cancel all current TITLE or FOOTNOTE statements, use the RESET= graphics option in a GOPTIONS statement:
Specifying RESET=GLOBAL or RESET=ALL also cancels all current TITLE and FOOTNOTE statements as well as other settings.

*Note:* The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

**Using the NOTE Statement**

NOTE statements are local, not global, and they must be defined within a procedure or RUN-group with which they are used. They remain in effect for the duration of the procedure that includes NOTE statements in any of its RUN-groups or until you end your SAS session. All notes defined in the current RUN group, as well as those defined in previous RUN-groups, are displayed in the output as long as the procedure remains active.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See “Using Multiple Options” on page 463.

**Using Multiple Options**

In each statement, you can use as many text strings and options as you want, but you must place the options before the text strings that they modify. Most options affect all text strings that follow them in the same statement, unless the option is explicitly reset to another value. In general, TITLE, FOOTNOTE, and NOTE statement options stay in effect until one of these events occurs:

- The end of the statement is reached.
- A new specification is made for that option.

For example, this statement specifies that one part of the note is red and another part is blue, but the height for all of the text is 4:

```sas
note height=4 color=red "Red Tide"
    color=blue "Effects on Coastal Fishing";
```

**Controlling Placement of Text**

The following options enable the positioning of text on the graphics output area:

- “JUSTIFY=LEFT | CENTER | RIGHT” on page 455 aligns the text string
- “MOVE=(x,y) <units>” on page 458 positions a line or lines of text anywhere on the graphics output area using x and y as absolute or relative coordinates.

Figure 24.25 on page 452 illustrates using the MOVE= option with relative coordinates to position a subsequent line of text
- “WRAP” on page 462 wraps the text to a second line

See the description of each option for usage details and restrictions.

**Setting Defaults**

You can set default characteristics for titles (including TITLE1 definitions), footnotes, and notes by using the following graphics options in a GOPTIONS statement:

- `CTITLE=color`
  - sets the default color for all titles, footnotes, and notes; overridden by the `COLOR=` option in a TITLE, FOOTNOTE, or NOTE statement.
CTEXT=text-color
    sets the default color for all text; overridden by the CTITLE= option for titles, footnotes, and notes.

FTITLE=title-font
    sets the default font for TITLE1 definitions; overridden by the FONT= option in the TITLE1 statement.

FTEXT=text-font
    sets the default font for all text, including the TITLE1 statement if the FTITLE= option is not used; overridden by the FONT= option a TITLE, FOOTNOTE, or NOTE statement.

HTITLE=height<units>
    sets the default height for TITLE1 definitions; overridden by the HEIGHT= option in the TITLE1 statement.

HTEXT=n<units>
    sets the default height for all text, including the TITLE1 statement if the HTITLE= option is not used; overridden by the HEIGHT= option a TITLE, FOOTNOTE, or NOTE statement.


Using Options That Can Reset Other Options
The ANGLE=, MOVE=, and JUSTIFY= options affect the position of the text and cause text breaks. (To cause a text break, the MOVE= option must have at least one absolute coordinate.) When a statement contains multiple text strings, the resulting text break can cause the following options to reset to their default values:

• BCOLOR=
• BLANK=
• BOX=
• BSPACE=
• LSPACE=
• UNDERLIN=.

Note: The LANGLE= option does not cause a text break.

In a TITLE, FOOTNOTE, or NOTE statement, before the first text string, you can use an option that can be reset (such as the UNDERLIN= option). Before the second string, if you use an option that resets it (such as the JUSTIFY= option), the first option does not affect the second string. In order for the first option to affect the second string, repeat the option and position it after the resetting option and before the text string.

For example, this statement produces a two-line title in which only the first line is underlined:

    title underlin=2 "Line 1" justify=left "Line 2";

To underline Line 2, repeat the UNDERLIN= option before the second text string and after the JUSTIFY= option:

    title underlin=2 "Line 1" justify=left
        underlin=2 "Line 2";
Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the “BY Statement” on page 370.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

Examples

Example 1: Ordering Axis Tick Marks with SAS Date Values

Features: AXIS statement options

LABEL=
OFFSET=
ORDER=

FOOTNOTE statement option

JUSTIFY=

SYMBOL statement options

INTERPOL=
WIDTH=

GOPTIONS statement options

BORDER
Sample library member: GAXTMDV1

Details
The code for this example is in SAS Sample Library member GAXTMDV1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example uses SAS datetime values with an AXIS statement's ORDER= option to set the major tick marks on the horizontal axis. It adjusts the position of the first and last major tick marks. The example also uses HILOCTJ interpolation in a SYMBOL statement to join minimum and maximum values.

Output

Program

goptions reset=all border;

data dowhlc;
    input date date9. high low close;
    format date date9.;
    datalines;
02JAN1997 6511.38 6318.96 6442.49
03JAN1997 6586.42 6437.10 6544.09
06JAN1997 6647.22 6508.30 6567.18
07JAN1997 6621.82 6481.75 6600.66
Program Description

Set the graphics environment. BORDER draws a border around the graph.

goptions reset=all border;
Create the data set. DOWHLC contains the high, low, and close values of the Dow
Jones Industrial index for each business day for a month.

```sas
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  6726.68
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.

```sas
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.

```sas
  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.

```sas
  symbol interpol=hiloctj
      cv=red
```
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.

```sas
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.

```sas
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.

```sas
proc gplot data=dowhlc2;
  plot dow*date / haxis=axis1
    vaxis=axis2;
run;
quit;
```

Example 2: Specifying Logarithmic Axes

**Features:**
- **AXIS statement options**
  - LABEL=
  - LENGTH=
  - LOGBASE=
  - LOGSTYLE=
  - MAJOR=
  - MINOR=
  - VALUE=
- **TITLE statement options**
  - MOVE=
- **GOPTIONS statement options**
  - GUNIT

**Sample library member:** GAXSPLA1

**Details**

The code for this example is in SAS Sample Library member GAXSPLA1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example illustrates the AXIS statement options LOGBASE= and LOGSTYLE=. The horizontal axis represents pH level. The vertical axis, which represents the concentration of the hydroxide ion expressed as moles per liter, is scaled logarithmically, using a base of 10. In addition, this example shows how the TICK= parameter of the
VALUE= option modifies individual tick marks. It also uses the MOVE= option in a TITLE statement to position the title's subscript and superscript text.

Output

Program

goptions reset=all gunit=pct;

data concentr;
   input pH conc;
   datalines;
1 1E-1
2 1E-2
3 1E-3
4 1E-4
5 1E-5
6 1E-6
7 1E-7
8 1E-8
9 1E-9
10 1E-10
11 1E-11
12 1E-12
13 1E-13
14 1E-14
;
run;
Program Description

**Set the graphics environment.** The GUNIT option specifies the default unit of measure to use with height specifications.

```plaintext
goptions reset=all  gunit=pct;
```

**Create the data set.** The data set is named CONCENTR and it contains the pH values and the concentration amount.

```plaintext
data concentr;
  input pH conc;
  datalines;
  1 1E-1
```

```plaintext
proc gplot data= concentr;
  plot conc*pH / haxis=axis1
       vaxis=axis2
       autohref chref=graydd
       autovref cvref=graydd;
run;
quit;
```
Define the title. The MOVE= option positions subscript 3 and superscript +. Each new position is relative to the last position specified by the MOVE= option.

```plaintext
title1 h=3.7 "Relationship of pH to H"  move=(-0,-.75) h=2 "3"  move=(+0,+.75) h=3.7 "0"  move=(+0,+1.5) h=2 "+"  move=(-0,-1.5) h=3.7 " Concentration";
```

Define the symbol characteristics.

```plaintext
symbol value=dot color=black height=2;
```

Define the horizontal axis characteristics. The LABEL= option uses the JUSTIFY= suboption to create a descriptive two-line label that replaces the variable name pH. MINOR=NONE removes all minor tick marks. The LENGTH= option controls the length of the horizontal axis. The OFFSET= option specifies the distance from the first and last major tick marks to the ends of the axis line.

```plaintext
axis1 label=(h=3 "Scale of pH Values"  justify=left color=red h=2 "More acid"  justify=right color=blue "More alkaline")  minor=none  length=60  offset=(2,2);
```

Define the vertical axis characteristics. LOGBASE=10 scales the vertical axis logarithmically, using a base of 10. Each major tick mark represents a power of 10. LOGSTYLE=EXPAND displays minor tick marks in logarithmic progression. The LABEL= option uses the ANGLE= suboption to place the label parallel to the vertical axis. The VALUE= option displays the major tick mark values as 10 plus an exponent. The HEIGHT= suboption for each TICK= specification affects only the text following it.

```plaintext
axis2 logbase=10  logstyle=expand  label=(angle=90 h=2 color=black  "Concentration (Moles/Liter)" )  value={tick=1 "10" height=1.2 "-14"  tick=2 "10" height=1.2 "-13"  tick=3 "10" height=1.2 "-12"  tick=4 "10" height=1.2 "-11"}
```
Generate the plot and assign AXIS definitions. AXIS1 modifies the horizontal axis and AXIS2 modifies the vertical axis. The AUTOHREF and AUTOVREF options draw reference lines at all major tick marks on both axes. The CHREF and CVREF options specify the color for these reference lines.

```
proc gplot data= concentr;
  plot conc*pH / haxis=axis1
       vaxis=axis2
       autohref chref=graydd
       autovref cvref=graydd;
run;
quit;
```

Example 3: Rotating Plot Symbols through the Color List

Features:
- GOTIONS statement options
  - COLORS=
  - BORDER
- LEGEND statement options
  - LABEL=
  - JUSTIFY=
- SYMBOL statement options
  - VALUE=
- TITLE statement options
  - JUSTIFY=
  - HEIGHT=

Sample library member:
GSYRPSC1

Details
The code for this example is in SAS Sample Library member GSYRPSC1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example specifies a plot symbol in a SYMBOL statement and rotates the symbol through the specified color list. Temperature values in the data are represented by the same plot symbol in a different color. The example also shows how default symbol sequencing provides a default plot symbol if a plot needs more plot symbols than are
defined. It also uses a LEGEND statement to specify a two-line legend label and to align the label with the legend values.

Output

Program

goptions reset=all border;

data bacteria;
  input temp div mass life @@;
  datalines;
10 3 10 1 20 22 46 0 30 23 20 9 40 42 16 16 50 33 20 6
10 1 11 2 20 01 44 2 30 21 31 10 40 41 14 12 50 31 21 7
10 4 14 3 20 13 32 4 30 24 34 9 40 43 22 14 50 34 24 2
10 2 09 2 20 12 40 6 30 26 29 8 40 42 20 16 50 26 29 4
10 3 08 3 20 09 33 8 30 24 38 11 40 39 23 18 50 34 38 2
10 2 09 1 20 08 38 1 30 25 47 14 40 38 18 12 50 43 44 1
10 4 10 3 20 15 42 3 30 29 30 14 40 35 22 14 50 39 20 8
10 3 11 2 20 20 36 5 30 28 31 9 40 40 26 15 50 28 31 0
10 2 15 3 20 19 35 7 30 26 25 11 40 39 25 17 50 26 15 4
10 4 12 3 20 14 33 2 30 27 22 8 40 36 23 12 50 27 22 3
10 4 13 3 20 12 37 4 30 26 33 9 40 42 27 14 50 26 33 5
10 2 17 1 20 10 39 6 30 25 43 13 40 40 29 16 50 35 43 7
10 3 14 1 20 08 38 4 30 28 34 8 40 38 28 14 50 28 34 4
10 1 12 1 20 06 41 2 30 26 32 14 40 36 21 12 50 21 22 2
10 1 11 4 20 09 32 2 30 27 31 8 40 39 22 12 50 37 31 2
10 1 20 2 20 11 32 5 30 25 32 16 40 41 22 15 50 35 22 5
10 4 09 2 20 13 39 1 30 28 29 12 40 43 19 15 50 28 29 1
10 3 02 2 20 09 32 5 30 26 32 9 40 39 22 15 50 36 22 5
10 2 05 3 20 07 35 4 30 24 35 15 40 37 25 14 50 24 35 4
10 3 08 1 20 05 38 6 30 23 28 9 40 35 28 16 50 33 28 6
;
proc sort data=bacteria;
  by temp;
run;

title1 ls=1.5 h=2 "Effect of Temperature on the Number" j=c "and Size of Bacterial Divisions";
footnote1 j=r "GSYRPSC1";
symbol1 value=dot;
axis1 label=("Size (in Angstroms)");
axis2 label=("Divisions");
legend1 label=(position=(top left) j=l "Temperature" j=1 "(Celsius)"
  "Temperature" j=1 "(Celsius)"
  );
proc gplot data=bacteria;
  plot div*mass=temp / haxis=axis1 vaxis=axis2 legend=legend1;
run;
quit;

Program Description

Set the graphics environment. The COLORS= graphics option specifies the color list. This list is used by the SYMBOL statement if a color option is not specified on that statement.

gooptions 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 16 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 16 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
Define title and footnote. LS= adds some space between the title and the border of the graphics output area. Positioning J= in the middle of the title breaks the title into two lines. H= specifies the size of the entire title.

```sas
   title1 ls=1.5 h=2 "Effect of Temperature on the Number"
                j=c "and Size of Bacterial Divisions";
   footnote1  j=r "GSYRPSC1";
```

Define symbol shape. The VALUE= option specifies a dot for the plot symbol. Because no color is specified, the symbol is rotated through the graphics option color list. Because the plot needs a fifth symbol, the default plus sign is rotated into the color list to provide that symbol.

```sas
   symbol1  value=dot;
```

Define axis characteristics.

```sas
   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.

```sas
   legend1 label=(position=(top left)
                  "Temperature" j=l *(Celsius));
```

Generate the plot.

```sas
   proc gplot data= bacteria;
   plot div*mass=temp / haxis=axis1
                   vaxis=axis2
                   legend=legend1;
   run;
   quit;
```

---

**Example 4: Creating and Modifying Box Plots**

**Features:**

- GOTOINS statement options
- BORDER
- AXIS statement options
  - LABEL=
  - LENGTH=
  - OFFSET=
  - VALUE=
- SYMBOL statement options
Sample library member: GSYCMBP1

Details

The code for this example is in SAS Sample Library member GSYCMBP1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example shows how to create box plots and how to specify SYMBOL definitions so that data outside the box-plot range can be represented with data points. It also shows how to change a box plot’s percentile range to see whether the new range encompasses the data.

Output

The first plot in the example uses a SYMBOL definition with INTERPOL=BOXT20. to specify a box plot with whisker tops at the 80th percentile. It also specifies whisker bottoms at the 20th percentile. Data points that are outside this percentile range are represented with squares.
As illustrated in the following output, the example then changes the SYMBOL definition to \texttt{INTERPOL=BOXT10}. This definition expands the whisker range to the 90th
percentile for tops and the 10th percentile for bottoms. There are no data points outside
the new percentile range.

Program

goptions reset=all border;

data grades;
  input section $ grade @@;
  datalines;
  A 74 A 89 A 91 A 76 A 87 A 93 A 93 A 96 A 55
  B 72 B 72 B 84 B 81 B 97 B 78 B 88 B 90 B 74
  C 62 C 74 C 71 C 87 C 68 C 78 C 80 C 85 C 82
  ;
  title1 "Comparison: Grades by Section";
  footnote1 j=r "GSYCMBP1(a) ";
symbol interpol=boxt20 /* box plot */
   co=blue /* box and whisker color */
   bwidth=4 /* box width */
   value=square /* plot symbol */
   cv=red /* plot symbol color */
   height=2; /* symbol height */

axis1 label=none
   value=(t=1 "Monday" j=c "section"
   t=2 "Wednesday" j=c "section"
   t=3 "Friday" j=c "section")
   offset=(5,5)
   length=50;

proc gplot data= grades;
   plot grade*section / haxis=axis1
   vaxis=50 to 100 by 10;
run;

footnote j=r GSYCMBP1(b);

symbol interpol=boxt10 width=2;
plot grade*section / haxis=axis1
   vaxis=50 to 100 by 10;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set. GRADES contains codes to identify each class section, and the grades scored by students in each section.

data grades;
   input section $ grade @@;
   datalines;
   A 74 A 89 A 91 A 76 A 87 A 93 A 93 A 96 A 55
   B 72 B 72 B 84 B 81 B 97 B 78 B 88 B 90 B 74
   C 62 C 74 C 71 C 87 C 68 C 78 C 30 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 */
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;
```
Details

The code for this example is in SAS Sample Library member GSYFAPL1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example shows how to fill the area between two plot lines by concatenating two data sets into one to form a polygon with the data points. It uses a SYMBOL statement to specify a pattern to fill the polygon and to determine the color of the area fill and the outline around the area.

The example plots yearly highs and lows for the Dow Jones Industrial Average. It separates the dependent variables HIGH and LOW to produce an upper plot line and a lower plot line. The dependent variable is named VALUE and the independent variable is named YEAR. When concatenated into one data set, AREA, the data sets form the polygon.

Output

![Dow Jones Industrial Average Graph](Image)

Source: 1997 World Almanac

Program

```sas
options reset=all border;

data stocks;
  input year @7 hdate date9. @17 high @26 ldate date9. @36 low;
  format hdate ldate date9.;
datalines;
```

Example 5: Filling the Area between Plot Lines

1981  27APR1981  1024.05  25SEP1981  824.01
1982  27DEC1982  1070.55  12AUG1982  776.92
1983  29NOV1983  1287.20  03JAN1983  1184.96
1984  16DEC1984  1553.10  04JAN1984  1184.96
1985  02DEC1985  1955.57  22JAN1985  1502.29
1986  25AUG1986  2722.42  19OCT1986  1738.74
1987  21OCT1987  2183.50  20JAN1987  1879.14
1988  09OCT1988  2791.41  03JAN1988  2144.64
1989  16JUL1989  2999.75  11OCT1989  2365.10
1990  31DEC1990  3168.83  09JAN1990  2470.30
1991  01JUN1991  3413.21  09OCT1991  3136.58
1992  29DEC1992  3794.33  20JAN1992  3241.95
1993  31JAN1993  3978.36  04APR1993  3593.35
1994  13DEC1994  5216.47  30JAN1994  3832.08

; data high(keep=year value)
  low(keep=year value);
set stocks;
  value=high; output high;
  value=low; output low;
run;
proc sort data=low;
    by descending year;
run;
data area;
set high low;
run;

  title1 "Dow Jones Industrial Average";
  title2 "Highs and Lows From 1980 to 1995";
  footnote " Source: 1997 World Almanac"
    j=r "GSYFAPL1 ";
  symbol interpol=m3n90
    cv=red
    co=blue;
  axis1 order=(1980 to 1995 by 5)
    label=none
    major=(height=2)
    minor=(number=4 height=1)
    offset=(2,2)
    width=3;
  axis2 order=(0 to 5500 by 500)
    label=none
    major=(height=1.5) offset=(0,0)
    minor=(number=1 height=1);
proc gplot data=area;
  plot value*year / haxis=axis1
    vaxis=axis2
    vref=(1000 3000 5000);
run;
quit;
Program Description

Set the graphics environment.

```sas
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.

```sas
data stocks;
  input year @7 hdate date9. @17 high @26 ldate date9. @36 low;
  format hdate ldate date9. ;
  datalines;
  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
  1989 09OCT1989 2791.41 03JAN1989 2144.64
  1990 16JUL1990 2999.75 11OCT1990 2365.10
  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.

```sas
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.

```sas
proc sort data=low;
  by descending year;
run;
```

Concatenate HIGH and LOW to create data set AREA.

```sas
data area;
  set high low;
run;
```

Define titles and footnote.

```sas
title1 "Dow Jones Industrial Average";
```
Define symbol characteristics. The INTERPOL= option specifies a map pattern or a plot pattern to fill the polygon formed by the data points. The pattern consists of medium-density parallel lines at 90 degrees. The CV= option colors the pattern fill. The CO= option colors the outline of the area. (If the CO= option is not used, the outline is the color of the area.)

```plaintext
symbol interpol=m3n90
cv=red
co=blue;
```

Define axis characteristics. The ORDER= option places the major tick marks at 5-year intervals.

```plaintext
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.

```plaintext
proc gplot data=area;
   plot value*year / haxis=axis1
      vaxis=axis2
      vref=(1000 3000 5000);
run;
quit;
```
Sample library member: GTIENTI1

Details

The code for this example is in SAS Sample Library member GTIENTI1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example illustrates some ways that you can format title text. The same options can be used to format footnotes.

Output

Program

```sas
options reset=all border;

title1 "This is TITLE1"
height=4;

title3 underlin=1
  "TITLE3 is Underlined"

title4 color=red
angle=-90
```

TITILE is Boxed

with Explicit Moves, and is Overlaid by TITLE9

TITLE9 is Not Overlaid BY TITLE10
"TITLE4 is Angled -90 and Positioned on the Right Side of the Output";

title5
    color=brown
    rotate=25
    "TITLE5 is Rotated";

title7 color=green
    box=1
    "TITLE7 is Boxed";

title9 color=black
    box=3
    blank=yes
    bcolor=yellow
    color=blue
    angle=-25
    "TITLE9 is Not Overlaid By TITLE10";

title10 color=purple
    draw=(20,35 20,27 58,27 58,35 20,35 20,35)
    move=(20,32)
    font=script
    "TITLE10 is in Script and *
    move=(20,27)
    height=2
    "is Partially Boxed, Positioned"
    move=(20,22)
    height=2
    "with Explicit Moves, and is Overlaid by TITLE9";

proc gslide;
    run;
    quit;

Program Description

**Set the graphics environment.** BORDER draws a border around the graph.

    goptions reset=all border;

**Define TITLE1.** TITLE1 uses the default font and height defined in the default style. The HEIGHT= option sets the height of the text.

    title1 *This is TITLE1"
    height=4;

**Define TITLE3.** The UNDERLIN= option underlines both text strings.

    title3 underlin=1
    "TITLE3 is"
    color=blue
    " Underlined";

**Define TITLE4.** The ANGLE= option tilts the line of text clockwise 90 degrees and places it at the right edge of the output, centered vertically.

    title4 color=red
    angle=-90
Define TITLE5. The ROTATE= option rotates each character in the text string at the specified angle.

```plaintext
title5
    color=brown
    rotate=25
    "TITLE5 is Rotated";
```

Define TITLE7. The BOX= option draws a green box around the text with the thinnest of the 4 available box lines.

```plaintext
title7 color=green
    box=1
    "TITLE7 is Boxed";
```

Define TITLE9. The BLANK= option prevents the boxed title from being overwritten by TITLE10. The first COLOR= option specifies the color of the box border, and the BCOLOR= option specifies the background color of the box. The second COLOR= option specifies the text color.

```plaintext
title9 color=black
    box=3
    blank=yes
    bcolor=yellow
    color=blue
    angle=-25
    "TITLE9 is Not Overlaid By TITLE10";
```

Define TITLE10. In this statement, the DRAW= option draws a box around the first two text strings. The BOX= option is turned off by the MOVE= option that uses absolute coordinates and causes a text break.

```plaintext
title10 color=purple
    draw=(20,35 20,27 58,27 58,35 20,35)
    move=(20,32)
    font=script
    "TITLE10 is in Script and *
    move=(20,27)
    height=2
    "is Partially Boxed, Positioned"
    move=(20,22)
    height=2
    "with Explicit Moves, and is Overlaid by TITLE9";
```

Display titles. All existing titles are automatically displayed by the procedure.

```plaintext
proc gslide;
    run;
quit;
```

Example 7: Using BY-group Processing to Generate a Series of Charts

**Features:**
- **AXIS statement options**
  - `LABEL=`
  - `MAJOR=`
Example 7: Using BY-group Processing to Generate a Series of Charts

MINOR=
NOPLANE
ORDER=
SPLIT=
STYLE=
VALUE=

BY statement
OPTIONS statement options
  NOBYLINE
PATTERN statement options
  COLOR=
TITLE statement
  #BYVAL

Sample library member: GBYGMSC1

Details

The code for this example is in SAS Sample Library member GBYGMSC1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example uses a BY statement with the GCHART procedure to produce a separate three-dimensional vertical bar chart for each value of the BY variable TYPE. The three charts, which are shown in Output 24.1 on page 490, Output 24.2 on page 491, and Output 24.3 on page 492, show leading grain producers for 1995 and 1996.

The program suppresses the default BY lines and instead uses #BYVAL in the TITLE statement text string to include the BY variable value in the title for each chart. The AXIS1 statement that is assigned to the vertical (response) axis is automatically applied to all three graphs generated by the BY statement. This AXIS statement removes all the elements of the response axis except the label. The same AXIS statement also includes an ORDER= option. Because this option is applied to all the graphs, it ensures that they all use the same scale of response values. Because no subgroups are specified and the PATTERNID= option is omitted, the color specified in the single PATTERN statement is used by all of the bars.
Leading Corn Producers 1995 and 1996

Millions of Metric Tons

- United States: 187
- China: 112
- Brazil: 68

Output 24.1 Output for BY Value Corn
Example 7: Using BY-group Processing to Generate a Series of Charts

Output 24.2  Output for BY Value Rice

Leading Rice Producers
1995 and 1996

Millions of Metric Tons

China: 185
India: 120
Indonesia: 50
goptions reset=all border;

data grainldr;
    length country $ 3 type $ 5;
    input year country $ type $ amount;
    megtons=amount/1000;
    datalines;
1995 BRZ Wheat 1516
1995 BRZ Rice 11236
1995 BRZ Corn 36276
1995 CHN Wheat 102207
1995 CHN Rice 185226
1995 CHN Corn 112331
1995 INS Wheat .
1995 INS Rice 49860
1995 INS Corn 8223
1995 USA Wheat 59494
1995 USA Rice 7888
1995 USA Corn 187300

Output 24.3  Output for BY Value Wheat

Leading Wheat Producers
1995 and 1996

Millions of Metric Tons

United States  China  India

121.59  102.21  62.62

GBYGMSC1

Program
1996 BRZ  Wheat    3302
1996 BRZ  Rice     10035
1996 BRZ  Corn     31975
1996 IND  Wheat    62620
1996 IND  Rice     120012
1996 IND  Corn     8660
1996 USA  Wheat    62099
1996 USA  Rice     7771

proc format;
  value $country "BRZ" = "Brazil"
    "CHN" = "China"
    "IND" = "India"
    "INS" = "Indonesia"
    "USA" = "United States";
run;

options nobyline;
title1 "Leading #byval(type) Producers" j=c "1995 and 1996";
footnote1 j=r "GBYGMSC1 ";
pattern1 color=green;
axis1 order=(0 to 550 by 100)
  label=(angle=90 "Millions of Metric Tons")
  major=none
  minor=none
  value=none
  style=0
  noplane;
axis2 label=none
  split=" ";
proc sort data=grainldr out=temp;
  by type;
run;
proc gchart data=temp (where=(megtons gt 31));
  by type;
  format country $country .;
  vbar3d country / sumvar=megtons
    outside=sum
    descending
    shape=hexagon
    width=8
    coutline=black
    cframe=grayaa
    maxis=axis2
    raxis=axis1 name="GBYGMSC1";
run;
quit;

Program Description

Set the graphics environment.
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.

```sas
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.

```sas
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.

```sas
options nobyline;
title1 "Leading #byval(type) Producers"
j=c "1995 and 1996";
footnote1 j=r "GBYGMSC1 ";
```

Specify a color for the bars.

```sas
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.

```plaintext
axis1 order=(0 to 550 by 100)
    label=(angle=90 "Millions of Metric Tons")
    major=none
    minor=none
    value=none
    style=0
    noplane;
```

**Define midpoint axis characteristics.** The SPLIT= option defines one or more characters, such as a blank, that the AXIS statement uses to break axis values into multiple lines.

```plaintext
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.

```plaintext
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.

```plaintext
proc gchart data=temp (where=(megtons gt 31));
    by type;
    format country $country.;
    vbar3d country / sumvar=megtons
        outside=sum
        descending
        shape=hexagon
        width=8
        coutline=black
        cframe=grayaa
        maxis=axis2
        raxis=axis1 name="GBYGMSC1";
run;
quit;
```

---

**Example 8: Combining Graphs and Reports in a Web Page**

**Features:**

- **AXIS statement options**
  - LENGTH=
  - MINOR=
  - ORDER=
  - VALUE=

- **BY statement**

- **GOPTIONS statement options**
  - BORDER
Details
The code for this example is in SAS Sample Library member GONCGRW1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example generates several graphs of sales data that can be accessed from a single web page. The graphs are two bar charts of summary sales data and three pie charts that break the data down by site. Each bar chart and an accompanying report is stored in a separate body file.

The three pie charts are generated with BY-group processing and are stored in one body file. The program suppresses the default BY lines and instead includes the BY variable value in the title for each chart. The SAS/GRAPH titles are displayed in the HTML output instead of in the graphics output.

Output
The web page contains two frames, one that displays a Table of Contents for all the graphs, and one that serves as the display area. Links to each piece of output appear in the table of contents, which is displayed in the left frame. Initially the frame file displays
the first body file, which contains a bar chart and a report, as shown in the following figure.

**Output 24.4**  Browser View of Bar Chart and Quarterly Sales Report
Notice that the chart title is displayed outside the graph as part of the HTML file. Select the link to **Total Department Sales** to display the second bar chart, as shown in the following figure.

**Output 24.5  Browser View of Bar Chart and Department Sales Report**
Selecting any link for **Department Sales** displays the corresponding pie chart as shown in the following figure. Because the pie charts are stored in one file, you can easily see all three by scrolling through the file.

**Output 24.6  Browser View of Pie Charts of Site Sales**

![Browser View of Pie Charts of Site Sales](image)

**Program**

```sas
filename odsout ".";
goptions reset=all border;
data totals;
  length Dept $ 7 Site $ 8;
  input Dept Site Quarter Sales;
datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools  Sydney 1 1775.74
Tools  Atlanta 1 2424.19
Tools  Paris   1 5914.25
Parts  Atlanta 2 11595.07
Parts  Paris   2 9558.29
Repairs Sydney 2 5505.31
Repairs Paris  2 7538.56
Tools  Atlanta 2 1903.99
Tools  Paris   2 7868.34
Parts  Sydney 3 8437.96
Parts  Paris   3 6789.85
Tools  Atlanta 3 3048.52
```
Tools   Paris   3 9017.96  
Parts   Sydney  4 6065.57  
Parts   Atlanta 4 9388.51  
Parts   Paris   4 8509.08  
Repairs Atlanta 4 2088.30  
Repairs Paris   4 5530.37  
;
ods _all_ close;
ods html path=odsout frame="sales_frame.html"  
   contents="sales_contents.html"  
   body="sales_body1.html"  
   nogtitle;

title1 "Total Sales By Quarter";
footnote j=r "salesqtr ";
axis1 order=(0 to 60000 by 20000)  
   minor=(number=1)  
   label=none;
axis2 label=none length=70pct  
   value=('1Q' '2Q' '3Q' '4Q');
legend1 label=none shape=bar(4,4);
proc gchart data=totals;
   format sales dollar8.;
   vbar3d quarter / discrete
      sumvar=sales
      shape=cylinder
      subgroup=site
      cframe=grayaa
      caxis=black
      width=12
      space=4
      legend=legend1
      maxis=axis2
      raxis=axis1
      des="Total Quarterly Sales"
      name="salesqtr";
run;
quit;
proc sort data=totals out=qtrsort;
   by quarter site;
run;
footnotel;
options nobyline;

title1 "Sales for Quarter #byval(quarter)";
proc report data=qtrsort nowindows;
   by quarter;
   column quarter site dept sales;
   define quarter / noprint group;
   define site    / display group;
   define dept    / display group;
   define sales   / display sum format=dollar8.;
   compute after quarter;
Example 8: Combining Graphs and Reports in a Web Page

```sas
site="Total";
endcomp;
break after site / summarize style=rowheader;
break after quarter / summarize style=rowheader;
run;

ods html body="sales_body2.html";
title1 "Total Sales By Department";
footnote1 j=r "salesdep ";
axis1 label=none
   minor=(number=1)
   order=(0 to 100000 by 20000);
axis2 label=none length=70pct;
proc gchart data=totals;
format sales dollar8.;
vbar3d dept / shape=cylinder
    subgroup=site
    cframe=grayaa
    width=12
    space=4
    sumvar=sales
    legend=legend1
    maxis=axis2
    raxis=axis1
    caxis=black
    des="Total Department Sales"
    name="salesdep";
run;
quit;
proc sort data=totals out=deptsort;
    by dept site;
run;
footnote1;
title1 "Sales for #byval(dept)";
proc report data=deptsort nowindows;
    by dept;
    column dept site quarter sales;
    define dept / noprprint group;
    define site / display group;
    define quarter / display group;
    define sales / display sum format=dollar8.;
    compute after dept;
        site="Total";
    endcomp;
    break after site / summarize style=rowheader;
    break after dept / summarize style=rowheader;
run;

ods html body="sales_body3.html"  gtitle;
proc sort data=totals out=sitesort;
    by site;
run;
title1 "Departmental Sales for #byval(site)";
```
footnote j=r "salespie ";

proc gchart data=sitesort;
    format sales dollar8.;
    by site;
    pie3d dept / noheading
        coutline=black
        sumvar=sales
        des="Department Sales"
        name="salespie";
run;
quit;
ods html close;

ods html; /* Not required in SAS Studio */

Program Description

Create a fileref for the output. The current working directory is specified in this example.

    filename odsout ".";

By default the ODS HTML statement to generate the graphics output as PNG files. The BORDER option is used so that the border around the graphics output area is compatible with the borders that are created for non-graphics output. Set the graphics environment.

    goptions reset=all border;

Create the data set TOTALS. The data set contains quarterly sales data for three manufacturing sites for one year.

    data totals;
        length Dept $ 7 Site $ 8;
        input Dept Site Quarter Sales;
        datalines;

Repairs Sydney  1 5592.82
Repairs Atlanta 1 9210.21
Tools   Sydney 1 1775.74
Tools   Atlanta 1 2424.19
Tools   Paris   1 5914.25
Parts   Atlanta 2 11595.07
Parts   Paris   2 9558.29
Repairs Sydney  2 5505.31
Repairs Paris   2 7538.56
Tools   Atlanta 2 1903.99
Tools   Paris   2 7868.34
Parts   Sydney  3 8437.96
Parts   Paris   3 6789.85
Tools   Atlanta 3 3048.52
Tools   Paris   3 9017.96
Parts   Sydney  4 6065.57
Parts   Atlanta 4 9388.51
Parts   Paris   4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris  4 5530.37
;

Close the currently open ODS destinations.

ods _all_ close;

---

Set the ODS HTML destination for file handling options. The PATH= option specifies
the fileref defined previously for the output. The FRAME= option names the HTML file
that integrates the contents and body files. The CONTENTS= option names the HTML
file that contains the table of contents to the HTML procedure output. The BODY=
option names the file for storing the HTML output. The contents file links to each of the
body files written to the HTML destination. The NOGTITLE option suppresses the
graphics titles from the SAS/GRAPH output and displays them through the HTML page.

ods html path=odsout frame="sales_frame.html"
   contents="sales_contents.html"
   body="sales_body1.html"
   nogtitle;

Define title and footnote.

title1 "Total Sales By Quarter";
footnote j=r  "salesqtr ";

Define axis characteristics for the first bar chart. In AXIS2, the LENGTH= option
specifies the length of the midpoint axis.

axis1 order=(0 to 60000 by 20000)
   minor=(number=1)
   label=none;
axis2 label=none length=70pct
   value=("1Q" "2Q" "3Q" "4Q");

Suppress the legend label and define the size of the legend values.

legend1 label=none shape=bar(4,4);

Generate the vertical bar chart of quarterly sales. The NAME= option specifies the
name of the catalog entry.

proc gchart data=totals;
   format sales dollar8.;
   vbar3d quarter / discrete
      sumvar=sales
      shape=cylinder
      subgroup=site
      cframe=grayaa
      caxis=black
      width=12
      space=4
      legend=legend1
      maxis=axis2
      raxis=axis1
      des="Total Quarterly Sales"
      name="salesqtr";
run;
quit;
Sort the data set for the report of quarterly sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```sas
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.

```sas
footnotel;
options nobyline;
```

Generate a report of quarterly sales. Because the HTML body file that references the GCHART procedure output is still open, the report is stored in that file. The chart and report are shown in Output 24.4 on page 496.

```sas
title1 "Sales for Quarter #byval(quarter)";
proc report data=qtrsort nowindows;
   by quarter;
   column quarter site dept sales;
   define quarter / noprint group;
   define site    / display group;
   define dept    / display group;
   define sales   / display sum format=dollar8.;
   compute after quarter;
      site="Total";
   endcomp;
   break after site    / summarize style=rowheader;
   break after quarter / summarize style=rowheader;
run;
```

Assign a new body file for the second bar chart and report. Assigning a new body file closes SALES_BODY1.HTML. The contents and frame files, which remain open, contains links to all body files.

```sas
ods html body="sales_body2.html";
```

Define title and footnote for second bar chart.

```sas
  title1 "Total Sales By Department";
  footnotel 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.

```sas
  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.

```sas
proc gchart data=totals;
   format sales dollar8.;
   vbar3d dept / shape=cylinder
               subgroup=site
               cframe=grayaa
```
width=12
space=4
sumvar=sales
legend=legend1
maxis=axis2
raxis=axis1
caxis=black
des="Total Department Sales"
name="salesdep";
run;
quit;

Sort the data set for the report of department sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

proc sort data=totals out=deptsort;
   by dept site;
run;

Reset the footnote, define a report title, and generate the report of department sales. #BYVAL inserts the value of the BY variable into the title of each report. The chart and report are shown in Output 24.4 on page 496.

footnote1;
title1 "Sales for #byval(dept)";
proc report data=deptsort nowindows;
   by dept;
   column dept site quarter sales;
   define dept     / noprint group;
   define site     / display group;
   define quarter / display group;
   define sales   / display sum format=dollar8.;
   compute after dept;
      site="Total";
   endcomp;
   break after site / summarize style=rowheader;
   break after dept / summarize style=rowheader;
run;

Assign a new body file for the pie charts. Assigning a new file as the body file closes SALES_BODY2.HTML. The contents and frame files remain open. GTITLE displays the titles in the graph.

ods html body="sales_body3.html" gtitle;

Sort data set in order of the BY variable before running the GCHART procedure with BY-group processing.

proc sort data=totals out=sitesort;
   by site;
run;

Define title and footnote. #BYVAL inserts the value of the BY variable SITE into the title for each output.

title1 "Departmental Sales for #byval(site)";
footnote j=r "salespie ";
Generate a pie chart for each site. All the procedure output is stored in one body file. Because BY-group processing generates multiple graphs from one PIE3D statement, the name assigned by the NAME= option is incremented to provide a unique name for each piece of output.

```sas
proc gchart data=sitesort;
  format sales dollar8.;
  by site;
  pie3d dept / noheading
    coutline=black
    sumvar=sales
    des="Department Sales"
    name="salespie";
run;
quit;
```

Close the ODS HTML destination. If you do not close the destination, then you are not able to view the HTML file specified by the FRAME option until you close your SAS session.

```sas
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```

---

**Example 9: Creating a Web Page with Drill-Down Functionality Using the ODS HTML Statement**

**Features:**
- GOPTIONS statement options
  - RESET=
  - TRANSPARENCY=
  - DEVICE=
- ODS HTML statement options
  - BODY=
  - NOGTITLE
  - PATH=

**Data set:**
TOTALS

**Sample library member:**
GONDDCW1

**Details**

The code for this example is in SAS Sample Library member GONDDCW1. The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example shows you how to create a drill-down graph in which the user can select an area of the graph in order to display additional information about the data. The program creates one vertical bar chart of total sales for each site and three reports that break down the sales figures for each site by department and quarter.
Output

The following figure shows the bar chart of sales.

Output 24.7  *Vertical Bar Chart of Total Sales*
Output 24.8 on page 508 shows the PROC REPORT output that appears when you click on the bar for Atlanta.

Output 24.8  PROC REPORT Output Displayed in a Web Browser

Sales Report for Atlanta

Site=Atlanta

<table>
<thead>
<tr>
<th>Dept</th>
<th>Quarter</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>2</td>
<td>$11,695</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$9,389</td>
</tr>
<tr>
<td>Repairs</td>
<td>1</td>
<td>$9,210</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$2,088</td>
</tr>
<tr>
<td>Tools</td>
<td>1</td>
<td>$2,424</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$1,904</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$3,049</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$39,659</td>
</tr>
</tbody>
</table>

Program

filename odsout ".";

goptions reset=all device=gif transparency noborder;
data totals;
   length Dept $ 7 Site $ 8;
   input Dept $ 7 Site $ 8 Quarter Sales;
datalines;

Repairs Sydney 1 5592.82
Repairs Atlanta 1 9210.21
Tools Sydney 1 1775.74
Tools Atlanta 1 2424.19
Tools Paris 1 5914.25
Parts Atlanta 2 11595.07
Parts Paris 2 9558.29
Repairs Sydney 2 5505.31
Repairs Paris 2 7538.56
Tools Atlanta 2 1903.99
Tools Paris 2 7868.34
Parts Sydney 3 8437.96
Parts Paris 3 6789.85
Tools Atlanta 3 3048.52
Tools Paris 3 9017.96
Parts Sydney 4 6065.57
Parts Atlanta 4 9388.51
Parts Paris 4 8509.08
Repairs Atlanta 4 2088.30
Repairs Paris 4 5530.37
;
data newtotal;
  set totals;
  length sitedrill $40;
if site="Atlanta" then
  sitedrill="HREF='report_deptsales.html#IDX'";
else if site="Paris" then
  sitedrill="HREF='report_deptsales.html#IDX1'";
if site="Sydney" then
  sitedrill="HREF='report_deptsales.html#IDX2'";
run;
ods _all_ close;
ods html
  path=odsout body="report_body.html"
  nogtitle;
title1 "Total Sales for All Sites";
footnote1  j=l "click on bars" j=r "REPORT3D ";
pattern color=cyan;
axis1 order=(0 to 80000 by 20000)
  minor=(number=1)
  label=none;
axis2 label=none offset=(9,9);
proc gchart data=newtotal;
  format sales dollar8.;
  vbar3d site / discrete
    width=15
    sumvar=sales
    inside=sum
    html=sitedrill
    coutline=black
    cframe=ltgray
    maxis=axis2
    raxis=axis1
    name="report3d ";
run;
quit;
proc sort data=newtotal;
  by site dept quarter;
run;
ods html body="report_deptsales.html";
goptions reset=footnote;
options nobyline;
title1 "Sales Report for #byval(site)";
proc report data=newtotal nowindows;
  by site;
    column site dept quarter sales;
  define site / noprprint group;
  define dept / display group;
  define quarter / display group;
define sales   / display sum format=dollar8.;
compute after site;
    dept="Total";
endcomp;
break after site / summarize style=rowheader page;
run;
ods html close;
ods html; /* Not required in SAS Studio */

Program Description

Create a fileref for the output. The current working directory is specified in this example.

    filename odsout ".";

Set the graphics environment. In the GOPTIONS statement, DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. The TRANSPARENCY option causes the graphics output to use the web-page background as the background of the graph.

    goptions reset=all device=gif transparency noborder;

Create the data set TOTALS. The data set contains quarterly sales data for three manufacturing sites for one year.

    data totals;
    length Dept $ 7 Site $ 8;
    input Dept Site Quarter Sales;
    datalines;
    Repairs Sydney 1 5592.82
    Repairs Atlanta 1 9210.21
    Tools Sydney 1 1775.74
    Tools Atlanta 1 2424.19
    Tools Paris 1 5914.25
    Parts Atlanta 2 11595.07
    Parts Paris 2 9558.29
    Repairs Sydney 2 5505.31
    Repairs Paris 2 7538.56
    Tools Atlanta 2 1903.99
    Tools Paris 2 7868.34
    Parts Sydney 3 8437.96
    Parts Paris 3 6789.85
    Tools Atlanta 3 3048.52
    Tools Paris 3 9017.96
    Parts Sydney 4 6065.57
    Parts Atlanta 4 9388.51
    Parts Paris 4 8509.08
    Repairs Atlanta 4 2088.30
    Repairs Paris 4 5530.37
    ;

Add the HTML variable to the TOTALS data set and create the NEWTOTAL data set. The HTML variable SITEDRILL contains the targets for the values of the variable SITE.
Each HREF value specifies the HTML body file and the name of the anchor within the body file that identifies the target graph.

```sas
data newtotal;
  set totals;
  length sitedrill $40;
  if site="Atlanta" then
    sitedrill="HREF='report_deptsales.html#IDX'";
  else if site="Paris" then
    sitedrill="HREF='report_deptsales.html#IDX1'";
  if site="Sydney" then
    sitedrill="HREF='report_deptsales.html#IDX2'";
run;
```

Close the currently open ODS destinations.

```sas
ods _all_ close;
```

Set the ODS HTML destination for file handling options. The PATH= option specifies the fileref that was defined previously for the output. The BODY= option names the file for storing HTML output. The NOGTITLE option suppresses the graph titles from the SAS/GRAPH output and displays them in the HTML.

```sas
ods html
  path=odsout body="report_body.html"
  nogtitle;
```

Define title and footnote.

```sas
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.

```sas
pattern color=cyan;
```

Define axis characteristics. The VBAR3D statement to follow assigns axis1 to the response axis and axis2 to the midpoint axis.

```sas
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.

```sas
proc gchart data=newtotal;
  format sales dollar8.;;
  vbar3d site / discrete
    width=15
    sumvar=sales
    inside=sum
    html=sitedrill
```

Example 9: Creating a Web Page with Drill-Down Functionality Using the ODS HTML Statement 511
coutline=black
cframe=ltgray
maxis=axis2
raxis=axis1
name="report3d ";

run;
quit;

Sort the data set NEWTOTAL. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=newtotal;
  by site dept quarter;
run;
```

Open the file for the PROC REPORT output. Assigning a new body file closes REPORT_BODY .HTML.

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

Clear the footnote.

```
goptions reset=footnote;
```

Suppress the default BY line and define a title that includes the BY-value. #BYVAL inserts the value of the BY variable SITE into the title of each report.

```
options nobyline;
title1 "Sales Report for #byval(site)";
```

Print a report of departmental sales for each site.

```
proc report data=newtotal nowindows;
  by site;
  column site dept quarter sales;
  define site / noprint group;
  define dept / display group;
  define quarter / display group;
  define sales / display sum format=dollar8.;
  compute after site;
    dept="Total";
  endcomp;
  break after site / summarize style=rowheader page;
run;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```
ods html; /* Not required in SAS Studio */
```
Details

Putting It All Together
This section provides additional information about the pieces of this program and how they work together to generate SAS/GRAPH output with drill-down functionality. It describes

• how an HREF value is built
• how the HTML= option creates an image map in the HTML file
• how the HTML file references the SAS/GRAPH output.

Building an HREF Value
In the DATA step, the variable SITEDRILL is assigned a string that defines the link target for a data value. An example follows.

```sql
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:

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Output</th>
<th>Anchor name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCHART</td>
<td>report3d.gif</td>
<td>IDX</td>
</tr>
<tr>
<td>REPORT</td>
<td>Atlanta report</td>
<td>IDX1</td>
</tr>
<tr>
<td>REPORT</td>
<td>Paris report</td>
<td>IDX2</td>
</tr>
<tr>
<td>REPORT</td>
<td>Sydney report</td>
<td>IDX3</td>
</tr>
</tbody>
</table>

Creating an Image Map
The HTML= option in the GCHART procedure is assigned the variable with the target information – in this case, SITEDRILL.

```sql
html=sitedrill
```

This option causes SAS/GRAPH to generate (in the HTML body file) the MAP and AREA elements that compose the image map. It loads the HREF attribute value from SITEDRILL into the AREA element. This image map is named `ggccke00k_map`. The map is stored in `report_body.html`. (ODS generates unique map names each time
you run the program, so the next time this program runs, the map name will be
different):

```xml
<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`:

```xml
<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 25

## Graphics Options and Device Parameters Dictionary

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to Graphics Options and Device Parameters</td>
<td>518</td>
</tr>
<tr>
<td>Specifying Graphics Options and Device Parameters</td>
<td>518</td>
</tr>
<tr>
<td>Using the GOPTIONS Statement and the GDEVICE Procedure</td>
<td>518</td>
</tr>
<tr>
<td>Specifying Units of Measurement</td>
<td>519</td>
</tr>
<tr>
<td>Options Used in Both GOPTIONS and OPTIONS Statements</td>
<td>519</td>
</tr>
</tbody>
</table>

### Dictionary

- ACCESSIBLE .............................................. 520
- ALTDESC ............................................... 522
- ASPECT .................................................. 523
- AUTOCOPY .............................................. 523
- AUTOFEED .............................................. 524
- AUTOSIZE .............................................. 525
- BOUNDING .............................................. 525
- BORDER ............................................... 526
- CBACK .................................................. 526
- CBY ..................................................... 527
- CELL .................................................... 527
- CHARACTERS ........................................... 528
- CHARTYPE ............................................. 529
- CIRCLEARC ............................................ 530
- CMAP ................................................... 531
- COLLATE ............................................... 531
- COLORS ............................................... 532
- COLORBL ............................................. 533
- COLORTYPE ........................................... 533
- COLS ................................................... 534
- CPATTERN ............................................. 534
- CSYMBOL ............................................. 535
- CTEXT .................................................. 535
- CTITLE ............................................... 536
- DASH .................................................. 537
- DASHLINE ............................................. 537
- DASHSCALE ............................................ 538
- DELAY .................................................. 538
- DESCRIPTION ......................................... 539
- DEVMAC ................................................ 539
- DEVICE ............................................... 540
- DEVMAP ............................................... 541
- DEVOPTS .............................................. 541
DEVTYPE ................................................................. 544
DISPLAY ................................................................. 545
DISPOSAL ................................................................. 545
DRVINIT ................................................................. 546
DRVQRY ................................................................. 546
DRVTERM ................................................................. 547
DUPLEX ................................................................. 547
ERASE ................................................................. 548
EXTENSION ........................................................... 549
FASTTEXT ............................................................. 549
FHY ................................................................. 550
FCACHE ................................................................. 550
FILECLOSE ............................................................. 551
FILEONLY ............................................................... 552
FILL ................................................................. 552
FILLINC ................................................................. 553
FONT NAME ............................................................ 553
FONTRES ................................................................. 554
FORMAT ................................................................. 555
FTEXT ................................................................. 555
FTITLE ................................................................. 556
FTRACK ................................................................. 556
GACCESS ................................................................. 557
GCOPIES ................................................................. 558
GEND ................................................................. 559
GEPLOG ................................................................. 560
GOUTMODE ............................................................ 561
GPLOG ................................................................. 561
GPROTOCOL ........................................................... 562
GRAPHRC ............................................................... 564
GSFLEN ................................................................. 564
GSFMODE ............................................................... 565
GSFNAME ............................................................... 566
GSFPROMPT ........................................................... 567
FSIZE ................................................................. 567
GSTART ................................................................. 567
GUNIT ................................................................. 568
GWAIT ................................................................. 569
HANDSHAKE .......................................................... 570
HBY ................................................................. 571
HEADER ................................................................. 571
HEADERFILE ............................................................ 572
HORIGIN ............................................................... 572
HOSTSPEC ............................................................. 573
HPPOS ................................................................. 573
HSIZE ................................................................. 574
HTEXT ................................................................. 575
HTITLE ................................................................. 575
IBACK ................................................................. 576
ID ................................................................. 577
IMAGEPRINT ........................................................... 577
IMAGESTYLE ............................................................ 577
INTERACTIVE .......................................................... 578
INTERLACED ............................................................ 579
INTERPOL ............................................................... 579
ITERATION ............................................................... 579
Introduction to Graphics Options and Device Parameters

This chapter provides a detailed description of all of the graphics options and device parameters used with SAS/GRAPH software. These include

- all graphics options used by the GOPTIONS statement
- all device parameters that can be specified as options in the ADD and MODIFY statements in the GDEVICE procedure
- all device parameters that appear as fields in the GDEVICE windows.

The descriptions provide the syntax, defaults, and required information for each option and parameter.

The graphics options and device parameters are intermixed and listed alphabetically. When the graphics option and device parameter have the same name, they are discussed in the same dictionary entry and the description uses only that name and does not distinguish between the option and the parameter except where the distinction is necessary.

For a list of all the graphics options, see “GOPTIONS Statement” on page 375. For a list of all the device parameters, see “ADD Statement” on page 1050.

If the syntax for the graphics option and the device parameter is different, both forms are shown. If the syntax is the same, one form is shown.

Specifying Graphics Options and Device Parameters

Using the GOPTIONS Statement and the GDEVICE Procedure

Use a GOPTIONS statement to specify the graphics options. Some graphics options can also be specified in an OPTIONS statement. Use the GDEVICE procedure to specify the device parameters. (See “GOPTIONS Statement” on page 375 and Chapter 37, “GDEVICE Procedure,” on page 1046 for details.)
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.

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.

The IN and CM units of measurement are not supported by Java.

---

**Options Used in Both GOPTIONS and OPTIONS Statements**

Several of the graphics options are also used as system options. These options include the following:

- these printer options:
  - **BINDING**
  - **PAPERSIZE**
  - **COLLATE**
  - **PAPERSOURCE**
  - **DUPLEX**
  - **PAPERTYPE**
  - **PAPERDEST**

- the DEVICE option

With the exception of the DEVICE= option, the effect of setting these options depends on whether they are set as graphics options or system options. For the printer options, when they are set as graphics options in a GOPTIONS statement, they affect the SAS/GRAPH devices. When they are set as system options in an OPTIONS statement, they affect the Universal Printers and the Universal Printer shortcut devices. (For
information about the Universal Printers, Universal Printer shortcut devices, and the SAS/GRAF devices, see Chapter 9, “Using Graphics Devices,” on page 79.) Setting a printer graphics option does not affect the corresponding system option, and vice versa. For example, here is a statement that sets the graphics option PAPERSIZE= to A4.

```sql
goptions papersize=A4;
```

This statement does not affect the PAPERSIZE= system option. As a result, it sets the paper size to A4 for the SAS/GRAF printer devices only. If you want to set the paper size for the Universal Printer devices or the Universal Printer shortcut devices, you must set the PAPERSIZE= system option instead of the graphics option as shown in the following example.

```sql
options papersize=A4;
```

For information about the GOPTIONS statement, see “GOPTIONS Statement” on page 375. For information about the OPTIONS statement, see SAS DATA Step Statements: Reference.

In contrast to the printer options listed previously, setting the DEVICE= option as a graphics option has the same effect as setting it as a system option. That is, setting DEVICE= as either a graphics option or a system option sets the SAS/GRAF output device driver. As a result, the following two statements have the same effect:

```sql
options device=pdf;
goptions device=pdf;
```

*Note:* Only one of the two statements is needed to set the device. You can use either one.

---

**Dictionary**

**ACCESSIBLE**

Generates descriptive text and summary statistics representing your graphics output.

- **Used by:** GOPTIONS statement
- **Default:** NOACCESSIBLE
- **Restriction:** Supported only by JAVA and ActiveX when used with the ODS HTML output destination.

**Syntax**

ACCESSIBLE | NOACCESSIBLE

**Parameter Values**

**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/GRAF writes accessibility information to the graph's output HTML file, and creates a left-justified footnote that provides a link to the information.
The information and the link are not visible in the output HTML. However, both are detected by accessibility aids, such as screen readers. You can also access the information by pressing the tab key and enter. The information will be displayed once you press enter on the link in the footnote. The information will also be displayed if you move your mouse over the location of the left-justified footnote, and click the link when the mouse pointer shape changes.

**Figure 25.1 ACCESSIBLE**

![Bar Chart of Class Data](Image)

This graph is a vertical bar chart of 'Age', with a mean response variable of 'Height'.

The following table contains the data from the graph:

<table>
<thead>
<tr>
<th>Age</th>
<th>Height (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>54.40</td>
</tr>
<tr>
<td>12</td>
<td>59.44</td>
</tr>
<tr>
<td>13</td>
<td>61.43</td>
</tr>
<tr>
<td>14</td>
<td>64.90</td>
</tr>
<tr>
<td>15</td>
<td>65.63</td>
</tr>
<tr>
<td>16</td>
<td>72.00</td>
</tr>
</tbody>
</table>

**NOACCESSIBLE**

toggles off the ACCESSIBLE option.
ALTDESC

Specifies whether to write the DESCRIPTION= statement text to the ALT= text in an HTML file.

Used by: GOPTIONS statement

Default: ALTDESC

Restriction: Supported only when used with the HTML output destination and the DESCRIPTION= option.

Syntax

ALTDESC | NOALTDESC

Parameter Values

ALTDESC

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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

**Default:** device-dependent

**Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

```
ASPECT=scaling-factor
```

**Parameter Values**

*scaling-factor*

is a nonnegative integer or real number that determines the ratio of width to height for graphics elements. If you specify ASPECT=1, each graphics element has equal horizontal and vertical scaling factors; ASPECT=2 scales the graphics element twice as wide as its height; and so on. If ASPECT= is not specified or is set to 0 or null, SAS/GRAPH uses the aspect ratio of the hardware device.

**Details**

The aspect ratio affects many graphics characteristics, such as the shape of software characters and the roundness of pie charts. Some graphics drivers do not produce correct output if the aspect ratio is anything other than the default. When you use a device that uses local scaling (that is, the device itself can scale the output—for example, some plotters), use ASPECT= to tell SAS/GRAPH the scaling factor.

*Note:* You can get more reliable results if you use the default aspect ratio and use the HSIZE= and VSIZE= graphics options to set the dimensions.

---

AUTOCOPY
Specifies whether to generate hard copy automatically.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Defaults:**

- GOPTIONS: NOAUTOCOPY
- GDEVICE: AUTOCOPY=N

**Restrictions:**

- This option is device-dependent.
- This option is not supported by Java or ActiveX.

**Syntax**

```
GOPTIONS: AUTOCOPY | NOAUTOCOPY
GDEVICE: AUTOCOPY=Y | N
```
Parameter Values

AUTOCOPY
AUTOCOPY=Y
prints a copy of the graph automatically.

NOAUTOCOPY
AUTOCOPY=N
suppresses printing a copy of the graph. A blank Autocopy field in the Parameters window is the same as AUTOCOPY=N.

Details

AUTOCOPY is used only for older terminals with printers attached directly to the device.

AUTOFEED

Specifies whether devices with continuous paper or automatic paper feed should roll or feed the paper automatically for the next graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Defaults: GOPTIONS: AUTOFEED (if a device is specified)
GDEVICE: AUTOFEED=Y

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

See: PPDFILE on page 597

Syntax

GOPTIONS: AUTOFEED | NOAUTOFEED
GDEVICE: AUTOFEED=Y | N

Parameter Values

AUTOFEED
AUTOFEED=Y
causes the device to feed new paper automatically for the next graph. A blank Autofeed field in the Parameters window is the same as AUTOFEED=Y.

NOAUTOFEED
AUTOFEED=N
suppresses the automatic paper feed.

Details

For PostScript devices, if AUTOFEED is unaltered, the PostScript file is unchanged. If you specify NOAUTOFEED and do not select a PPD file with the PPDFILE option, a PostScript Level 1 MANUALFEED command is added to the driver output. If you specify NOAUTOFEED and select a PPD that contains a MANUALFEED option, the procedure code for that MANUALFEED option is sent. If there is no MANUALFEED option in the PPD, no MANUALFEED code is sent. See “PPDFILE” on page 597.
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 by:** GOPTIONS statement

**Default:** device-dependent

**Restriction:** This option is not supported by Java or ActiveX.

**See:** DEVOPTS on page 541

### Syntax

AUTOSIZE=ON | OFF | DEFAULT

### Parameter Values

**ON**
changes the cell size in order to preserve the number of rows and columns.

**OFF**
preserves the device's original cell size and temporarily changes the number of rows and columns.

**DEFAULT**
uses the default setting (ON or OFF) that is controlled by DEVOPTS bit 50 (see “DEVOPTS” on page 541).

### 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 by:** GOPTIONS statement, OPTIONS statement

**Default:** DEFAULTEDGE

**Restrictions:** PostScript and PCL printers only. PostScript printers require a PPD file. Not supported by Java or ActiveX.

**See:** DUPLEX on page 547

PPDFILE on page 597

“Options Used in Both GOPTIONS and OPTIONS Statements” on page 519
Syntax

BINDING=DEFAULTEDGE | LONGEDGE | SHORTEDGE

Comparisons

BINDING controls how the page is flipped when DUPLEX is in effect. It does not change the orientation of the graph. DEFAULTEDGE refers to the hardware's factory-default setting. LONGEDGE and SHORTEDGE refer to the paper's long and short edges.

For PostScript printers, a PPD file must also be specified, using the PPDFILE= option. The PPD file contains the command that SAS/GRAPH needs to request the appropriate binding method on the printer being used. If a PPD file is not specified, the BINDING= option is ignored because SAS/GRAPH will lack the command needed to request the binding method.

---

BORDER

Specifies whether to draw a border around the graphics output area.

**Used by:** GOPTIONS statement

**Default:** NOBORDER

**Restriction:** This option is ignored when the JAVAIMG, ACTIVEX, or ACTXIMG device is used.

---

Syntax

BORDER | NOBORDER

Details

The placement of the border on the display is defined by the HSIZE= and VSIZE= graphics options, if used. Otherwise, the placement is defined by the XMAX and YMAX device parameters.

---

CBACK

Specifies the background color of the graphics output.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Gcolors window

**Default:** as specified in the Gcolors window

Syntax

CBACK=background-color

Parameter Values

**background-color**

Details

The CBACK= option is valid on all devices but can be ignored by some (for example, plotters). Specify the default in the Gcolors window of the device entry.

Note: This option overrides the Background and Foreground style attributes in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide*.

If you explicitly specify a background color with the CBACK= option, the background color that you select should contrast with the foreground colors.

If the IBACK= option is in effect, an image will appear in the background in place of the color specified with the CBACK= option.

CBY

Selects the color of the BY lines that appear in the graphics output.

- **Used by:** GOPTIONS statement
- **Defaults:**
  1. CTEXT= graphics option, if used
  2. first color in current color list
- **Restriction:** This option is not supported by Java or ActiveX.

Syntax

`CBY=By line-color`

Parameter Values

*By line-color*


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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

Syntax

`GOPTIONS: CELL | NOCELL`
GDEVICE: CELL=Y | N

**Parameter Values**

**CELL**

CELL=Y

causes the device to use cell alignment. In that case, SAS/GRAPH attempts to place hardware (or simulated hardware) characters inside character cells. This restriction on the location of characters means that in some cases the SAS/GRAPH procedure can generate axes that do not occupy the entire procedure output area or might be unable to create the requested graph. A blank *Cell* field in the Parameters window is the same as CELL=Y.

**NOCELL**

CELL=N

suppresses cell alignment, causing the procedure to use the entire procedure output area and place axis and tick mark labels without regard to cell alignment.

**Details**

Specify N in the device entry or use NOCELL in a GOPTIONS statement if you want to preview a graph on a cell-aligned display but intend to produce the final graph on a device that is not cell-aligned, such as a pen plotter.

---

**CHARACTERS**

Specifies whether the device-resident font is used when no font or FONT=NONE is specified in a SAS statement.

**Used by:**

- GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Defaults:**

- GOPTIONS: CHARACTERS
- GDEVICE: CHARACTERS=Y

**Restriction:**

This option is not supported by Java or ActiveX.

**Syntax**

GOPTIONS: CHARACTERS | NOCHARACTERS

GDEVICE: CHARACTERS=Y | N

**Parameter Values**

**CHARACTERS**

CHARACTERS=Y

causes SAS/GRAPH to use the device-resident font when you do not specify a font in a SAS program. A blank *Characters* field in the Parameters window is the same as CHARACTERS=Y.

**NOCHARACTERS**

CHARACTERS=N

causes SAS/GRAPH to draw the characters using the SIMULATE font and suppresses the use of all device—resident fonts, regardless of values that you specify in other SAS statements.
Details

The device—resident font is not used if you changed the HPOS= and VPOS= graphics options from the default, or if you used the HEIGHT= option in a SAS statement and the device does not have scalable characters.

CHARREC

Specifies a device-resident font by associating a CHARTYPE number with a device-resident font. Also defines a default size to use with that font.

Used by: GDEVICE procedure
Default: device-dependent

Syntax

CHARREC=(charrec-list(s))

Parameter Values

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 530 for more information.)

rows

is the number of rows of text in the font that will fit on the display. (See “ROWS” on page 611 for more information.)

cols

is the number of columns of text in the font that will fit on the display. (See “COLS” on page 534 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 553 for more information.)

Y

represents a scalable font. A scalable font can be displayed at any size. (See “SCALABLE” on page 611 for more information.)

N

represents a nonscalable font. A nonscalable font can be displayed only at a fixed size. (See “SCALABLE” on page 611 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 by:**
- GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:**
- device-dependent

**Restriction:**
- This option is not supported by Java or ActiveX.

**Syntax**

CHARTYPE=hardware-font-chartype

**Parameter Values**

*hardware-font-chartype*

is a nonnegative integer from 0 to 999. *hardware-font-chartype* refers to the actual number for the device-resident font that you want to use as listed in the Chartype window of the device entry for the selected device driver. By default, CHARTYPE is 0, which is the default device-resident font for the device.

CIRCLEARC

Specifies whether SAS/GRAPH should use the device's hardware circle-drawing capability, if available.

**Used by:**
- GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:**
- device-dependent

**Restriction:**
- This option is not supported by Java or ActiveX.

**Syntax**

GOPTIONS: CIRCLEARC | NOCIRCLEARC
GDEVICE: CIRCLEARC=Y | N

**Parameter Values**

*CIRCLEARC*

*CIRCLEARC=Y*

causes SAS/GRAPH to use the built-in hardware circle- and arc-drawing capability of the device. A blank Circlearc field in the Parameters window is the same as CIRCLEARC=Y.

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 by:** GDEVICE procedure, GDEVICE Colormap window

**Syntax**

CMAP=('from-color : to-color' <..., 'from-color-n : to-color-n'> )

**Parameter Values**

- **from-color**
  - Specifies the name that you want to assign to the color designated by the color value.
  - In the Colormap window, enter this value in the From field.

- **to-color**
  - Specifies any SAS/GRAPH color name up to eight characters long. In the Colormap window, enter this value in the To field. See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313 for information about specifying colors.

**Details**

Once you have defined the color mapping, you use the new color name in any color option. For example, if your device entry maps the color name DAFFODIL to the SAS color value PAOY, you can specify the following:

```plaintext
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 by:** GOPTIONS statement, OPTIONS statement

**Default:** NOCOLLATE

**Restrictions:** This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

**See:** GPROLOG on page 561  
PPDFILE on page 597  
“Options Used in Both GOPTIONS and OPTIONS Statements” on page 519

**Syntax**

COLLATE | NOCOLLATE
Comparisons
A limited number of printers can collate output, which means to separate each copy of printed output when you print multiple copies of output.

For PostScript printers, if a device's PPD file has Collate defined as “True”, the COLLATE option is supported.

For PCL printers that support collation, use the GPROLOG= option to specify a Printer Job Language (PJL) command to enable the collation. For information about the appropriate PJL command, consult the Printer Commands section of your printer's user manual.

COLORS
Specifies the foreground colors used to produce your graphics output if you do not specify colors explicitly in program statements.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Gcolors window

**Default:** device-dependent

**Example:** “Example 3: Rotating Plot Symbols through the Color List” on page 473

**Syntax**

GOPTIONS: COLORS=<\(\)colors-list | NONE>\(\)>

GDEVICE: COLORS=\(\)\(<\)colors-list\> \(\)\)

**Parameter Values**

**colors-list**

specifies one or more SAS color names. If you specify more than one color, separate each name with a blank. See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313 for information about specifying colors and using a color list.

To change some of the colors in the color list and retain others, you can use a null value for colors that you do not want to change. For example, to change COLORS=(RED GREEN BLUE) to COLORS=(WHITE GREEN BROWN), you can specify COLORS=(WHITE,BROWN).

**NONE**

tells SAS/GRAPH to use only the colors that you explicitly specify in program statements and to ignore the device's default color list.

*Note:* If you specify COLORS=(NONE) 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.

**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 398 for more information about pattern rotation.

**TIP** You can use the COLOR= option on the global PATTERN statement, the CTEXT= option, or the CTITLE= option in the GOPTIONS statement to control the color.

---

### COLORTBL

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 by:** GDEVICE procedure, GDEVICE Metagraphics window

**Default:** NAME

**Syntax**

```
COLORTYPE=NAME | RGB | RGBA | HLS | GRAY | CMY | CMYK | HSV | HSB
```

**Parameter Values**

See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313 for a description of these color types.

**NAME**

SAS predefined color names.

**RGB**

red-green-blue (RGB) color specifications.

**RGBA**

red-green-blue (RGB) color specifications with an alpha channel to control color opacity.
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).

Details

Use the COLOTYPE 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.

COLO

Sets the number of columns that the device-resident font uses.

  **Used by:**  GDEVICE Chartype window, GDEVICE procedure, CHARREC= option
  **Default:**  0
  **See:**  CHARREC on page 529

Details

For information about the syntax, see “CHARREC” on page 529.

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 by:**  GOPTIONS statement
  **Default:**  first color in current color list
  **Restriction:**  This option is not supported by Java or ActiveX.
Syntax
CPATTERN=pattern-color

Parameter Values
pattern-color

Details
CPATTERN= is overridden by any color specification in a PATTERN statement. For details about how CPATTERN= affects the PATTERN statement, see “The Effect of the CPATTERN= Graphics Option” on page 409.

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 398 for more information about pattern rotation.

CSYMBOL
Specifies the default color for SYMBOL definitions when a color has not been specified.

Used by: GOPTIONS statement
Default: first color in current color list
Restriction: This option is not supported by Java or ActiveX.

Syntax
CSYMBOL=symbol-color

Parameter Values
symbol-color

Details
CSYMBOL= is overridden by any color specification in a SYMBOL statement. See “SYMBOL Statement” on page 412.

CTEXT
Selects the default color for all text and the border.

Used by: GOPTIONS statement
Defaults: Java and ActiveX devices: black
all other devices: the first color in current color list
Restriction: This option is partially supported by Java.
CTITLE

Selects the default color for all titles, footnotes, and notes, and the border.

**Used by:** GOPTIONS statement

**Defaults:**

1. color specified by CTEXT=, if used
2. based on device:
   - Java and ActiveX: black
   - all other devices: the first color in current color list

**See:** CTEXT on page 535

**Syntax**

CTITLE=title-color

**Parameter Values**

*title-color*


**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 101 for more information.

---

**DASH**

Specifies whether to use the device's hardware dashed-line capability, if available.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:** device-dependent

**Restriction:** This option is not supported by Java or ActiveX.

**See:** DASHLINE on page 537

**Syntax**

```
GOPTIONS: DASH | NODASH
GDEVICE: DASH=Y | N
```

**Parameter Values**

**DASH**

**DASH=Y**

causes SAS/GRAPH to use the built-in hardware dashed-line drawing capability of the device when generating graphics output. A blank Dash field in the Parameters window is the same as DASH=Y. 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 by:** GDEVICE procedure, GDEVICE Parameters window

**Default:** device-dependent

**See:** DASH on page 537

**Syntax**

```
DASHLINE='dashed-line-hex-string'X
```

DASHLINE 537
**Parameter Values**

*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 24.21 on page 443 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 the following:

```
dashline='c000000000000000'x
```

Bit 1 should always be on because it corresponds to a solid line.

**Details**

If the DASH device parameter is N in the device entry or if NODASH is used in a  
GOPTIONS statement, SAS/GRAPH ignores the hexadecimal string in the DASHLINE  
device parameter.

---

**DASHSCALE**

Scales the lengths of the dashes in a dashed line.

- **Used by:** GOPTIONS statement  
- **Default:** DASHSCALE=1  
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

```
DASHSCALE=scaling-factor
```

**Parameter Values**

*scaling-factor*  
can be any number greater than 0. For example, GOPTIONS DASHSCALE=.5  
reduces any existing dash length by one-half.

**Details**

Only dashes or spaces with lengths greater than one pixel are scaled. Dots are not scaled  
because their length is effectively zero. DASHSCALE= always uses system line styles  
instead of the device's dashed line capabilities.

---

**DELAY**

Controls the amount of time between graphs in the animation sequence.

- **Used by:** GOPTIONS statement  
- **Default:** 0
Restrictions: This option applies to the GIF and SVGANIM device drivers only. This option is not supported by all browsers.

See: Chapter 15, “Generating Animations,” on page 167
“Example: Creating an SVG Animation” on page 183

Syntax
DELAY=delay-time

Parameter Values
delay-time
specifies the length of time between graphs in units of 0.01 seconds. For example, to specify a delay of .03 seconds, specify DELAY=3.

Details
SAS/GRAPH puts the DELAY= value into the image file. Based on this value, the browser determines how to display the series of graphs.

DESCRIPTION
Provides a description of the device entry.

Used by: GDEVICE procedure, GDEVICE Detail window
Alias: DES
Default: none

Syntax
DESCRIPTION='text-string'

Parameter Values
text-string
is a string up to 256 characters long. This is a comment field and does not affect the graphics output.

DEVADDR
Specifies the location of the device to which the output of device drivers is sent.

Used by: GOPTIONS statement
Default: host-dependent
Restriction: IBM mainframe systems only
**Syntax**

DEVADDR=\textit{device-address}

**Parameter Values**

\textit{device-address}

specifies the address of the device to which the device driver output is to be sent.

---

**DEVICE**

Specifies the device driver to which SAS/GRAPH sends the procedure output. The device driver controls the format of graphics output.

- **Used by:** GOPTIONS statement, OPTIONS statement
- **Alias:** DEV
- **Default:** device-dependent
- **Note:** When you are running SAS without a terminal, you must either specify the DEVICE= option in the GOPTIONS statement, or specify the DEVICE= or PRINTER= option in the ODS destination statement. A terminal is not associated with your SAS session if you are running in batch mode on z/OS, if you are running SAS on UNIX without an X Server, or if you have specified the NOTERMINAL system option.

- **See:** “Options Used in Both GOPTIONS and OPTIONS Statements” on page 519

**Syntax**

DEVICE=\textit{device-entry}

**Parameter Values**

\textit{device-entry}

specifies the name of a device entry that is stored in a device catalog.

**Details**

A device driver can direct graphics output to a hardware device, such as a terminal or a printer, or can create an external file in another graphics file format, such as TIF, GIF, or PostScript. Some device drivers also generate both graphics files and HTML files that can be viewed with a web browser.

Usually, a device driver is assigned by default. If a default driver is not assigned or if you specify RESET=ALL in a GOPTIONS statement, and you do not specify a device driver, SAS/GRAPH prompts you to enter a driver name when you execute a procedure that produces graphics output. If you are producing a graph to the screen and the Graph window is active, SAS/GRAPH selects the display driver for you automatically.

For a description of device drivers and for more information about selecting a device entry and changing device parameters, see Chapter 9, “Using Graphics Devices,” on page 79.

For information about using device drivers to display and print graphics output, see Chapter 11, “SAS/GRAPH Output,” on page 105.
For information about using device drivers to export graphics output to external files, see “Specifying the Graphics Output File Type for Your Graph” on page 109. For information about using device drivers to create output for the web, see “Developing Web Presentations with the PNG, SVG, and GIF Devices” on page 156 and “Developing Web Presentations with the JAVA IMG and ACTX IMG Devices” on page 157.

**DEVMAP**

Specifies the device map to be used when device-resident fonts are used.

- **Used by:** OPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

DEVMAP=\device-map-name | NONE

**Parameter Values**

- **device-map-name**
  - is a string up to eight characters long that is the name of the device map entry.

- **NONE**
  - specifies that you do not want to use a device map. This can cause text to be displayed incorrectly or not at all.

**Details**

Device maps usually are used only when national characters appear in the text and you want them to display properly.

**DEVOPTS**

Specifies the hardware capabilities of the device.

- **Used by:** GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent

**Syntax**

DEVOPTS=\hardware-capabilities-hex-stringX

**Parameter Values**

- **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>
<thead>
<tr>
<th>Bit On</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hardware circle generation</td>
</tr>
<tr>
<td>1</td>
<td>hardware pie fill supported</td>
</tr>
<tr>
<td>2</td>
<td>scalable hardware characters</td>
</tr>
<tr>
<td>3</td>
<td>device is a CRT-type (See TYPE device parameter)</td>
</tr>
<tr>
<td>4</td>
<td>translate table needed for non-ASCII hosts</td>
</tr>
<tr>
<td>5</td>
<td>hardware polygon fill available</td>
</tr>
<tr>
<td>6</td>
<td>hardware characters cell-aligned</td>
</tr>
<tr>
<td>7</td>
<td>user-definable colors supported</td>
</tr>
<tr>
<td>8</td>
<td>hardware polygons with multiple boundaries supported</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
</tr>
<tr>
<td>11</td>
<td>adjustable hardware line width</td>
</tr>
<tr>
<td>12</td>
<td>double-byte font (non-US) supported</td>
</tr>
<tr>
<td>13</td>
<td>hardware repaint supported</td>
</tr>
<tr>
<td>14</td>
<td>hardware characters supported</td>
</tr>
<tr>
<td>15</td>
<td>no hard limit on x coordinate</td>
</tr>
<tr>
<td>16</td>
<td>no hard limit on y coordinate</td>
</tr>
<tr>
<td>17</td>
<td>not used</td>
</tr>
<tr>
<td>18</td>
<td>ability to justify proportional text</td>
</tr>
<tr>
<td>19</td>
<td>driver can produce dependent catalog entries</td>
</tr>
<tr>
<td>20</td>
<td>device cannot draw in default background color</td>
</tr>
<tr>
<td>21</td>
<td>flush device buffer when filled</td>
</tr>
<tr>
<td>22</td>
<td>colors defined using HLS</td>
</tr>
<tr>
<td>23</td>
<td>colors defined using RGB</td>
</tr>
<tr>
<td>Bit On</td>
<td>Capability</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>24</td>
<td>not used</td>
</tr>
<tr>
<td>25</td>
<td>polyline supported</td>
</tr>
<tr>
<td>26</td>
<td>polymarker supported</td>
</tr>
<tr>
<td>27</td>
<td>graphics clipping supported</td>
</tr>
<tr>
<td>28</td>
<td>not used</td>
</tr>
<tr>
<td>29</td>
<td>linkable device driver</td>
</tr>
<tr>
<td>30</td>
<td>select CHARTYPE by name in CHARREC entries</td>
</tr>
<tr>
<td>31</td>
<td>device-dependent pattern support</td>
</tr>
<tr>
<td>32</td>
<td>treat SCALABLE=Y CHARREC as metric</td>
</tr>
<tr>
<td>33</td>
<td>size CHARTYPE as HW from CHARREC entries</td>
</tr>
<tr>
<td>34</td>
<td>device supports rotated arcs</td>
</tr>
<tr>
<td>35</td>
<td>device supports target fonts</td>
</tr>
<tr>
<td>36</td>
<td>device supports drawing images</td>
</tr>
<tr>
<td>37</td>
<td>device supports multiple color maps</td>
</tr>
<tr>
<td>38</td>
<td>image rotation direction</td>
</tr>
<tr>
<td>39</td>
<td>device requires sublib for image rotation</td>
</tr>
<tr>
<td>40</td>
<td>device is a 24 bit truecolor machine</td>
</tr>
<tr>
<td>41</td>
<td>device supports setting font attributes</td>
</tr>
<tr>
<td>42</td>
<td>use scan line font rendering</td>
</tr>
<tr>
<td>43</td>
<td>device can scale images</td>
</tr>
<tr>
<td>44</td>
<td>text clipping supported</td>
</tr>
<tr>
<td>45</td>
<td>non-interactive color device</td>
</tr>
<tr>
<td>46</td>
<td>driver does prolog processing</td>
</tr>
<tr>
<td>47</td>
<td>driver does epilog processing</td>
</tr>
<tr>
<td>48</td>
<td>driver output uses only a file</td>
</tr>
</tbody>
</table>
### Details

Each capability in the table corresponds to a bit in the value of the DEVOPTS device parameter. For example, if your device can generate hardware pie fills, the second bit in the first byte of the DEVOPTS string should be turned on if you want the driver to use that capability. If your device is capable of generating only hardware circles and pie fills, specify a value of `'C000000000000000'X` as your DEVOPTS value (the first byte is `'1100` so that the first 2 bits of the first byte are set to 1). Many of the hardware capabilities specified in the DEVOPTS string are overridden by graphics options or other device parameters.

**CAUTION:**
Do not modify the DEVOPTS device parameter unless you are building a Metagraphics driver. If you want to prevent a driver supplied by SAS from using certain hardware capabilities, change the specific device parameter or use the corresponding graphics option.

If the DEVOPTS string indicates that a capability is available, the driver uses it unless it is explicitly disabled by another device parameter or graphics option. If the DEVOPTS string indicates that the capability is not available, it is not used by the driver, even if the corresponding device parameter or graphics option indicates that it should be used. For example, if the DEVOPTS value indicates that the device can do a hardware pie fill, the driver uses the hardware pie fill capability unless the PIEFILL device parameter is set to N or NOPIEFILL has been specified in a GOPTIONS statement. However, if the DEVOPTS device parameter indicates that the device cannot do a hardware pie fill, the driver does not attempt to use one, even if the PIEFILL device parameter is set to Y or PIEFILL is used in a GOPTIONS statement.

### DEVTYPE

*Specifies the information required by SAS/GRAPH routines to determine the nature of the output device.*

**Used by:** GDEVICE procedure, GDEVICE Host File Options window
Default: device-dependent

Syntax
DEVTYPE=\textit{device-type}

Parameter Values
\textit{device-type}
is a string eight characters long containing either blanks or some token name that is interpreted by the host. \textit{Device-type} can be:

\begin{itemize}
  \item \textbf{GTERM} indicates that the output device is a graphics device that will be receiving graphics data; most device drivers use this value.
  \item \textbf{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.
\end{itemize}

\textbf{Note:} GTERM and G3270 are SAS/GRA PH device types. Other valid values depend on your operating environment. DEVTYPE supports any of the device-type values supported in the FILENAME statement. Refer to the SAS Help facility for the device types the FILENAME statement supports in your operating environment. In most cases, this field should not be changed.

\textbf{DISPLAY}

Specifies whether output is displayed on the graphics device but does not affect whether a graph is placed in a catalog.

\begin{itemize}
  \item \textbf{Used by:} GOPTIONS statement
  \item \textbf{Default:} DISPLAY
  \item \textbf{Restriction:} This option is not supported by Java or ActiveX.
\end{itemize}

\textbf{Syntax}
DISPLAY | NODISPLAY

\textbf{Comparisons}

In most cases, NODISPLAY suppresses all output except the catalog entry written to the catalog selected in the GOUT= option. Therefore, you usually specify NODISPLAY when you want to generate a graph in a catalog but do not want to display the graph on your monitor or terminal while the catalog entry is being produced.

\textbf{DISPOSAL}

Specifies what happens to the graphic after it is displayed.

\begin{itemize}
  \item \textbf{Used by:} GOPTIONS statement
  \item \textbf{Default:} NONE
\end{itemize}
Restriction: GIF driver only

Syntax

DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED

Parameter Values

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.

DRVINIT

Specifies host commands to be executed before driver initialization.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

Restriction: This option is not supported by Java or ActiveX.

Syntax

DRVINIT1='system-command(s)'
DRVINIT2='system-command(s)'

Parameter Values

system-command(s)
specifies a character string 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 by: GDEVICE procedure, GDEVICE Detail window
**Default:** device-dependent

---

**Syntax**

DRVQRY | NODRVQRY

**Comparisons**

Generally, this setting is device-dependent and you should not change it.

---

**DRVTERM**

Specifies host commands to be executed after the driver terminates.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

**Restriction:** This option is not supported by Java or ActiveX.

---

**Syntax**

DRVTERM1='*system-command(s)*'

DRVTERM2='*system-command(s)*'

---

**Parameter Values**

*system-command(s)*

specifies a character string 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 by:** GOPTIONS statement, OPTIONS statement

**Default:** NODUPLEX

**Restriction:** duplex printers only

**See:** BINDING on page 525

GSFMODE on page 565

PPDFILE on page 597

“Options Used in Both GOPTIONS and OPTIONS Statements” on page 519
Syntax
DUPLEX | NODUPLEX

Comparisons
When DUPLEX is on, the driver sets up the printer for duplex operation. Before producing the first graph, set GSFMODE=REPLACE in the GOPTIONS statement, and DUPLEX on an OPTIONS or GOPTIONS statement. You can also use the BINDING= option in conjunction with DUPLEX. Before producing the second graph, set GSFMODE=APPEND in the GOPTIONS statement so that the driver knows to place succeeding graphs on the next available side of paper.

If DUPLEX is in effect, the page's inside (binding) margin is set equal to the current HORIGIN setting, and the outside margin is set equal to

\[ \text{XMAX} - \text{HSIZE} - \text{HORIGIN} \]

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 by:**
- GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Defaults:**
- GOPTIONS: NOERASE
- GDEVICE: ERASE=N

**Restriction:**
This option is not supported by Java or ActiveX.

**Syntax**

GOPTIONS: ERASE | NOERASE
GDEVICE: ERASE=Y | N

**Parameter Values**

- **ERASE**
- **ERASE=Y**
  - causes the graph to be erased when you press Enter after the graph has been displayed.
NOERASE
ERASE=N
causes the graph to remain on the display when you press Enter after the graph has been displayed. A blank Erase field in the Parameters window is the same as ERASE=N.

Details
ERASE is useful for those devices that overlay the graphics area and the message area—that is, those devices that have separate dialog box and graphics areas. On other devices, the graph is erased.

EXTENSION
Specifies the file extension for an external graphics file.

- Used by: GOPTIONS statement
- Default: device-dependent
- Restriction: This option is not supported by Java or ActiveX.
- See: GACCESS on page 557
  GSFNAME on page 566

Syntax
EXTENSION='file-type'

Parameter Values

- file-type
  a string up to eight characters long that is a file extension, such as GIF or CGM, that you want to append to an external file.

Details
The extension specified on EXTENSION= is used when the output destination is a storage location. The extension is ignored when the output destination is a file. To specify the output destination, you can use a FILENAME statement, or the graphics options GACCESS= or GSFNAME=.

Assuming that the output destination is a storage location,

- if EXTENSION='.', no extension is added to the 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 by: GOPTIONS statement
Default: NOFASTTEXT
Restriction: This option is not supported by Java or ActiveX.
Note: When the FASTTEXT option is specified and you use the Annotate facility, some round symbols in the Marker font, such as W, might appear noncircular, especially if you use the PNG device. In that case, use the NOFASTTEXT option instead.

**Syntax**

FASTTEXT | NOFASTTEXT

**FBY**
Selects the font for BY lines.

*Used by:* GOPTIONS statement

*Defaults:* (1) font specified by FTEXT=, if used
(2) deviceresident font
(3) simulate font

*Restriction:* This option is not supported by Java or ActiveX.

*See:* “BY Statement” on page 370

**Syntax**

FBY=By line-font

**Parameter Values**

*By line-font*

specifies the font for all BY lines on the graphics output. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 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 by:* GOPTIONS statement

*Default:* FCACHE=3

*Restriction:* This option is not supported by Java or ActiveX.
Syntax
FCACHE=number-fonts-open

Parameter Values
number-fonts-open
specifies the number of system fonts to keep open. Number-fonts-open must be greater than or equal to zero.

Details
Each font requires from 4K to 10K memory. Graphs that use many fonts can run faster if you set the value of number-fonts-open to a higher number. However, graphs that use multiple fonts might require too much memory on some computer systems if all the fonts are kept open. In such cases, set the value of number-fonts-open to a lower number to conserve memory.

FILECLOSE
Controls when the graphics stream file (GSF) is closed when you are using the device driver to send graphics output to a hard copy device.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
Default: DRIVERTERM (if a device is specified)
Restriction: This option is not supported by Java or ActiveX.
See: “Specifying the Graphics Output File Type for Your Graph” on page 109

Syntax
FILECLOSE=DRIVERTERM | GRAPHEND

Parameter Values
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 location or a storage location is the default destination for graphics output.

**Used by:** GOPTIONS statement

**Default:** device-dependent

**Restrictions:**
- This option is ignored if the device requires the output destination to be a storage location.
- This option is not supported by Java or ActiveX.

**See:**
- DEVOPTS on page 541
- GSFNAME on page 566

**Syntax**

FILEONLY | NOFILEONLY

**Parameter Values**

**FILEONLY**
- specifies that a file rather than a storage location is the default destination for graphics output.

**NOFILEONLY**
- specifies that a storage location is the default destination for graphics output, unless a file of the same name exists.

**Details**

Most devices use FILEONLY as the default. However, devices that require the output destination to be a storage location use NOFILEONLY as the default. For example, the HTML device requires a storage location because it produces two types of output (HTML files and GIF image files) that cannot be written to the same file.

To determine what the default is for a particular device, look at the settings for DEVOPTS bits 48 and 49.

For more information, see “Specifying the Graphics Output File Type for Your Graph” on page 109.

FILL

Specifies whether to use the device's hardware rectangle-fill capability.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:** device-dependent

**Restriction:** This option is not supported by Java or ActiveX.
Syntax

GOPTIONS: FILL | NOFILL
GDEVICE: FILL=Y | N

Parameter Values

FILL
FILL=Y
causes SAS/GRAPH to use the built-in hardware rectangle-filling capability of the device. A blank Fill field in the Parameters window is the same as FILL=Y.

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 by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
Default: device-dependent
Restriction: This option is not supported by Java or ActiveX.
See: FILL on page 552
PIEFILL on page 593
POLYGONFILL on page 595

Syntax

FILLINC= 0...9999

Comparisons

In order for FILLINC to have any effect, a software fill must be used. To force a software fill, use the options NOFILL, NOPIEFILL, and NOPOLYGONFILL in a GOPTIONS statement.

If FILLINC is set to 0 or 1, adjacent lines are used (solid fill with no gaps). If FILLINC is set to 2, a pixel-width line is skipped before drawing the next line of a fill.

This option can be useful for keeping plotters from over saturating a solid area and for speeding the plotting. Some inks spread on paper. The type of paper used can also affect ink spread.

FONT NAME

Specifies the device-resident font associated with CHARTYPE.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option
Requirement: if adding or modifying a CHARREC

See: CHARREC on page 529

Details

For information about the syntax, see “CHARREC” on page 529.

Use FONT NAME if you are adding to or modifying the device-resident fonts available for a particular device driver. The fonts that you specify must be valid for the output device. If you are using a device entry supplied by SAS, this parameter already is set for most available device-resident fonts.

FONTRES

Controls the resolution of Bitstream fonts.

Used by: GOPTIONS statement

Default: NORMAL

Restriction: Not supported by Java or ActiveX

See: FASTTEXT on page 549
     FCACHE on page 550
     RENDER on page 606
     RENDERLIB on page 607
     SWFONTRENDER on page 613

Syntax

FONTRES=NORMAL | PRESENTATION

Parameter Values

NORMAL

renders fonts in memory using integer rendering routines, which improves character drawing speed for most host systems. NORMAL has the same effect as specifying the default values for these graphics options:

    render=memory
    renderlib=work
    fasttext
    fcache=0

PRESENTATION

disables the storage or use of rendered versions of Bitstream fonts, but produces the fonts at their highest resolution. FONTRES=PRESENTATION has the same effect as specifying these graphics options:

    render=none
    renderlib=work
    nofasttext
    fcache=3
FORMAT

Sets the file format of the metacode file produced by the part of the Metagraphics device driver that is supplied by SAS.

Used by: GDEVICE procedure, GDEVICE Metagraphics window
Default: CHARACTER
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax

FORMAT=CHARACTER | BINARY

Comparisons

A blank field defaults to CHARACTER. For information about Metagraphics drivers, contact Technical Support.

FTEXT

Sets the default font for all text.

Used by: GOPTIONS statement
Default: Default deviceresident font (except the first title)
Restriction: This option is partially supported by Java and ActiveX.
See: FTITLE on page 556

Syntax

FTEXT=text-font

Parameter Values

text-font

specifies the font for all text on the graphics output. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 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 101 for more information.
**FTITLE**

Selects the default font for the first TITLE line.

**Used by:** GOPTIONS statement

**Defaults:**
1. Font specified by FTEXT=, if used
2. Value of the style variable
3. Device-resident font
4. Simulate font

**See:** FTEXT on page 555

---

### Syntax

FTITLE=\textit{title-font}

### Parameter Values

\textit{title-font}

specifies the font for the TITLE1 statement. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for information about specifying fonts.

### Details

\textit{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 101 for more information.

---

**FTRACK**

Controls the amount of space between letters in the Bitstream fonts supplied by SAS (Brush, Century, Swiss, and Zapf).

**Used by:** GOPTIONS statement

**Default:** TIGHT

**Restriction:** This option is not supported by Java or ActiveX.

---

### Syntax

FTRACK=LOOSE | NONE | NORMAL | TIGHT | TOUCH | V5

### Parameter Values

LOOSE

leaves the most visible space between characters and produces a longer string.
NONE
spacing depends on the size of the font. NONE might produce a shorter or longer
string than LOOSE for the same font at different point sizes, because some sizes add
space between the characters while others remove it.

NORMAL
is the recommended setting.

TIGHT
reduces the space between characters.

TOUCH
leaves the least visible space between characters.

V5
places a fixed amount of space between the characters and does not adjust for the
shape of the character. That is, it does not support kerning. This spacing is
compatible with Version 5 Bitstream fonts.

Details
The spacing that you specify with FTRACK= affects all Bitstream text in a graph. For
example, you cannot produce TIGHT Century type and LOOSE Zapf type
simultaneously. This option has no effect on other font types.

Because the value of FTRACK= is stored with the graph, the spacing that you specify
when the graph is created is always used when the graph is replayed.

GACCESS
Specifies the format or the destination or both of graphics data written to a traditional SAS/GRAPH device
driver or to a graphics stream file (GSF).

Used by:
GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default:
device-dependent

Restriction:
This option is not supported by Java, ActiveX, or shortcut devices.

See: Chapter 9, “Using Graphics Devices,” on page 79 for more information about
devices.

Syntax
GACCESS=output-format | 'output-format destination'

Parameter Values

output-format
specifies the format or the destination (the SAS log or a fileref) of the graphics data. Output-format
varies according to the operating environment. These values can be
specified in all operating environments:

SASGASTD
specifies that a continuous stream of data is written. SASGASTD is the default
for most devices and is typically appropriate when the output file will be sent
directly to a device. If you specify GACCESS=SASGASTD, use the
GSFNAME= and GSFMODE= graphics options or device parameters to direct
your graphics output to a GSF.
SASGAEDT
specifies that the file be host-specific edit format. Some hosts allow editing by inserting characters at the end of each record. SASGAEDT is typically used when the output file is to be edited later. If you specify GACCESS=SASGAEDT, use the GSFNAME= and GSFMODE= graphics options or device parameters to direct your graphics output to a GSF.

SASGAFIX
specifies that fixed-length records be written. (The record length is controlled by the value of the GSFLEN= graphics option or device parameter or the sixth byte of the PROMPTCHARS value.) The records are padded with blanks where necessary. SASGAFIX is typically used when the output file will be transferred to a computer that requires fixed-length records. If you specify GACCESS=SASGAFIX, use the GSFNAME= and GSFMODE= graphics options or device parameters to direct your graphics output to a GSF.

Note: The value of the GPROTOCOL= graphics option or device parameter can greatly affect the length of the records; for example, if GPROTOCOL=SASGPLCL, the length of the records is doubled.

SASGALOG
specifies that records are to be written to the SAS log.

GSASFILE
specifies that the records are to be written to the destination whose fileref is GSASFILE. The fileref can point to a specific external file or to an aggregate file location. See “FILENAME Statement” on page 47 for more information about specifying a fileref.

'output-format destination'
specifies the destination in addition to one of these output format values: SASGASTD, SASGAEDT, or SASGAFIX. Destination is the physical name of an external file or aggregate file location, or of a device. For details about specifying the physical name of a destination, see the SAS documentation for your operating environment.

This form is not available in all operating environments. See “Specifying the Graphics Output File Type for Your Graph” on page 109 for more information about creating graphics stream files.

Note: In the Gaccess field of the Host File Options window, you can specify a destination without an output format. In that case, the format defaults to SASGASTD. When you specify a value in the Gaccess field, you do not need to quote it.

Operating Environment Information
Depending on your operating environment, you might be able to specify other values for GACCESS=. See the SAS companion for your operating environment for additional values.

**GCOPIES**
Sets the current and maximum number of copies to be printed.

**Used by:** GOPTIONS statements, GDEVICE Parameters window, GDEVICE procedure, OPTIONS statement

**Defaults:** GOPTIONS: GCOPIES=(0,20)
GDEVICE: GCOPIES=0
Restriction: This option is not supported by Java or ActiveX.

Syntax
GOPTIONS: GCOPIES=(<current-copies>,<max-copies>)
GDEVICE: GCOPIES=current-copies

Parameter Values

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.

Details
If you do not specify GCOPIES, a default number of copies is searched for in this order:
1. the number of copies specified on an OPTIONS COPIES setting
2. 0 current copies, and 20 maximum copies.

Not all devices have the capability to print multiple copies. See the Gcopies field in the Parameters window for your device to determine its capabilities.

GEND
Appends an ASCII string to every graphics data record that is sent to a device or file.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Gend window
Restriction: This option is not supported by Java or ActiveX.
See: GSTART on page 567

Syntax
GEND='string' <...'string-n'>

Parameter Values

'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 \texttt{x}. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

**Details**

GEND is useful if you are creating a file and want to insert a carriage return at the end of every record. You can also use GEND in conjunction with the GSTART= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GEND= option rather than the GDEVICE Gend window because it is easier to code the string as text with GEND= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gend window.

*Note:* On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

**GEPILOG**

Sends a string to a device or file after all graphics commands are sent.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Gepilog window

**Restriction:** This option is not supported by Java or ActiveX.

**See:** PREGEPILOG on page 598

POSTGEPILOG on page 595

**Syntax**

\texttt{GEPILOG='string' <...'string-n'>}

**Parameter Values**

\texttt{'string'}

can be either of the following:

- \texttt{'hex-string'}\texttt{x}
- \texttt{'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 \texttt{x}. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.
Details

GEPILOG can be used in conjunction with the GPROLOG= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GEPILOG= option rather than the Gepilog window because it is easier to code the string as text with GEPILOG= than it is to convert the string to its ASCII representation, which is required to enter the string in the Gepilog window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEPILOG that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEPILOG is the recommended method.

GOUTMODE

Appends to or replaces the graphics output catalog.

<table>
<thead>
<tr>
<th>Used by:</th>
<th>GOPTIONS statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>APPEND</td>
</tr>
<tr>
<td>Restriction:</td>
<td>This option is not supported by Java or ActiveX.</td>
</tr>
</tbody>
</table>

Syntax

GOUTMODE=APPEND | REPLACE

Parameter Values

APPEND

adds each new graph to the end of the current catalog.

REPLACE

replaces the contents of the catalog with the graph or graphs produced by a single procedure.

CAUTION:

If you specify REPLACE, the entire contents of the catalog are replaced, not just graphs of the same name. Graphs are added to the catalog for the duration of the procedure, but when the procedure ends and a new procedure begins, the contents of the catalog are deleted and the new graph or graphs are added.

GPROLOG

Sends a string to device or file before graphics commands are sent.

<table>
<thead>
<tr>
<th>Used by:</th>
<th>GOPTIONS statement, GDEVICE procedure, GDEVICE Gprolog window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This option is not supported by Java or ActiveX.</td>
</tr>
<tr>
<td>See:</td>
<td>PREGPROLOG on page 598</td>
</tr>
<tr>
<td></td>
<td>POSTGPROLOG on page 596</td>
</tr>
</tbody>
</table>
Syntax

GPROLOG='string' <...'string-n'>

Parameter Values

'string'
    can be either of the following:
    • 'hex-string'X
    • 'character-string'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GPROLOG= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the GPROLOG window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details

GPROLOG can be used in conjunction with the GEPILOG= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GPROLOG= option rather than the GDEVICE Gprolog window because it is easier to code the string as text with GPROLOG= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gprolog window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

GPROTOCOL

Specifies the protocol module to use when routing output directly to a printer or creating a graphics stream file (GSF) to send to a device attached to your host by a protocol converter.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

Default: host-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

GPROTOCOL=module-name
**Parameter Values**

*module-name* can be one of these

- SASGPADE*
- SASGPAGL*
- SASGPASC
- SASPAXI*
- SASPCAB*
- SASPCHK*
- SASPDAT*
- SASPDCA*
- SASPHEX
- SASPHYD*
- SASPIDA*
- SASPIDX*
- SASPIMP*
- SASPIOC*
- SASPISI*
- SASPI24*
- SASPLCL*
- SASPNET*
- SASPMIC*
- SASPRTM*
- SASPSCS*
- SASPSTD
- SASPSTE*
- SASPTCX*
- SASPVAT*
- SASP497*
- SASP71

*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 by:** GOPTIONS statement

**Default:** GRAPHRC

**Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

GRAPHRC | NOGRAPHRC

**Parameter Values**

**GRAPHRC**

allows a return code at procedure termination. If the return code is not 0, the entire job might terminate.

**NOGRAPHRC**

always returns a step code of 0, even if the SAS/GRAPH program produced errors. As a result, the entire job's return code is unaffected by errors in any graphics procedure. NOGRAPHRC also overrides the ERRABEND system option.

**Details**

You typically use this option when you are running multiple jobs in a batch environment. It is useful primarily in an z/OS batch environment.

---

**GSFLEN**

Controls the length of records written to the graphics stream file (GSF).

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

**Default:** device-dependent

**Restriction:** This option is not supported by Java or ActiveX.

**See:** PROMPTCHARS on page 602

**Syntax**

GSFLEN=record-length

**Parameter Values**

**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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

**Default:** REPLACE

**Restriction:** This option is not supported by Java or ActiveX.

**See:** GACCESS on page 557  
GSFNAME on page 566

**Syntax**

GSFMODE=APPEND | PORT | REPLACE

**Parameter Values**

**APPEND**

adds the records to the end of a GSF designated by the GACCESS= or GSFNAME= graphics option or device parameter.

*Note:* The APPEND option is useful only with the GIF graphics output device, and the PCL and PostScript Universal Printer devices. When used with other devices, such as PNG, SVG, or PDF, only the first graph is displayed in the viewer.

If the file does not already exist, it is created. The destination can be either a specific file or an aggregate file storage location. If the destination of the GSF is a specific file and you specify APPEND, SAS/GRAPH will add the new records to an existing GSF of the same name. If the destination of the GSF is a file location and not a specific file, SAS/GRAPH will add the records to an external file whose name matches the name of the newly created catalog entry. For more information about how SAS/GRAPH names catalog entries, see “Specifying the Graphics Output File Type for Your Graph” on page 109.

*Note:* Some viewers of bitmapped output can view only one graph, even though multiple graphs are stored in the file. Therefore, it might appear that a file contains only one graph when in fact, it contains multiple graphs.

**PORT**

sends the records to a device or communications port. The GACCESS= graphics option or device parameter should point to the desired port or device.

**REPLACE**

replaces the existing contents of a GSF designated by the GACCESS= or GSFNAME= graphics option or device parameter. If the file does not exist, it is
created. REPLACE is always the default, regardless of the destination of the GSF. If the destination of the GSF is a specific file and you specify REPLACE, SAS/GRAPH will replace an existing GSF with the contents of a newly created GSF of the same name. If the destination of the GSF is a file location and not a specific file, SAS/GRAPH will replace an external file whose name matches the name of the newly created catalog entry. For more information about how SAS/GRAPH names catalog entries, see “Specifying the Graphics Output File Type for Your Graph” on page 109.

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 109 for more information about 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window

**Restriction:** Not valid for IBM32xx, linkable, Metagraphics, Java, or ActiveX drivers.

**See:**
- GACCESS on page 557
- GSFMODE on page 565

**Syntax**

GSFNAME=fileref

**Parameter Values**

fileref

specifies a fileref that points to the destination for the graphics stream file (GSF) output. Fileref must be a valid SAS fileref up to eight characters long and must be assigned with a FILENAME statement before running a SAS/GRAPH procedure that uses that fileref. The destination specified by the FILENAME statement can be either a specific file or an aggregate file location. See “FILENAME Statement” on page 47 for additional information about the FILENAME statement.

Details

Whether the resulting graphs are stored as one file or many files depends on both the type of destination and the setting of the GSFMODE= option.

If you specify a fileref with GSFNAME= and forget the FILENAME statement that defines the fileref, and if a destination is specified by the GACCESS= graphics option or device parameter, SAS/GRAPH assigns that destination to the fileref and sends the graphics output there. See also “GACCESS” on page 557.

See “Specifying the Graphics Output File Type for Your Graph” on page 109 for more information about creating graphics stream files.
**GSFPROMPT**

Specifies whether to write prompt messages to the graphics stream file (GSF).

- **Used by:** GOPTIONS statement
- **Default:** NOGSFPROMPT
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

GSFPROMPT | NOGSFPROMPT

**Comparisons**

When the GSF is processed by another program, that program can display the prompt messages. The default, NOGSFPROMPT, is compatible with Release 6.06.

Although the prompt messages appear if the graphics device is in eavesdrop mode, they do not wait for user response. If GSFPROMPT is on, the prompt messages are sent with the GSF to the device, regardless of the status of the graphics options PROMPT, GACCESS=, GSFMODE=, or GSFNAME=.

---

**FSIZE**

Sets the number of lines of display used for graphics for devices whose displays can be divided into graphics and text areas.

- **Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

FSIZE=lines

**Parameter Values**

*lines*

specifies the number of lines to be used for graphics. *Lines* is a nonnegative integer up to three digits long (0...999), and can be larger or smaller than the total number of lines that can be displayed at one time. If the number is larger, scroll the graph to see it all. If FSIZE 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Gstart window
Default: none
Restriction: This option is not supported by Java or ActiveX.
See: GEND on page 559

Syntax
GSTART='string <...'string-n'>

Parameter Values
'string'
can be either of the following:

• 'hex-string'X
• 'character-string'

In a GOPTIONS statement or in the GDEVICE procedure ADD or MODIFY statement, you can specify multiple strings with the GSTART= option. In this case, you can mix the formats, specifying some as ASCII hexadecimal strings and some as character strings. Multiple strings are concatenated automatically.

In the GSTART window, enter the hexadecimal string without either quotation marks or a trailing x. Note, however, that the string must be entered as a hexadecimal string.

PROC GOPTIONS always reports the value as a hexadecimal string.

Details
GSTART is useful when sending a file to a device that requires each record be prefixed with some character. You can use GSTART= in conjunction with the GEND= graphics option or device parameter.

If you must specify the long and complicated initialization strings required by some devices (for example, PostScript printers), it is easier to use the GOPTIONS GSTART= option rather than the GDEVICE Gstart window because it is easier to code the string as text with GSTART= than it is to convert the string to its ASCII representation, which is required to enter the string in the GDEVICE Gstart window.

Note: On non-ASCII hosts, only ASCII hexadecimal strings produce consistent results in all instances because of how the character strings are translated. In addition, the only way to specify a value for GEND that can be used by all hosts is to use an ASCII hexadecimal string. Therefore, using an ASCII hexadecimal string to specify a value for GEND is the recommended method.

GUNIT
Specifies the default unit of measure to use with height specifications.

Used by: GOPTIONS statement
Default: CELLS
Restrictions: This option is partially supported by Java and ActiveX.
This option is not honored by the ODS styles.
Syntax

GUNIT=units

Parameter Values

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, GUNIT is ignored and inches will be used.

GWAIT

Specifies the time between each graph displayed in a series.

Used by:  GOPTIONS statement
Default:  GWAIT=0
Restriction:  This option is not supported by Java or ActiveX.

Syntax

GWAIT=seconds

Parameter Values

seconds
  specifies the number of seconds between graphs. Seconds can be any reasonable positive integer. By default, GWAIT=0, which means that you must press the Enter key between each display in a series of graphs.

Details

GWAIT= enables you to view a series of graphs without having to press the Enter key between each display. For example, if you specify GWAIT=5, five seconds elapse between the display of each graph in a series. If you use the NOPROMPT graphics option, the GWAIT= graphics option is disabled.
HANDSHAKE
Specifications the type of flow control used to regulate the flow of data to a hard copy device.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:** host-dependent

**Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF

**Parameter Values**

**HARDWARE**

**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 that 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 by:** GOPTIONS statement
- **Default:** One cell unless the HTEXT= option is used
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** “BY Statement” on page 370

**Syntax**

HBY=By line-height <units>

**Parameter Values**

- **By line-height <units>**
  - specifies the height of BY line text; by default By line-height is 1. If you specify HBY=0, the BY headings are suppressed. For a description of units, see “Specifying Units of Measurement” on page 519.
  
  **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 by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers.
- **See:** HEADERFILE on page 572

**Syntax**

HEADER='command'
Parameter Values

command specifies a command that runs a user-written program that creates the file of HEADER records. Command is a string up to 40 characters long.

Details

For information about Metagraphics drivers, contact Technical Support.

HEADERFILE

Specifies the fileref for the file from which the Metagraphics driver reads HEADER records.

Used by: GDEVICE procedure, GDEVICE Metagraphics window
Restriction: Used only with user-supplied Metagraphics drivers.
See: HEADER on page 571

Syntax

HEADERFILE=fileref

Parameter Values

fileref specifies a valid SAS fileref up to eight characters long. Fileref must have been previously assigned with a FILENAME statement or a host command before running the Metagraphics driver. See “FILENAME Statement” on page 47 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 by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window
Restriction: This option is not supported by Java or ActiveX.
See: VORIGIN on page 620

Syntax

HORIGIN=horizontal-offset <IN | CM | PT>

Parameter Values

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 70 for details.

HOSTSPEC

Stores FILENAME statement options in the device entry.

**Used by:** GDEVICE procedure, GDEVICE Host File Options window

**Syntax**

HOSTSPEC='text-string'

**Parameter Values**

*text-string*

specifies FILENAME statement options that are valid for the operating environment. *Text-string* accepts characters in upper or lower case. See the SAS documentation for your operating environment for details.

**Details**

HOSTSPEC can be used when the driver dynamically allocates a graphics stream file or spool file. It can specify the attributes of the file, such as record format or record length. It cannot be used with Metagraphics drivers.

HPOS

Specifies the number of columns in the graphics output area.

**Used by:** GOPTIONS statement

**Default:** device-dependent: the value of the LCOLS or PCOLS device parameter

**Restriction:** This option is not supported by Java or ActiveX.

**See:** LCOLS on page 581
PCOLS on page 591
Syntax

HPOS=columns

Parameter Values

columns  
specifies the number of columns in the graphics output area, which is equivalent to the number of hardware characters that can be displayed horizontally. Specifying HPOS=0 causes the device driver to use the default hardware character cell width for the device.

Details

The HPOS= graphics option overrides the values of the LCOLS or PCOLS device parameters and temporarily sets the number of columns in the graphics output area. HPOS= does not affect the width of the graphics output area but merely divides it into columns. Therefore, you can use HPOS= to control cell width.

The values specified in the HPOS= and VPOS= graphics options determine the size of a character cell for the graphics output area and consequently the size of many graphics elements, such as device–resident text. The larger the size of the HPOS= and VPOS= values, the smaller the size of each character cell.

See “Overview” on page 69 for more information.

HSIZE

Sets the horizontal size of the graphics output area.

Used by:  
GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

Restriction:  
This option is partially supported by Java and ActiveX.

See:  
VSIZE on page 622  
XMAX on page 623

Syntax

HSIZE=horizontal-size <IN | CM | PT>

Parameter Values

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 by:** GOPTIONS statement

**Default:** One cell

**Restriction:** This option is partially supported by Java.

**Syntax**

HTEXT=\text-height <\textpack>`

**Parameter Values**

\text-height <\textpack>`

specifies the height of the text; by default \text-height is 1. For a description of \textpack, see “Specifying Units of Measurement” on page 519.

**Note:** If a value for \textpack 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 101 for more information.

---

**HTITLE**

Selects the default height used for the first TITLE line.

**Used by:** GOPTIONS statement

**Default:** Two cells unless HTEXT= is used

**Syntax**

HTITLE=\title-height <\textpack>`

**Parameter Values**

\title-height <\textpack>`

specifies the height of the text in the TITLE1 statement. By default, \title-height is 2. For a description of \textpack, see “Specifying Units of Measurement” on page 519.
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 101 for more information.

---

**IBACK**

Specifies an image file to display in a graph's background area.

**Used by:** GOPTIONS statement

**Restriction:** This option is partially supported by Java.

**See:** CBACK on page 526

**IMAGESTYLE on page 577**

---

**Syntax**

IBACK=fileref | 'external-file' | 'URL' | " "

**Parameter Values**

fileref

specifies a fileref that points to the image file that you want to use. Fileref must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement.

external-file

specifies the complete filename of the image file that you want to use. The format of external-file varies across operating environments.

URL

specifies the URL of the image file that you want to use.

**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 “Graph, Map, Tile Chart, and Contour Applets” in SAS/GRAPH: Java Applets and ActiveX Control User’s Guide 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 about graph styles, refer to the TEMPLATE procedure documentation in SAS Output Delivery System: Procedures Guide.
For a list of the file types that you use, see “Image File Types Supported by SAS/GRAPH” on page 331.

ID
Specifies the description string used by the Metagraphics driver.

**Used by:** GDEVICE procedure, GDEVICE Metagraphics window

**Restriction:** Used only with user-supplied Metagraphics drivers.

**Syntax**

```
ID='description'
```

**Parameter Values**

- `description` is a character string up to 70 characters long. If this field is blank, the name and description of the graph as specified in the PROC GREPLAY window of the GREPLAY procedure are used.

**Details**

For information about Metagraphics drivers, contact Technical Support.

---

**IMAGEPRINT**
Enables or disables image output

**Used by:** GOPTIONS statement

**Default:** IMAGEPRINT

**Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

```
IMAGEPRINT | NOIMAGEPRINT
```

**Parameter Values**

- `IMAGEPRINT` default value specifies that any images are to be included in graphics output.
- `NOIMAGEPRINT` specifies that images are to be withheld from graphics output.

---

**IMAGESTYLE**
Specifies how to display the image file that is specified on the IBACK= option.

**Used by:** GOPTIONS statement
### IMAGESTYLE

**Default:** TILE  
**Restriction:** This option is not supported by Java.

**Syntax**

IMAGESTYLE= TILE | FIT

**Parameter Values**

**TILE**

tile the image within the specified area. This copies the images as many times as needed to fit the area.

**FIT**

fit the image within the background area. This stretches the image, if necessary.

**Details**

*Note:* This option overrides the BackGroundImage and Image styles attribute in the graph styles. For more information about graph styles, refer to the TEMPLATE procedure documentation in *SAS Output Delivery System: Procedures Guide.*

---

### INTERACTIVE

Sets level of interactivity for Metagraphics driver.

**Used by:** GDEVICE procedure, GDEVICE Metagraphics window  
**Default:** USER  
**Restriction:** Used only with user-supplied Metagraphics drivers.

**Syntax**

INTERACTIVE=USER | GRAPH | PROC

**Parameter Values**

**USER**

specifies that the user-written part of the driver be executed outside of SAS/GRAPH.

**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 by:** GOPTIONS statement
- **Default:** NONINTERLACED
- **Restriction:** driver-dependent, GIF series of drivers only

**Syntax**

INTERLACED | NONINTERLACED

**Comparisons**

With interlacing it is possible to get a rough picture of what a large image will look like before it is completely drawn in your browser. Your browser might enable you to set an option that will determine how images are displayed.

**INTERPOL**

Sets the default interpolation value for the SYMBOL statement.

- **Used by:** GOPTIONS statement
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

INTERPOL=interpolation-method

**Parameter Values**

interpolation-method

specifies the default interpolation to be used when the INTERPOL= option is not specified in the SYMBOL statement. See “SYMBOL Statement” on page 412 for the complete syntax of all interpolation methods.

**ITERATION**

Specifies the number of times to repeat the GIF or SVG animation loop.

- **Used by:** GOPTIONS statement
- **Default:** 0
- **Restriction:** GIF and SVGANIM drivers only
- **See:** DELAY on page 538

Chapter 15, “Generating Animations,” on page 167

“Example: Creating an SVG Animation” on page 183
Syntax

The syntax for GIF animations is as follows:
ITERATION=0 | iteration-count

The syntax for SVG animations is as follows:
ITERATION=0 | 1

Parameter Values

0 | iteration-count

specifies the number of times that your complete GIF animation loop is repeated. It is assumed that the animation is always played once; this option specifies how many times the animation is repeated. A value of 0 causes the animation to loop continuously. The value of iteration-count can be up to 65535.

0 | 1

specifies whether your SVG animation loop is repeated continuously (0) or is played only once (1). When ITERATION=1, your users can use the SVG Viewer reset and play buttons to replay the animation as desired.

Note: If a value other than 0 or 1 is specified, a value of 1 is assumed when the SVG animation is replayed.

When an SVG animation plays, each page is held in view for a period of time. After that time, the page display fades out as the next page fades in. You can use the DELAY= graphics option to change the length of time that each page is held in view.

KEYMAP

Selects the key map to use.

Used by: GOPTIONS statement

Default: installation-dependent

Restriction: This option is not supported by Java or ActiveX.

Syntax

KEYMAP=key-map-name | NONE

Parameter Values

key-map-name

specifies the name of a key map.

NONE

suppresses the key map assigned by default to a non-U.S. keyboard. If you specify KEYMAP=NONE, text might be displayed incorrectly or not at all.

Details

Non-default key maps usually are used only with non-U.S. Keyboards.
LCOLS
Sets the number of columns in the graphics output area for landscape orientation.

**Used by:** GDEVICE procedure, GDEVICE Detail window

**Default:** device-dependent

**See:** HPOS on page 573
LROWS on page 582
PCOLS on page 591

---

**Syntax**

LCOLS=landscape-columns

**Parameter Values**

*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 69 for more information.

---

LFACCTOR
Selects the default hardware line thickness.

**Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:** device-dependent

**Restrictions:** Used only with devices that can draw hardware lines of varying thicknesses.
Not supported by Java or ActiveX.

---

**Syntax**

LFACCTOR=line-thickness-factor

**Parameter Values**

*line-thickness-factor*

can range from 0 through 9999. A value of 0 for LFACCTOR 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 by:** GDEVICE procedure, GDEVICE Detail window
- **Default:** device-dependent
- **See:** LCOLS on page 581, PROWS on page 604, VPOS on page 621

**Syntax**

LROWS=landscape-rows

**Parameter Values**

*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 69 for more information.

MAXCOLORS

Sets the total number of colors that can be displayed at once.

- **Used by:** GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **See:** PENMOUNTS on page 592

**Syntax**

MAXCOLORS=number-of-colors

**Parameter Values**

*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 by:** GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent

**Syntax**
MAXPOLY=number-of-vertices

**Parameter Values**
- **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 by:** GDEVICE procedure, GDEVICE Detail window
- **Default:** device-dependent

**Syntax**
MODEL=model-number

**Parameter Values**
- **model-number** is a nonnegative integer up to five digits long that is the model number designated by SAS for the corresponding device. It is not the same as a manufacturer's model number.

**Details**
Do not change this field in drivers supplied by SAS or in drivers that you copy from drivers supplied by SAS.

MODULE
Specifies the name of the corresponding executable driver module for the device.
Used by: GDEVICE procedure, GDEVICE Detail window  
Default: device-dependent

**Syntax**

\[ \text{MODULE=driver-module} \]

**Parameter Values**

*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 drivers supplied by SAS or in drivers that you copy from drivers supplied by SAS.

---

**NAK**

Specifies the negative response for software handshaking for Metagraphics drivers.  

**Used by:** GDEVICE procedure, GDEVICE Metagraphics window  
**Restriction:** Used only with user-supplied Metagraphics drivers.

**Syntax**

\[ \text{NAK='negative-handshake-response'}X \]

**Parameter Values**

*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 by:** GOPTIONS statement  
**Default:** (0.0625, 0.0625) IN  
**Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

\[ \text{OFFSHADOW=(x <units>, y <units>) | (x,y) <units>} \]
Parameter Values

\(x, y\)
specify the width \((x)\) and depth \((y)\) of the drop shadow generated by the LEGEND statement.

Details

If a value for \textit{units} is not specified, the current units associated with the GUNIT graphics option are used. For a description of \textit{units}, see “Specifying Units of Measurement” on page 519.

The values specified by OFFSHADOW= are used with the CSHADOW= and CBLOCK= options in a LEGEND statement. For details, see “LEGEND Statement” on page 377.

PAPERDEST

Specifies which output bin the printer should use if multiple bins are available on the device.

\textbf{Used by:}\n\quad GOPTIONS statement, OPTIONS statement

\textbf{Default:}\n\quad 1 (the upper output bin)

\textbf{Restrictions:}\n\quad This option is hardware-dependent.
\quad PostScript printers require a PPD file.
\quad This option is not supported by Java or ActiveX.

\textbf{See:}\n\quad PAPERSOURCE on page 588
\quad PPDFILE on page 597
\quad “Options Used in Both GOPTIONS and OPTIONS Statements” on page 519

\textbf{Syntax}

\texttt{PAPERDEST=bin}

\textbf{Parameter Values}

\textit{bin}

specifies the name or number of the output bin. Values for \textit{bin} depend on the type of printer and can be one of the following:

\textit{bin}

\quad the name or number of the output bin – for example, PAPERDEST=4,
\quad PAPERDEST=BIN2, PAPERDEST=SIDE

\textit{long bin name}

\quad a character string that is the name of the output bin – for example,
\quad PAPERDEST=‘Top Output Bin’. Names with blanks or special characters must be quoted.

For PostScript printers, the value for \textit{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\(\text{n}\)', 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

**Default:** PAPERFEED=0.0 IN

**Restrictions:**
- This option is device-dependent.
- This option is not supported by Java or ActiveX.

**Syntax**

PAPERFEED=feed-increment <IN | CM>

**Parameter Values**

feed-increment <IN | CM>

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 by:** GOPTIONS statement

**Default:** maximum dimensions specified in the device driver

**Restriction:** ZETA plotters and KMW rasterizers

**Syntax**

PAPERLIMIT=width <IN | CM>
Parameter Values

width <IN | CM>

specifies the paper width 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 by: GOPTIONS statement, OPTIONS statement

Default: device-dependent

Restrictions: This option is hardware-dependent.

PostScript printers require a PPD file.

This option is not supported by Java or ActiveX.

See: PAPERSOURCE on page 588

PPDFILE on page 597

“Options Used in Both GOPTIONS and OPTIONS Statements” on page 519

Syntax

PAPERSIZE='size-name'

Parameter Values

size-name

specifies the name of a paper size, such as LETTER, LEGAL, or A4.

If you do not specify the PAPERSIZE= option, the PAPERSIZE= option setting in an OPTIONS statement is used. If no OPTIONS statement sets a paper size, the value for paper size is device-dependent:

• The universal printing devices use the size specified in the Page Setup dialog box.
• All other printer devices use the LETTER paper size.

Details

Typically, you might use the PAPERSIZE= option with the Output Delivery System (ODS). For some printers, the PAPERSIZE= option overrides the PAPERSOURCE= option selection.

For PostScript devices, the name must match the name of a paper size in the PPD file. Refer to the PPD file for a list of valid names. Size-name is case-insensitive and can contain a subset of the full name. For example, if the name in the PPD file is *PageSize
A4/A4, you can specify PAPERSIZE='A4'. If a PPD file is not specified, the PAPERSIZE= option is ignored.

For PCL devices, the device driver searches the SAS Registry for supported paper size values. To see the supported list of sizes, submit the following statements:

```sas
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 by:** GOPTIONS statement, OPTIONS statement
- **Default:** device-dependent
- **Restrictions:**
  - This option is hardware-dependent.
  - PostScript printers require a PPD file.
  - This option is not supported by Java or ActiveX.

**See:**
- PAPERDEST on page 585
- PAPERSIZE on page 587
- PPDFILE on page 597
- "Options Used in Both GOPTIONS and OPTIONS Statements" on page 519

### Syntax

```
PAPERSOURCE=tray
```

### Parameter Values

- **tray**
  - specifies the name or number of the paper tray. Values for tray depend on the type of printer and can be one of the following:
    - *tray*
      - 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\textsubscript{n}', where \(n\) is a number from 2 to 21
• 'MANUAL'
• 'MANUAL\_ENVELOPE'
• 'TRAY\textsubscript{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 by:** GOPTIONS statement, OPTIONS statement

**Default:** PLAIN

**Restrictions:** This option is hardware-dependent.
PostScript printers require a PPD file.
This option is not supported by Java or ActiveX.

**See:** PPDFILE on page 597
“Options Used in Both GOPTIONS and OPTIONS Statements” on page 519

### Syntax

PAPERTYPE='\textit{type-name}'

### Parameter Values

\textit{type-name}

specifies the name of a paper type. Valid values depend on the type of printer.

For PostScript devices, \textit{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. \textit{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, \textit{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 by: GDEVICE procedure, GDEVICE Metagraphics window
Default: PATH=0
Restriction: Used only with user-supplied Metagraphics drivers.

Syntax
PATH=angle-increment

Parameter Values
angle-increment
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 by: GOPTIONS statement
Default: NOPCLIP
Restriction: This option is not supported by Java or ActiveX.
See: POLYGONCLIP on page 594

Syntax
PCLIP |NOPCLIP

Parameter Values
PCLIP
stores clipped polygons with the graph in the default catalog WORK.GSEG, or in the catalog that you specify.

NOPCLIP
stores the unclipped form of the polygon and causes the polygon to be clipped when replayed.

Details
The effects of this option are seen only when you use the graphics editor to edit a graph.
When a procedure produces a graph with intersecting polygons or blanking areas, it clips portions of the polygons to prevent the ones behind from showing through. When the graph is created and stored in a catalog, if PCLIP is in effect, the clipped form of the polygon is stored with it. If NOPCLIP is specified, the complete polygon is stored in the catalog and the graph is clipped each time it is replayed.

For example, suppose you create a block map like the one in Figure 25.3 on page 591.

![Figure 25.3 Intersecting Polygons](image)

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 25.4 on page 591.

![Figure 25.4 Clipped Polygon with PCLIP Option](image)

NOPCLIP stores the map area in its unclipped form, as shown in Figure 25.5 on page 591.

![Figure 25.5 Clipped Polygon with NOPCLIP Option](image)

In this case, when the graph is recalled from the catalog, the map area polygon must be clipped before it is displayed with the block. If you plan to edit the graph with the graphics editor, specify NOPCLIP so that polygons retain their original form.

**PCOLS**

Sets the number of columns in the graphics output area for portrait orientation.

- **Used by:** GDEVICE procedure, GDEVICE Detail window
- **Default:** device-dependent
See: HPOS on page 573  
LCOLS on page 581  
PROWS on page 604

Syntax

PCOLS=portrait-columns

Parameter Values

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 69 for more information.

PENMOUNTS

Specifies the number of active pens or colors.

Used by: GOPTIONS statement  
Default: device-dependent  
Restriction: This option is not supported by Java or ActiveX.

See: MAXCOLORS on page 582

Syntax

PENMOUNTS=active-pen-mounts

Parameter Values

active-pen-mounts specifies the number of pens for a plotter with multiple pens. After the specified  
number of pens have been used, you are prompted to change the pens.

Details

For devices that are not pen plotters, PENMOUNTS= can be used to indicate the number  
of colors that can be displayed at one time. In this case, PENMOUNTS= performs the  
same function as the MAXCOLORS device parameter except that the value specified for  
MAXCOLORS includes the background color and PENMOUNTS refers to foreground  
colors only. Thus, PENMOUNTS=4 implies MAXCOLORS=5.

PENMOUNTS= overrides the value of the MAXCOLORS device parameter. You can  
specify MAXCOLORS= in a GOPTIONS statement as a synonym for PENMOUNTS=.
**PENSORT**

Specifies whether plotters draw graphics elements in order of color.

- **Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

GOPTIONS: PENSORT | NOPENSORT

GDEVICE: PENSORT=Y | N

**Parameter Values**

**PENSORT**

**PENSORT=Y**

causes the plotter to draw all graphics elements of one color at one time. For example, it draws all the red elements in the output, then all the blue elements, and so on. This specification is compatible with previous releases. Use it for plotters with real pens.

**NOPENSORT**

**PENSORT=N**

causes the plotter to draw each element as it is encountered, regardless of its color. For example, the plotter might draw a red circle, then a blue line, and then a red line, and so on. This method is best for electrostatic printers implemented with Metagraphics drivers of TYPE=PLOTTER. In addition, NOPENSORT enables you to specify nonstandard color names.

**PIEFILL**

Specifies whether to use the device's hardware pie-fill capability.

- **Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

GOPTIONS: PIEFILL | NOPIEFILL

GDEVICE: PIEFILL=Y | N
Parameter Values

PIEFILL

PIEFILL=Y causes SAS/GRAPH to use the built-in hardware capability of the device, if available, to fill pies and pie sections. A blank Piefill field in the Parameters window is the same as PIEFILL=Y.

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 by: GOPTIONS statement

Default: device-dependent

Restriction: This option is not supported by Java or ActiveX.

See: PCLIP on page 590

Syntax

POLYGONCLIP | NOPOLYGONCLIP

Parameter Values

POLYGONCLIP

specifies polygon clipping, which enables a clipped polygon to be filled with a hardware pattern. POLYGONCLIP affects only graphs that have blanking areas or intersecting polygons.

NOPOLYGONCLIP

specifies line clipping; a polygon that has been line-clipped cannot use a hardware pattern.

Details

Clipping is the process of removing part of one polygon when two polygons intersect. For example, in a block map, a block might overlap the boundary of its map area. In this case, the polygon that makes up the map area is clipped so that you do not see the boundary line behind the block. (See Figure 25.3 on page 591 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

<table>
<thead>
<tr>
<th>GOPTIONS: POLYGONFILL</th>
<th>NOPOLYGONFILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDEVICE: POLYFILL=Y</td>
<td>N</td>
</tr>
</tbody>
</table>

**Parameter Values**

- **POLYGONFILL POLYFILL=Y** causes SAS/GRAPH to use the built-in hardware capability of the device to fill polygons. A blank Polyfill field in the Parameters window is the same as POLYGONFILL.

  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 by:** GOPTIONS statement
- **Default:** Null string
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** GEPILOG on page 560
  PREGEPILOG on page 598

**Syntax**

POSTGEPILOG='string'
PARAMETER VALUES

'string'

can be either of the following:

- 'hex-string'
- '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 by:** GOPTIONS statement

**Default:** Null string

**Restriction:** This option is not supported by Java or ActiveX.

**See:** GPROLOG on page 561

**Syntax**

POSTGPROLOG='string'

PARAMETER VALUES

'string'

can be either of the following:

- 'hex-string'
- '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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window

**Restriction:** This option is not supported by Java or ActiveX.

**See:** FILECLOSE on page 551

**Syntax**

POSTGRAPH1='system-command(s)'

POSTGRAPH2='system-command(s)'

596  Chapter 25  •  Graphics Options and Device Parameters Dictionary
Parameter Values

system-command(s)
specifies one or more valid system commands. The string can contain upper- or lowercase characters. Separate multiple commands with a command delimiter, which is host-specific; for example, some operating environments use a semicolon. The total length of the string cannot exceed 72 characters. The commands are executed right after the graph is produced.

Details

If you want to use a host command to send output to the device after each graph executes, use the POSTGRAPH parameter with FILECLOSE=GRAPHEND.

PPDFILE

Specifies the location of an external file containing PostScript Printer Description (PPD) information.

Used by: GOPTIONS statement

Restriction: PostScript printers only

See: BINDING on page 525
     COLLATE on page 531
     DUPLEX on page 547
     PAPERDEST on page 585
     PAPERSIZE on page 587
     PAPERSOURCE on page 588
     PAPERTYPE on page 589
     REVERSE on page 608

Syntax

PPDFILE=fileref | 'external-file'

Parameter Values

fileref
specifies a fileref that points to the PPD file that you want to use. Fileref must be a valid SAS fileref up to eight characters long and must have been previously assigned with aFILENAME statement.

external-file
specifies the complete filename of the PPD file that you want to use. The format of external-file varies across operating environments. For details, see the SAS documentation for your operating environment.

Details

A PostScript Printer Description (PPD) file is a text file that contains commands required to access features of the device. These files are available from Adobe. Also, many printer manufacturers provide the appropriate PPD file for their PostScript printers.
PREGEPILOG

Specifies data to send immediately before the data that is stored in the Gepilog field of the device entry is sent.

- **Used by:** GOPTIONS statement
- **Default:** Null string
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** GEPILOG on page 560
  POSTGEPILOG on page 595

**Syntax**

PREGEPILOG='string'

**Parameter Values**

- **'string'**
  - can be either of the following:
    - **'hex-string'**
    - **'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 by:** GOPTIONS statement
- **Default:** Null string
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** GPROLOG on page 561
  POSTGPROLOG on page 596

**Syntax**

PREGPROLOG='string'

**Parameter Values**

- **'string'**
  - can be either of the following:
    - **'hex-string'**
    - **'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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Host Commands window
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** FILECLOSE on page 551

**Syntax**

```
PREGRAPH1='system-command(s)'
PREGRAPH2='system-command(s)'
```

**Parameter Values**

- **system-command(s)**
  - Specifies one or more valid system commands. The string can contain upper- or lowercase characters. Separate multiple commands with a command delimiter, which is host-specific; for example, some operating environments use a semicolon. The total length of the string cannot exceed 72 characters. The commands are executed immediately before the graph is produced.

**Details**

The PREGRAPH parameter should be used with FILECLOSE=GRAPHEND.

---

**PROCESS**

Specifies the command that translates the metafile into commands for the device.

- **Used by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers.
- **See:** INTERACTIVE on page 578

**Syntax**

```
PROCESS='command'
```

**Parameter Values**

- **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 by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers.

Syntax

```
PROCESSINPUT=fileref
```

Parameter Values

- **fileref**
  
  specifies a valid SAS fileref up to eight characters long. Fileref must be assigned with a FILENAME statement or a host command before running the Metagraphics driver. See “FILENAME Statement” on page 47 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 by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers.

Syntax

```
PROCESSOUTPUT=fileref
```

Parameter Values

- **fileref**
  
  specifies a valid SAS fileref up to eight characters long. Fileref must be assigned with a FILENAME statement or a host command before running the Metagraphics driver. See “FILENAME Statement” on page 47 SAS/GRAPH: Reference for additional information.

Details

For information about Metagraphics drivers, contact Technical Support.
**PROMPT**

Specifies whether prompts are issued.

**Used by:**  
GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

**Default:**  
device-dependent

**Restriction:**  
This option is not supported by Java or ActiveX.

### Syntax

GOPTIONS: PROMPT | NOPROMPT

GDEVICE: PROMPT=0...7

### Parameter Values

**PROMPT**  
causes all prompts to be displayed.

**NOPROMPT**  
suppresses all prompts. NOPROMPT overrides the GWAIT= graphics option.

**PROMPT=0...7**  
in the GDEVICE procedure, specifies the level of prompting:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>provides no prompting</td>
</tr>
<tr>
<td>1</td>
<td>issues start-up messages only. Start-up messages are messages such as PLEASE PRESS RETURN TO CONTINUE.</td>
</tr>
<tr>
<td>2</td>
<td>signals end of graph if device is a video display or sends message to change paper if device is a plotter.</td>
</tr>
<tr>
<td>3</td>
<td>combines the effects of 1 and 2.</td>
</tr>
<tr>
<td>4</td>
<td>sends a message to mount pens if the device is a plotter.</td>
</tr>
<tr>
<td>5</td>
<td>combines the effects of 4 and 1.</td>
</tr>
<tr>
<td>6</td>
<td>combines the effects of 4 and 2.</td>
</tr>
<tr>
<td>7</td>
<td>sends all prompts</td>
</tr>
</tbody>
</table>

**Note:** If you specify either 0 for the PROMPT device parameter or NOPROMPT in a GOPTIONS statement for a display device, the display clears immediately after the graph is drawn.
Details

In the GDEVICE Parameters window, the PROMPT parameter consists of four fields that describe the type of prompt:

- **start up**
  issues a message to turn the device on (if the device is a 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** host-dependent
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** GSFLEN on page 564
  HANDSHAKE on page 570

**Syntax**

PROMPTCHARS='prompt-chars-hex-string'X

**Parameter Values**

- **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
Select an XON or DC1 character as a prompt. If the host does not send a special character for a prompt, set this byte to 00.

**byte 2**

is the ASCII code of the echo-terminator character (for software handshaking). This character is sent at the beginning of each record.

**byte 3**

prevents splitting commands across records if the value is 01. If you are creating a graphics stream file to send to a device at a later time, and there is the possibility that extra characters will be added between records during transmission, setting the third byte to 01 reduces the likelihood that the extra characters are interpreted as graphics commands and cause stray lines or other device characters. If the third byte is set to 00, the driver makes the records as long as possible and splits device commands across records if necessary. Setting the third byte to 00 is more efficient but is more likely to 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 570), but it can also be used to control the length of records written by most drivers. You can also use the GSFLEN= graphics option for this purpose. PROMPTCHARS can be used to control the length of records written by most drivers. You can also use the GSFLEN= graphics option for this purpose.
PROWS
Sets the number of rows in the graphics output area for portrait orientation.

**Used by:** GDEVICE procedure, GDEVICE Detail window
**Default:** device-dependent
**See:** LROWS on page 582
PCOLS on page 591
VPOS on page 621

**Syntax**
PROWS=portrait-rows

**Parameter Values**
*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 69 for more information.

QMSG
Specifies whether log messages are held until after the graphics output is displayed.

**Used by:** GDEVICE procedure, GDEVICE Detail window
**Default:** device-dependent

**Syntax**
GOPTIONS: QMSG | NOQMSG
GDEVICE: QMSG=Y | N

**Parameter Values**
*QMSG*
- QMSG=Y
  queues driver messages while the device is in graphics mode (default for video devices).
*NOQMSG*
- NOQMSG
  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 by:** GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **See:** FILL on page 552

**Syntax**

```
RECTFILL='rectangle-fill-hex-string'X
```

**Parameter Values**

*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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Fill pattern</th>
<th>Bit</th>
<th>Fill pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R1</td>
<td>9</td>
<td>L4</td>
</tr>
<tr>
<td>2</td>
<td>R2</td>
<td>10</td>
<td>L5</td>
</tr>
<tr>
<td>3</td>
<td>R3</td>
<td>11</td>
<td>X1</td>
</tr>
<tr>
<td>4</td>
<td>R4</td>
<td>12</td>
<td>X2</td>
</tr>
<tr>
<td>5</td>
<td>R5</td>
<td>13</td>
<td>X3</td>
</tr>
<tr>
<td>6</td>
<td>L1</td>
<td>14</td>
<td>X4</td>
</tr>
<tr>
<td>7</td>
<td>L2</td>
<td>15</td>
<td>X5</td>
</tr>
<tr>
<td>8</td>
<td>L3</td>
<td>16</td>
<td>S</td>
</tr>
</tbody>
</table>

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 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 398 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 by:** GOPTIONS statement

**Default:** MEMORY

**Restriction:** This option is not supported by Java or ActiveX.

**See:** RENDERLIB on page 607

**Syntax**

RENDER=APPEND | DISK | MEMORY | NONE | READ

**Parameter Values**

**APPEND**

creates files to store rendered versions of Bitstream fonts if the files do not already exist, reads previously rendered characters from the font files, and appends rendered versions of new characters to the font files when the SAS/GRAPH procedure terminates.

**DISK**

creates files to store rendered versions of Bitstream fonts if the files do not already exist, reads previously rendered characters from the font files, and appends rendered versions of new characters to the font files as they are encountered. This method is slower on some hosts, but it can work in memory-constrained conditions where the other rendering methods fail.

**MEMORY**

renders all fonts in memory without creating any font files on disk. Font files are not used even if they already exist. New characters are not written to existing font files when SAS/GRAPH procedures terminate.

This is the default and should be the fastest method on hosts that support virtual memory.

**NONE**

disables the font rendering features.

**READ**

reads existing rendered font files but does not create new font files or write new characters to existing font files. This is useful only when font files already exist in the rendered font library.
Details
The memory capacity and input/output characteristics of your host system determine which value for the RENDER= option provides the best performance.

RENDERLIB
Specifies the SAS library in which rendered font files are stored.

- **Used by:** GOPTIONS statement
- **Default:** WORK
- **Restriction:** This option is not supported by Java or ActiveX.

See: RENDER on page 606

Syntax
RENDERLIB=libref

**Parameter Values**
libref
specifies a previously defined libref that identifies the SAS library. The default library is WORK. See “LIBNAME Statement” on page 47 for more information about assigning a libref.

REPAINT
Specifies how many times to redraw the graph.

- **Used by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.

Syntax
REPAINT=redraw-factor

**Parameter Values**
redraw-factor
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 default values, cancels global statements, or both.

**Used by:**
- GOPTIONS statement

**Example:**
*“Example 9: Creating a Web Page with Drill-Down Functionality Using the ODS HTML Statement” on page 506*

**Syntax**

RESET=ALL | GLOBAL | statement-name | (statement-name(s))

**Parameter Values**

**ALL**
- sets all graphics options to defaults and cancels all global statements.

**GLOBAL**
- cancels all global statements (AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, and TITLE). Options in the GOPTIONS statement are unaffected.

**statement-name**
- resets or cancels only the specified global statements. For example, RESET=PATTERN cancels all PATTERN statements only. To cancel several statements at one time, enclose the statement names in parentheses. For example, RESET=(TITLE FOOTNOTE AXIS) cancels all TITLE, FOOTNOTE, and AXIS statements.

**Note:** RESET=GOPTIONS sets all graphics options to defaults but does not cancel any global statements.

**Details**

RESET=ALL or RESET=GOPTIONS must be the first option specified in the GOPTIONS statement. Otherwise, the graphics options that precede the RESET= option in the GOPTIONS statement are reset. Other options can follow the RESET= graphics option in the statement.

REVERSE

Specifies whether to print the output in reverse order, if reverse printing is supported by the device.

**Used by:**
- GOPTIONS statement

**Default:**
- NOREVERSE

**Restrictions:**
- This option is hardware-dependent.
- PostScript printers require a PPD file.
- This option is not supported by Java or ActiveX.

**See:**
- PPDFILE on page 597
Syntax
REVERSE | NOREVERSE

Comparisons
The purpose of REVERSE is to control the stacking order of printer output, depending on how the printer stacks paper. On some printers, reverse implies using the alternate output bin (back of the printer).

For PCL devices, REVERSE sends output to the LOWER out bin, which is the face-up output bin.

For PostScript devices, if the PPD file has an “OutputOrder” entry and one of its entries is “Reverse,” the device supports reverse order printing and the appropriate PostScript code to activate reverse will be used. If the PPD file does not have an “OutputOrder” entry but does have a “PageStackOrder” entry and corresponding OutputBin value, then reverse order printing is supported indirectly, using the PPD file's PageStackOrder or OutputBin entries.

Note: Some PostScript devices implement Reverse as the default output mode for one of the output bins. In this case, selecting either the “reverse” output bin or specifying REVERSE mode produces identical results.

### ROTATE
Specifies whether and how to rotate the graph.

**Used by:** 
- GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

**Restrictions:** 
This option is supported by printer-type Universal Printer devices only, which include PDF, POSTSCRIPT, and PCL.

This option is not supported by the SVG, SVGZ, SVGVIEW, SVGT, and SVGANIM devices.

This option is partially supported by the display devices.

**Syntax**
- GOPTIONS: ROTATE=LANDSCAPE | PORTRAIT
- GOPTIONS: ROTATE | NOROTATE
- GDEVICE: ROTATE=LANDSCAPE | PORTRAIT

**Parameter Values**
- **ROTATE | NOROTATE**
  - specifies whether to rotate the graph 90 degrees from its default orientation.
- **ROTATE=LANDSCAPE**
  - specifies landscape orientation (the graph is wider than it is high).
- **ROTATE=PORTRAIT**
  - specifies portrait orientation (the graph is higher than it is wide).

**Details**
If you do not specify a rotation, a default orientation is searched for in this order:
1. the ORIENTATION setting in an OPTIONS statement
2. the device-dependent default

By default, many of the graphics output devices such as PNG and GIF use the landscape orientation while the printer devices use the portrait orientation. When you direct the output of a landscape device to printer device, the graph is scaled to fit the portrait dimensions, which might not be desirable. You can use the ROTATE or ROTATE=LANDSCAPE graphics option to rotate the graph 90 degrees on the portrait-oriented printer easel in order to maintain the original dimensions of the graph. For example:

```sas
    goptions reset=all rotate=landscape;
```

However, because the graph is being rendered by devices with different orientations in that case, unexpected results might occur. When directing landscape output to a printer device, the best approach is to use the ORIENTATION=LANDSCAPE system option rather than the ROTATE or ROTATE=LANDSCAPE graphics option. For example:

```sas
    options orientation=landscape;
```

The ORIENTATION=LANDSCAPE system option sets the orientation of all of the output devices to landscape, which eliminates any problems associated with different device orientations.

See “ORIENTATION= System Option” in SAS System Options: Reference for information about the ORIENTATION= system option.

---

**ROTATION**

Sets the increment of the angle by which the device can rotate any given letter in a string of text in a Metagraphics driver.

- **Used by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Default:** ROTATION=0
- **Restriction:** Used only with user-supplied Metagraphics drivers.

**Syntax**

```
ROTATION=angle-increment
```

**Parameter Values**

- **angle-increment**

  Specifies the increment of the angle at which to rotate individual characters (for example, every 5 degrees, every 45 degrees, and so on). *Angle-increment* is an integer in the range 0 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 by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option
Default: 0
See: CHARREC on page 529

Details
For information about the syntax, see “CHARREC” on page 529.

If you are using a device driver from SASHELP.DEVICES, this parameter already is set for device–resident fonts that have been defined for your installation. For scalable fonts, you can specify 1 for ROWS, and the actual number of rows are computed based on the current text width. If you are adding to or modifying device-resident fonts available for a particular device driver, specify a positive value for the ROWS device parameter. If ROWS is greater than 0, it overrides the values of the LROWS and PROWS device parameters.

SCALABLE
Specifies whether a font is scalable.

Used by: GDEVICE Chartype window, GDEVICE procedure, CHARREC= option
Default: device-dependent
See: CHARTYPE on page 530

Details
See “CHARREC” on page 529 for information about the syntax.

A device-resident font is scalable if it can be used with any combination of rows and columns. Use the SCALABLE device parameter if you are adding to or modifying the fonts available for a particular device driver. If you are using a device driver from SASHELP.DEVICES, this parameter already is set for device-resident fonts that have been defined for your installation.

SIMFONT
Specifies a SAS/GRAPH font to use if the default device-resident font cannot be used.

Used by: GOPTIONS statement
Default: SIMULATE
Restriction: This option is not supported by Java or ActiveX.
Syntax

SIMFONT=SAS/GRAPH-font

Parameter Values

SAS/GRAPH-font
specifies a SAS/GRAPH font to use instead of the default device-resident font. 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.


SPEED

Selects pen speed for plotters with variable speed selection.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
Default: device-dependent
Restriction: This option is not supported by Java or ActiveX.

Syntax

SPEED=pen-speed

Parameter Values

pen-speed
specifies a percentage (1 through 100) of the maximum pen speed for the device. For example, SPEED=50 slows the drawing speed by half. In general, slowing the drawing speed produces better results.

By default, the value of SPEED is the normal speed for the device.

SWAP

Specifies whether to reverse BLACK and WHITE in the graphics output.
### SWFONTRENDER

Specifies the method used to render system fonts.

**Used by:**
- GOPTIONS statement

**Default:**
- device-dependent

**Restriction:**
- This option is not supported by Java or ActiveX.

**Syntax**

```
SWFONTRENDER = POLYGON | SCANLINE
```

**Parameter Values**

```
SWFONTRENDER = POLYGON
uses polygon rendering
```

---

**SWFONTRENDER**

Specifies the method used to render system fonts.

**Used by:**
- GOPTIONS statement

**Default:**
- device-dependent

**Restriction:**
- This option is not supported by Java or ActiveX.

**Syntax**

```
SWFONTRENDER = POLYGON | SCANLINE
```

**Parameter Values**

```
SWFONTRENDER = POLYGON
uses polygon rendering
```

---

**Syntax**

```
GOPTIONS: SWAP | NOSWAP
GDEVICE: SWAP=Y | N
```

**Parameter Values**

- **SWAP**
  - **SWAP=Y**
    - swaps BLACK for WHITE and vice versa.
  - **SWAP=N**
    - does not swap the colors. A blank **Swap** field in the Parameters window is the same as **SWAP=N**.

**Details**

SWAP does not affect the background color and affects only BLACK and WHITE foreground colors specified as predefined SAS color names. SWAP ignores BLACK and WHITE specified in HLS, RGB, or gray-scale format. This option is useful when you want to preview a graph on a video device and send the final copy to a printer that uses a white background.

```
goptions reset=all cback=blue ctitle=black swap;
title1 h=8 'swap test';
title2 h=8 'another title';
proc gslide border;
run;
```
SWFONTRENDER = 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 by:** GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** SYMBOLS on page 614

**Syntax**
GOPTIONS: SYMBOL | NOSYMBOL
GDEVICE: SYMBOL=Y | N

**Parameter Values**
SYMBOL
SYMBOL=Y
causes SAS/GRAPH to use the built-in symbol-drawing capability of the device, if available. A blank Symbol field in the Parameters window is the same as SYMBOL=Y.

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 by:** GDEVICE procedure, GDEVICE Parameters window
- **Default:** device-dependent
- **See:** “SYMBOL Statement” on page 412
Syntax

SYMBOLS='hardware-symbols-hex-string'X

Parameter Values

*hardware-symbols-hex-string*

is a hexadecimal string that is 16 characters long and must be completely filled. This table shows which bit position (left-to-right) within the hexadecimal string controls each hardware symbol.

**Table 25.4  Bit Positions That Control Hardware Symbols**

<table>
<thead>
<tr>
<th>Bit to Turn On</th>
<th>Symbol Name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PLUS</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>×</td>
</tr>
<tr>
<td>3</td>
<td>STAR</td>
<td>✴</td>
</tr>
<tr>
<td>4</td>
<td>SQUARE</td>
<td>□</td>
</tr>
<tr>
<td>5</td>
<td>DIAMOND</td>
<td>◊</td>
</tr>
<tr>
<td>6</td>
<td>TRIANGLE</td>
<td>△</td>
</tr>
<tr>
<td>7</td>
<td>HASH</td>
<td>№</td>
</tr>
<tr>
<td>8</td>
<td>Y</td>
<td>¥</td>
</tr>
<tr>
<td>9</td>
<td>Z</td>
<td>№</td>
</tr>
<tr>
<td>10</td>
<td>PAW</td>
<td>⋯</td>
</tr>
<tr>
<td>11</td>
<td>POINT</td>
<td>.</td>
</tr>
<tr>
<td>12</td>
<td>DOT</td>
<td>•</td>
</tr>
<tr>
<td>13</td>
<td>CIRCLE</td>
<td>○</td>
</tr>
</tbody>
</table>

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.

- **Used by:** GOPTIONS statement
- **Alias:** TARGET
- **Restriction:** This option is not supported by Java or ActiveX.

**Syntax**

```
TARGETDEVICE=target-device-entry
```

**Parameter Values**

`target-device-entry`

specifies the name of a device entry in a catalog.

**Details**

Use TARGETDEVICE= to specify a device driver when you want to do the following:

- preview graphics output on your monitor as it would appear on a different output device. For details, see “Previewing Output” on page 131.
- print output from the Graph window or the Graphics Editor window with the PRINT command. For details, see “Printing Your Graph” on page 131.
- 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 by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers
- **See:** TRAILERFILE on page 617

**Syntax**

```
TRAILER='command'
```
Parameter Values

`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 by:** GDEVICE procedure, GDEVICE Metagraphics window
- **Restriction:** Used only with user-supplied Metagraphics drivers
- **See:** TRAILER on page 616

Syntax

```
TRAILERFILE=fileref
```

Parameter Values

`fileref`

specifies a valid SAS fileref up to eight characters long. `Fileref` must have been previously assigned with a FILENAME statement or a host command before running the Metagraphics driver. See “FILENAME Statement” on page 47 for additional information about 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 by:** GOPTIONS statement
- **Default:** NOTRANSPARENCY
- **Restriction:** This option is supported by the following device drivers only:
  - the PNG device driver
  - the SVG device drivers
  - the GIF device drivers
  - the ACTIVEX and ACTXIMG device drivers only when the output is used in a Microsoft PowerPoint presentation
Syntax

TRANSPARENCY | NOTRANSPARENCY

Comparisons

When the image is displayed and TRANSPARENCY is in effect, the browser's background color replaces the driver's background color, causing the image to appear transparent.

Note: It is recommended that you set the background color of your GIF output to match the background color of the presentation in which you want to use the GIF image.

TRANTAB

Selects a translate table for your system that performs ASCII-to-EBCDIC translation.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Host File Options window
Default: host-dependent
Restriction: This option is not supported by Java or ActiveX.

Syntax

TRANTAB=table | user-defined-table

Parameter Values

table

specifies a translate table stored as a SAS/GRAPH catalog entry. Table can be one of the following:

• SASGTAB0 (default translate table for your operating environment)
• GTABVTAM
• GTABTCAM

user-defined-table

specifies the name of a user-created translate table.

Details

TRANTAB is set by the SAS Installation Representative and is needed when an EBCDIC host sends data to an ASCII graphics device. See the SAS/GRAPH installation instructions for details. You can also create your own translate table using the TRANTAB procedure. For a description of the TRANTAB Procedure, see Base SAS Procedures Guide.

TYPE

Specifies the type of output device to which graphics commands are sent.

Used by: GDEVICE procedure, GDEVICE Detail window
Default: device-dependent
Syntax
TYPE=CAMERA | CRT | EXPORT | PLOTTER | PRINTER

Parameter Values
CAMERA
specifies a film-recording device.

CRT
specifies a monitor or terminal.

EXPORT
identifies the list in which the device appears under SAS/ASSIST software. This is used for drivers that produce output to be exported to other software applications, such as CGM or HPGL.

PLOTTER
specifies a pen plotter.

PRINTER
specifies a printer

Details
You should not modify this value for device drivers supplied by SAS.

UCC
Sets the user-defined control characters for the device.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Parameters window

Restrictions: This option is device-dependent.
This option is not supported by Java or ActiveX.

Syntax
UCC='control-characters-hex-string'X

Parameter Values
control-characters-hex-string
is a hexadecimal string that can be up 32 bytes (64 characters) long. You need to specify only up to the last nonzero byte; the remaining bytes will be set to zero.

Details
Not all devices support this feature, and the meaning of each byte of the string varies from device to device.

Typically, the UCC byte position is indicated by a bracketed value. For example, UCC[2] refers to the second byte of the string. For assistance with determining UCC values for your specific device, please contact SAS Technical Support.
USERINPUT

Determines whether user input is enabled for the device.

Used by: GOPTIONS statement
Default: NOUSERINPUT
Restrictions: This option is supported by the GIF driver only.
This option is not supported by all browsers.

Syntax

USERINPUT | NOUSERINPUT

Parameter Values

USERINPUT
enables user input

NOUSERINPUT
disables user input

Details

When user input is enabled, processing of the animation is suspended until a carriage
return, mouse click, or some other application-dependent event occurs. The user input
feature works with the delay time setting so that processing continues when user input
occurs or the delay time has elapsed, whichever comes first.

VORIGIN

Sets the vertical offset from the lower left corner of the display area to the lower left corner of the graph.

Used by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window
Restriction: This option is not supported by Java or ActiveX.
See: HORIGIN on page 572

Syntax

VORIGIN=vertical-offset <IN | CM | PT>

Parameter Values

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 70 for details.

VPOS

Sets the number of rows in the graphics output area.

- **Used by:** GOPTIONS statement
- **Default:** device-dependent: the value of the LROWS or PROWS device parameter
- **Restriction:** This option is not supported by Java or ActiveX.
- **See:** HPOS on page 573, LROWS on page 582, PROWS on page 604

Syntax

VPOS=rows

Parameter Values

- **rows** specifies the number of rows in the graphics output area, which is equivalent to the number of hardware characters that can be displayed vertically. Specifying VPOS=0 causes the device driver to use the default hardware character cell height for the device.

Details

The VPOS= graphics option overrides the values of the LROWS or PROWS device parameters and temporarily sets the number of columns in the graphics output area. VPOS= does not affect the height of the graphics output area but merely divides it into rows. Therefore, you can use VPOS= to control cell height.

The values specified in the HPOS= and VPOS= graphics options determine the size of a character cell for the graphics output area and consequently the size of many graphics elements, such as hardware text. The larger the size of the HPOS= and VPOS= values, the smaller the size of each character cell.

See “Overview” on page 69 for more information.
**VSIZE**

Sets the vertical size of the graphics output area.

**Used by:**
- GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window

**Restriction:**
This option is partially supported by Java and ActiveX.

**See:**
- HSIZE on page 574
- YMAX on page 625

**Syntax**

\[ \text{VSIZE=} \text{vertical-size} \ \text{<IN | CM | PT>} \]

**Parameter Values**

\( \text{vertical-size} \ <\text{IN | CM | PT}> \)

specifies the height of the graphics output area; \( \text{vertical-size} \) 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
   \[ \text{YMAX} - \text{BOTTOMMARGIN} - \text{TOPMARGIN} \]
   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 by:**
- GOPTIONS statement

**Default:**
NOV6COMP

**Restrictions:**
This option is partially supported by Java and ActiveX.
This option is ignored unless OPTIONS NOGSTYLE is also specified.

**Syntax**

\[ \text{V6COMP} | \text{NOV6COMP} \]

**Parameter Values**

\( \text{V6COMP} \)
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 by: GOPTIONS statement, GDEVICE procedure, GDEVICE Detail window
Restriction: Ignored by default display drivers, universal printing drivers, Java, and ActiveX
See: HSIZE on page 574
      PAPERSIZE on page 587
      XPIXELS on page 624

Syntax
XMAX=width <IN | CM | PT>

Parameter Values
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:

\[ x\text{-resolution} = \frac{\text{XPIXELS}}{\text{XMAX}} \]

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:

\[
\begin{align*}
\text{XPIXELS}=1500 & \quad \text{and} \quad \text{XMAX}=5 \\
\text{XPIXELS}=1800 & \quad \text{and} \quad \text{XMAX}=6
\end{align*}
\]

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:

\[
\text{HSIZE} = \text{XMAX} - \text{margins}
\]

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 69 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 by:** GOPTIONS statement; GDEVICE procedure; GDEVICE Detail window
- **Default:** device-dependent
- **Restriction:** This option is partially supported by Java and ActiveX.
- **See:** XMAX on page 623

**Syntax**

\[
\text{XPIXELS}=\text{width-in-pixels}
\]
Parameter Values

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 about graph styles, refer to the TEMPLATE procedure documentation in SAS Output Delivery System: Procedures Guide.

The value of XPIXELS is used in calculating the resolution of the device:

\[ x\text{-resolution} = \frac{\text{XPIXELS}}{\text{XMAX}} \]

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:

\[ \text{HSIZE} = \text{XMAX} - \text{margins} \]

Note: margins are not device parameters, but represent the value of HORIGIN (the left margin) plus the right margin. The right margin is whatever is left over when you subtract HSIZE and HORIGIN from XMAX. The values of margins are always based on the original XMAX and HSIZE values that are stored in the device entry.

If HSIZE= is specified and its value is larger than XMAX, HSIZE= is ignored.

- If you use the GDEVICE procedure to permanently change the value of the XPIXELS device parameter in the device entry, SAS/GRAPH automatically recalculates the resolution of the device is using the value of XMAX device parameter.

- If you change the values of both XMAX= and XPIXELS=, SAS/GRAPH recalculates the resolution of the device using both of the specified values.

Note: When SAS/GRAPH recalculates the resolution, the resolution does not necessarily change. For example, both of these pairs of values produce the same resolution, 300dpi:

\[ \text{XPIXELS}=1500 \text{ and } \text{XMAX}=5 \]
\[ \text{XPIXELS}=1800 \text{ and } \text{XMAX}=6 \]

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.
YMAX=

Parameter Values

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.

See Also

“XMAX” on page 623

YPIXELS

Specifies the height of the addressable display area in pixels and in conjunction with YMAX determines the horizontal resolution for the device.

Syntax

YPIXELS=height-in-pixels

Parameter Values

height-in-pixels

is a positive integer up to eight digits long (0...99999999).
Details

*Note:* This option overrides the `OutputHeight` style attribute in the graph styles. For more information about graph styles, refer to the `TEMPLATE` procedure documentation in *SAS Output Delivery System: Procedures Guide.*

See Also

“`XPIXELS`” on page 624
Chapter 26
SAS System Options Used by SAS/GRAPH

Introduction to System Options

This chapter provides a detailed description of the system options used with SAS/GRAPH software. The descriptions provide the syntax, defaults, and related options for each option.

The system options are listed alphabetically.

System options are instructions that affect the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. Examples of items that are controlled by SAS system options include the appearance of SAS output, the handling of some files that are used by SAS, the use of system variables, and the processing of observations in SAS data sets.

For detailed information about using SAS System Options, see SAS System Options: Reference.

Dictionary

DEVICE= System Option

Specifies the device driver to which SAS/GRAPH sends procedure output.

Valid in: SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window
          SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Graphics: Driver Settings
**PROC OPTIONS**

GROUP= GRAPHICS

**Alias:** DEV=

**Requirement:** This option is required for the LISTING destination when you run SAS in a non-interactive mode.

**Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.


---

**Syntax**

DEVICE= device-driver-specification

**Syntax Description**

*device-driver-specification* specifies the name of a device driver.

**Details**

If you omit the device-driver name, the Output Delivery System selects a device driver for the opened destinations. If you specify a device that is not compatible with the opened destinations, the Output Delivery System selects a device that is valid. If you enter an invalid device driver, SAS prompts you to enter a device driver when you execute a procedure that produces graphics.

A best practice is to let the Output Delivery System select the device driver, except when you use the LISTING destination. When the LISTING destination is open, the default device is the Graph window.

**See Also**

Chapter 9, “Using Graphics Devices,” on page 79

---

**GSTYLE System Option**

Specifies whether ODS styles can be used to generate graphs that are stored as GRSEG catalog entries.

**Valid in:** SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

**Categories:** Graphics: Driver Settings

Log and Procedure Output Control: ODS Printing

**PROC OPTIONS**

GROUP= GRAPHICS

ODSPRINT

**Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.
Syntax

GSTYLE | NOGSTYLE

Syntax Description

GSTYLE

specifies that ODS styles can be used in the generation of graphs that are stored as GRSEG catalog entries. If no style is specified, the default style for the given output destination is used. This is the default.

NOGSTYLE

specifies to not use ODS styles in the generation of graphs that are stored as GRSEG catalog entries.

Tip
Use NOGSTYLE for compatibility of graphs generated prior to SAS 9.2.

Details

The GSTYLE system option affects only graphic output that is generated using GRSEGs. The GSTYLE option does not affect the use of ODS styles in graphs that are generated by the following means:

• Java device driver
• ActiveX device driver
• SAS/GRAPH statistical graphic procedures
• SAS/GRAPH template language
• ODS GRAPHICS ON statement

GWINDOW System Option

Specifies whether SAS displays SAS/GRAPH output in the GRAPH window.

Valid in:
SAS 9.4: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window
SAS Viya: Configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category:
Graphics: Driver Settings

PROC OPTIONS GROUP=

GRAPHICS

Note:
This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

GWINDOW | NOGWINDOW
Syntax Description

GWINDOW
  displays SAS/GRAPH software output in the GRAPH window, if your site licenses SAS/GRAPH software and if your personal computer has graphics capability.

NOGWINDOW
  displays graphics outside of the windowing environment.
## Part 6

### The Annotate Facility

**Chapter 27**

*Using Annotate Data Sets* .............................................. 635

**Chapter 28**

*Annotate Function Dictionary* ........................................ 667

**Chapter 29**

*Annotate Variable Dictionary* ........................................ 705

**Chapter 30**

*Annotate Macro Dictionary* ........................................... 749
# Chapter 27

Using Annotate Data Sets

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview: The Annotate Facility</strong></td>
<td>636</td>
</tr>
<tr>
<td>Overview of the Annotate Facility</td>
<td>636</td>
</tr>
<tr>
<td>Enhancing Existing Graphs</td>
<td>636</td>
</tr>
<tr>
<td>Creating Custom Graphs</td>
<td>637</td>
</tr>
<tr>
<td>Creating Annotate Graphics</td>
<td>638</td>
</tr>
<tr>
<td><strong>The Annotate Function, Variable, and Macro Dictionaries</strong></td>
<td>638</td>
</tr>
<tr>
<td>About the Annotate Data Set</td>
<td>638</td>
</tr>
<tr>
<td>Overview of Annotate Data Sets</td>
<td>638</td>
</tr>
<tr>
<td>Structure of an Annotate Data Set</td>
<td>639</td>
</tr>
<tr>
<td>Annotate Variables</td>
<td>640</td>
</tr>
<tr>
<td>Annotate Functions</td>
<td>643</td>
</tr>
<tr>
<td><strong>About Annotate Graphics</strong></td>
<td>645</td>
</tr>
<tr>
<td>Overview of Annotate Graphics</td>
<td>645</td>
</tr>
<tr>
<td>Graphics Elements</td>
<td>645</td>
</tr>
<tr>
<td>Coordinates</td>
<td>646</td>
</tr>
<tr>
<td>Coordinate Systems</td>
<td>646</td>
</tr>
<tr>
<td>Internal Coordinates</td>
<td>648</td>
</tr>
<tr>
<td>Attribute Variables</td>
<td>649</td>
</tr>
<tr>
<td><strong>Creating an Annotate Data Set</strong></td>
<td>650</td>
</tr>
<tr>
<td>Overview of Creating an Annotate Data Set</td>
<td>650</td>
</tr>
<tr>
<td>Using the DATA Step</td>
<td>650</td>
</tr>
<tr>
<td>Using Annotate Macros in the DATA Step</td>
<td>651</td>
</tr>
<tr>
<td>Effect of Missing Values</td>
<td>651</td>
</tr>
<tr>
<td><strong>Producing Graphics Output from Annotate Data Sets</strong></td>
<td>651</td>
</tr>
<tr>
<td>Overview of Producing Graphics Output</td>
<td>651</td>
</tr>
<tr>
<td>Including Annotate Graphics with Procedure Output</td>
<td>652</td>
</tr>
<tr>
<td>Producing Only Annotate Graphics Output</td>
<td>652</td>
</tr>
<tr>
<td>Using the Annotate Variables for Web Output</td>
<td>652</td>
</tr>
<tr>
<td><strong>Annotate Processing Details</strong></td>
<td>652</td>
</tr>
<tr>
<td>Order in Which Graphics Elements Are Drawn</td>
<td>652</td>
</tr>
<tr>
<td>Using BY-Group Processing with the Annotate Facility</td>
<td>653</td>
</tr>
<tr>
<td>Using the LIFO Stack</td>
<td>654</td>
</tr>
<tr>
<td>Debugging</td>
<td>654</td>
</tr>
<tr>
<td><strong>List of Annotate Examples</strong></td>
<td>655</td>
</tr>
<tr>
<td><strong>Annotate Error Messages</strong></td>
<td>656</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>660</td>
</tr>
</tbody>
</table>
Overview: The Annotate Facility

Overview of the Annotate Facility

The Annotate facility enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an Annotate data set. You can generate custom graphics by using an Annotate data set with GANNO (Chapter 32, “GANNO Procedure,” on page 779), or GSLIDE (Chapter 44, “GSLIDE Procedure,” on page 1333). You can enhance your graphics output by applying an Annotate data set to graphics that you generated using procedures such as GCHART (Chapter 35, “GCHART Procedure,” on page 872), GPLOT (Chapter 41, “GPLOT Procedure,” on page 1134), and GMAP (“GMAP Procedure” in SAS/GRAPH and Base SAS: Mapping Reference), among others.

In addition, SAS/GRAPH supports using annotation in the following procedures with the Java or ActiveX devices: GCHART, GCONTOUR, GMAP, GPLOT, GRADAR, and G3D.

Enhancing Existing Graphs

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

- label points on a map using map coordinates
- label bars on horizontal and vertical bar charts
- label points on a plot
- create animated text labels
- create a legend for a three-dimensional graph
This figure shows the GCHART procedure using the VBAR statement to create annotated vertical bars. The units ordered, the city, and type text are all annotated.

![Orders Received](image)

See “Example 1: Labeling Subgroups in a Vertical Bar Chart” on page 660 for the program that generated this bar chart.

**Creating Custom Graphs**

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

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

Figure 27.1 on page 637 is an example of a custom graph that uses Annotate commands to draw the graphics elements.

![Custom Graphics Using Only Annotate Commands](image)

The program that creates this output is in “Example 2: Drawing a Circle of Stars” on page 663.
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 638. Once you understand how the data set works, you can follow these three steps to create Annotate graphics:

1. Determine what you want to draw, and where (location) and how (coordinate system) you want to position it on the graphics output. (See “About Annotate Graphics” on page 645.)

2. Build an Annotate data set of graphics commands using the Annotate variables and functions. (See “Creating an Annotate Data Set” on page 650.)

3. Submit a SAS/GRAPH procedure to produce the graphics output. (See “Producing Graphics Output from Annotate Data Sets” on page 651.)

The Annotate Function, Variable, and Macro Dictionaries

You can find detailed information about how to use the Annotate facility functions, variables, and macros in the Annotate dictionaries. The following table lists the dictionaries that are available for the Annotate facility:

<table>
<thead>
<tr>
<th>Functions, Variables, or Macros</th>
<th>Dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>Chapter 29, “Annotate Variable Dictionary,” (p. 705)</td>
</tr>
<tr>
<td>Macros</td>
<td>Chapter 30, “Annotate Macro Dictionary,” (p. 749)</td>
</tr>
</tbody>
</table>

You can find information about error messages that might appear when you invoke an Annotate function or macro in “Annotate Error Messages” on page 656.

About the Annotate Data Set

Overview of Annotate Data Sets

In an Annotate data set, each observation represents a command to draw a graphics element or perform an action. The observations use a set of predefined annotate variables on page 706. Annotate functions on page 667 determine what is to be done with each observation. Annotate macros on page 750 simplify the process of drawing a graphics element. Annotate error messages on page 656 are sent to the SAS log. For
usage information and example programs, refer to “Using Annotate Macros” on page 750 and Chapter 27, “Using Annotate Data Sets,” on page 635.

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 650.

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 27.1 on page 639 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 650.)

<table>
<thead>
<tr>
<th>OBS</th>
<th>FUNCTION</th>
<th>X</th>
<th>Y</th>
<th>HSYS</th>
<th>XSYS</th>
<th>YSYS</th>
<th>STYLE</th>
<th>COLOR</th>
<th>POSITION</th>
<th>SIZE</th>
<th>LINE</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>label</td>
<td>20</td>
<td>85</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>swissb</td>
<td>green</td>
<td>6</td>
<td>6.0</td>
<td>.</td>
<td>Sample Annotate Graphics</td>
</tr>
<tr>
<td>2</td>
<td>move</td>
<td>28</td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>swissb</td>
<td>green</td>
<td>6</td>
<td>6.0</td>
<td>.</td>
<td>Sample Annotate Graphics</td>
</tr>
<tr>
<td>3</td>
<td>draw</td>
<td>68</td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>swissb</td>
<td>red</td>
<td>6</td>
<td>0.8</td>
<td>1</td>
<td>Sample Annotate Graphics</td>
</tr>
<tr>
<td>4</td>
<td>draw</td>
<td>48</td>
<td>70</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>swissb</td>
<td>red</td>
<td>6</td>
<td>0.8</td>
<td>1</td>
<td>Sample Annotate Graphics</td>
</tr>
<tr>
<td>5</td>
<td>draw</td>
<td>28</td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>swissb</td>
<td>red</td>
<td>6</td>
<td>0.8</td>
<td>1</td>
<td>Sample Annotate Graphics</td>
</tr>
</tbody>
</table>

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

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

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

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

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

Figure 27.2 on page 640 shows the green title and the red triangle produced by the TRIANGLE data set and displayed with the GANNO on page 779 procedure. Notes on the figure in black contain the X and Y coordinates of the graphics elements.

Figure 27.2  Annotate Output from the TRIANGLE Data Set

Annotate Variables

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

an action variable
tells what to do. The only action variable is FUNCTION, which specifies what graphics element to draw (graphics primitive) or what action to take (programming function).

positioning variables
tell where to do it. The positioning variables specify the point at which to draw the graphics element.

attribute variables
tell how to do it. The attribute variables specify the characteristics of the graphics element (for example, color, size, line style, text font).

There is also an HTML variable that is used to generate a drill-down graph that is viewed in a web browser.

When used with text labels and SVG devices, the HTML variable enables you to animate the text. For more information, see “LABEL Function” on page 681.

Table 27.1 on page 641 lists all Annotate variables, grouped by task, and briefly describes each one. See “About the Annotate Variables” on page 706 for a complete description of each variable.
### Table 27.1 Summary of Annotate Variables

<table>
<thead>
<tr>
<th>Task Group</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable that defines an action</td>
<td>FUNCTION</td>
<td>specifies a drawing or programming action; Table 27.2 on page 644 describes these actions.</td>
</tr>
<tr>
<td>Positioning variables that determine coordinate values</td>
<td>GROUP</td>
<td>uses the value of the GCHART GROUP= option in place of X or Y</td>
</tr>
<tr>
<td></td>
<td>MIDPOINT</td>
<td>uses the value of the GCHART MIDPOINT= option in place of X or Y</td>
</tr>
<tr>
<td></td>
<td>SUBGROUP</td>
<td>uses the value of the GCHART SUBGROUP= option in place of X or Y</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>specifies a numeric horizontal coordinate</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>specifies a numeric vertical coordinate</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>specifies a numeric third dimensional coordinate; used with G3D procedure only</td>
</tr>
<tr>
<td></td>
<td>XC</td>
<td>specifies a horizontal character coordinate; only used with data coordinate systems 1, 2, 7, 8</td>
</tr>
<tr>
<td></td>
<td>YC</td>
<td>specifies a vertical character coordinate; only used with data coordinate systems 1, 2, 7, 8</td>
</tr>
<tr>
<td>Positioning variables that contain internal coordinates</td>
<td>XLAST, YLAST</td>
<td>contain the X and Y coordinates of the last nontext function</td>
</tr>
<tr>
<td></td>
<td>XLSTT, YLSTT</td>
<td>contain the X and Y coordinates of the last text function</td>
</tr>
<tr>
<td>Positioning variables that specify coordinate systems</td>
<td>HSYS</td>
<td>specifies type of units for the SIZE variable</td>
</tr>
<tr>
<td></td>
<td>XSYS</td>
<td>specifies coordinate system for X or XC coordinates</td>
</tr>
<tr>
<td></td>
<td>YSYS</td>
<td>specifies coordinate system for Y or YC coordinates</td>
</tr>
<tr>
<td></td>
<td>ZSYS</td>
<td>specifies coordinate system for Z coordinate (G3D procedure only)</td>
</tr>
<tr>
<td>Attribute variables</td>
<td>ANGLE</td>
<td>angle of text label or starting angle of a pie slice</td>
</tr>
<tr>
<td>Task Group</td>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>CBORDER</td>
<td>colored border around text or symbol</td>
</tr>
<tr>
<td></td>
<td>CBOX</td>
<td>colored box behind text or symbol</td>
</tr>
<tr>
<td></td>
<td>COLOR</td>
<td>color of a graphics primitive</td>
</tr>
<tr>
<td></td>
<td>IMGPATH</td>
<td>path to an image file to be displayed.</td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td>line type to use in drawing or special control over pies and bars</td>
</tr>
<tr>
<td></td>
<td>POSITION</td>
<td>placement and alignment for text strings</td>
</tr>
<tr>
<td></td>
<td>ROTATE</td>
<td>angle at which to place individual characters in a text string or the delta angle (sweep) of a pie slice</td>
</tr>
<tr>
<td></td>
<td>SIZE</td>
<td>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)</td>
</tr>
<tr>
<td></td>
<td>STYLE</td>
<td>font or pattern for a graphics element, depends on the FUNCTION variable</td>
</tr>
<tr>
<td></td>
<td>TEXT</td>
<td>text to use in a label, symbol, or comment</td>
</tr>
<tr>
<td></td>
<td>WHEN</td>
<td>whether a graphics element is drawn before or after procedure graphics output</td>
</tr>
<tr>
<td>web variable</td>
<td>HTML</td>
<td>specifies link information for a drill-down graph for text labels, can be used to animate the text when using SVG devices</td>
</tr>
</tbody>
</table>

See Figure 27.3 on page 643 for a table that shows you which Annotate functions are used with which Annotate variables.
Figure 27.3  Annotate Variables Used with Annotate Functions

<table>
<thead>
<tr>
<th>Variables</th>
<th>ARROW</th>
<th>BAR</th>
<th>CNTL2TXT</th>
<th>COMMENT</th>
<th>DEBUG</th>
<th>DRAW</th>
<th>DRAW2TXT</th>
<th>FRAME</th>
<th>IMAGE</th>
<th>LABEL</th>
<th>MOVE</th>
<th>PIE</th>
<th>PECONTR</th>
<th>PIEAY</th>
<th>POINT</th>
<th>POLY</th>
<th>POLYCNTL</th>
<th>POP</th>
<th>PUSH</th>
<th>SWAP</th>
<th>SYMBOL</th>
<th>TXT2CNTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBORDER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTION</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW2TXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEBUG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW2TXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW2TXT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 645.

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 648, “Using the LIFO Stack” on page 654, and “Debugging” on page 654.

Table 27.2 on page 644 summarizes the tasks that are performed by the Annotate functions. See “About the Annotate Functions” on page 667 for a complete description of the FUNCTION variable and its values.
<table>
<thead>
<tr>
<th>Task Group</th>
<th>If you want to...</th>
<th>Use this function...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphics tasks</strong></td>
<td>begin to draw a polygon (starting point) and specify a fill color and pattern</td>
<td>POLY</td>
</tr>
<tr>
<td></td>
<td>continue drawing a polygon (additional vertex) and specify an outline color of the polygon</td>
<td>POLYCONT</td>
</tr>
<tr>
<td></td>
<td>draw an arrow from the current (X,Y) position (see MOVE and TXT2CNTL)</td>
<td>ARROW</td>
</tr>
<tr>
<td></td>
<td>draw a line from the current (X,Y) position (see MOVE and TXT2CNTL)</td>
<td>DRAW</td>
</tr>
<tr>
<td></td>
<td>draw a point</td>
<td>POINT</td>
</tr>
<tr>
<td></td>
<td>draw a rectangle from the current (X,Y) position (see MOVE and TXT2CNTL) and fill with a pattern</td>
<td>BAR</td>
</tr>
<tr>
<td></td>
<td>draw a symbol</td>
<td>SYMBOL</td>
</tr>
<tr>
<td></td>
<td>draw line from (XLAST, YLAST) coordinates to (XLSTT, YLSTT) coordinates</td>
<td>DRAW2TXT</td>
</tr>
<tr>
<td></td>
<td>draw pie slice, circle, or arc</td>
<td>PIE</td>
</tr>
<tr>
<td></td>
<td>draw text; animate the text</td>
<td>LABEL</td>
</tr>
<tr>
<td></td>
<td>move to the specified point (X,Y)</td>
<td>MOVE</td>
</tr>
<tr>
<td></td>
<td>put a frame around the area defined by XSYS and YSYS and fill with a pattern</td>
<td>FRAME</td>
</tr>
<tr>
<td><strong>Programming tasks</strong></td>
<td>insert a comment in the data set (no action); documentation aid</td>
<td>COMMENT</td>
</tr>
<tr>
<td></td>
<td>copy (XLAST, YLAST) coordinates to (XLSTT, YLSTT) coordinates</td>
<td>CNTL2TXT</td>
</tr>
<tr>
<td></td>
<td>copy (XLSTT, YLSTT) coordinates to (XLAST, YLAST) coordinates</td>
<td>TXT2CNTL</td>
</tr>
<tr>
<td></td>
<td>exchange LSTT and LAST coordinates</td>
<td>SWAP</td>
</tr>
<tr>
<td></td>
<td>get coordinates of a point on a pie slice outline</td>
<td>PIEXY</td>
</tr>
<tr>
<td></td>
<td>get values for LAST and LSTT coordinates from LIFO stack</td>
<td>POP</td>
</tr>
<tr>
<td></td>
<td>put current values of LAST and LSTT coordinates onto LIFO stack</td>
<td>PUSH</td>
</tr>
<tr>
<td>Task Group</td>
<td>If you want to...</td>
<td>Use this function...</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td>set pie radius and coordinates for center; does not draw a pie</td>
<td>PIECNTR</td>
</tr>
<tr>
<td></td>
<td>turn on trace of previous values and LIFO stack</td>
<td>DEBUG</td>
</tr>
</tbody>
</table>

See Figure 27.3 on page 643 for a table that shows you which Annotate functions work with which Annotate variables.

### About Annotate Graphics

#### Overview of Annotate Graphics

When you create Annotate graphics, you specify these things:
- what to draw (graphics elements)
- where to draw those elements (the coordinates of the position on the output)
- how to draw (characteristics of the element such as size or color).

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

#### Graphics Elements

In an Annotate data set, the FUNCTION variable determines the graphics element that is drawn.

The particular graphics elements that you can draw are shown in Figure 27.4 on page 645 along with the value of the FUNCTION variable or Annotate macro that draws them.

**Figure 27.4  Annotate Graphics Elements**

You can control the position of graphics elements in the following ways:
• explicitly, using coordinates that you supply.
• dependently, based on the location of features in the SAS/GGRAPH output. For example, when you use the GCHART procedure, you can label the parts of a subgrouped vertical bar chart by using the SUBGROUP variable in your Annotate data set. The Annotate facility enables you to label subgroups without having to specify the actual coordinates of the subgroup bar.
• dependently, based on values that are supplied from other data sets. For example, you can label the ending point of a plot line in the GPLOT procedure by extracting the value of the last point in the sorted input data set.

Coordinates

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

• X, Y, and sometimes Z are used for numeric coordinates.
• XC and YC are used for character coordinates.
• GROUP, MIDPOINT, and SUBGROUP can be used when you annotate output from procedures such as GCHART. Use these variables to specify coordinates for horizontal or vertical bar charts.

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

Coordinate Systems

Overview of Coordinate Systems

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

You also specify a coordinate system for the SIZE variable using the HSYS variable. HSYS takes the same types of values as XSYS, YSYS, and ZSYS.

The SIZE variable can be used to specify the following:

• size of a graphics element, such as the width of lines (for example, FRAME)
• radius of pie slices (for example, PIE, PIECNTR, and PIEXY)
• height of text (for example, LABEL and SYMBOL)

These are the important components of the Annotate coordinate systems:

• Area: Each coordinate system refers to one of three drawing areas: data area, procedure output area, and graphics output area. Coordinates are measured from a different origin for each area; they also have different limits. Figure 27.5 on page 647 shows the areas on the graphics output and the coordinate systems that use them.
Figure 27.5  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 648.
  - 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 27.3 on page 647 describes the coordinate system values for the XSYS, YSYS, ZSYS, and HSYS variables.

Table 27.3  Coordinate System Values for XSYS, YSYS, ZSYS, and HSYS Variables

<table>
<thead>
<tr>
<th>Type of Coordinates</th>
<th>Area</th>
<th>Units</th>
<th>Range</th>
<th>Value for XSYS, YSYS, ZSYS, HSYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute</td>
<td>data</td>
<td>%</td>
<td>0-100% of axis</td>
<td>'1' *</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td>values</td>
<td>minimum to maximum of axis</td>
<td>'2' *</td>
</tr>
<tr>
<td></td>
<td>graphics output area</td>
<td>%</td>
<td>0-100% of graphics output area</td>
<td>'3'</td>
</tr>
<tr>
<td></td>
<td>graphics output area</td>
<td>cells</td>
<td>0 to limit of graphics output area</td>
<td>'4'</td>
</tr>
<tr>
<td></td>
<td>procedure output area</td>
<td>%</td>
<td>0-100% of procedure output area</td>
<td>'5'</td>
</tr>
<tr>
<td></td>
<td>procedure output area</td>
<td>cells</td>
<td>0 to limit of procedure output area</td>
<td>'6'</td>
</tr>
</tbody>
</table>
### Ranges for Cells

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

**graphics output area**

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

**procedure output area**

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

See “Overview” on page 69 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

---

<table>
<thead>
<tr>
<th>Type of Coordinates</th>
<th>Area</th>
<th>Units</th>
<th>Range</th>
<th>Value for XSYS, YSYS, ZSYS, HSYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative</td>
<td>data</td>
<td>%</td>
<td>0-100% of axis</td>
<td>’7’ *</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td>values</td>
<td>minimum to maximum of axis</td>
<td>’8’ *</td>
</tr>
<tr>
<td></td>
<td>graphics output area</td>
<td>%</td>
<td>0-100% of graphics output area</td>
<td>’9’</td>
</tr>
<tr>
<td></td>
<td>graphics output area</td>
<td>cells</td>
<td>0 to limit of graphics output area</td>
<td>’A’</td>
</tr>
<tr>
<td></td>
<td>procedure output area</td>
<td>%</td>
<td>0-100% of procedure output area</td>
<td>’B’</td>
</tr>
<tr>
<td></td>
<td>procedure output area</td>
<td>cells</td>
<td>0 to limit of procedure output area</td>
<td>’C’</td>
</tr>
<tr>
<td>none</td>
<td>Text font point size</td>
<td>none</td>
<td>0 to limit of graphics output area</td>
<td>’D’**</td>
</tr>
</tbody>
</table>

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

**Coordinate system D is used only for text functions such as LABEL. For functions that do not create text, a warning appears in the log and the 4-coordinate system is used.
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:

\[ x\text{last}=50; \]

However, you can use several functions to directly manipulate the values of the internal coordinates. The functions are shown in Figure 27.6 on page 649.

**Figure 27.6 Programming Functions That Manipulate System Variables**

For a complete description, see “Annotate Internal Coordinates” on page 706.

**Attribute Variables**

Attribute variables control the appearance of the graphics elements. Each function uses only a subset of these variables. See Table 27.1 on page 641 for a list of attribute variables.

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

For a complete description of the attribute variables and the aspect of the graphics elements that they control, see “About the Annotate Variables” on page 706.
Creating an Annotate Data Set

Overview of Creating an Annotate Data Set

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

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

Most of the examples in this documentation use a DATA step with assignment statements. For more information about creating SAS data sets, see SAS Data Sets in 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 27.1 on page 639:

```sas
data triangle;
    /* declare variables */
    length function style color $ 8 text $ 25;
    retain hsys xsys ysys *3*;

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

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

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

Notice that a RETAIN statement sets the values of the HSYS, XSYS, and YSYS variables. RETAIN statements are useful when you want to select the values for variables that are required for many functions and the value is the same for all of them.
The SIZE, LINE, and COLOR variables are included with only the first DRAW function. Using this method to create the data set, the values set in the first DRAW function carry over to subsequent DRAW functions.

The PROC GANNO takes as input the annotate data set “triangle” created by the previous DATA step and creates the output shown in Figure 27.2 on page 640.

**Using Annotate Macros in the DATA Step**

A set of Annotate macros is provided in the SAS sample library.

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

You can use macro calls in a DATA step to create observations in an Annotate data set. You can also use Annotate macros and explicit variable assignments together in the same DATA step. For complete information, see “About the Annotate Macros” on page 750 and “Using Annotate Macros” on page 750.

**Effect of Missing Values**

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

Variables that have a missing value use a default value. For example, if the COLOR variable has a missing value, then the first color in the color list that is defined by the COLORS= graphics option is used. If the COLORS= option is not specified, then the device's default color list is used. If the FUNCTION variable has a missing value, LABEL is used. If the X variable is missing, the value of the XLSTT internal coordinate is used for text functions and the XLAST internal coordinate is used for nontext functions. See “About the Annotate Variables” on page 706 for the default value of each Annotate variable.

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

Sometimes missing values are required to produce the desired results. If you have calculated the coordinates of a point and have stored the values in (XLAST,YLAST) or (XLSTT,YLSTT), then you can force Annotate to use the internal coordinates. You do this by supplying missing values for the X and Y variables. See “Annotate Internal Coordinates” on page 706 for details about using the (XLAST,YLAST) and (XLSTT,YLSTT) internal coordinates.

---

**Producing Graphics Output from Annotate Data Sets**

**Overview of Producing Graphics Output**

You can display Annotate graphics in two ways:

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

Including Annotate Graphics with Procedure Output

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

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

Producing Only Annotate Graphics Output

To produce Annotate graphics without other procedure output, use the GANNO procedure or the GSLIDE procedure:

• The GANNO procedure produces graphics output consisting only of Annotate graphics. See Chapter 32, “GANNO Procedure,” on page 779 for information about displaying or storing Annotate graphics.

• The GSLIDE procedure can also produce graphics output consisting only of Annotate graphics. In addition, you can enhance the graphics output with TITLE, NOTE, and FOOTNOTE statements. See Chapter 44, “GSLIDE Procedure,” on page 1333 for details.

Using the Annotate Variables for Web Output

Most of the annotate variables can be used in programs that generate output for the web. For more information about the annotate functions and variables, see Chapter 28, “Annotate Function Dictionary,” on page 667 and Chapter 29, “Annotate Variable Dictionary,” on page 705. For information about using annotate data sets in web output, see “When to Apply Annotate Data Sets to Web Presentations” on page 201.

Annotate Processing Details

Order in Which Graphics Elements Are Drawn

Overview of Drawing Order

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

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

Controlling the Processing with the WHEN Variable

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

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

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

Within each set, graphics elements are drawn in the order in which they appear in the Annotate data set. The graphics elements overlay each other as appropriate (on devices that demonstrate overlay behavior). For details, see the description of the WHEN variable on “WHEN Variable” on page 736.

Order of Annotate Layers

When you annotate a graph, you create additional transparent layers that are interspersed with the existing layers (starting from the bottom):

1. the graph background (back wall)
2. the WHEN='B' layer for all annotations except the LABEL and SYMBOL functions
3. axis frame area (chart wall)
4. the WHEN='B' layer for the LABEL and SYMBOL functions
5. the graphical data (plot points, join lines, bars, pie slices, and so on)
6. axes (axis lines, tick values, tick marks, labels)
7. the WHEN='A' layer for all annotations except the LABEL and SYMBOL functions
8. the WHEN='A' layer for the LABEL and SYMBOL functions

Using BY-Group Processing with the Annotate Facility

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

• Both the input data set for the procedure and the Annotate data set must contain the same BY variable.
The BY variable must be defined as the same type (character or numeric) and length in both data sets.

If a label or format is associated with a BY variable in one data set, the same association must occur in the other data set.

Both data sets must be sorted by the BY variable.

The ANNOTATE= option must be specified in an action statement in the procedure. If you specify the ANNOTATE= option in the PROC statement, the Annotate graphics are used for all graphs that are generated by the procedure. Specifying ANNOTATE= in an action statement in the procedure generates Annotate graphics for unique values of the BY variable.

See “BY Statement” on page 370 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 27.7 on page 654 illustrates how PUSH and POP functions work together.

Figure 27.7 Using PUSH and POP to Store and Retrieve Coordinate Values

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 27.1 on page 639 provides complete information about the value that you specify for each variable in every observation.

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

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

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

  NOTE: ERROR DETECTED IN ANNOTATE= libref.dataset

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

  PROBLEM IN OBSERVATION number-message

  where message is the text of the error message.

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

  ERROR LIMIT REACHED IN ANNOTATE PROCESS

  20 TOTAL ERRORS

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

---

List of Annotate Examples

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

• “Using GfK GeoMarketing Map Data to Specify Response Levels in a Block Map” in SAS/GRAF and Base SAS: Mapping Reference

• “Example 1: Labeling Subgroups in a Vertical Bar Chart” on page 660

• “Example 2: Drawing a Circle of Stars” on page 663

Other examples that use Annotate data sets are as follows:

• “Example 1: Scaling Data-Dependent Output” on page 784 (and others in that chapter)

• “Mapping and Annotating Values from the GINSIDE Procedure” in SAS/GRAF and Base SAS: Mapping Reference

• “Example 2: Displaying Annotate Graphics” on page 1342

• “Projecting an Annotate Data Set” in SAS/GRAF and Base SAS: Mapping Reference

• Example Code 28.1 on page 685
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 might accompany the message for invalid coordinate system specification.
  User Action: Check for an invalid or misspecified coordinate system value, or x or y values outside displayed range.

A PERCENTAGE VALUE LIES OUTSIDE 0 TO 100 BOUNDARIES
  Explanation: The x or y value requested is negative or greater than 100%. This message is informational.
  User Action: Check requested value for accuracy.

ANNOTATE MIDPOINT DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
  Explanation: The MIDPOINT variable in the Annotate data set is character, and the GCHART midpoint is numeric or vice versa.
  User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.

ANNOTATE GROUP DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
  Explanation: The GROUP variable in the Annotate data set is character, and the GCHART group is numeric or vice versa.
  User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.
ANNOTATE SUBGROUP DATATYPE DOES NOT MATCH GCHART- INPUT WAS #
Explanation: The SUBGROUP variable in the Annotate data set is character, and the GCHART subgroup is numeric or vice versa.
User Action: Check for misspelling or wrong variable assignment, or check for quotation marks in the assignment statement.

BOTH OLD AND NEW VARIABLE NAMES ENCOUNTERED IN ANNOTATE= DATA SET
Explanation: Variables named both MIDPOINT and MIDPNT or SUBGROUP and SUBGRP occur in the Annotate data set.
User Action: Determine which variable has the proper values for the Annotate data set and either delete the other variable or rename MIDPNT to MIDPOINT and SUBGRP to SUBGROUP.

CALCULATED COORDINATES LIE COMPLETELY OFF THE VISIBLE AREA
Explanation: Both the x and y coordinates supplied are outside the visible display area.
User Action: Check for improper or inappropriate coordinate system specification or coordinates out of range.

CANNOT HAVE MISSING GROUP VALUE IF GROUPS ARE PRESENT
Explanation: The GROUP variable in the Annotate data set contains a missing value.
User Action: If the GROUP= option is specified in the GCHART procedure, the Annotate GROUP variable cannot contain missing values. Remove the missing value from the request. Check reference system for data-dependent request.

CANNOT HAVE SUBGROUP AND X/Y MISSING IN GCHART STREAM
Explanation: Data coordinate system was requested and the X, Y, and SUBGROUP variables contain missing values.
User Action: The X, Y, or SUBGROUP variable must have a value if a data coordinate system is requested. Check stream for improper request.

CANNOT OMIT GROUP VARIABLE IF GCHART GROUPS ARE PRESENT
Explanation: You used a data coordinate system and specified GROUP= in the GCHART procedure, but the Annotate data set does not contain the GROUP variable.
User Action: Supply the GROUP variable in the Annotate data set.

CHARACTER VALUE SHOWN IS NOT ON THE HORIZONTAL AXIS
Explanation: The specified value of the XC variable is not on the x axis of the graph or chart. The observation is ignored.
User Action: Check for misspelling, for uppercase or lowercase conflict, or for exclusion in an axis specification.

CHARACTER VALUE SHOWN IS NOT ON THE VERTICAL AXIS
Explanation: The specified value of the YC variable does not occur on the y axis of the graph or chart. The observation is ignored.
User Action: Check for misspelling, for uppercase or lowercase conflict, or for exclusion in an axis specification.

CONFLICT BETWEEN PROCEDURE AXIS TYPE AND ANNOTATE DATA TYPE
Explanation: The axis type is character and the x and y coordinates are numeric or vice versa.
User Action: Check values for proper type matching.
DATA SYSTEM NOT SUPPORTED FOR THIS STATEMENT
Explanation: The data coordinate systems 1, 2, 7, 8 are not permitted for this statement.
User Action: Choose a different reference system for this observation.

DATA SYSTEM REQUESTED, BUT POINT IS NOT ON GRAPH
Explanation: The coordinate specified is not on displayed graph, and data coordinate system placement has been requested.
User Action: Check for improper specification of data value or graph axis parameters, or incorrect system specification. If this occurs, you might be able to use percent of the data area to position Annotate graphics.

G3D DATA SYSTEM REQUESTED, ALL SYSTEMS NOT DATA DEPENDENT
Explanation: Not all requested XSYS, YSYS, and ZSYS variable values are data values.
User Action: If one variable in G3D annotation is data-dependent, all variables must be data-dependent. Either specify all points in the data coordinate system or use another reference system value.

G3D DATA SYSTEM REQUESTED, VARIABLE CONTAINED MISSING VALUE
Explanation: The X, Y, or Z variable contained a missing value.
User Action: All values in G3D data placement requests must be specified. Remove the missing value from the request.

INTERNAL SYSTEM STACK OVERFLOW- TOO MANY PUSH FUNCTIONS
Explanation: The limit of stack positions has been exhausted. The maximum number of stack positions is system-dependent. Each PUSH operation uses one position; each POP frees one position for reuse.
User Action: Rewrite the program section to decrease the number of values stored in the stack.

INTERNAL SYSTEM STACK UNDERFLOW- TOO MANY POP FUNCTIONS
Explanation: The POP function has been issued with no values in the LIFO stack.
User Action: Check for unequal numbers of PUSH versus POP functions. They can be unequal, but you cannot move more values with the POP function than are stored with the PUSH function. At least one PUSH must occur before a POP can be issued.

LABEL FUNCTION REQUESTED, BUT TEXT VARIABLE NOT ON DATA SET
Explanation: A TEXT variable has not been found for the LABEL function.
User Action: If FUNCTION='LABEL', the TEXT variable must contain the string to be placed in the display area. Check for misspelling of variable name or specification of the wrong Annotate data set.

LINE VALUE SPECIFIED IS NOT WITHIN LIMITS- 0<=L<=3
Explanation: An invalid special line value has been specified.
User Action: The LINE value specified was not acceptable for FUNCTION='BAR' or the RECT macro. Check function for definition of line values or previous value used in DATA step prior to this observation.

LINE VALUE SPECIFIED IS NOT WITHIN LIMITS- 1<=L<=46
Explanation: The LINE value specified is not in the range 1 through 46.
User Action: Check for improper specification of data value. Line styles represented by the LINE values can be found in the "Specifying Line Types" on page 442.
MINIMUM VARIABLES NOT MET--AMBIGUITY PREVENTS SELECTION.
Explanation: The combinations of available X, Y, XC, YC, GROUP, MIDPOINT, and SUBGROUP variables do not identify the data-dependent values uniquely.

User Action: Check variable requirements and respecify.

MINIMUM VARIABLES NOT MET- MUST HAVE X/XC,Y/YC IN DATA SET
Explanation: The X, XC, Y, or YC variables have not been found in the Annotate data set.

User Action: The X or XC and Y or YC variables must be in the data set. This message represents a minimum validity check of the supplied Annotate data set.

POLYCONT ENCOUNTERED BEFORE POLY
Explanation: The POLYCONT function was encountered with no POLY function specification.

User Action: Probable sequencing error. Check for missing POLY command, improper ordering of polygon points, or interruption of POLY type commands by other valued functions. Also, check the value of WHEN for a mismatch.

"POLYCONT" INTERRUPTED
Explanation: A POLYCONT definition has been interrupted and resumed in the Annotate data set. This usually accompanies the error message POLYCONT ENCOUNTERED BEFORE POLY

User Action: Check data stream for proper order.

POSITION VALUE INVALID- MUST BE ONE OF “0123...9ABCDEF”
Explanation: The value of the POSITION variable is not in range '0' through '9' or 'A' through 'F' or '<', '>', or ' ' in a LABEL command.

User Action: Check desired value in POSITION description and correct.

REQUESTED POLYGON CONTAINS TOO MANY VERTICES (OBSERVATIONS)
Explanation: The maximum allocation for polygon points is exhausted. The maximum number of vertices is limited by a device's memory.

User Action: Define polygon with fewer points or break polygon into sections.

SYSTEM VALUE INVALID- MUST BE ONE OF “0123...9ABC”
Explanation: The value supplied for the XSYS, YSYS, or HSYS variable is not valid.

User Action: Check the desired value and correct the data set.

TEXT STRING EXTENDS BEYOND BOUNDARY OF SYSTEM DEFINED
Explanation: The text string is too long.

User Action: Check for excessive SIZE value or shorten the string. This error could be caused by HSYS='4' and a small value of the VPOS graphics option.

USE THE XC VARIABLE FOR DATA VALUES WHEN TYPE IS CHARACTER
Explanation: The X variable is character type in the Annotate data set when it should be numeric.

User Action: If character data are being plotted, use the XC variable to specify any data-related points pertaining to character values. If data are not character, omit quotation marks in X data value assignment.

USE THE YC VARIABLE FOR DATA VALUES WHEN TYPE IS CHARACTER
Explanation: The Y variable is character type in the Annotate data set when it should be numeric.
User Action: If character data are being plotted, use the YC variable to specify any data-related points pertaining to character values. If data are not character, omit quotation marks in Y data value assignment.

VALUE SHOWN IS NOT A VALID FONT OR PATTERN TYPE
Explanation: The value of the STYLE variable is not a valid font or pattern.
User Action: Check the value supplied for misspelling, truncation, and support in the FUNCTION description.

VALUE SHOWN IS NOT A VALID FUNCTION
Explanation: The value in the FUNCTION variable is not recognized as an available function.
User Action: Check for misspellings or truncation of value. Truncation can be corrected by specifying a length of 8 bytes in the LENGTH statement in the DATA step that generates the data set.

VALUE SHOWN IS NOT A VALID SIZE FACTOR
Explanation: The SIZE value of the variable is negative or excessive.
User Action: Check request or calculation for positive value result.

VARIABLE SHOWN HAS IMPROPER LENGTH IN ANNOTATE= DATA SET
Explanation: The length is incorrect for variable indicated. Either the length of the character string exceeds the length for the variable specified in a LENGTH statement, or the variable was not specified in a LENGTH statement.
User Action: Make sure the variable length is defined in a length statement and that the length specified adequately covers the length of the character strings that are used.

VARIABLE SHOWN IS NOT OF THE PROPER DATA TYPE
Explanation: The data type does not match required type for variable listed. Either variable type is character where a numeric is required, or numeric where a character is required.
User Action: Specify proper type for variable as described in “About the Annotate Variables” on page 706.

Examples

Example 1: Labeling Subgroups in a Vertical Bar Chart

<table>
<thead>
<tr>
<th>Features:</th>
<th>Annotate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LABEL</td>
</tr>
<tr>
<td></td>
<td>Annotate Variables</td>
</tr>
<tr>
<td></td>
<td>MIDPOINT</td>
</tr>
<tr>
<td></td>
<td>POSITION</td>
</tr>
<tr>
<td></td>
<td>SUBGROUP</td>
</tr>
</tbody>
</table>

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

See Chapter 35, “GCHART Procedure,” on page 872 for more information about creating bar charts.

Output 27.2 Bar Chart with Annotated Labels on Subgroups

Program

```plaintext
goptions reset=all border;

data sold;
   length type $ 10;
   input city $ units type $ ;
   datalines;
   Atlanta  99 Printers
   Atlanta 105 Plotters
   Atlanta  85 Terminals
   Paris   182 Printers
   Paris   150 Plotters
   Paris   157 Terminals
   Sydney  111 Printers
   Sydney  136 Plotters
   Sydney  100 Terminals
; run;
```
Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set SOLD.

data sold;
  length type $ 10;
  input city $ units type $ ;
  datalines;
Atlanta  99 Printers
Atlanta 105 Plotters
Atlanta 85 Terminals
Paris  182 Printers
Paris  150 Plotters
Paris  157 Terminals
Sydney 111 Printers
Sydney 136 Plotters
Sydney 100 Terminals
;run;
data barlabel;
  length color style $ 8;
  retain color "white" when "a" style "arial"
    xsys ysys "2" position "E" size 4 hsys "3";
  set sold;
  midpoint=city;
  subgroup=type;
  text=left(put(units,5.));
run;
title "Orders Received";
footnote j=r "GANVBAR";
axis1 label=none major=none minor=none style=0
  value=none;
axis2 label=none;
proc gchart data=sold;
  vbar city / type=sum
    sumvar=units
    subgroup=type
    width=17
    raxis=axis1
    maxis=axis2
    annotate=barlabel;
run;
quit;
Define the title and footnote.

```
title "Orders Received";
footnote j=r "GANVBAR";
```

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

```
axis1 label=none major=none minor=none style=0
    value=none;
axis2 label=none;
```

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

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

---

**Example 2: Drawing a Circle of Stars**

**Features:** Annotate Functions
- BAR
- CNTL2TXT
- FRAME
- LABEL
- MOVE
- PIECNTR
- PIEXY
- SYMBOL

Annotate Variables
- COLOR
- HSYS, XSYS, YSY
- LINE
- STYLE
- TEXT
- X and Y
- XLAST and YLAST
- XLSTT and YLSTT

**Sample library member:** GANCIRCL

**Note:** The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
This example shows how to use an Annotate data set to draw a flag that consists of a rectangle and four stars. The stars are positioned by placing them on an imaginary circle. The program uses the PIECNTR and PIEXY functions to find the points on the circle and the CNTL2TXT programming function to transfer coordinate values. It also processes Annotate assignment statements in a DO loop. The GANNO procedure displays the Annotate graphics.

**Output 27.3  Stars Positioned in a Circle with GANNO**

```
Program

goptions reset=all border;

data flag;
    length function style color $ 8 text $ 30;
    retain xsys ysys hsys "3";
    function="frame"; output;
    function="label"; x=50; y=90; text="Flag of Micronesia";
        style=""; size=6; output;
    function="move"; x=20; y=30; output;
    function="bar"; x=80; y=80; color="blue";
        line=3; style="solid"; output;
    do star_ang=0 to 270 by 90;
        function="piecntr"; x=50; y=55; size=15; output;
        function="piexy"; size=1; angle=star_ang; output;
        function="cntl2txt"; output;
        function="symbol"; style="marker"; text="V";
            angle=0; color="white"; size=10; x=.; y=.;
            output;
    end;
run;
```
Program Description

Set the graphics environment.

goptions reset=all border;

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

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

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

    function="frame"; output;

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

    function="label"; x=50; y=90; text="Flag of Micronesia";
    style=""; size=6; output;

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

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

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

    do star_ang=0 to 270 by 90;
    function="piecntr"; x=50; y=55; size=15; output;
    function="piexy"; size=1; angle=star_ang; output;

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="cntl2txt"; output;
    function="symbol"; style="marker"; text="V";

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

```sas
proc ganno annotate=flag;
run;
quit;
```
# Chapter 28

## Annotate Function Dictionary

### About the Annotate Functions

In an Annotate data set, the value of the FUNCTION variable specifies what action the observation performs. Annotate functions act in conjunction with Annotate variables that determine where and how to perform the action. Many of these variables are function-dependent, that is, what they do depends on the function that they are used with. For example, with the LABEL function the STYLE variable specifies a font; with the BAR function, STYLE specifies a pattern.

This section describes all of the values of the FUNCTION variable. For each function it

- describes the function's action.

### Dictionary

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARROW Function</td>
<td>668</td>
</tr>
<tr>
<td>BAR Function</td>
<td>671</td>
</tr>
<tr>
<td>CNTL2TXT Function</td>
<td>673</td>
</tr>
<tr>
<td>COMMENT Function</td>
<td>675</td>
</tr>
<tr>
<td>DEBUG Function</td>
<td>675</td>
</tr>
<tr>
<td>DRAW Function</td>
<td>675</td>
</tr>
<tr>
<td>DRAW2TXT Function</td>
<td>677</td>
</tr>
<tr>
<td>FRAME Function</td>
<td>678</td>
</tr>
<tr>
<td>IMAGE Function</td>
<td>679</td>
</tr>
<tr>
<td>LABEL Function</td>
<td>681</td>
</tr>
<tr>
<td>MOVE Function</td>
<td>688</td>
</tr>
<tr>
<td>PIE Function</td>
<td>689</td>
</tr>
<tr>
<td>PIECNTR Function</td>
<td>692</td>
</tr>
<tr>
<td>PIEXY Function</td>
<td>693</td>
</tr>
<tr>
<td>POINT Function</td>
<td>695</td>
</tr>
<tr>
<td>POLY Function</td>
<td>696</td>
</tr>
<tr>
<td>POLYCONT Function</td>
<td>698</td>
</tr>
<tr>
<td>POP Function</td>
<td>700</td>
</tr>
<tr>
<td>PUSH Function</td>
<td>701</td>
</tr>
<tr>
<td>SWAP Function</td>
<td>701</td>
</tr>
<tr>
<td>SYMBOL Function</td>
<td>701</td>
</tr>
<tr>
<td>TXT2CNTL Function</td>
<td>704</td>
</tr>
</tbody>
</table>
• notes whether the function updates the internal coordinate variables XLAST, YLAST and XLSTT, YLSTT.
• describes how other Annotate variables behave with the function. For a complete description of each variable, see “About the Annotate Variables” on page 706

For a summary of drawing and programming tasks performed by the FUNCTION variable, see Table 27.2 on page 644.

The variables that are available for use with each function are listed in Figure 27.3 on page 643.

---

**Dictionary**

**ARROW Function**

Draws an arrow in the graphics output from the (XLAST, YLAST) coordinates to the (X,Y) coordinates specified in the function.

**Updates:** XLAST, YLAST

**Tip:** For best results, specify a graphics device driver in the GOPTIONS statement.

**Syntax**

FUNCTION='ARROW';

**Associated Variables**

**ANGLE=angle-value**

specifies the angle for the tip of the arrowhead. You can specify any number for the angle. If the angle that you specify is not between 0 and 180, the absolute value of \( \text{mod}(\text{angle-value}, 180) \) is used. For example, the values -45, 45, and 225 all produce the same result.

Default 30

See “ANGLE Variable” on page 706

**COLOR='color'**

specifies the color of the arrow that is being drawn. Color can be any SAS/GRAPH color name.

See “COLOR Variable” on page 709

**GROUP=group-value**

**MIDPOINT=midpoint-value**

**SUBGROUP=subgroup-value**

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711
HSYS='coordinate-system'
specifies the coordinate system for the SIZE variable.
See “HSYS Variable” on page 714

HTML='link-string'
specifies the text that defines the drill-down link. The drill-down information is applied to the arrowhead.
See “HTML Variable” on page 715

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.

Figure 28.1 Sample Filled Arrowhead Sizes Used with the SIZE Variable

SIZE=line-thickness
specifies the thickness of the line that is being drawn. The units depend on the value of the HSYS variable. For example, if HSYS='3', the SIZE variable is in units of percent of the graphics output area. If HSYS='4', the SIZE variable is in units of cells of the graphics output area.

Figure 28.2 Sample Line Thicknesses Used with the SIZE Variable

STYLE= 'CLOSED' | 'FILLED' | 'OPEN'
specifies the type of arrowhead.
**Figure 28.3  Arrowhead Styles CLOSED, FILLED, and OPEN**

![Arrowhead Styles Diagram]

**Default** OPEN

See “STYLE Variable (Arrows)” on page 731

**WHEN=’B’ | ’A’**

specifies when to draw the line in relation to other procedure output.

See “WHEN Variable” on page 736

**X=horizontal-coordinate**

**Y=vertical-coordinate**

**Z=depth-coordinate**

**XC=’character-type-horizontal-coordinate’**

**YC=’character-type-vertical-coordinate’**

specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

**Restriction** The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738

“Y Variable” on page 744

“Z Variable” on page 747

“XC Variable” on page 739

“YC Variable” on page 745

**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 741

**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 746

**ZSYS=’coordinate-system’**

specifies the coordinate system for the Z variable.

**Restriction** The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747
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.

See “COLOR Variable” on page 709

**GROUP=group-value**

**MIDPOINT=midpoint-value**

**SUBGROUP=subgroup-value**

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711

“MIDPOINT Variable” on page 722

“SUBGROUP Variable” on page 733

---

**HTML='link-string'
**

specifies the text that defines the drill-down link.

See “HTML Variable” on page 715

---

**LINE=0...3**

specifies the direction in which to adjust the outline of the bar. Use LINE values 1 and 2 to offset a particular bar from an axis or adjoining area.

Default 1

See “LINE Variable” on page 721

---

**SIZE=line-thickness**

specifies a line thickness for the rectangle
STYLE='SOLID' | 'EMPTY' | style<density>
specifies the pattern that fills the bar.

Default EMPTY

Restriction Java and ActiveX support only SOLID and EMPTY.

WHEN='B' | 'A'
specifies when to draw the bar in relation to other procedure output.

X=horizontal-coordinate
Y=vertical-coordinate
Z=depth-coordinate
XC='character-type-horizontal-coordinate'
YC='character-type-vertical-coordinate'
define the upper right corner of a bar (rectangle) whose lower left corner is (XLAST,YLAST). Figure 28.5 on page 673 illustrates the use of these coordinates.

Restriction The Z= variable is used with the G3D procedure only.

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 741

YSYS='coordinate-system'
specifies the coordinate system for Y or YC variable. The YC variable can be used with YSYS='2' only.

See “YSYS Variable” on page 746

ZSYS='coordinate-system'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.
Details

Figure 28.5 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 28.5  Points Used to Construct a Bar

CNTL2TXT Function

Copies the values of the internal coordinates stored in the variable pairs (XLAST, YLAST) to (XLSTT, YLSTT).

**Updates:**  XLSST, YLSST

**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 28.8 on page 674.

First, use the PIE function to draw the pie slice:

data pie label;
   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 28.6 on page 674.

/* find a point that is half of the arc (rotate*.5) */
/* and is 4 units beyond the radius (size=1.1) */
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 28.7 on page 674 shows the results.

Now you can use the LABEL function to write the label as shown in Figure 28.8 on page 674. 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.

```sas
/* 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 28.8  Labeled Pie Slice
COMMENT Function

Inserts comments within the Annotate data set. The observations generated by the COMMENT function are ignored when the data set is processed.

Syntax

FUNCTION='COMMENT';

Associated Variables

TEXT='text-string'

specifies the comment to write to the data set.

See “TEXT Variable” on page 735

DEBUG Function

Writes the values of internal coordinates and Annotate variables to the SAS log before and after processing the next command (unless it is DEBUG) in the Annotate DATA step.

Syntax

FUNCTION='DEBUG';

DRAW Function

Draws a line in the graphics output from the (XLAST, YLAST) coordinates to the (X, Y) coordinates specified in the function.

Updates: XLAST, YLAST

Syntax

FUNCTION='DRAW';

Associated Variables

COLOR='color'

specifies the color of the line that is being drawn. Color can be any SAS/GRAPH color name.

See “COLOR Variable” on page 709
GROUP=group-value
MIDPOINT=midpoint-value
SUBGROUP=subgroup-value

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use
these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711
“MIDPOINT Variable” on page 722
“SUBGROUP Variable” on page 733

HSYS='coordinate-system'
specifies the coordinate system for the SIZE variable.

See “HSYS Variable” on page 714

LINE=1...46
specifies the line type of the line that is being drawn.

See “LINE Variable” on page 721
“Specifying Line Types” on page 442 for an illustration of the line types.

SIZE=line-thickness
specifies the thickness of the line that is being drawn. The units depend on the value
of the HSYS variable. For example, if HSYS='3', the SIZE variable is in units of
percent of the graphics output area. If HSYS='4', the SIZE variable is in units of cells
of the graphics output area.

Figure 28.9  Sample Line Thicknesses Used with the SIZE Variable

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

See “SIZE Variable” on page 728

WHEN='B' | 'A'
specifies when to draw the line in relation to other procedure output.

See “WHEN Variable” on page 736

X=horizontal-coordinate
Y=vertical-coordinate
Z=depth-coordinate
XC='character-type-horizontal-coordinate'
YC='character-type-vertical-coordinate'
specify the endpoint of a line drawn from (XLAST, YLAST) to (X,Y).

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
“Y Variable” on page 744
“Z Variable” on page 747
### 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 741

### 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 746

### ZSYS='coordinate-system'

specifies the coordinate system for the Z variable.

**Restriction**
The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

---

**DRAW2TXT Function**

Draws a line from (XLAST, YLAST) to (XLSTT, YLSTT) without updating any of those variables.

**Syntax**

FUNCTION='DRAW2TXT';

**Associated Variables**

**COLOR='color'**

specifies the line color. Color can be any SAS/GRAPH color name.

See “COLOR Variable” on page 709

**HSYS='coordinate-system'**

specifies the coordinate system for the SIZE variable.

See “HSYS Variable” on page 714

**LINE=1...46**

specifies the line type of the line that is being drawn.

See “LINE Variable” on page 721

“Specifying Line Types” on page 442 for an illustration of the line types.

**SIZE=line-thickness**

specifies the thickness of the line that is being drawn.
Figure 28.10  Sample Line Thicknesses Used with the SIZE Variable

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

See “LINE Variable” on page 721

“DRAW Function” on page 675 for details

WHEN='B' | 'A'
specifies when to draw the line in relation to generation of the procedure output.

See “WHEN Variable” on page 736

Details

DRAW2TXT is useful for underlining text.

DRAW2TXT does not update the (XLAST, YLAST) or (XLASTT, YLASTT) coordinates; neither can it interrupt a POLYCONT sequence.

FRAME Function

Draws a border around the portion of the display area defined by the XSYS and YSYS variables. Can specify a background color for the framed area.

Note: The FRAME function is not supported by Java.

Syntax

FUNCTION='FRAME';

Associated Variables

COLOR='color'
specifies the frame color and, if the STYLE variable is specified, fills the interior of the frame. Color can be any SAS/GRAPH color name.

See “COLOR Variable” on page 709

HSYS='coordinate-system'
specifies the coordinate system for the SIZE variable.

See “HSYS Variable” on page 714

HTML='link-string'
specifies the text that defines the drill-down link.

See “HTML Variable” on page 715

LINE=1...46
specifies the line type with which to draw the frame.

See “LINE Variable” on page 721
“Specifying Line Types” on page 442 for an illustration of the line types

SIZE=\textit{line-thickness}

specifies the thickness of the line with which to draw the frame.

\textit{Figure 28.11} \textit{Sample Line Thicknesses Used with the SIZE Variable}

\begin{tabular}{ccc}
1 & 2 & 3 \\
\end{tabular}

See “SIZE Variable” on page 728

“DRAW Function” on page 675 for details.

\textbf{STYLE=}'SOLID' | 'EMPTY' \textit{style\textless;density\textgreater;}

specifies the pattern that fills the area that is bounded by the frame.

Default \texttt{EMPTY}

Restriction Java and ActiveX support only SOLID and EMPTY.

See “STYLE Variable (Patterns)” on page 731

the discussion of fill patterns for bars and blocks in the documentation for the VALUE= option for the “PATTERN Statement” on page 398.

\textbf{WHEN=}'B' | 'A'

specifies when to draw the frame in relation to other procedure output.

See “WHEN Variable” on page 736

\textbf{XSYS=}'coordinate-system'
\textbf{YSYS=}'coordinate-system'

define the area to be enclosed by the frame.

See “XSYS Variable” on page 741

“YSYS Variable” on page 746

\textbf{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 526 for more information about CBACK=.

\textbf{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.

\textbf{Updates:} XLAST, YLAST
Syntax

FUNCTION='IMAGE';

Associated Variables

**HTML='link-string'**

specifies the text that defines the drill-down link.

See “HTML Variable” on page 715

**IMGPAT H='external-file'**

specifies the image file to be displayed in the graphics output. The syntax of external file specifications varies across operating environments.

Note Copying and pasting the image works only if an absolute path is specified instead of a relative path, or if the file into which the image is being pasted is opened from the directory to which the image is relative.

See “IMGPATH Variable” on page 720

**STYLE = 'TILE' | 'FIT' | 'SINGLE';**

specifies how the image is to be applied to fill the specified area of the graphics output.

- **TILE**
  replicates the image to fill the area.

- **FIT**
  stretches a single instance of the image to fill the area.

- **SINGLE**
  centers a single instance of the image on the specified coordinates. Any part of the image that extends beyond the data area is clipped. SINGLE is valid starting with SAS 9.4M5.

Default **TILE**

See “STYLE Variable (Images)” on page 730

**X=horizontal-coordinate**

**Y=vertical-coordinate**

**Z=depth-coordinate**

specifies the coordinates that determine the size of the image displayed in the graphics output.

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738

“Y Variable” on page 744

“Z Variable” on page 747

**ZSYS='coordinate-system'**

specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.
The following example shows how the `IMAGE` function adds a single stretched instance of an image to the graphics output. The image begins at the current coordinates and ends at the specified coordinates.

```
x=10; y=5; function="move"; output;
x=35; y=15; imgpath="/images/gifs/picture.gif"; style="fit";
function="image"; output;
```

For a list of the file types that you can use, see “Image File Types Supported by SAS/GRA" on page 331.

**LABEL Function**

Places text in the graphics output. Associated variables can control the color, size, font, base angle, and rotation of the characters displayed. You can also animate labels using the `HTML=` variable. When you specify the `HTML=` variable, you include the animation action and the parameters that control the animation. The animated label feature is available only when you use one of the SVG device drivers, except `SVGANIM`.

**Updates:** XLISTT, YLISTT

**Syntax**

```
FUNCTION='LABEL';
```

**Associated Variables**

**ANGLE=0...360**

specifies the baseline angle of the character string with respect to the horizontal. The pivot point is at `(X, Y)`, and the rotation is in a counterclockwise direction.

See “ANGLE Variable” on page 706

**CBORDER='color' | 'CTEXT'**

draws a colored border around the text. `Color` can be any SAS/GRA color name.

See “CBORDER Variable” on page 707

**CBOX='color' | 'CBACK'**

draws a solid, colored box behind the text. `Color` can be any SAS/GRA color name.

See “CBOX Variable” on page 708

**COLOR='color'**

specifies the color of the text. `Color` can be any SAS/GRA color name.

See “COLOR Variable” on page 709
GROUP=group-value
MIDPOINT=midpoint-value
SUBGROUP=subgroup-value

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711
See “MIDPOINT Variable” on page 722
See “SUBGROUP Variable” on page 733

HSYS='coordinate-system'
specifies the coordinate system for the SIZE variable.

See “HSYS Variable” on page 714

HTML='link-string'
specifies the text that defines the drill-down link.

See “HTML Variable” on page 715

HTML="ANIMATE=animate-action(parameters)"
animates the text. You specify the animation action and the parameters that control the animation.

See “"ANIMATE=animate-action(parameters)"” on page 715

POSITION='text-position' | '0'
controls the text string placement and alignment. Text-position can be one of the characters 1 through 9, A through F, <, +, or >. Invalid or missing values default to POSITION='5'. POSITION should always be a character variable of length 1.

See “POSITION Variable” on page 724

ROTATE=rotation-angle
specifies the rotation angle of each character in the string. It is equivalent to the ROTATE= option in the FOOTNOTE, NOTE, and TITLE statements.

See “ROTATE Variable” on page 727

SIZE=height
specifies the height of the text string. The SIZE variable units are based on the value of the HSYS variable.

See “SIZE Variable” on page 728

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 729

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.

X=horizontal-coordinate
Y=vertical-coordinate
Z=depth-coordinate
XC='character-type-horizontal-coordinate'
YC='character-type-vertical-coordinate'
specify the start point of the text string. You can also modify the placement of the
text string with the POSITION variable.

Restriction
The Z= variable is used with the G3D procedure only.

XSYS='coordinate-system'
specifies the coordinate system for the X or XC variable. Use the XC variable only
with XSYS='2'.

YSYS='coordinate-system'
specifies the coordinate system for the Y or YC variable. Use the YC variable only
with YSYS='2'.

ZSYS='coordinate-system'
specifies the coordinate system for the Z variable.

Restriction
The ZSYS= variable is used with the G3D procedure only.

Details

Animating Text Labels
When you create text labels using the Annotate facility, you can animate those labels if
you use one of the SAS/GRAPH SVG device drivers, except SVGANIM. Various
animation actions are available, including moving text on a path, growing or shrinking
text, moving text, and blinking text.

To animate labels, add the HTML variable to the LABEL function in the annotate data
set. The HTML variable accepts attributes that specify the animation action and the
parameters that control the animation.
In the following example, the HTML variable specifies that the text “ALERT: Here is a moving news ticker” moves continuously across the bottom of the screen. The code for this example can be found in Example Code 28.1.

The figure captures a still frame of the output.

**Figure 28.12  Example: Adding a Moving Text Annotation to Charts**

Here is the HTML variable statement that created the moving text.

`html="animate=move('ALERT: Here is a moving news ticker',
100, +0, 0, 5, repeat)";`

The following items describe the parts of the statement:

1. **ANIMATE=MOVE(parameters)** specifies the type of animation to perform. In this example, the MOVE animation is performed. The HTML variable specified must be enclosed in double quotation marks.

2. `'ALERT: Here is a moving news ticker'` specifies the text that is moved. The text must be enclosed in single quotation marks. In addition, the text must match the text string that is provided for the TEXT variable in the LABEL function.

3. `100` specifies the X coordinate that the text moves to in percentage units of the display or the SVG space.

4. `+0` specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space. The + indicates that the text moves the specified amount relative to the Y value that is specified for the label.
5 0, 5 specifies that the animation begins 0 seconds after the SVG page has been loaded and has a duration of 5 seconds.

6 REPEAT specifies that the animation repeats continuously.

The attributes that determine the text color, background color, and so on, are specified separately in the data set.

Here are the main steps for producing the graph in the example.

1. In your annotation data set, after you specify FUNCTION='LABEL', add the HTML variable that specifies the animation action and associated parameters.

   For complete syntax, see “HTML Variable” on page 715.

2. Use the GOPTIONS statement to specify one of the SVG device drivers, except SVGANIM.

   goptions dev=svgview;

3. Specify the ODS HTML5, ODS HTML, or the ODS LISTING destination.

   TIP If using one of the ODS HTML destinations, see “Browser Support for Viewing SVG Files” on page 149 to determine whether to use HTML5 or HTML.

4. After you have set the options and opened the ODS destination, proceed with your SAS code to create the graph. The graph code should be designed to display the annotation. For more information, see “Producing Graphics Output from Annotate Data Sets” on page 651.

SAS/GRAPH generates the output as an SVG graphic. To view the output, open the SVG file in a browser.

Note: If your browser does not render the graphic, see “Browser Support for Viewing SVG Files” on page 149.

The following code produces the output shown in the previous example.

**Example Code 1 Code for Adding a Moving Text Annotation to Charts**

```sas
/* Create a file reference for the ODS output */
filename odsout ".";

/* Specify the graphics options */
options nodate nonumber nofontembedding;
goptions reset=all device=svgview;
on _all_ close;
ods html path=odsout file="animlabel.html";

/* Create the annotation data set */
/* that includes animation for the label */
data moving;
  length color $ 8;
xsys='3'; ysys='3'; hsys='D'; position='6'; function='label';
x=0; y=5; color='black'; style='Arial'; size=18;
text='ALERT: Here is a moving news ticker';
html="animate=move('ALERT: Here is a moving news ticker',
  100, +0, 0, 5, repeat)";
output;
function='move'; x=0; y=3; hsys='3';
text=''; html=''; output;
function='bar'; x=100; y=5; hsys='3'; style='solid'; line=0;
```
/* Create the graph data by extracting information for */
/* Canada and Germany from sashelp.prdsale. */
data work.qsales;
set sashelp.prdsale(where=(country="CANADA" or country="GERMANY")
keep=Actual Country Product Quarter Year);
run;

/* Sort the data by quarter */
proc sort data=work.qsales;
by quarter;
run;

/* Generate the first set of graphs */
title1 "1993 Sales";
footnote h=2 ' ';
axis1 order=(0 to 30000 by 5000);
proc gchart data=work.qsales(where=(year=1993)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;

/* Generate the second set of graphs */
title1 "1994 Sales";
proc gchart data=work.qsales(where=(year=1994)) anno=moving;
vbar3d country / sumvar=actual subgroup=product sum raxis=axis1
shape=hexagon;
where product in ("BED" "TABLE" "CHAIR");
by quarter;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

You can also use Base SAS Universal Printing to generate SVG output. For more information, see “Creating SVG (Scalable Vector Graphics) Files Using Universal Printing” in SAS Language Reference: Concepts.

For another example of label text animation, see “Example: Moving Text on a Path” on page 687.

See Also
Chapter 13, “Using SVG Graphics,” on page 141
Example: Moving Text on a Path

This example moves text on a path that is defined in the annotate data set using the DRAW function. The path moves from left to right in a zig-zag motion, first up, then down, and back up again. The following figure shows a snapshot of the moving text.

In the figure, the path defined by the DRAW function is made visible so that you can see how the text moves along the defined text path. Here is the code for this example. In the output from this code, the path that is defined by the DRAW function is completely transparent.

```sas
filename odsout ".\";

data annotp;
length function $8 color $ 10 style $ 12 text $ 50 html $ 100;
xsys='3'; ysys='3'; hsys='D';

/* Label observation to create moving text on a path */
function='label'; x=0; y=50; size=12; color='blue'; style='Arial/bold';
text='Up  and  down  and  UP  UP  UP';
html="animate=textpath('Up  and  down  and  UP  UP  UP', 0, 100, 0, 5, repeat)";
output;

/* Move and Draw observations to create transparent line */
function='move'; x=0; y=0; color='AFFFFFF00'; html=''; text='';
size=.; style='';
hsys=''; position=''; output;
function='draw'; x=30; y=50; color='AFFFFFF00'; output;
function='draw'; x=45; y=30; color='AFFFFFF00'; output;
function='draw'; x=85; y=90; color='AFFFFFF00'; output;

run;

ods _all_ close;
ods html path=odsout file="movingtext.html";

options nofontembedding;
```
MOVE Function

Moves the drawing pointer to a specific location without drawing a line.

**Updates:** XLAST, YLAST

**Syntax**

FUNCTION='MOVE';

**Associated Variables**

GROUP=group-value
MIDPOINT=midpoint-value
SUBGROUP=subgroup-value

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711
“MIDPOINT Variable” on page 722
“SUBGROUP Variable” on page 733

WHEN='B' | 'A'

specifies when to perform the move in relation to other procedure output.

See “WHEN Variable” on page 736

X=horizontal-coordinate
Y=vertical-coordinate
Z=depth-coordinate
XC='character-type-horizontal-coordinate'
YC='character-type-vertical-coordinate'

specify the coordinates to which the pen is to be moved.

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
“Y Variable” on page 744
“Z Variable” on page 747
“XC Variable” on page 739


“YC Variable” on page 745

**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 741

**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 746

**ZSYS=’coordinate-system’**

specifies the coordinate system for the Z variable.

Restriction: The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

**Details**

Use MOVE to prepare for a DRAW command, a BAR command, or programming functions.

---

**PIE Function**

Draws pie slices in the graphics output.

**Updates:** XLAST, YLAST to coordinates for center of the slice.

**Syntax**

FUNCTION=’PIE’;

**Associated Variables**

**ANGLE=starting-angle**

specifies the starting angle of the slice arc. The default is 0.00 (horizontal) if the ANGLE variable is not specified for the first slice. After the first slice, the default is the ending angle of the slice arc just drawn if ANGLE= (missing). Therefore, you can specify consecutive pie slices more easily by omitting the start and end calculations that are otherwise required. If you want the next slice to start at an angle that is different from the ending angle of the previous slice, specify a value for the ANGLE variable.

See “ANGLE Variable” on page 706

**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.

HTML='link-string'
specifies the text that defines the drill-down link.

LINE=0...3
specifies which slice line (or lines) to draw.

ROTATE=rotation-angle
specifies the angle of rotation or the delta angle of the slice arc.

For example, if you specify these statements, the slice arc that is drawn begins at 90 degrees (vertical) and ends at 135 degrees (90+45):

function="pie"; angle=90; rotate=45; output;

The ANGLE variable is internally updated to the end value, 135 degrees. The value is modified only internally. If a second PIE is used and the ANGLE variable contains a missing value, the start angle is assumed to be the previous end, or 135 degrees. The arc continues from that point.

If you specify the previous statements and then specify these statements, the slice begins at 135 degrees (the end angle from the previous slice). The slice extends another 45 degrees to the end point, 180 degrees.

function="pie"; angle=.; rotate=45; output;

This action repeats for every missing angle in the sequence.

Default 0

SIZE=radius
specifies the radius of the circle being drawn. The SIZE variable uses units that are determined by the HSYS variable.
STYLE='PSOLID' | 'PEMPTY' | \texttt{Density<\texttt{Style<\texttt{Angle}>}}

specifies the value of the pattern that fills the pie slices.

- **Default**: PEMPTY
- **Restriction**: Java and ActiveX support only PSOLID and PEMPTY.
- **See**: “STYLE Variable (Patterns)” on page 731

\texttt{WIDTH='line-thickness'}

specifies the thickness of the outline around the pie slice.

- **See**: “WIDTH Variable” on page 737

\texttt{WHEN='B' | 'A'}

specifies when to draw the pie slice in relation to other procedure output.

- **See**: “WHEN Variable” on page 736

\texttt{X=horizontal-coordinate}
\texttt{Y=vertical-coordinate}
\texttt{Z=depth-coordinate}
\texttt{XC='character-type-horizontal-coordinate'}
\texttt{YC='character-type-vertical-coordinate'}

define the center of the slice. The pivot point for all slices is the point referenced by X, Y, and Z (with the G3D procedure only). The first PIE command that is issued sets the center at the (X,Y) value. If subsequent values for X and Y are missing, the coordinates of the center point are used.

- **Restriction**: The Z= variable is used with the G3D procedure only.
- **See**: “X Variable” on page 738
  “Y Variable” on page 744
  “Z Variable” on page 747
  “XC Variable” on page 739
  “YC Variable” on page 745

\texttt{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 741

\texttt{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 746
ZSYS='coordinate-system'
specifies the coordinate system for the Z variable.

Restriction  The ZSYS= variable is used with the G3D procedure only.

See  “ZSYS Variable” on page 747

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.

Syntax
FUNCTION='PIECNTR';

Associated Variables
GROUP=group-value
MIDPOINT=midpoint-value
SUBGROUP=subgroup-value
specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See  “GROUP Variable” on page 711
     “MIDPOINT Variable” on page 722
     “SUBGROUP Variable” on page 733

HSYS='coordinate-system'
specifies the coordinate system for the SIZE variable.

See  “HSYS Variable” on page 714

SIZE=radius
specifies the new radius of the pie slice. The new radius is used by a subsequent PIEXY function.

Tip  The HSYS variable determines the SIZE variable units.

See  “SIZE Variable” on page 728

WHEN='B' | 'A'
specifies when to draw the pie slice in relation to other procedure output.

See  “WHEN Variable” on page 736
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).
SIZE = radius-multiplier
determines the distance from the center of the slice to the point that is being calculated. The point's distance is the current value of the SIZE variable multiplied by the radius (that is, the SIZE variable) of the previously drawn slice. To position a graphics element inside the pie slice, set the SIZE variable to less than 1; to position it outside of the pie slice, set the SIZE variable to greater than 1. For example, if you specify these statements, the point calculated is 1.1 times the radius. (The radius is taken from the SIZE variable that is used with the previous FUNCTION='PIE' or FUNCTION='PIECNTR' observation.)

function="piexy"; size=1.1; output;

WHEN='B' | 'A'
specifies when to update the internal coordinate pair (XLAST, YLAST) in relation to other procedure output.

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 28.13 on page 694 shows a pie slice that is drawn with the PIE function. Figure 28.14 on page 695 shows a point beyond the arc that was calculated using the PIEXY function.
**POINT Function**

Places a single point at the (X, Y) coordinates in the color that you specify. The point is one visible pixel in size.

**Updates:** XLAST, YLAST

**Syntax**

FUNCTION='POINT';

**Associated Variables**

**COLOR='color'**

specifies the color of the point to be drawn. Color can be any SAS/GRAPH color name.

See “COLOR Variable” on page 709

**GROUP=group-value**

**MIDPOINT=midpoint-value**

**SUBGROUP=subgroup-value**

specify coordinates when used with HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711

“MIDPOINT Variable” on page 722

“SUBGROUP Variable” on page 733

**WHEN='B' | 'A'**

specifies when to draw the point in relation to other procedure output.

See “WHEN Variable” on page 736

**X=horizontal-coordinate**

**Y=vertical-coordinate**

**Z=depth-coordinate**

**XC='character-type-horizontal-coordinate'**

**YC='character-type-vertical-coordinate'**

specify the coordinates of the point that is to be drawn.
Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
“Y Variable” on page 744
“Z Variable” on page 747
“XC Variable” on page 739
“YC Variable” on page 745

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 741

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 746

ZSYS='coordinate-system'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

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/GRAFH color name.

See “COLOR Variable” on page 709

HTML='link-string'
specifies the text that defines the drill-down link.

See “HTML Variable” on page 715
LINE=1...46
specifies the line type that outlines the polygon.

See “LINE Variable” on page 721
“Specifying Line Types” on page 442 for an illustration of the line types.

STYLE='MSOLID' | 'MEMPTY' | \texttt{M}density<\texttt{style}<\texttt{angle}>>
specifies the value of the pattern that fills the polygon.

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 \textit{fill-pattern} value M5N15 specifies that the lines use the heaviest density, are parallel, and are drawn at a 15-degree angle from the horizontal.

Default MEMPTY

Restriction Java and ActiveX support only MSOLID and MEMPTY.

See “STYLE Variable (Patterns)” on page 731

\textit{WHEN}='B' | 'A'
specifies when to begin the polygon in relation to other procedure output.

See “WHEN Variable” on page 736

\textit{X}=\textit{horizontal-coordinate}
\textit{Y}=\textit{vertical-coordinate}
\textit{Z}=\textit{depth-coordinate}
\textit{XC}='\textit{character-type-horizontal-coordinate}'
\textit{YC}='\textit{character-type-vertical-coordinate}'
specify the initial point of the polygon that is being created.

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
“Y Variable” on page 744
“Z Variable” on page 747
“XC Variable” on page 739
“YC Variable” on page 745

\textit{XSYS}='\textit{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 741

\textit{YSYS}='\textit{coordinate-system}'
specifies the coordinate system for the Y or YC variable. Use the YC variable only with YSYS='2'.

\textit{POLY Function} 697
See “YSYS Variable” on page 746

\[
\text{ZSYS='coordinate-system'}
\]

specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

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 698

---

**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.

See “COLOR Variable” on page 709

**WHEN='B' | 'A'**

specifies when to draw the polygon in relation to other procedure output.

See “WHEN Variable” on page 736

\[
\begin{align*}
X&=\text{horizontal-coordinate} \\
Y&=\text{vertical-coordinate} \\
Z&=\text{depth-coordinate} \\
\text{XC}&='\text{character-type-horizontal-coordinate}' \\
\text{YC}&='\text{character-type-vertical-coordinate}'
\end{align*}
\]

specify a point on the outline of the polygon that is being created.

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
XSYS='coordinate-system'
specifies the coordinate system for the X and XC variable.

See “XSYS Variable” on page 741

YSYS='coordinate-system'
specifies the coordinate system for the Y and YC variable.

See “YSYS Variable” on page 746

ZSYS='coordinate-system'
specifies the coordinate system for the Z variable.

Restriction The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

Details
The polygon definition is terminated by a new POLY command or by any of these functions:

- BAR
- LABEL
- PIEXY
- DRAW
- MOVE
- POINT
- DRAW2TXT
- PIE
- SYMBOL
- FRAME
- PIECNTR

Use POLY and POLYCONT together to draw a polygon. The (X, Y) observation from the POLY function and the last (X, Y) observation from POLYCONT are assumed to connect. Thus, you are not required to respecify the first point. For example, these statements draw a pentagon like the one in Figure 28.15 on page 700:

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;
Missing values for the X and Y variables that are specified with POLYCONT are interpreted differently from how they are interpreted with the other functions. Other functions use the missing values to request a default value. POLYCONT interprets a missing value as a discontinuity (that is, a hole) in the polygon. If you are not using the data coordinate system and you specify an X or Y value of –999 in a POLYCONT observation, the default of (XLAST, YLAST) is used. Missing values indicate holes and are handled identically in the Annotate facility and the GMAP procedure. See “Displaying Map Areas and Response Data” in SAS/GRAPH and Base SAS: Mapping Reference for more information about handling missing values.

### POP Function

Removes the (XLAST, YLAST) and (XLSTT, YLSTT) values from the LIFO stack and updates the internal coordinate pairs with the retrieved values.

**Updates:** (XLAST, YLAST) and (XLSTT, YLSTT)

**Syntax**

FUNCTION='POP';

**Comparisons**

Use POP when you want to access the values of (XLAST, YLAST) and (XLSTT, YLSTT) that you most recently stored with the PUSH function. See the PUSH function for a description of the LIFO stack.
PUSH Function

Adds current (XLAST, YLAST) and (XLSTT, YLSTT) values to the LIFO stack.

Syntax

FUNCTION='PUSH';

Comparisons

The LIFO (last-in-first-out) stack is a storage area where you can keep internal coordinate values for later use by utility functions without recalculating those values. LIFO stacks manage the stored data so that the last data stored in the stack is the first data removed from the stack.

Use the stack to save the current values of (XLAST, YLAST) and (XLSTT, YLSTT) and use them with functions later in the DATA step. You store and retrieve these values from the stack with the PUSH and POP functions. The PUSH function copies the current values of XLAST, YLAST, XLSTT, and YLSTT onto the stack. The POP function copies values from the stack into XLAST, YLAST, XLSTT, and YLSTT.

SWAP Function

Exchanges values of (XLAST, YLAST) with (XLSTT, YLSTT) and vice versa.

Updates: (XLAST, YLAST) and (XLSTT, YLSTT)

Syntax

FUNCTION='SWAP';

Comparisons

Use SWAP when you want to use both the (XLAST, YLAST) and (XLSTT, YLSTT) coordinates for text and nontext functions, respectively.

SYMBOL Function

Places symbols in the graphics output. Associated variables can specify the color, font, and height of the symbols displayed.

Updates: XLSTT, YLSTT

Syntax

FUNCTION='SYMBOL';
**Associated Variables**

**CBORDER=’color’ | ’CTEXT’**

draws a colored border around the text. *Color* can be any SAS/GRAPH color name.

See “CBORDER Variable” on page 707

**CBOX=’color’ | ’CBACK’**

draws a solid, colored box behind the text. *Color* can be any SAS/GRAPH color name.

See “CBOX Variable” on page 708

**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 Variable” on page 709

“SYMBOL Statement” on page 412

**GROUP=group-value**

**MIDPOINT=midpoint-value**

**SUBGROUP=subgroup-value**

specify coordinates for HBAR and VBAR charts from the GCHART procedure. Use these variables only with the data coordinate systems 1, 2, 7, and 8.

See “GROUP Variable” on page 711

“MIDPOINT Variable” on page 722

“SUBGROUP Variable” on page 733

**HSYS=’coordinate-system’**

specifies the coordinate system for the SIZE variable.

See “HSYS Variable” on page 714

**HTML=’link-string’**

specifies the text that defines the drill-down link.

See “HTML Variable” on page 715

**SIZE=height**

specifies the height of the symbol that is being drawn, using units determined by the HSYS variable. The SIZE variable is equivalent to the HEIGHT= option in the SYMBOL statement.

See “SIZE Variable” on page 728

“SYMBOL Statement” on page 412

**STYLE=’font-specification’ | ’NONE’;**

specifies the font that is used to draw the symbol that is specified by the TEXT variable.

When the STYLE variable is used with the SYMBOL function, it behaves the same as the FONT= option in the SYMBOL statement. By default, no font is specified and
the symbol that is specified by the TEXT variable is taken from the special symbol table. If you use STYLE to specify a symbol font, such as Marker, the string that is assigned by the TEXT variable is the character code for a symbol. If you use STYLE to specify a text font, such as Swiss, the string assigned by the TEXT variable is displayed as text.

See “STYLE Variable (Arrows)” on page 731
See “STYLE Variable (Fonts)” on page 729

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.

For ActiveX, the following values are supported: plus, X, star, square, diamond, triangle, dot, circle, ", #, $, %. If a symbol is not supported, a plus sign (+) is drawn instead.

For Java, the following values are supported: plus, X, star, square, diamond, triangle, dot (draws a circle), circle, *, +, >. If a symbol is not supported, a plus sign (+) is drawn instead.

If you also specify a text font with the STYLE variable, you can specify a text string that is displayed as the symbol. The maximum length for text-string is 200 characters.

When the TEXT variable is used with the SYMBOL function, it behaves the same as the VALUE= option in the SYMBOL statement.

See “TEXT Variable” on page 735
See “SYMBOL Statement” on page 412

WHEN='B' | 'A'

specifies when to draw the symbols in relation to other procedure output.

See “WHEN Variable” on page 736

X=horizontal-coordinate
Y=vertical-coordinate
Z=depth-coordinate
XC='character-type-horizontal-coordinate'
YC='character-type-vertical-coordinate'
specify the point at which the symbol is placed.

Restriction The Z= variable is used with the G3D procedure only.

See “X Variable” on page 738
See “Y Variable” on page 744
See “Z Variable” on page 747
See “XC Variable” on page 739
See “YC Variable” on page 745
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 741

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 746

ZSYS='coordinate-system'
specifies the coordinate system for the Z variable

Restriction The ZSYS= variable is used with the G3D procedure only.

See “ZSYS Variable” on page 747

Details
SYMBOL is similar to the LABEL function with these exceptions:

- SYMBOL draws symbols. If you do not specify a font, SYMBOL can use the symbols found in Table 24.6 on page 433.
- The text cannot be rotated or angled.
- The text string cannot be longer than eight characters.
- The text string is always centered with respect to x and y.

TXT2CNTL Function
Copies the values (XLSTT, YLSTT) to (XLAST, YLAST), replacing previous values of (XLAST, YLAST).

Syntax
FUNCTION='TXT2CNTL';

Comparisons
TXT2CNTL allows nontext functions to use the ending position of a text string as a starting or ending point.
Chapter 29
Annotate Variable Dictionary

About the Annotate Variables .................................................. 706
Annotate Internal Coordinates ................................................ 706
Dictionary .............................................................................. 706
  ANGLE Variable ................................................................. 706
  CBORDER Variable ............................................................ 707
  CBOX Variable ................................................................. 708
  COLOR Variable ................................................................. 709
  FUNCTION Variable ........................................................... 710
  GROUP Variable ............................................................... 711
  HSYS Variable ................................................................. 714
  HTML Variable ............................................................... 715
  IMGPATH Variable ............................................................ 720
  LINE Variable ................................................................. 721
  MIDPOINT Variable .......................................................... 722
  POSITION Variable .......................................................... 724
  ROTATE Variable ............................................................. 727
  SIZE Variable ................................................................. 728
  STYLE Variable (Fonts) ....................................................... 729
  STYLE Variable (Images) ..................................................... 730
  STYLE Variable (Arrows) .................................................... 731
  STYLE Variable (Patterns) ................................................... 731
  SUBGROUP Variable ........................................................ 733
  TEXT Variable ............................................................... 735
  WHEN Variable ............................................................... 736
  WIDTH Variable .............................................................. 737
  X Variable .................................................................... 738
  XC Variable ................................................................. 739
  XLAST, YLAST Variables .................................................... 740
  XLSTT, YLSTT Variables .................................................... 741
  XSYS Variable ............................................................... 741
  Y Variable .................................................................... 744
  YC Variable ................................................................. 745
  YSYS Variable ............................................................... 746
  Z Variable .................................................................... 747
  ZSYS Variable .............................................................. 747
About the Annotate Variables

When an Annotate data set is processed, the Annotate facility looks at the values of specific variables in order to draw graphics. This section describes all of the Annotate variables in alphabetical order. Not all variables are used with all functions. Refer to the description of the individual functions in “About the Annotate Functions” on page 667 for more information about how each variable is used with each function. For a summary of Annotate variables and their uses, see Table 27.1 on page 641.

Annotate Internal Coordinates

The Annotate facility maintains two sets of internal coordinates that are stored in the variable pairs (XLAST, YLAST) and (XLSTT, YLSTT). One set of variables (XLAST, YLAST) stores coordinate values that are generated by nontext functions and the other set (XLSTT, YLSTT) stores coordinates generated by text functions. These two variable pairs supply default values when the X or Y variable contains a missing value.

Both pairs are initially set to 0 and remain 0 until a function updates the values. You cannot assign explicit values to these variables, but you can manipulate their values with some of the Annotate functions.

Dictionary

ANGLE Variable

Specifies the angle at which the graphics output is drawn.

<table>
<thead>
<tr>
<th>Type</th>
<th>numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>function dependent</td>
</tr>
</tbody>
</table>

Syntax

ANGLE= angle-value;

Variable Values

angle-value

specifies the angle at which the graphics output is to be drawn.

Details

The ANGLE variable is function dependent.
### CBORDER Variable

Draws a colored border around text or symbols.

**Type:** character  
**Length:** 8 for color codes and up to 64 for color names  
**See:** CBOX

#### Syntax

`CBORDER='color' | 'CTEXT';`

#### Variable Values

**color**

specifies the color that outlines the box. The `color` value can be any SAS/GRAPH color name. See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313 for more information about specifying colors.

Specifying a null value for the `color` value (`CBORDER=''`) cancels the CBORDER variable.

**CTEXT**

draws the border in the same color as the text or symbol. The text color is determined by (1) the COLOR variable or (2) the CTEXT=graphics option or (3) the first color in the color list.

<table>
<thead>
<tr>
<th>Function</th>
<th>What the ANGLE variable specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARROW</td>
<td>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.</td>
</tr>
<tr>
<td>LABEL</td>
<td>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.</td>
</tr>
<tr>
<td>PIE</td>
<td>the starting angle of the slice arc, measured counterclockwise. The valid values are from –360 to 360. The default for the first PIE function is ANGLE=0 (horizontal, or 3:00 position), or is the ending point of the arc of the previous slice. Specify a value for the ANGLE variable if you want the next slice to start at an angle that is different from the edge of the previous slice. Also, specify a value if you want the first slice to start at an angle other than horizontal.</td>
</tr>
<tr>
<td>PIEXY</td>
<td>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.</td>
</tr>
</tbody>
</table>
Details

Using the CBORDER Variable
Once you have specified CBORDER, it remains in effect for all subsequent observations that use the LABEL or SYMBOL function and draws a border around all text or symbols. To turn off the border for subsequent text or symbols, specify CBORDER=' '.

To fill the area defined by CBORDER, use the CBOX variable in conjunction with CBORDER.

Functions
You can use the CBORDER variable with these functions:

- LABEL
- SYMBOL

CBOX Variable
Draws a solid box behind the text or symbol and fills the box with the specified color.

Type: character
Length: 8 for color codes and up to 64 for color names
Note: The size of the box that is drawn behind some symbols might change when UTF-8 encoding is used in the SAS session.
See: CBORDER

Syntax
CBOX='color' | 'CBACK';

Variable Values

- color
  specifies the color that fills the box. Color is any SAS/GRAIi color name. See Chapter 22, “Using Colors in SAS/GRAIi Programs,” on page 313 for more information about specifying colors.
  Specifying a null value for color (CBOX=' ') cancels the CBOX variable.

- CBACK
  fills the box with the same color as the background color of the graph. The background color is either (1) the color specified by the CBACK= graphics option or (2) the default background color for the device.

Details

Using the CBOX Variable
Once you have specified CBOX, it remains in effect for all subsequent observations that use the LABEL or SYMBOL function.

The color of the text or symbol within the box is controlled by the COLOR variable.
By default, the solid box has no border. To add a colored border to the box, use the CBORDER variable in conjunction with CBOX.

Functions
You can use the CBOX variable with these functions:

- LABEL
- SYMBOL

COLOR Variable
Specifies the color used by the function.

- **Type:** character
- **Length:** 8 for color codes and up to 64 for color names
- **Default:**
  1. COLORS= graphics option, if specified; 2. first color in device’s default color list

Syntax
COLOR='color';

Variable Values

*color*


Details
The COLOR variable is function dependent.

<table>
<thead>
<tr>
<th>Function</th>
<th>What the COLOR variable specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR</td>
<td>The color that outlines and, as an option., fill the bar if a pattern is specified in the STYLE (patterns) on page 731 variable. If no pattern is specified, the color value is applied only to the outline of the bar.</td>
</tr>
<tr>
<td>ARROW</td>
<td>The color of the arrow.</td>
</tr>
<tr>
<td>DRAW, DRAWTXT</td>
<td>The color of the line.</td>
</tr>
<tr>
<td>FRAME</td>
<td>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.</td>
</tr>
<tr>
<td>LABEL</td>
<td>The color of the text.</td>
</tr>
<tr>
<td>PIE</td>
<td>The color for the pie slice if a pattern is specified with the STYLE (patterns) on page 731 variable. If no pattern is specified, color determines the color of the outline of the pie slice.</td>
</tr>
</tbody>
</table>
### Function Variable

Specifies a graphics command or programming function for the Annotate facility to perform.

- **Type:** character
- **Length:** 8
- **Default:** LABEL

#### Syntax

FUNCTION='function-name';

#### Variable Values

*function-name*

specifies the name of an Annotate function. The *function-name* value can be any of the following.

- **BAR** draws and fills a rectangle (optional).
- **CNTL2TXT,** **DRAW2TXT** copies (XLAST, YLAST) to (XLSTT, YLSTT), overwriting the previous values of (XLSTT, YLSTT).
- **COMMENT** places comments in your data set. The observation is ignored when the data set is processed.
- **DEBUG** writes the values of all Annotate variables to the SAS log before and after the next observation.
- **DRAW** draws a line in the graphics output.
- **FRAME** draws a border around the area defined by XSYS and YSYS and specifies a background color for the framed area.
- **IMAGE** displays an image in the graphics output from the current (X,Y) coordinates to the coordinates that are associated with the IMGPATH variable.
- **LABEL** draws text and is the default for the FUNCTION variable.
MOVE moves to the specified point (does not draw a line).
PIE draws a pie slice, arc, or circle that can be filled.
PIECNTR sets new center and radius values. The PIEXY function can use this information in a later observation.
PIEXY returns the coordinates of a point on a pie slice. Other functions can use this information in a later observation.
POINT draws a point.
POLY begins drawing a polygon (first vertex). Use the POLYCONT function in successive observations to supply the remaining vertices.
POLYCONT continues drawing a polygon.
POP gets values from the LIFO stack and changes the current value of (XLAST, YLAST) and (XLSTT, YLSTT) to those values.
PUSH puts the current values for (XLAST, YLAST) and (XLSTT, YLSTT) in the LIFO stack.
SWAP exchanges the values of (XLAST, YLAST) and (XLSTT, YLSTT).
SYMBOL draws a symbol.
TXT2CNTL copies the values (XLSTT, YLSTT) to (XLAST, YLAST), overwriting the previous values of (XLAST, YLAST).

For a list of the symbols that can be used with the SYMBOL variable, see Table 24.6 on page 433.

All other variables in the observation that contain the function act as parameters for the action. For a detailed description of each function and the Annotate variables that can be used in conjunction with it, see “About the Annotate Functions” on page 667.

---

**GROUP Variable**

Positions graphics elements on the bars of a vertical or horizontal bar chart drawn using the GROUP= option in the GCHART procedure.

- **Type:** Numeric or character; must match the type of the GROUP= variable used in the GCHART procedure.
- **Length:** Should match the length of GROUP= variable in the GCHART procedure.
- **Default:** none
- **Restriction:** Used only with vertical or horizontal bar charts produced by the GCHART procedure.

**Syntax**

GROUP=group-value;
Variable Values

*group-value*

references value(s) of the variable that is identified by the GROUP= option in the GCHART procedure either as a variable name or as an explicit data value. *Group-value* can be one of the following:

*group-variable*

the name of a group variable.

*group-data-value*

a specific numeric data value.

'*group-data-value'*

a specific character data value.

To annotate all the bars in a horizontal or vertical bar chart, specify a variable name. To annotate a bar chart for a specific value of the GROUP variable, specify a specific value.

Details

Using the GROUP Variable

Using the GROUP variable is similar to using the X and Y variables with data system coordinates to position graphics elements in a vertical or horizontal bar chart.

Figure 29.1 on page 713 shows how the GROUP variable works with the SUBGROUP and MIDPOINT variables to label the bars of a vertical bar chart.
The label showing the number of units that were sold in Dallas in the year 1997 is positioned by the values that are assigned to these Annotate variables:

- GROUP=YEAR (where YEAR is a variable in the GCHART data set)
- MIDPOINT=CITY (where CITY is a variable in the GCHART data set)
- SUBGROUP=ITEM (where ITEM is a variable in the GCHART data set).

**Functions**

You can use the GROUP variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

- BAR
- MOVE
- POINT
- DRAW
- PIE
- SYMBOL
- LABEL
- PIECNTR
HSYS Variable

Defines the coordinate system and area of the output used by the SIZE variable to display the Annotate graphics. In addition, you can use the HSYS variable with Java or ActiveX to control the marker size and line size for the BAR, DRAW, DRAW2TXT, POLY, and SYMBOL functions.

**Type:** character  
**Length:** 1  
**Default:** 4

**Syntax**

HSYS='coordinate-system';

**Variable Values**

*coordinate-system*

specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C as shown in the following table:

<table>
<thead>
<tr>
<th>Absolute Systems</th>
<th>Relative Systems</th>
<th>Coordinate System Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>percentage of data area</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>data values</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>percentage of graphics output area</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>cell in graphics output area</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>percentage of procedure output area</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>cell in procedure output area</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>point size (text only)</td>
</tr>
</tbody>
</table>

These values are also used by the XSYS and YSYS variables. See “Coordinate Systems” on page 646 for a description of the areas and coordinate systems.

**Details**

**About the Coordinate System**

The coordinate system that you specify with the HSYS variable affects how the function interprets the value of the SIZE variable. For example, if you use HSYS='3' and SIZE=10 with the DRAW function, the thickness of the line is 10% of the graphics output area. If you use HSYS='1' and SIZE=10 with DRAW, the thickness of the line is 10% of the data area.
For text only, HSYS='D' specifies that text sizes are in points. For example, if you use HSYS='D' and SIZE=10 for the LABEL function, the label text uses a 10-point font.

If you use HSYS='D' with a function that does not create text, a warning appears in the log and the HSYS='4' coordinate system is used.

**Functions**

You can use HSYS with these functions, all of which also use the SIZE variable:

- DRAW
- LABEL
- SYMBOL
- DRAW2TXT
- PIE
- FRAME
- PIECNTR

---

**HTML Variable**

Defines a link in the HTML file that is created for a drill-down graph. This link is associated with an area of the graph and contains valid HTML syntax that can link to a report or another graph. The report or graph is displayed when the user clicks on the area.

- **Type:** character
- **Length:** no limit
- **Default:** none
- **Tip:** When used with an SVG graphics device, the variable can create particular drill-down events. When used with the LABEL function and an SVG graphics device, this variable can be used to animate the label text.

**Syntax**

```
HTML='link-string';
HTML="ANIMATE=animate-action(parameters)";
HTML=enhanced-link-options;
```

**Variable Values**

- **link-string**

  specifies the text that defines the drill-down link. For more information about drill-down graphs and how to specify the link string, see “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192. For using the HTML variable for data tips, see “Adding Custom Data Tips with the HTML= Option” on page 191.

- **Restriction**

  The ACTXIMG device does not support nested quotation marks in drill-down link URLs for annotations. If you need to nest quotation marks in your annotation drill-down link URLs, use the PNG device instead.

"ANIMATE=animate-action(parameters)"

When used with the LABEL function and an SVG graphics device, the ANIMATE= keyword animates the label text. When you specify HTML=ANIMATE, you include the animation action and the parameters that control the animation.
Note: "ANIMATE=animate-action(parameters)" must be enclosed in double quotation marks. If you use single quotation marks, an error is generated.

When you animate the label text, the output is saved as an SVG graphic.

*animate-action* can be one of the following:

**BLINK**

causes the text in the label observation to blink at a specified speed.

Here is the syntax:

```
BLINK('text-string', begin, speed, REPEAT|repeat-value)
```

Provide all of the following parameters in the order listed here.

- `'text-string'` specifies the value of the label to blink. This parameter must exactly match the text variable in the corresponding label observation in the annotate data set. In addition, `'text-string'` must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

- `begin` specifies after how many seconds to begin the blinking action after the SVG page has been loaded.

- `speed` controls how fast the blinking should occur. You specify how many seconds should pass before the text becomes visible after it has become hidden.

- `REPEAT | repeat-value` specifies whether to repeat the blink action indefinitely or a specified number of times. Specify either the `REPEAT` keyword or a numeric value.

**Examples**

This example causes the text label 'blinking text' to blink when the page is loaded into the browser. The text blinks at a speed of one second, and repeats indefinitely.

```
HTML="animate=blink('blinking text', 0, 1, repeat)"
```

This example causes the text label 'text blinking for 10 times' to blink when the page is loaded into the browser. The text blinks 10 times at a speed of 2 seconds, and then stops blinking.

```
HTML="animate=blink('text blinking for 10 times', 0, 2, 10)"
```

**SIZE**

causes the text in the label observation to increase or decrease in size by modifying the font size of the text over a time interval.

Here is the syntax:

```
SIZE('text-string', begin, duration, REPEAT|repeat-value)
```

Provide all of the following parameters in the order listed here.

- `'text-string'` specifies the value of the label to resize. This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, `'text-string'` must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

- `font-size` specifies the new font size in pixels. The text changes to this size during the specified duration.
**begin** specifies after how many seconds to begin the sizing action after the SVG page has been loaded.

**duration** specifies the duration in seconds during which the text sizing should occur.

**REPEAT | repeat-value** specifies whether to repeat the size action indefinitely or a specified number of times. Specify either the REPEAT keyword or a numeric value.

**Examples**

This example causes the text label ‘growing text’ to increase to an 80-pixel font size. The increase occurs three seconds after the page is loaded into the browser over a duration of five seconds. The action repeats indefinitely.

HTML="animate=size('growing text', 80, 3, 5, repeat)"

This example causes the text label ‘growing text’ to decrease to 0 pixel font size. The increase occurs five seconds after the page is loaded into the browser over a duration of three seconds, and then the animation stops.

HTML="animate=size('shrinking text', 0, 5, 3, norepeat)"

**MOVE**

moves the text in the label observation to a specified coordinate in the SVG output within a specified time interval.

Here is the syntax:

MOVE('text-string', toX, toY, begin, duration, REPEAT|repeat-value)

Provide all of the following parameters in the order listed here.

*text-string* specifies the value of the label to move. This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, *text-string* must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

toX specifies the X coordinate that the text moves to in percentage units of the display or the SVG space. You can specify either a relative or absolute value. (See the following note for the MOVE action.)

toY specifies the Y coordinate that the text moves to in percentage units of the display or the SVG space. You can specify either a relative or absolute value. (See the following note for the MOVE action.)

begin specifies after how many seconds to begin the move action after the SVG page has been loaded. You can specify either a relative or absolute value. (See the following Note.)

duration specifies the duration in seconds during which the text move should occur.

**REPEAT | repeat-value** specifies whether to repeat the move action indefinitely or a specified number of times. Specify either the REPEAT keyword or a numeric value.
Note

The toX and toY coordinates can be specified in either a relative or absolute value from the original X and Y coordinates of the annotated text string. To indicate a relative move, add + or – prior to the coordinate. The symbol causes the text to move the specified amount in a positive or negative direction relative from the original text coordinate. To move the text to an absolute position, specify the percentage-based coordinates without the + or –.

Examples

This example moves the text label 'moving text' to the bottom right corner of the screen when the page is loaded into the browser. The text moves over a duration of four seconds, and repeats indefinitely.

HTML="animate=move('moving text', 100, 0, 0, 4, repeat)"

This example moves the text label across the bottom of the screen when the page is loaded into the browser. The text moves over a duration of five seconds, and repeats indefinitely. The X value of 100 is interpreted as an absolute value and moves the text 100% across the display. The Y value of +0 is relative to the original Y coordinate. The 0 value has the effect of not moving the text vertically.

HTML="animate=
move('ALERT: SVG can create a moving news ticker!',
100, +0, 0, 5, repeat)"

**TEXTPATH**

causes the text in the label observation to move along a path at a specified speed. The path that the text follows must also be defined in the annotate data set. (See the requirement for TEXTPATH.)

Here is the syntax:

SIZE('text-string', from-offset, to-offset, begin, duration, REPEAT|repeat-value)

Provide all of the following parameters in the order listed here.

*text-string* specifies the value of the label to move. This parameter must exactly match the text variable in the label observation in the annotate data set. In addition, *text-string* must be enclosed in single quotation marks. If you use double quotation marks, an error is generated.

from-offset specifies the starting position of the text string relative to the path. This value is expressed as a percentage of the path.

to-offset specifies the ending position of the text string relative to the path. This value is expressed as a percentage of the path.

begin specifies after how many seconds to begin the move action after the SVG page has been loaded.

duration specifies the duration in seconds during which the move should occur.

REPEAT | repeat-value specifies whether to repeat the move action indefinitely or a specified number of times. Specify either the REPEAT keyword or a numeric value.

Restriction

There can be only one TEXTPATH action per SVG file.
### Requirement
The line that the text follows must be added to the annotate data set using the MOVE and DRAW functions. The line annotation must immediately follow the label observation in the data set.

### Example
This example causes the text label ‘moving text on a path’ to move from the 0% to 100% offset of the path. The animation occurs when the page is loaded into the browser. The text moves across the annotated line over a duration of five seconds, and repeats indefinitely.

```html
HTML="animate=textpath('moving text on a path', 0, 100, 0, 5, repeat)"
```

### Restriction
Available only when used with the LABEL function and an SVG graphics device.

### See
Chapter 13, “Using SVG Graphics,” on page 141

### enhanced-link-options
When you generate output as an SVG graphic, you have additional options for enhancing your drill-down links.

The SVG graphics devices provide the following `enhanced-link-options`.

- `ONMOUSEOVER=` specifies particular events that occur when the mouse pointer is positioned over a linked shape.
- `ONCLICK=` specifies particular events that occur when a linked shape is clicked.

For the complete syntax with examples, see “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198.

### See
Chapter 13, “Using SVG Graphics,” on page 141

### Details

#### Setting the Length of the HTML Variable
Use a LENGTH statement to set the length of the HTML variable to the longest string that you need for the link string. Be sure to set the HTML value to a null if you continue writing observations to the annotate data set after you are done assigning links. For example, the following code defines link information for two squares, but then sets the HTML variable to null when drawing a frame. Otherwise, the background area within the frame will use the link information from the last defined HTML value and become a hot zone in the graph.

```syntax
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 */
```
Functions
You can use the HTML variable with these functions:

- ARROW
- BAR
- FRAME
- IMAGE
- LABEL
- PIE
- POLY
- SYMBOL

IMGPATH Variable
Specifies an image to be displayed from the current (X,Y) coordinates to the (X,Y) coordinates that are associated with this variable.

Type: character
Length: 255

Syntax
IMGPATH = 'external-file';

Variable Values

external-file
specifies the full path or full filename of an external image file. The format of the external file specification varies between operating environments.

You can also specify the URL for a graphic on a web page. For example:

'http://mywebsite.com/myImage.png'

Note  Copying and pasting the image works only if an absolute path is specified instead of a relative path, or if the file into which the image is being pasted is opened from the directory to which the image is relative.
Details

The IMGPATH variable can be used only with the “IMAGE Function” on page 679.

The manner in which the specified image is to be displayed is determined by the “STYLE Variable (Images)” on page 730.

For a list of the file types that you use, see “Image File Types Supported by SAS/GRAPH” on page 331.

LINE Variable

Controls the drawing of a line by determining either the type of line to draw or the relative position of the line.

<table>
<thead>
<tr>
<th>Type:</th>
<th>numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>1</td>
</tr>
</tbody>
</table>

Syntax

LINE=line-type;

Variable Values

line-type

specifies the line type. The line type that can be used depends on the function. See “Details” on page 721.

Details

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.

Figure 29.2 Sample LINE Values for Filled Arrowheads

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 29.3  LINE Values for Bars

DRAW, DRAW2TXT, FRAME, POLY
Valid values are whole numbers from 1 to 46. A value of 1 specifies a solid line. The remaining values specify different segmented lines, as illustrated in Figure 24.21 on page 443.

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 29.4 on page 722.

Figure 29.4  LINE Values Used with the PIE Function

MIDPOINT Variable
Positions graphics elements on the bars of a vertical or horizontal bar chart drawn by the GCHART procedure.

Type: Numeric or character; must match the type of the midpoint variable in the GCHART procedure.
Length: Should match the length of the midpoint variable in the GCHART procedure.
Default: none
Restriction: Used only with vertical or horizontal bar charts produced by the GCHART procedure.

Syntax
MIDPOINT=midpoint-value;

Variable Values
midpoint-value
references midpoint data value(s) in the GCHART procedure either as a variable name or as an explicit data value. Midpoint-value can have one of the following forms:
**midpoint-variable**
the name of a midpoint variable.

**midpoint-data-value**
a specific numeric data value.

'**midpoint-data-value**'
a specific character data value.

Generally, specify a variable name if you want to annotate all of the bars in a horizontal or vertical bar chart. To annotate a bar chart for a specific value of the MIDPOINT variable, specify a specific value.

**Details**

**Using the MIDPOINT Variable**
Using the MIDPOINT variable is similar to using the X and Y variables to position graphics elements in a vertical or horizontal bar chart when using data system coordinates. For example, suppose you produce a vertical bar chart in which the chart variable CITY produces a bar for each city in a data set. The height of each bar is determined by the value of the SUMVAR= variable, UNITS.

You can label these bars by assigning the chart variable CITY to the Annotate MIDPOINT variable. The MIDPOINT variable provides the x coordinate for the label. By default, Annotate assigns the statistic variable, in this case the SUMVAR= variable, UNITS, to the Annotate Y variable, which provides the y coordinate for the label.

[Figure 29.5 on page 723](#) 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 29.5  Using the MIDPOINT Variable to Position a Label in a Bar Chart*

The labels in this figure are positioned by the values that are assigned to these Annotate variables:

- MIDPOINT=CITY (where CITY is the chart variable); the MIDPOINT variable provides the horizontal coordinate in the vertical bar chart.
• Y=UNITS (where UNITS is the SUMVAR= variable); the Y variable provides the vertical coordinate. By specifying Y=units/2, you can vertically center the label in the bar.

*Note:* In a horizontal bar chart, the MIDPOINT variable controls the $y$ coordinate and the statistic variable controls the $x$ coordinate.

**CAUTION:**

**Be careful when using MIDPOINT and X and Y variables in the same data set.**

Using the MIDPOINT and X variables in an Annotate data set that is used to annotate a VBAR chart can cause unexpected results. The same is true when using the MIDPOINT and Y variables in the same data set that is used to annotate an HBAR chart.

When annotating a VBAR chart, the Annotate facility uses the MIDPOINT variable as the horizontal coordinate if it exists in the Annotate data set and ignores the X variable. Consequently, you should use the MIDPOINT variable as the horizontal coordinate for all observations in an Annotate data set if you use it for one. A similar behavior occurs if you use both the MIDPOINT and Y variables in an Annotate data set that is used to annotate HBAR charts. The MIDPOINT variable is always used, regardless of whether it has a missing value, and the Annotate facility ignores the Y variable. In this case, as well, use the MIDPOINT variable for the vertical coordinate for all observations in an Annotate data set if you use it for one.

**Functions**

You can use the MIDPOINT variable only with the data coordinate systems 1, 2, 7, and 8, and with these functions:

- BAR
- MOVE
- POINT
- DRAW
- PIE
- SYMBOL
- LABEL
- PIECNTR

**POSITION Variable**

Controls placement and alignment of a text string specified by the LABEL function.

- **Type:** character
- **Length:** 1
- **Default:** 5

**Syntax**

POSITION='text-position' | '0';

**Variable Values**

- **text-position**

  specifies the placement of the text string in relation to the position that is defined by the X and Y variables. Text-position can be one of the characters 1 through 9, A through F, <, +, or >. These characters represent the positions that are described in the following table.
These positions are illustrated in Table 29.1 on page 725.

'0'
specifies a pause in the string in order to change an attribute, such as the color of the text.

**Details**

*Positioning Text Strings*

Table 29.1 on page 725 shows the effect of the POSITION= variable values on text string, "Text." In each image, The red dot represents the location specified for the text string. The gray box surrounding the text string indicates the bounding box for the text.

**Table 29.1 Effect of POSITION Values on Text Strings**
The <, +, and > positions are recommended for labels that are numbers. The + position is also recommended for annotating font-based markers. The <, +, and > positions are especially useful when you are labeling a horizontal bar chart. You can use <, +, or > if the numbers in your font are significantly smaller than the text and you are having trouble centering labels. If the numbers in your font are the same or nearly the same height as the text, you can use positions 4, 5, and 6 to center the labels.
Stacking Text Strings
To stack text strings, specify a different position value of each string. Figure 29.6 on page 727 shows two ways to stack text. The text bounding boxes are shaded with different colors so that you can see how they overlap when text strings are stacked.

Figure 29.6 Combining POSITION Values to Stack Text

Note: You cannot stack <, +, and > positions as you can 4, 5, and 6 positions.

Changing Attributes in the Middle of a Text String
0 is a special value to use when you want to pause and then continue a text string. With this value, you can change colors, fonts, and so on, in the middle of a line, while retaining the exact position of the text at the pause. When POSITION=’0’, the combined text string is left-justified beginning at the point that is defined by the X and Y variables. However, you must define missing values for X for the continuation string. The following Annotate data set changes the font in the middle of the string. The result is shown in Figure 29.7 on page 727.

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 29.7 Using POSITION=’0’ to Change the Attributes of a Text String

ROTATE Variable
Specifies the angle at which to rotate the graphics element.
**Syntax**

ROTATE=rotation-angle;

**Variable Values**

*rotation-angle*

specifies the angle at which the graphics element is to be rotated. This variable is function dependent. See “Details” on page 728.

**Details**

The ROTATE variable is function dependent:

<table>
<thead>
<tr>
<th>Function</th>
<th>What the variable does</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIE</td>
<td>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.</td>
</tr>
<tr>
<td>LABEL</td>
<td>Rotates the individual text characters with respect to the baseline.</td>
</tr>
</tbody>
</table>

---

**SIZE Variable**

Determines the size of the graphics element with which it is used.

- **Type:** numeric
- **Length:** 8
- **Default:** 1.00 (2 when HSYS=3)
- **Interaction:** For the LABEL function, the value of the HTEXT= goption is used as the default. However, the value of the GUNIT= goption affects the default value that is used by the SIZE= variable.

**Syntax**

SIZE=size-factor;

**Variable Values**

*size-factor*

specifies the size of the graphics element. This variable is function dependent. See “Functions” on page 729.
Details

About the SIZE Variable
The SIZE variable uses the coordinate system that is specified by the “HSYS Variable” on page 714, which specifies the type of coordinate system used to generate the graph.

For lines, as the thickness of the line increases, it might be impossible to center around a given coordinate. For example, if you specify a thickness of value 2 and HSYS='4', the first line is drawn at the (X, Y) coordinates. The second is drawn slightly above the first. The exact amount varies by device, but it is always one pixel in width. A thickness of value 3 produces one line above, one line at, and one line below the (X, Y) coordinate position.

The SIZE variable is equivalent to the HEIGHT= option in the SYMBOL statement. See “SYMBOL Statement” on page 412 for details.

See Figure 29.8 on page 729 for examples of line thicknesses.

Figure 29.8  Sample Line Thicknesses Used with the SIZE Variable

| 1 | 2 | 3 |

Functions
The SIZE variable is function dependent.

<table>
<thead>
<tr>
<th>Function.</th>
<th>What the variable does</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARROW</td>
<td>Determines the thickness of the arrow being drawn.</td>
</tr>
<tr>
<td>DRAW, DRAW2TXT, or FRAME</td>
<td>Determines the thickness of the line being drawn.</td>
</tr>
<tr>
<td>LABEL</td>
<td>Specifies the height of the text.</td>
</tr>
<tr>
<td>PIE or PIECNTR</td>
<td>Determines the radius of the pie.</td>
</tr>
<tr>
<td>PIEXY</td>
<td>Sets the radius multiplier.</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>Selects the height of the symbol.</td>
</tr>
</tbody>
</table>

STYLE Variable (Fonts)
Specifies a font for text or symbols produced by the LABEL or SYMBOL functions.

Type: character
Length: Depends on specification.
Default: default device-resident font
Restriction: Partially supported by ActiveX and not supported by Java

Syntax
STYLE='font-specification' | 'NONE';

Variable Values

font-specification

specifies a font. You can specify a GRSEG catalog entry that is supplied by SAS (for example, CENTB) or a system font that is available in your operating environment. A device-resident font can be specified by using either of these forms:

• HWxxxxn
• font-name

NONE

specifies the default device-resident font.

Details

When the STYLE variable is used with the SYMBOL function, it behaves the same as the FONT= option in the SYMBOL statement. By default, no font is specified and the symbol that is specified by the TEXT variable is taken from the special symbol table. If you use STYLE to specify a symbol font, such as Marker, the string that is assigned by the TEXT variable is the character code for a symbol. If you use STYLE to specify a text font, such as Swiss, the string assigned by the TEXT variable is displayed as text. See the FONT= option of the SYMBOL statement for details.

Note: Java does not support the STYLE variable. However, you can use special symbols from the MARKER font by using the SYMBOL function.

See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for more information about specifying fonts.

If the value of the STYLE variable is missing, SAS/GRAPH software searches for a font specification in this order:
1. the font specified by the FTEXT= graphics option
2. the device-resident font, if the device supports one
3. the SIMULATE font.

STYLE Variable (Images)

Determines the appearance of images specified with the IMGPATH variable and the IMAGE function.

<table>
<thead>
<tr>
<th>Type</th>
<th>character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>‘TILE’</td>
</tr>
</tbody>
</table>

Syntax

STYLE='TILE' | 'FIT' | 'SINGLE';
Variable Values

'TILE'
uses copies of the image to fill the image area.

'FIT'
stretches one instance of the image to fill the image area.

'SINGLE'
centers a single instance of the image on the specified coordinates. Any part of the image that extends beyond the data area is clipped. SINGLE is valid starting with SAS 9.4M5.

Details

This version of the STYLE variable can be used only with the “IMAGE Function” on page 679.

STYLE Variable (Arrows)

Specifies the type of arrowhead for arrows.

Type: character
Length: 8
Default: OPEN

Syntax

STYLE='CLOSED' | 'FILLED' | 'OPEN';

Details

The following figure shows an example of each arrowhead style.

STYLE Variable (Patterns)

Specifies a pattern for bars, pies, frames, and rectangles

Type: character
Length: 8
Default: EMPTY | PEMPTY | MEMPTY
Restriction: Partially supported by Java and ActiveX

Syntax

STYLE="fill-pattern";

Variable Values

*fill-pattern*

specifies a pattern to use with the graphics element. The value for *fill-pattern* is function dependent as shown in the following table.

<table>
<thead>
<tr>
<th>Function</th>
<th>Fill Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR, FRAME</td>
<td>SOLID</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>EMPTY</td>
<td>E</td>
</tr>
<tr>
<td>style &lt;density&gt;</td>
<td>style</td>
<td>R for right-slanted fill lines, L for left-slanted fill lines, or X for crossing fill lines.</td>
</tr>
<tr>
<td></td>
<td>density</td>
<td>Whole numbers 1 through 5 specify increasing thickness for the fill lines.</td>
</tr>
<tr>
<td>PIE</td>
<td>PSOLID</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>PEMPTY</td>
<td>PE</td>
</tr>
<tr>
<td>P&lt;density&gt;&lt;style&gt;&lt;angle&gt;&gt;</td>
<td>density</td>
<td>Whole numbers 1 through 5 specify increasing thickness for the fill lines.</td>
</tr>
<tr>
<td></td>
<td>style</td>
<td>N, the default, can specify parallel fill lines; X can specify crossed fill lines.</td>
</tr>
<tr>
<td></td>
<td>angle</td>
<td>can specify the angle of the fill lines. Values range from 0 to 360. The angle is measured counterclockwise from the horizontal. The default is 0°, which draws horizontal lines.</td>
</tr>
</tbody>
</table>

For example, if STYLE="P5N15", a pie slice with a fill of parallel lines is produced. The fill uses the heaviest density to draw the lines, and the parallel lines are drawn at a 15-degree angle from perpendicular to the radius of the pie slice.

Note: Java and ActiveX support only PSOLID and PEMPTY values and default to PEMPTY if any other value is used.

An illustration of these pattern styles is provided in the definition of the VALUE= option of the PATTERN statement.
<table>
<thead>
<tr>
<th>Function</th>
<th>Fill Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLY</td>
<td>MSOLID</td>
<td>Fill with a solid color.</td>
</tr>
<tr>
<td>MEMPTY</td>
<td>ME</td>
<td>No fill.</td>
</tr>
</tbody>
</table>

- **Mdensity <style><angle>>**
  - **density**: Whole numbers 1 through 5 specify increasing thickness for the fill lines.
  - **style**: N, the default, can specify parallel fill lines; X can specify crossed fill lines.
  - **angle**: Can specify the angle of the fill lines. Values range from 0 to 360. The angle is measured counterclockwise from the vertical. The default is 0°, which draws vertical lines.

**Note:** Java and ActiveX support only MSOLID and MEMPTY and default to MEMPTY if any other value is used.

An illustration of these pattern styles is provided in the definition of the VALUE= option of the PATTERN statement.

* 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 VALUE= option description for the PATTERN statement on page 398.

---

**SUBGROUP Variable**

Positions graphics elements within subgrouped bars of a vertical or horizontal bar chart produced by the GCHART procedure.

- **Type:** Numeric or character; must match the type of the SUBGROUP variable used in the GCHART procedure.
- **Length:** Should match the length of the SUBGROUP= variable in the GCHART procedure.
- **Default:** none
- **Restriction:** The bar charts must have been produced using the SUBGROUP= option.

**Syntax**

```
SUBGROUP=subgroup-value;
```
Variable Values

\textit{subgroup-value}

references value(s) of the \texttt{SUBGROUP=} variable in the \texttt{GCHART} procedure either as a variable name or as an explicit data value. \textit{Subgroup-value} can have one of the following forms:

\textit{subgroup-variable}

the name of a subgroup variable.

\textit{subgroup-data-value}

a specific numeric data value.

\textit{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 \texttt{SUBGROUP} variable, specify a specific value.

Details

\textbf{Using the \texttt{SUBGROUP} Variable}

The \texttt{SUBGROUP} variable positions the graphics elements in subgroup segments in vertical and horizontal bar charts. Using the \texttt{SUBGROUP} variable is similar to using the \texttt{X} and \texttt{Y} variables with data system coordinates. For example, in a vertical bar chart that produces a bar for each city in a data set, you can easily label the subgroups in each bar by setting \textit{subgroup-variable} to the \texttt{GCHART} variable by which the bar is being subgrouped. This variable provides the \(y\) coordinate of the label. Therefore, you do not specify a competing value for \(y\), but instead specify \(y=\) or \(y=y\).

The \texttt{MIDPOINT} variable works well with the \texttt{SUBGROUP} variable to provide the \(x\) coordinate. In this example, if you set the \texttt{MIDPOINT} variable to the \texttt{GCHART} variable that contains the names of the cities, the \texttt{MIDPOINT} variable provides your \(x\) coordinate. Rather than providing the \texttt{X} and \texttt{Y} variables, you would use the \texttt{SUBGROUP} and \texttt{MIDPOINT} variables. \textbf{Figure 29.9 on page 735} shows how the \texttt{SUBGROUP} variable works with the \texttt{MIDPOINT} variable to label the bars of a vertical bar chart.
The label showing the number of printers sold in Atlanta is positioned by the values that are assigned to these Annotate variables:

- MIDPOINT=CITY (where CITY is a variable in the GCHART data set)
- SUBGROUP=ITEM (where ITEM is a variable in the GCHART data set)

**Functions**

You can use the SUBGROUP variable only with the data coordinate system 1, 2, 7, or 8, and with these functions:

- BAR
- MOVE
- POINT
- DRAW
- PIE
- SYMBOL
- LABEL
- PIECNTR

**TEXT Variable**

Specifies the text or symbol to be placed on the graphics output.

- **Type:** character
- **Length:** up to 200
- **Default:** blank string

**Syntax**

```
TEXT='text-string' | 'special-symbol';
```
**Variable Values**

*text-string*

specifies the text that is used as a label (LABEL or COMMENT function) or symbol (SYMBOL function). The maximum length for *text-string* is 200 characters.

*special-symbol*

specifies the name of a symbol from the special symbol table that is illustrated in Table 24.6 on page 433. The maximum length for *special-symbol* is eight characters.

**Details**

**Defining the TEXT Variable**

Define the TEXT variable with sufficient length to contain all of the characters in your text string. If you need longer strings, use separate observations and POSITION='0' to continue the text.

Use a LENGTH statement to set the length of the TEXT variable if the length of a text string is longer than one character.

**Functions**

You can use the TEXT variable with these functions:

- COMMENT
- LABEL
- SYMBOL

---

**WHEN Variable**

Specifies when the function is performed in relation to generating other graphics output for the procedure or in relation to generating other Annotate graphics.

**Type:** character

<table>
<thead>
<tr>
<th>Length</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
</tr>
</tbody>
</table>

**Syntax**

WHEN='B' | 'A' ;

**Variable Values**

*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.

**Default** B

**Notes**

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:

```plaintext
xsys="3"; ysys="3"; when="b";
function="move"; x=0; y=0; output;
```
Note that your graph might conceal your annotations unless your annotations are specified with the variable value WHEN=A. Specifying ‘A’ 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 might be made visible by default.

### Details

**The WHEN Variable and Annotate Data Set Processing**

Normally, observations in an Annotate data set are processed sequentially. If you use the WHEN variable, all those observations with a WHEN value of B are processed first. The procedure output is then processed (if one is to be produced), and finally the observations with a WHEN value of A are processed.

**Functions**

You can use the WHEN variable with these functions:

- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- DRAW2TXT
- PIECNTR
- SYMBOL
- FRAME
- PIEXY
- LABEL
- POINT

### WIDTH Variable

Determines the thickness of a line.

- **Type:** numeric
- **Length:** 8
- **Default:** 1

**Syntax**

`WIDTH=line-thickness;`

**Variable Values**

- `line-thickness`
  
  specifies the thickness of the line that is to be drawn.

**Details**

The WIDTH variable can be used only with the PIE function.

The WIDTH variable always specifies a width in pixels. The coordinate system that you specify with the HSYS variable does not affect the WIDTH variable.
Note: The WIDTH variable is not supported by Java when your graph contains a depth axis (for example, graphs that are created by the SCATTER statement of the G3D procedure).

Note: For ActiveX output, the maximum line thickness is ten pixels. If you specify a greater value, then the value is reduced to 10.

X Variable

Identifies the x coordinate of where a graphics element is to be drawn.

- **Type:** numeric
- **Default:** value of XLAST or XLSTT

**Syntax**

\[ X=\text{horizontal-coordinate}; \]

**Variable Values**

- **horizontal-coordinate** specifies the X coordinate of where the graphics element is to be drawn.

**Details**

**Using the X Variable**

Specify a corresponding vertical coordinate when using the X variable. This vertical coordinate can be specified with the Y, YC, MIDPOINT, or SUBGROUP variables, depending on the type of graph that you are annotating.

The X variable uses the units that are specified in the XSYS variable. If you use XSYS='2' and the data axis is entered as character, use the XC variable instead of the X variable.

If the value of the X variable is missing for a function that requires it, the value of the XLAST variable or the XLSTT variable is used. The XLAST variable is used with nontext functions, and the XLSTT variable is used with text functions.

**Functions**

You can use the X variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- IMAGE
- PIECNTR
- SYMBOL

Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.
**XC Variable**

Identifies the x coordinate of a graphics element when the coordinate value is character.

- **Type:** character
- **Length:** Should match that of the plot variable in the procedure. The maximum length is 32 characters in SAS 9.4M2 and in earlier releases. Starting with SAS 9.4M3, the maximum length is 256 characters.
- **Default:** the value of XLAST or XLSTT
- **Restriction:** Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

**Syntax**

```plaintext
XC='character-type-horizontal-coordinate';
```

**Variable Values**

- `character-type-horizontal-coordinate`
  - specifies the X coordinate of the graphics element to be drawn when the coordinate value is a character.

**Details**

**Using the XC Variable**

The XC variable is the character equivalent of the X variable. Use XC when the axis values are character. You must also specify a value of 2 (absolute data values) for the XSYS variable. (See also “XSYS Variable” on page 741.) If you use a value other than 2 for the XSYS variable, the graphics output is not displayed properly.

Figure 29.10 on page 740 illustrates the XC variable.
Figure 29.10  Using the XC and YC Variables with Character Data

Note: The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.

CAUTION: Do not use the X and XC variables in the same data set. Using both X and XC variables in the same data set can cause unpredictable results.

Functions
You can use the XC variable with these functions:

BAR    PIE    POLYCONT
DRAW   PIECNTR  SYMBOL
LABEL   POINT
MOVE    POLY

XLAST, YLAST Variables
Track the last values specified for the X and Y variables when X and Y are used with nontext functions.

Comparisons
The coordinate values that are stored in the (XLAST, YLAST) variables are automatically updated by these nontext functions: BAR, DRAW, MOVE, PIE, and POINT. These values are then available for use by other nontext functions that follow in the DATA step. (The DRAW2TXT graphics function uses XLAST and YLAST but does not update them.)
Because (XLAST, YLAST) are updated internally, you cannot specify values for them. However, their values can be manipulated by these programming functions:

- CNTL2TXT
- PUSH
- PIECNTR
- SWAP
- PIEXY
- TXT2CNTL
- POP

### XLSTT, YLSTT Variables

Track the last position for the X and Y variables when X and Y are used with text-handling functions.

### Comparisons

The coordinate values stored in the (XLSTT, YLSTT) variables are automatically updated by the LABEL and SYMBOL text functions. These values are then available for use by other text functions that follow in the DATA step.

Because (XLSTT, YLSTT) are updated internally, you cannot specify values for them. However, their values can be manipulated by these programming functions:

- CNTL2TXT
- PUSH
- DRAW2TXT
- SWAP
- POP
- TXT2CNTL

### XSYS Variable

Defines the coordinate system and area of the output used by the X and XC variables to display the Annotate graphics.

- **Type:** character
- **Length:** 1
- **Default:** 4

#### Syntax

```
XSYS='coordinate-system';
```

#### Variable Values

`coordinate-system`

specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C as shown in the following table:
### Using the XSYS Variable with the X and XC Variables

The coordinate system that you specify with the XSYS variable affects how the function interprets the value of the X or XC variable.

**Note:** Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.

### Functions

You can use the XSYS variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- FRAME
- PIECNTR
- SYMBOL

The behavior of the XSYS variable is function dependent for the 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.
To draw a frame that encloses the entire graphics output area, specify XSYS='3' and YSYS='3', as shown in the following figure.

*Figure 29.12  Frame Created When XSYS='3' and YSYS='3'*

To limit the size of the frame to the size of the procedure output area, specify a value of 5 for XSYS and YSYS.

Note that the values of XSYS and YSYS can differ. You can specify a frame that occupies the entire width of the graphics output area and only the vertical width of the procedure output area. To do this, specify XSYS='3' and YSYS='5', as shown in the following figure.
Y Variable

Identifies the \( y \) coordinate of where a graphics element is to be drawn.

**Type:** numeric

**Default:** value of YLAST or YLSTT

---

**Syntax**

\[
Y = \text{vertical-coordinate};
\]

**Variable Values**

- **vertical-coordinate**
  - Specifies the \( Y \) coordinate of where the graphics element is to be drawn.

---

**Details**

**Using the Y Variable**

Specify a corresponding horizontal coordinate when using the Y variable. You can specify the horizontal coordinate with the \( X \), \( XC \), MIDPOINT, or SUBGROUP variable, depending on the type of graph that you are annotating.

The Y variable uses the units specified in the YSYS variable. If you use YSYS='2' and the axis data is type character, use the YC variable instead of the Y variable.

If the value of the Y variable is missing for a function that requires it, the value of the YLAST variable or the YLSTT variable is used. The YLAST variable is used for nontext functions, and the YLSTT variable is used for text functions.

**Functions**

You can use the Y variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
YC Variable

Identifies the y coordinate of a graphics element when the coordinate value is character.

- **Type:** character
- **Length:** Should match that of the plot variable in the procedure. The maximum length is 32 characters in SAS 9.4M2 and in earlier releases. Starting with SAS 9.4M3, the maximum length is 256 characters.
- **Default:** YLAST | YLSTT
- **Restriction:** Used only with output from the GCHART and GPLOT procedures. Ignored if the axes are numeric.

**Syntax**

YC='character-type-vertical-coordinate';

**Variable Values**

`character-type-vertical-coordinate` specifies the Y coordinate of the graphics element to be drawn when the coordinate value is a character.

**Details**

**Using the YC Variable**

The YC variable is the character equivalent of the Y variable. Use YC when the axis values are character. You must also specify a value of 2 (absolute data values) for the YSYS variable. (See “YSYS Variable” on page 746.) If you use a value other than 2 for the YSYS variable, the graphics output is not displayed properly.

See Figure 29.10 on page 740 for an illustration of the YC variable.

**Note:** The X or XC variable is required unless either the MIDPOINT, GROUP, or SUBGROUP variable provides the horizontal coordinate.

**CAUTION:**

Do not use Y and YC variables in the same data set. Using both Y and YC variables in the same data set can cause unpredictable results.

**Functions**

You can use the YC variable with these functions:

- BAR
- PIE
- POLYCONT
- IMAGE
- PIECNTR
- SYMBOL
- LABEL
- POINT
YSYS Variable

Defines the coordinate system and area of the output used by Y and YC to display the Annotate graphics.

Type: character
Length: 1
Default: 4

Syntax

YSYS='coordinate-system';

Variable Values

coordinate-system
specifies a value that represents a coordinate system. Values can be 1 through 9 and A through C, as shown in the following table:

<table>
<thead>
<tr>
<th>Absolute Systems</th>
<th>Relative Systems</th>
<th>Coordinate System Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>Percentage of data area</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Data values</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>Percentage of graphics output area</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>Cell in graphics output area</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>Percentage of procedure output area</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>Cell in procedure output area</td>
</tr>
</tbody>
</table>

These values are also used by the HSYS and XSYS variables. See “Coordinate Systems” on page 646 for a description of the areas and coordinate systems.

Details

Using the YSYS Variable with the Y and YC Variables

The coordinate system that you specify with the YSYS variable affects how the function interprets the value of the Y or YC variable.

Note: Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.
**Functions**
The YSYS variable is function dependent, as defined in the “XSYS Variable” on page 741.

You can use the YSYS variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- FRAME
- PIECNTR
- SYMBOL

---

**Z Variable**
Identifies the z coordinate of where a graphics element is to be drawn.

**Type:** numeric  
**Length:** 8  
**Default:** none  
**Restriction:** For Java or ActiveX, you can use the Z variable with GMAP, GCHART, GCONTOUR, GPLOT, and G3D, for example, to add annotations above the plane of the map. For other devices, the Z variable is used only with output from the G3D procedure.

**Syntax**

\[ Z = \text{depth-coordinate}; \]

**Variable Values**

- **depth-coordinate**  
  specifies the Z coordinate of where the graphics element is to be drawn.

**Note**  
The Z variable uses the units that are specified in the ZSYS variable.

**Details**

You can use the Z variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- IMAGE
- PIECNTR
- SYMBOL

---

**ZSYS Variable**
Defines the coordinate system and area of the output used by Z variable to display the Annotate graphics.

**Type:** character
Syntax

ZSYS='coordinate-system';

Variable Values

coordinate-system

specifies a value that represents a coordinate system. Values can be 1, 2, 7, or 8 as shown in the following table:

<table>
<thead>
<tr>
<th>Absolute Systems</th>
<th>Relative Systems</th>
<th>Coordinate System Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>Percentage of data area</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Data values</td>
</tr>
</tbody>
</table>

See “Coordinate Systems” on page 646 for a description of the areas and coordinate systems.

Details

Using the ZSYS Variable with the Z Variable

The coordinate system that you specify with the ZSYS variable affects how the function interprets the value of the Z variable.

Note: Not all coordinate systems can be used with all Annotate variables. For any restrictions, see the individual variables in this section.

Functions

You can use the ZSYS variable with these functions:

- ARROW
- LABEL
- POINT
- BAR
- MOVE
- POLY
- DRAW
- PIE
- POLYCONT
- IMAGE
- PIECNTR
- SYMBOL
Chapter 30

Annotate Macro Dictionary

About the Annotate Macros .......................................................... 750
Using Annotate Macros .............................................................. 750
    Macro-Call Syntax .............................................................. 750
    Making the Macros Available .............................................. 750
    Annotate Macro Task Summary ............................................ 751

Dictionary ................................................................. 752
    %ANOMAC Macro ............................................................... 752
    %ARROW Macro .................................................................. 752
    %BAR, %BAR2 Macros ......................................................... 753
    %CENTROID Macro .......................................................... 754
    %CIRCLE Macro ................................................................ 755
    %CNTL2TXT Macro .......................................................... 756
    %COMMENT Macro ........................................................... 756
    %DCLANNO Macro ............................................................ 757
    %DRAW Macro ................................................................. 757
    %DRAW2TXT Macro .......................................................... 758
    %FRAME Macro ................................................................ 758
    %LABEL Macro ................................................................. 759
    %LINE Macro ................................................................... 760
    %MAPLABEL Macro .......................................................... 761
    %MOVE Macro .................................................................. 762
    %PIEXY Macro .................................................................. 762
    %POLY, %POLY2 Macro ....................................................... 763
    %POLYCONT Macro ............................................................ 764
    %POP Macro ..................................................................... 764
    %PUSH Macro ................................................................. 765
    %RECT Macro ................................................................... 765
    %SCALE Macro ................................................................. 766
    %SCALET Macro ............................................................... 768
    %SEQUENCE Macro .......................................................... 769
    %SLICE Macro ................................................................. 770
    %SWAP Macro ................................................................. 771
    %SYSTEM Macro ............................................................. 772
    %TXT2CNTL Macro ............................................................. 772
About the Annotate Macros

You can use Annotate macros within a SAS DATA step to simplify the process of creating Annotate observations. With a macro, you specify a function and assign variable values in one step without having to write explicit variable assignment statements. You can mix assignment statements and macro calls in the same DATA step.

This section describes all of the Annotate macros including the complete syntax and a description of the parameters. For more information about accessing and using macros, and for a summary of operations performed by the Annotate macros, see “Using Annotate Macros” on page 750.

Using Annotate Macros

Macro-Call Syntax

The general syntax of an Annotate macro call is

```sas
%macro-name(parameter1, parameter2, ...);
```

In general, the macro name represents a function and the parameters contain the values for the variables that can be used with the function. All macros except DCLANNO, SYSTEM, and SEQUENCE output one or more observations.

The Annotate macro parameters are positional parameters, which must be specified in the proper order in the macro call. The parameters are either numeric or character. Numeric parameters can be numeric constants or numeric variable names that have been initialized to the appropriate value. Most character parameters must be expressed as literal character strings without quotation marks. Exceptions are the text values that are used with the COMMENT and LABEL macros, which can be expressed as character strings enclosed in quotation marks or as character variable names.

When generating observations, an Annotate macro assigns the macro parameter values to the corresponding Annotate variables. The resulting observations are equivalent to those that you would create manually using assignment statements. For example, the following two statements are equivalent:

```sas
%LABEL(10, 15, "Graph", black, 0, 0, centb, 8);
function="label"; x=10; y=15; text="Graph"; color="black"; style="centb"; position="8"; output;
```

Making the Macros Available

Before you can use the Annotate macros, you must run the %ANNOMAC autocall macro to compile the Annotate macros as follows:

```sas
%annomac;
```

When executed, the %ANNOMAC macro writes a message to the SAS log indicating that the Annotate macros are available. The message also describes how to get help for using the macros. The macros remain available for the duration of your SAS session.
When you start a new SAS session, you must run the %ANNOMAC macro again to make the Annotate macros available in the new session.

**Annotate Macro Task Summary**

The following table summarizes the tasks performed by the Annotate macros.

**Table 30.1 Tasks with Annotate Macros**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Macro to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign values of (XLSTT,YLSTT) to (XLAST,YLAST)</td>
<td>%TXT2CNTL;</td>
</tr>
<tr>
<td>Begin drawing a polygon</td>
<td>%POLY(x, y, color, style, line);</td>
</tr>
<tr>
<td>Continue drawing a polygon</td>
<td>%POLYCONT(x, y, color);</td>
</tr>
<tr>
<td>Copy (XLAST,YLAST) to (XLSTT,YLSTT)</td>
<td>%CNTL2TXT;</td>
</tr>
<tr>
<td>Declare all variables</td>
<td>%DCLANNO;</td>
</tr>
<tr>
<td>Draw an arrow</td>
<td>%ARROW(x1, y1, x2, y2, color, line, size, angle, style);</td>
</tr>
<tr>
<td>Draw a bar</td>
<td>%BAR(x1, y1, x2, y2, color, line, style);</td>
</tr>
<tr>
<td>Draw a circle</td>
<td>%CIRCLE(x, y, size, color);</td>
</tr>
<tr>
<td>Draw a frame</td>
<td>%FRAME(color, line, size, style);</td>
</tr>
<tr>
<td>Draw a line from (XLAST,YLAST) to (XLSTT,YLSTT)</td>
<td>%DRAW2TXT(color, line, size);</td>
</tr>
<tr>
<td>Draw a line from previous point</td>
<td>%DRAW(x, y, color, line, size);</td>
</tr>
<tr>
<td>Draw a line</td>
<td>%LINE(x1, y1, x2, y2, color, line, size);</td>
</tr>
<tr>
<td>Draw a pie slice or arc</td>
<td>%SLICE(x, y, angle, rotate, size, color, style, line);</td>
</tr>
<tr>
<td>Draw a rectangle</td>
<td>%RECT(x 1,y 1,x 2,y 2, color, line, size);</td>
</tr>
<tr>
<td>Draw text</td>
<td>%LABEL(x, y, text, color, angle, rotate, size, style, position);</td>
</tr>
<tr>
<td>Exchange the values of (XLAST,YLAST) and (XLSTT,YLSTT)</td>
<td>%SWAP;</td>
</tr>
<tr>
<td>Move to a point near a pie slice</td>
<td>%PIEXY(angle, size);</td>
</tr>
<tr>
<td>Move to a point without drawing</td>
<td>%MOVE(x, y);</td>
</tr>
</tbody>
</table>
### Dictionary

%ANNOMAC Macro

Compiles Annotate macros and makes them available for use.

**Variables written out:**

- none directly

**Note:** Starting with SAS 9.4M6, it is not necessary to run the %ANNOMAC macro before using the %CENTROID macro.

### Syntax

%ANNOMAC;

### Comparisons

In a SAS session, you must submit the ANNOMAC macro before you can use the Annotate macros.

%ARROW Macro

Draws an arrow from (X1, Y1) to (X2, Y2).

**Updates:**

- XLAST, YLAST

**Variables written out:**

- ANGLE, COLOR, FUNCTION, LINE, SIZE, STYLE, X, Y
Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%ARROW (x1, y1, x2, y2, color, line, size, angle, style);

Parameters

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 738.

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 738.

color
specifies the color of the line using a character string without quotation marks. For details, see the Annotate “COLOR Variable” on page 709. 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 721 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 728 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 706 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 731.

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.

Updates: XLAST, YLAST

Variables written out: COLOR, FUNCTION, LINE, STYLE, X, Y
Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%BAR (x1, y1, x2, y2, color, line, style);
%BAR2(x1, y1, x2, y2, color, line, style, width);

Parameters

x1, y1
specify the location of the first corner of the bar. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

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 709.

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 721 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 731 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 728 for the DRAW function.

%CENTROID Macro

Retrieves the centroid positions of map data set polygons.

Variables written out: X, Y, ID variables

Requirement: In SAS 9.4M5 and in earlier releases, you must run the %ANNOMAC macro before using the %CENTROID annotate macro. See “Making the Macros Available” on page 750. Starting with SAS 9.4M6, it is not necessary to run the %ANNOMAC macro before using the %CENTROID macro.

Note: Starting with SAS 9.4M6, the %CENTROID autocall macro is available with Base SAS. The macro is no longer part of SAS/GRAPH, nor is SAS/GRAPH required to be installed to use this macro.

Tip: Use this macro as an aid in adding labels to a map. It can also be used in conjunction with the Base SAS GEODIST function when computing distances.
the “GEODIST Function” in *SAS Functions and CALL Routines: Reference* for more information.

## Syntax

%CENTROID (input-data-set, output-data-set, list-of-id-variables <, SEGONLY=n>);

### Parameters

**input-data-set**
- specifies a map data set. The input map data set must be sorted by the ID variables.

**output-data-set**
- contains the ID variables and the X and Y variables.

**list-of-id-variables**
- specifies the variables, each of which is assigned the centroid coordinates of each observation in the input-data-set. There is one observation for each unique set of ID values. If you specify more than one ID variable, then separate each variable with a space.

**SEGONLY=n**
- specifies that only the nth segment of each map area is used to calculate the centroid. For example, if you specify SEGONLY=1, then only the first segment of each area is used.

*Note:* Be sure to end the list of ID variables with a comma before specifying a SEGONLY= value.

---

## %CIRCLE Macro

Draws an empty circle with the center at \((x, y)\).

**Updates:** XLAST, YLAST

**Variables written out:** ANGLE, FUNCTION, ROTATE, SIZE, STYLE, X, Y

**Requirement:** You must run the %ANNOTMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

### Syntax

%CIRCLE \((x, y, size, color)\);

### Parameters

**x, y**
- specify coordinates for the center of the circle. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.
**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 728.

**color**

specifies the color of the circle using a character string without quotation marks. For details, see the Annotate “COLOR Variable” on page 709. Use an asterisk (*) to specify the previous value of the color parameter.

**See Also**

“%SLICE Macro” on page 770 to draw a filled circle

---

**%CNTL2TXT Macro**

Copies the values of the internal coordinates (XLAST, YLAST) to the text coordinate (XLSTT, YLSTT).

**Updates:**

- XLSTT, YLSTT

**Variables written out:**

- FUNCTION

**Requirement:**

You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

**Syntax**

```
%CNTL2TXT;
```  

**Comparisons**

The %CNTL2TXT macro is useful when you are calculating the position of labels on a graph. For an example, see “CNTL2TXT Function” on page 673.

---

**%COMMENT Macro**

Inserts a comment into an Annotate data set.

**Variables written out:**

- FUNCTION, TEXT

**Requirement:**

You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

**Syntax**

```
%COMMENT (text-string);
```
### Parameters

**text-string**

specifies the text to insert in the Annotate data set. The value can be a character string enclosed in quotation marks or the name of a character variable. For details, see the Annotate “TEXT Variable” on page 735.

---

### %DCLANNO Macro

Automatically sets the correct length and data type for all Annotate variables except the TEXT variable.

**Variables written out:** None

**Requirement:** You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

### Syntax

```
%DCLANNO;
```

---

### %DRAW Macro

Draws a line from (XLAST, YLAST) to the specified coordinate.

**Updates:** XLAST, YLAST

**Variables written out:** COLOR, FUNCTION, LINE, SIZE, X, Y

**Requirement:** You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

### Syntax

```
%DRAW (x, y, color, line, size);
```

---

### Parameters

**x, y**

specify coordinates for the end point of the line. Values can be coordinate numbers, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

**color**

specifies the color of the line using a character string without quotation marks. For details, see the Annotate “COLOR Variable” on page 709. 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 721 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 728 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

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax
%DRAW2TXT (color, line, size);

Parameters

color
specifies the color of the line using a character string without quotation marks. For details, see the Annotate “COLOR Variable” on page 709. 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 721 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 728 for the DRAW function.

%FRAME Macro
Draws a border around the portion of the display area defined by the reference system and can fill the area.

Variables written out: COLOR, FUNCTION, LINE, SIZE, STYLE

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.
Syntax

%FRAME (color, line, size, style);

Parameters

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 709. 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 721 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 728 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 731 for the FRAME function.

Details

See “%SYSTEM Macro” on page 772 for information about setting the reference system.

%LABEL Macro

Places a text label at the specified coordinates.

Parameters

x, y

specifies the location of the text string. Values can be coordinate numbers, numeric constants, or numeric variables. The position of the text string relative to x, y is determined by the position parameter. For details, see the Annotate “X Variable” on page 738.
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 735.

color
specifies the color of the text string using a character string without quotation marks. For details, see the Annotate “COLOR Variable” on page 709. 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 706 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 727.

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 728 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 729.

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 724.

%LINE Macro
Draws a line between two sets of coordinates.

Updates:  XLAST, YLAST
Variables written out:  COLOR, FUNCTION, LINE, SIZE, X, Y
Requirement:  You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax
%LINE (x1, y1, x2, y2, color, line, size);

Parameters
x1, y1
specify the coordinates of the start of the line. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738 variable.
\(x_1, y_2\)

specify the coordinates of the end of the line. Values can be numbers, numeric constants, or numeric variables.

\textit{color}

specifies the color of the line using a character string without quotation marks. For valid values, see the Annotate “COLOR Variable” on page 709. Use an asterisk (*) to specify the previous value of the \textit{color} parameter.

\textit{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 721 for the DRAW function.

\textit{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 728 for the DRAW function.

\%MAPLABEL Macro

Creates an output data set that can be used with the ANNO= option for PROC GMAP.

<table>
<thead>
<tr>
<th>Variables written out:</th>
<th>FUNCTION, STYLE, COLOR, SIZE, HSYS</th>
</tr>
</thead>
</table>

Requirement: You must run the \%ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

\textbf{Syntax}

\%MAPLABEL (map-dataset, attr-dataset, output-dataset, label-var, id-list, font=font_name, color=n, size=n, hsys=n);

\textbf{Parameters}

\textit{map-dataset}

the name of the map to be annotated.

\textit{attr-dataset}

the name of the data set containing the text to be shown on each ID value.

\textit{output-dataset}

the name of the Annotate data set created by the macro.

\textit{label-var}

the name of the label variable to place on the map (the text for annotate).

\textit{id-list}

the list of ID variables that you would issue in PROC GMAP to create the map. These values need to be on both the map and attribute data sets. If you also supply the SEGMENT variable, then every polygon will get a value. Without the SEGMENT variable, only one label per ID set will be shown over the collection of polygons. For example, Hawaii with SEGMENT gets a label on each island, whereas without SEGMENT, there is only one label centered on the entire set of islands.
**font**
specifies a font name for the STYLE variable.

**color**
specifies a value for the COLOR variable.

**size**
specifies a value for the SIZE variable. Defaults to 2.

**hsys**
specifies a value for the HSYS variable. Defaults to 3.

---

### %MOVE Macro

Moves to the \((x, y)\) coordinate.

- **Updates:** XLAST, YLAST
- **Variables written out:** FUNCTION, X, Y
- **Requirement:** You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

---

### Syntax

\%
MOVE \((x, y)\);

### Parameters

\(x, y\)
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 738.

---

### %PIEXY Macro

Calculates a point in relation to the latest pie slice.

- **Updates:** XLAST, YLAST
- **Variables written out:** ANGLE, FUNCTION, SIZE, X, Y
- **Requirement:** You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

---

### Syntax

\%
PIEXY \((angle, size)\);
Parameters

angle
specifies the angle used to calculate the point, relative to the center of the latest pie slice. The value can be a number, a numeric constant, or a numeric variable. For details, see the Annotate “ANGLE Variable” on page 706 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 728 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 692 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,

Requirement:
You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%POLY (x, y, color, style, line);
%POLY2(x, y, color, style, line, width);

Parameters

x, y
specify the starting point for a new polygon. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate or the names of the Annotate variables “X Variable” on page 738.

color
specifies the optional polygon fill color using a character string without quotation marks. For valid values, see the Annotate “COLOR Variable” on page 709. 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 764.

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 731 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 721 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 728 for the POLY function.

See Also
“POLY Function” on page 696

%POLYCONT Macro
Continues drawing the polygon to the next specified coordinates.

Variables written out: COLOR, FUNCTION, X, Y

Requirement: You must run the %ANOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax
%POLYCONT (x, y, color);

Parameters
x, y
specify the end point of the next line in the polygon. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

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 709. 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 763.

%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.
Updates: XLAST, YLAST, XLSTT, YLSTT
Variables written out: FUNCTION
Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax
%POP;

Comparisons
Use the %POP macro when you want to access the values of (XLAST on page 740, YLAST) and (XLSTT on page 741, YLSTT variables that you previously stored with the “%PUSH Macro” on page 765 or function.

%PUSH Macro
Enters the coordinates (XLAST, YLAST) and (XLSTT, YLSTT) in a LIFO system stack.

Updates: XLAST, YLAST, XLSTT, YLSTT
Variables written out: FUNCTION, internal coordinates
Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax
%PUSH;

Comparisons
The last-in, first-out (LIFO) stack provides a way to save previously calculated coordinates. It enables you to retain coordinate values for later use by utility functions without recalculating those values. In order to save coordinate values in the stack, you must explicitly push them onto the stack. See “Using the LIFO Stack” on page 654 for a description of the LIFO stack.

%RECT Macro
Draws a rectangle with diagonal corners at two specified points.

Updates: XLAST, YLAST
Variables written out: COLOR, FUNCTION, LINE, SIZE, X, Y
Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.
Syntax

%RECT (x1, y1, x2, y2, color, line, size)

Parameters

x1, y1
specify the coordinates of the first corner of the rectangle. Values can be numeric coordinates, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

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 709. 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 721 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 728 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 753.

%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

Requirement:
You must run the %ANNO MAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%SCALE (ptx, pty, x1, y1, x2, y2, vx1, vy1, vx2, vy2);
**Parameters**

\( ptx, pty \)

specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

\( x1, y1 \)

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 768.

The following example uses the \%SCALE macro to reduce \( x \) and \( y \) coordinates by 50%, as shown in Figure 30.1 on page 767.

\%SCALE\{x, y, 0, 0, 100, 100, 0, 0, 50, 50\};

**Figure 30.1 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

Requirement: You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%SCALET (ptx, pty, x1, y1, x2, y2, vx1, vy1, vx2, vy2);

Parameters

ptx, pty
specifies the coordinates to scale. Values can be numbers, numeric constants, or numeric variables. For details, see the Annotate “X Variable” on page 738.

x1, y1
specifies the minima of the original range. Values can be numbers, numeric constants, or numeric variables.

x1, y2
specifies the maxima of the original range. Values can be numbers, numeric constants, or numeric variables.

vx1, vy1
specifies the minima of the second range using numeric values. Values can be numbers, numeric constants, or numeric variables. These coordinates are also used as the origin against which the scaled point is plotted.

vx2, vy2
specifies the maxima of the second range. Values can be numbers, numeric constants, or numeric variables.

Details

The %SCALET macro reduces or enlarges Annotate graphics elements that use two-dimensional numeric coordinates. The %SCALET macro does not affect graphics elements that are drawn with text functions.

The difference between the %SCALET and %SCALE on page 766 macros is that the SCALET macro plots the new coordinates with respect to minima of the second range (vx1, vy1). The %SCALE macro plots the new coordinates with respect to the origin (0, 0).

The following example uses the %SCALET macro reduces x and y coordinates by 50% and plots the new coordinates with respect to (50, 0), as shown in Figure 30.2 on page 769:

%SCALET(x, y, 0, 0, 100, 100, 50, 0, 100, 50);
%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

Requirement: You must run the %ANOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%SEQUENCE (when);

Parameters

when

Values can be BEFORE or AFTER, as defined for the Annotate “WHEN Variable” on page 736.
%SLICE Macro

Draws a arc, pie slice, or circle, with available line types, colors, and fill types.

**Updates:** XLAST, YLAST

**Variables written out:** ANGLE, COLOR, FUNCTION, LINE, ROTATE, SIZE, STYLE, X, Y

**Requirement:** You must run the %ANNOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

**Syntax**

%SLICE (x, y, angle, rotate, size, color, style, line);

**Parameters**

*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 738.

*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 706 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 727 for the PIE function.

Interaction  
When LINE=WHOLE on page 770 is specified, the %SLICE macro always specifies ROTATE=360 for the PIE function in the Annotate data set.

*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 728.

*color*  
specifies the color of the arc outline and optional fill using a character string without quotation marks. See the Annotate COLOR variable on page 709 for valid values. 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 731 for the PIE function.

*line*  
specifies which lines of a pie slice are to be drawn. The value can be one of the following, which are shown in Figure 30.3 on page 771:

- **WHOLE**  
draws the entire 360 degree outside arc, starting at the specified angle, with no radius lines.
NONE draws the outside arc from the specified starting angle through the specified rotation with no radius lines.

LEAD draws the outside arc from the specified starting angle through the specified rotation and a radius line from the lead point to the center.

TRAIL draws the outside arc from the specified starting angle through the specified rotation and a radius line from the trail point to the center.

BOTH draws the outside arc from the starting angle through the specified rotation, a lead radius line, and a trail radius line.

The following figure illustrates each of the line values.

Figure 30.3 LINE Values Used with the %SLICE Macro

![Diagram of the line values]

In the Annotate data set, these values are converted to the PIE function LINE values shown in the following table.

<table>
<thead>
<tr>
<th>Macro %SLICE LINE Value</th>
<th>PIE Function LINE Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHOLE</td>
<td>0 *</td>
</tr>
<tr>
<td>NONE</td>
<td>0</td>
</tr>
<tr>
<td>LEAD</td>
<td>1</td>
</tr>
<tr>
<td>TRAIL</td>
<td>2</td>
</tr>
<tr>
<td>BOTH</td>
<td>3</td>
</tr>
</tbody>
</table>

* When LINE=WHOLE is specified, LINE=0 and ROTATE=360 are specified for the PIE function in the Annotate data set.

See LINE Variable on page 721 for the PIE function.

---

%SWAP Macro

Exchanges control between (XLAST, YLAST) and text (XLSTT, YLSTT) coordinates.

**Updates:** XLAST, YLAST, XLSTT, YLSTT

**Variables written out:** FUNCTION

**Requirement:** You must run the %ANNO macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.
%SYSTEM Macro

Defines the Annotate reference systems and the XSYS, YSYS, and HSYS variables.

Variables written out:

HSYS, XSYS, YSYS

Requirement:

You must run the %ANOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%SYSTEM (xsys, ysys, hsys);

Parameters

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 741. 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 “About the Annotate Functions” on page 667 for any restrictions that apply to the variable that you want to use.

The ZSYS variable on page 747 cannot be set through this macro. Use an explicit variable assignment instead:

zsys="value"; output;

See Coordinate Systems on page 646 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.

Updates: XLAST, YLAST

Variables written out:

FUNCTION

Requirement:

You must run the %ANOMAC macro before using this or any of the other SAS/GRAPH annotate macros. See “Making the Macros Available” on page 750.

Syntax

%TXT2CNTL;
Comparisons

Use the %TXT2CNTL macro when you want nontext functions to use the ending position of a text string as a starting or ending point.
Part 7

SAS/GRAPH Procedures

Chapter 31
SAS Mapping Procedures Documented in Another SAS Publication ........................................... 777

Chapter 32
GANNO Procedure ................................................................. 779

Chapter 33
GAREABAR Procedure ....................................................... 795

Chapter 34
GBARLINE Procedure .......................................................... 815

Chapter 35
GCHART Procedure .............................................................. 871

Chapter 36
GCONTOUR Procedure ........................................................ 1007

Chapter 37
GDEVICE Procedure ............................................................. 1045

Chapter 38
GFONT Procedure ............................................................... 1067

Chapter 39
GKPI Procedure ................................................................. 1099

Chapter 40
GOPTIONS Procedure ........................................................ 1125

Chapter 41
GPLOT Procedure ............................................................... 1133

Chapter 42
GRADAR Procedure ............................................................ 1243

Chapter 43
GREPLAY Procedure .......................................................... 1279
Chapter 44
  GSLIDE Procedure ............................................ 1333

Chapter 45
  GTILE Procedure ............................................. 1345

Chapter 46
  G3D Procedure .............................................. 1381

Chapter 47
  G3GRID Procedure ........................................... 1417
Chapter 31
SAS Mapping Procedures
Documented in Another SAS Publication

SAS Non-Mapping Procedures

The following mapping procedures are documented with related subject matter in SAS/GRAPH and Base SAS: Mapping Reference. The mapping procedures include GEOCODE, GINSIDE, GMAP, GPROJECT, GREduce, GREMOVE, and MAPIMPORT.

SAS Non-Mapping Procedures

Non-mapping SAS procedures are documented in this reference document.
Chapter 32
GANNO Procedure

Overview: GANNO Procedure
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 27, “Using Annotate Data Sets,” on page 635.

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 32.1 on page 780 displays output from an Annotate data set.
The program for this graph is in “Example 1: Scaling Data-Dependent Output” on page 784.

Syntax: GANNO Procedure

Restriction: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

Requirement: An input Annotate data set is required.

```
PROC GANNO ANNOTATE=Annotate-data-set
  <DATASYS>
  <DESCRIPTION='description'>
  <GOUT=<libref.;output-catalog>.
  <IMAGEMAP= output-data-set>
  <NAME= "name">;
```

PROC GANNO Statement

Identifies the Annotate data set and draws the graphics output defined by that data set. It can also scale the output to accommodate data-dependent coordinate values and specify an output catalog.

Syntax

```
PROC GANNO ANNOTATE=Annotate-data-set
  <DATASYS>
  <DESCRIPTION='description'>
  <GOUT=<libref.;output-catalog>.
  <IMAGEMAP= output-data-set>
  <NAME= "name">;
```

Summary of Optional Arguments

DATASYS
indicates that absolute or relative data-dependent coordinates occur in the 
Annotate data set and scales the coordinates to fit the graphics output area.

**DESCRIPTION='description'**

specifies a description of the output.

**GOUT=<libref>:output-catalog**

specifies the SAS catalog in which to save the graphics output produced by 
the GANNO procedure.

**IMAGEMAP= output-data-set**

creates a temporary SAS data set that is used to generate an image map.

**NAME="name"**

specifies the name of the GRSEG catalog entry and the name of the graphics 
output file, if one is created.

---

### Required Argument

**ANNOTATE=Annotate-data-set**

**ANNO=Annotate-data-set**

specifies a data set that includes Annotate variables that identify graphics commands 
and parameters.

See Chapter 27, “Using Annotate Data Sets,” on page 635

---

### Optional Arguments

Options in the GANNO statement affect all graphs produced by that statement. You can 
specify as many options as you want and list them in any order.

**DATASYS**

indicates that absolute or relative data-dependent coordinates occur in the Annotate 
data set and scales the coordinates to fit the graphics output area. Use the DATASYS 
option only with Annotate data sets in which the coordinate system variables XSYS, 
YSYS, and HSYS specify the values 1, 2, 7, or 8.

Use the DATASYS option when graphics elements that were created with data-
dependent variables do not fit in the graphics output area. This happens when the 
coordinate values generated by the data exceed a range of 0 to 100.

If you omit the DATASYS option, the GANNO procedure attempts to draw each 
graphics element according to the data values assigned to it, without scaling the 
values. If the range of data values is too large, some graphics elements will not be 
displayed.

See “Using the DATASYS Option to Scale Graphs” on page 783

Example “Example 1: Scaling Data-Dependent Output” on page 784

**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 189 for more information.

- the Table of Contents that is generated when you use the CONTENTS= option in 
an ODS HTML statement, assuming that the output is generated while the 
contents page is open.

- the description and the properties for the output in the Results window.
the description and properties for the catalog entry in the Explorer.

- the **Description** field of the PROC GREPLAY window.

**Alias** DES=

**Default** OUTPUT FROM PROC GANNO

**GOUT=<libref.>output-catalog**

specifies the SAS catalog in which to save the graphics output produced by the GANNO procedure. If you omit the libref, the SAS/GRAPH software looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See “Specifying the Catalog Name and Entry Name for Your GRSEGs” on page 120

Example “Example 2: Storing Annotate Graphics” on page 787

**IMAGEMAP= output-data-set**

creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192 and “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198

**NAME="name"**

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

**For the GRSEG entry name:**

- The name is truncated to eight characters.
- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

**For the graphics output filename:**

- The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
All characters are represented in lowercase.

If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.

If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.

If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default: GANNO

Examples

“Example 2: Storing Annotate Graphics” on page 787

“Example 3: Using the NAME= Option to Produce Multiple Graphs” on page 789

Details

Using the DATASYS Option to Scale Graphs

If your Annotate data set specifies a coordinate system that is based on data values (that is, XSYS, YSYS, and HSYS are assigned the values 1, 2, 7, or 8), the data values determine the size and location of the graphics elements on the output.

If the procedure that specifies the annotation generates axes (such as GPLOT or GCHART), by default the axes are scaled to accommodate the full range of data values and to fit in the procedure output area. Because all values are included in the axes, the graph displays all the Annotate output that is dependent on data values.

However, if the annotation displays with the GSLIDE or GANNO procedure, which do not generate axes, the data values might generate coordinate values that exceed the limits of the graphics output area.

In this case, you can use the DATASYS option to tell the procedure that the Annotate data set contains data-dependent coordinates and to scale the output accordingly. For an illustration of this process, see “Example 1: Scaling Data-Dependent Output” on page 784.

When you use the DATASYS option, the GANNO procedure reads the entire input data set before drawing the graph and creates an output environment that is data dependent. That is, the environment is based on the minimum and maximum values that are contained in the data set. It then scales the data to fit this environment so that all graphics elements can be drawn.

Although the DATASYS option enables you to generate graphs using one of the data-dependent coordinate systems, it requires that the procedure scan the entire data set to determine the minimum and maximum data values. You can save this extra pass of the data set by using data-dependent values only in procedures that generate axes. Annotate coordinate system 5 (percent of the procedure output area) is recommended for use with
the GANNO procedure. This coordinate system works equally well with the GSLIDE procedure if you decide to display the annotation with titles and footnotes.

Examples: GANNO Procedure

Example 1: Scaling Data-Dependent Output

Features: PROC GANNO statement options ANNOTATE= and DATASYS

Sample library member: GANSSCALE

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

This example uses an Annotate data set to scale data-dependent output with the DATASYS option and create a vertical bar chart of sales for each of six sites. The values that determine the height of each bar range from 137 to 999. The range of values is so large that the GANNO procedure cannot fit all of the bars in the output area without scaling the output. This program uses the DATASYS option to scale the data values so that the bars fit in the graphics output area.

Output 32.1  Scaled GANNO Output

![Projected Sales](image)

Program

```sas
goptions reset=all border;

data wrldtotl;
   length sitename $ 10;
   input sitename $ 1-10 mean 12-15;
   datalines;
   Paris 999
   Munich 271
   Tokyo 137
   London 546
   Frankfurt 273
   New York 931
```

Example 1: Scaling Data-Dependent Output

Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set WRLDTOTL. WRLDTOTL contains sales data for six sites. SITENAME contains the names of the sites. MEAN contains the average sales for each site.

data wrldtotl;
    length sitename $ 10;
    input sitename $ 1-10 mean 12-15;
    datalines;
Paris         999
Munich        571
Tokyo         137
London        273
Frankfurt      546
New York      991
;
run;

data wrldanno;
    length function color $ 8 text $ 20;
    retain line 0 xsys "2" hsys "3" x 8;
    set wrldtotl end=end;
    function="move"; x=x+8; y=20; output;
    function="bar"; y=y+(mean); x=x+9;
        style="empty"; color="red"; output;
    function="label"; y=0; x=x-4; size=3.5;
        position="E"; style="swiss";
        color="blue"; text=sitename; output;
    function="move"; y=y+(mean)-3; output;
    function="label"; x=x-1; text=left(put(mean,3.));
        position="5"; style="swiss"; size=3; output;
if end then do;
    function="move"; x=10; y=20; output;
    function="draw"; x=90; y=20; line=1;
        size=.5; color="blue"; output;
    function="label"; x=50; y=95; text="Projected Sales";
        xsys="3"; ysys="3"; position="5"; style="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;

proc ganno annotate=wrldanno
    datasys;
run;
quit;
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;
run;
```

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= option identifies the data set that contains the graphics commands. DATASYS tells the procedure to use the maximum and minimum data values to construct the output environment. In addition, the values of X and Y are scaled to fit the environment and all of the bars are displayed on the graph.

```
proc ganno annotate=wrldanno
```
Example 2: Storing Annotate Graphics

**Features:**
- PROC GANNO statement options DESCRIPTION=, GOUT=, and NAME=

**Sample library member:**
- GANSQUAR

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

This example creates an Annotate data set that draws four colored squares, displays the data set as a single graphics output, and stores the output as a catalog entry in a permanent catalog. In this example, the NAME= option specifies a text string that identifies the name that is stored with the graphics output in the catalog.

**Output 32.2**  Four Squares

Program

```sas
goptions reset=all border;
data squares;
length function style color $8 text $15;
xsys="3"; ysys="3";
  color="green";
  function="move"; x=10; y=65; output;
  function="bar"; x=30; y=95; style="solid"; output;
  function="label"; x=10; y=63; position="6";
      style="swissb"; size=2; text="Green"; output;
  color="red";
  function="move"; x=60; y=65; output;
```

function="bar"; x=80; y=95; output;
function="label"; x=60; y=63; position="6";
    style="swissb"; size=2; text="Red"; output;

color="blue";
function="move"; x=10; y=15; output;
function="bar"; x=80; y=45; output;
function="label"; x=10; y=12; position="6";
    style="swissb"; size=2; text="Blue"; output;

color="gray";
function="move"; x=60; y=15; output;
function="bar"; x=80; y=45; output;
function="label"; x=60; y=12; position="6";
    style="swissb"; size=2; text="Gray"; output;
x=88; y=5; position="5"; size=1.5; style="swiss";
    text="GANSQUAR"; output;

function="frame"; color="red"; when="b";
    style="empty"; output;
run;

proc ganno annotate=squares
    gout=excat
    name="GANSQUAR"
    description="Four squares"
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the Annotate data set, SQUARES. XSYS and YSYS specify coordinate system
3 (percent of the graphics output area) for X and Y.

data squares;
    length function style color $ 8 text $ 15;
    xsys="3"; ysys="3";

Draw the first square. The COLOR variable assigns the color for the square. The
FUNCTION variable selects the operation to be performed by the Annotate facility. The
X and Y variables contain coordinate values. The BAR function draws the square. When
the STYLE variable is used with the BAR function, it selects the fill pattern for the bar.

    color="green";
    function="move"; x=10; y=65; output;
    function="bar"; x=30; y=95; style="solid"; output;

Label the first square. The LABEL function creates the label. The POSITION value of
6 left-justifies the text with respect to X and Y. The TEXT variable specifies the text
string to be written.

    function="label"; x=10; y=63; position="6";
    style="swissb"; size=2; text="Green"; output;
Draw and label the second square.

```
color="red";
function="move"; x=60; y=65; output;
function="bar"; x=80; y=95; output;
function="label"; x=60; y=63; position="6";
    style="swissb"; size=2; text="Red"; output;
```

Draw and label the third square.

```
color="blue";
function="move"; x=10; y=15; output;
function="bar"; x=30; y=45; output;
function="label"; x=10; y=12; position="6";
    style="swissb"; size=2; text="Blue"; output;
```

Draw and label the fourth square.

```
color="gray";
function="move"; x=60; y=15; output;
function="bar"; x=80; y=45; output;
function="label"; x=60; y=12; position="6";
    style="swissb"; size=2; text="Gray"; output;
```

Add a footnote.

```
x=88; y=5; position="5"; size=1.5; style="swiss";
    text="GANSQUAR"; output;
```

Draw a red frame.

```
function="frame"; color="red"; when="b";
    style="empty"; output;
run;
```

Display the annotate graphics. GOUT= assigns the catalog in which the graphics output is stored. NAME= assigns a name to the entry stored in the WORK.EXCAT catalog. DESCRIPTION= assigns a description to the catalog entry.

```
proc ganno annotate=squares
gout=excat
    name="GANSQUAR"
    description="Four squares"
run;
quit;
```
In this example, the GANNO procedure uses the NAME= option to generate multiple
graphs from one Annotate data set. Since NAME= is assigned the variable COLOR, the
GANNO procedure generates separate graphics output for each value of the COLOR, as
shown in Output 32.3 on page 790, Output 32.4 on page 790, Output 32.5 on page 791 and Output 32.6 on page 791.

Each output is stored as a separate entry in the temporary output catalog
WORK.EXCAT. The entries are named according to the values of COLOR: BLUE, GRAY, GREEN, and RED. Note that the output for GRAY includes the footnote shown in
“Example 2: Storing Annotate Graphics” on page 787. The output for RED shows the
frame that is generated by the Annotate data set. The black borders in the other outputs
are not generated by the code.

Output 32.3  Output for COLOR Value BLUE (WORK.EXCAT.BLUE.GRSEG)

Output 32.4  Output for COLOR Value GRAY (WORK.EXCAT.GRAY.GRSEG)
**Output 32.5** Output for COLOR Value GREEN (WORK.EXCAT.GREEN.GRSEG)

![Green Color Example](image)

**Output 32.6** Output for COLOR Value RED (WORK.EXCAT.RED.GRSEG)

![Red Color Example](image)

**Program**

```sas
goptions reset=all border;

data squares;
  length function style color $ text $ 15;
  xsys="3"; ysys="3";
  color="green";
  function="move"; x=10; y=65; output;
  function="bar"; x=30; y=95; style="solid"; output;
  function="label"; x=10; y=63; position="6";
    style="swissb"; size=2; text="Green"; output;
  color="red";
  function="move"; x=60; y=65; output;
  function="bar"; x=80; y=95; output;
```

Example 3: Using the NAME= Option to Produce Multiple Graphs
function="label"; x=60; y=63; position="6";
   style="swissb"; size=2; text="Red"; output;

color="blue";
function="move"; x=10; y=15; output;
function="bar"; x=30; y=45; output;
function="label"; x=10; y=12; position="6";
   style="swissb"; size=2; text="Blue"; output;

color="gray";
function="move"; x=60; y=15; output;
function="bar"; x=80; y=45; output;
function="label"; x=60; y=12; position="6";
   style="swissb"; size=2; text="Gray"; output;
x=88; y=5; position="5"; size=1.5; style="swiss";
text="GANSQUAR"; output;
function="frame"; color="red"; when="b";
   style="empty"; output;
run;

proc ganno annotate=squares
   name=color
   gout=excat
   description="Individual squares";
run;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the Annotate data set, SQUARES. XSYS and YSYS specify coordinate system
3 (percent of the graphics output area) for X and Y.

data squares;
   length function style color $ 8 text $ 15;
   xsys="3"; ysys="3";

Draw the first square. The COLOR variable assigns the color for the square. The
FUNCTION variable selects the operation to be performed by the Annotate facility. The
X and Y variables contain coordinate values. The BAR function draws the square. When
the STYLE variable is used with the BAR function, it selects the fill pattern for the bar.

color="green";
function="move"; x=10; y=65; output;
function="bar"; x=30; y=95; style="solid"; output;

Label the first square. The LABEL function creates the label. The POSITION value of
6 left-justifies the text with respect to X and Y. The TEXT variable specifies the text
string to be written.

function="label"; x=10; y=63; position="6";
   style="swissb"; size=2; text="Green"; output;

Draw and label the second square.

color="red";
Draw and label the third square.

```plaintext
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.

```plaintext
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.

```plaintext
x=88; y=5; position="5"; size=1.5; style="swiss";
   text="GANSQUAR"; output;
```

Draw a red frame.

```plaintext
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.

```plaintext
proc ganno annotate=squares
   name=color
   gout=excacat
   description="Individual squares";
run;
```
Overview: GAREABAR Procedure

The GAREABAR procedure produces an area bar chart displaying two statistics for each category of data. For example, in the following chart, for each bar, the width, and the height of each bar represent different values, proportionally. The chart creates one bar for each unique value of the SITE variable. The height of each bar represents the SUM of the sales for that SITE. The width of each bar represents the number of salespersons generating revenue for that site.
The GAREABAR procedure produces a chart based on the values of a chart variable, a width variable, and a sum calculation variable specified by the SUMVAR= option. The chart variable can be either character or numeric. All values of the chart variable are treated as discrete. The chart values are displayed in data order. PROC GAREABAR does not calculate a midpoint.

For the VBAR statement, the width variable defines the width of the bar along the horizontal axis. The SUMVAR= variable determines the height of the bar on the vertical axis.

For the HBAR statement, the width variable defines the width of each bar on the vertical axis. The SUMVAR= variable determines the length of the bar on the horizontal axis.

The width variable, the SUMVAR= option variable, and the SUBGROUP= option variable can be calculated, and displayed as a percentage of the total or as a sum. The default is sum.

An example using the SUBGROUP option is shown in “Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages” on page 810.

Syntax: GAREABAR Procedure

Restrictions: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.
The only devices supported for the GAREABAR procedure are ACTIVEX and ACTXIMG. These devices are recommended for use with the HTML and RTF destinations only.

**Requirements:**
- You must specify either the ACTIVEX or ACTXIMG device in the GOPTION statement.
- At least one HBAR, HBAR3D, VBAR, or VBAR3D statement is required.

**Global statements:**
- FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

**Supports:**
- Run-group processing

**Note:**
The procedure can include the SAS/GRAPH statements BY on page 370 and NOTE on page 447, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

**Tips:**
- When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See “About Filename Indexing” on page 119. To reset the current index value and overwrite output files that already exist, use the following statement:

```plaintext
ods graphics / reset=index;
```

---

**PROC GAREABAR Statement**

Identifies the data set containing the chart variables.

**Requirement:**
- An input data set is required.

**Syntax**

```plaintext
PROC GAREABAR <DATA=\text{input-data-set};>
   HBAR | HBAR3D | VBAR | VBAR3D \text{chart-variable}*\text{width-variable} / 
   \text{SUMVAR=}\text{numeric-variable}<\text{options}>
```

---

**Optional Argument**

PROC GAREABAR statement options affect all graphs produced by the procedure.

**DATA=\text{input-data-set};**

specifies the SAS data set that contains the variable(s) to chart. By default the procedure uses the most recently created SAS data set.
HBAR, HBAR3D, VBAR, and VBAR3D Statement

Create horizontal or vertical bar charts in which the length or height of the bar represents the value of a chart statistic for each category of data. A second statistic is represented by the width of each bar.

**Requirement:** You must specify at least one chart variable, one width variable, and the SUMVAR= option.

**Global statements:** FOOTNOTE, GOPTIONS, LEGEND, PATTERN, TITLE

**Syntax**

```
HBAR | HBAR3D | VBAR | VBAR3D chart-variable*width-variable / SUMVAR=numeric-variable <options>;
```

**Summary of Optional Arguments**

**Appearance options**

- `CFRAME=background-color` specifies a background color for the graph.
- `COUTLINE=bar-outline-color | SAME` outlines all of the bars with the color specified.
- `CTEXT=text-color` specifies a color for all text on the chart.
- `FRAME | NOFRAME` specifies whether the two-dimensional axis area frame or the three-dimensional backplane is drawn.
- `LEGEND=LEGEND<1...99>` assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.
- `NOLEGEND` suppresses the legend that is automatically generated by the SUBGROUP= option.

**Catalog entry description options**

- `DESCRIPTION="description"` specifies the description of the plot.
- `NAME="name"` specifies the name of the graphics output file created.

**Midpoint options**

- `CONTINUOUS` specifies that the graph data be treated as continuous.
- `DISCRETE` treats the chart variable axis data as discrete data.
- `SUBGROUP=subgroup-variable` divides the bars into segments according to the values of the subgroup-variable column.
Statistic options

\[ \text{WIDTHSTAT=} \text{FIRST | LAST | SUM | PCT | PERCENT} \]

specifies the statistic to be applied to the width-variable.

**Required Arguments**

*chart-variable*

specifies the variable that defines the categories of data to chart. This variable can be either character or numeric. Each unique value of the chart variable results in a separate bar.

*width-variable*

specifies the variable that defines the width of each bar. The width-variable is always numeric. The width of each bar represents the sum of the width-variable values for that category. The default statistic is sum.

*SUMVAR=numeric-variable*

specifies the numeric variable used for statistical calculations. The resulting statistics are represented by the height of each vertical bar or length of each horizontal bar. The default statistic is sum.

**Optional Arguments**

The options in an HBAR, HBAR3D, VBAR, and VBAR3D statement affect all bars that are produced by the statement. You can specify as many options as you want and list them in any order.

*CFRAME=background-color*

specifies a background color for the graph. The specified color must be a valid SAS/GRAPH color name.

<table>
<thead>
<tr>
<th>Alias</th>
<th>CFR=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style reference</td>
<td>Color attribute of the GraphBackground element</td>
</tr>
<tr>
<td>Restriction</td>
<td>Not supported by Java</td>
</tr>
</tbody>
</table>

**CONTINUOUS**

specifies that the graph data be treated as continuous. Continuous data can take any of an infinite number of values between whole numbers, so it might not be measured accurately. The default is discrete.

| Restriction | Not supported by Java |

*COUTLINE=bar-outline-color | SAME*

outlines all of the bars with the color specified. SAME specifies that the outline color of a bar is the same as the interior pattern color. The default color is the outline color of the current style.

<table>
<thead>
<tr>
<th>Alias</th>
<th>CO=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style reference</td>
<td>Color attribute of the GraphOutlines element</td>
</tr>
<tr>
<td>Restriction</td>
<td>Not supported by Java</td>
</tr>
</tbody>
</table>

*CTEXT=text-color*

specifies a color for all text on the chart. The GAREABAR procedure looks for the text color in the following order:
1. colors specified for labels and values on assigned LEGEND statements, which override the CTEXT= option specified in the GAREABAR statement
2. the color specified by the CTEXT= option in the GAREABAR statement
3. the color specified by the CTEXT= option in a GOPTIONS statement
4. the color specified in the current style

**Alias**
CT=

**Style reference**
Color attribute of the GraphLabelText and GraphValueText elements

**Restriction**
Not supported by Java

**DESCRIPTION=** "description"

specifies the description of the plot. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is displayed as follows:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open
- the description and the properties for the output in the Results window
- the description and properties for the catalog entry in the SAS Explorer
- the Description field of the PROC GREPLAY window

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on statements such as TITLE, FOOTNOTE, and NOTE. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

**Alias**
DES=

**Default**
GAREABAR of chart-variable

**Restriction**
Not supported by Java

**DISCRETE**

treats the chart variable axis data as discrete data. Discrete data is characterized as data in which the variable can take only one of a finite set of values. The GAREABAR procedure creates a separate bar for each unique value of the chart variable. If the chart variable has a format associated with it, each formatted value is treated as a unique value. The default is discrete.

**Restriction**
Not supported by Java

**FRAME | NOFRAME**

specifies whether the two-dimensional axis area frame or the three-dimensional backplane is drawn. The default is FRAME, which draws a frame around the axis areas (in two-dimensional bar charts) or generates a colored three-dimensional backplane (in three-dimensional bar charts). For three-dimensional charts,
NOFRAME removes the backplane color, and leaves the backplane grid, the vertical axis and plane, and the horizontal axis and plane.

The NOFRAME option overrides the CFRAME= option.

**Alias** FR | NOFR

**Restriction** Not supported by Java

**LEGEND=**LEGEND<1...99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does not generate a legend.

LEGEND= is ignored if any of the following are true:

- The SUBGROUP= option is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.

**Restrictions** The LEGEND statement options are partially supported by ActiveX.

Not supported by Java

**See** “LEGEND Statement” on page 377

**NAME=”name”**
specifies the name of the graphics output file created.

The following applies to name:

- **For the graphics output filename:**
  - All characters are represented in lowercase.
  - The ACTXIMG device must be used in order to generate a graphics output file.
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.
  - The name can include special characters.
  - Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique filename for the output. See “About Filename Indexing” on page 119.

**Default** graph.png

**Restriction** Not supported by Java

**NOLEGEND**
suppresses the legend that is automatically generated by the SUBGROUP= option. The NOLEGEND option is ignored if the SUBGROUP= option is not used.

**Restriction** Not supported by Java

**SUBGROUP=**subgroup-variable
divides the bars into segments according to the values of the subgroup-variable column. The subgroup-variable column can be either character or numeric, and is
always treated as a discrete variable. The SUBGROUP= option creates a separate segment within each bar for each unique value of the subgroup variable.

**Restriction**
Not supported by Java

**WIDTHSTAT= FIRST | LAST | SUM | PCT | PERCENT**
specifies the statistic to be applied to the width-variable. Values for WIDTHSTAT= are as follows:

- **FIRST** specifies that the width of each bar is determined solely by the value of the first observation of the width-variable in the data for that bar.

- **LAST** specifies that the width of each bar is determined solely by the value of the last observation of the width-variable in the data for that bar.

- **SUM** specifies that the width of each bar is the sum of the values of the width-variable for that bar.

- **PCT** | **PERCENT** specifies that the width of each bar is the sum of the values of the width-variable for that bar. The width is expressed as a percentage of the sum of the values of the width-variable for all of the bars.

**Alias**
WSTAT=

**Default**
SUM

**Restriction**
Not supported by Java

**Details**
The HBAR, HBAR3D, VBAR, and VBAR3D statements specify the variables that define the categories, and width of each bar. The SUMVAR= option variable calculates the length or height of each bar. These statements do the following;

- calculate the chart statistic for each bar (the default is SUM)
- scale the response axes and the bars according to the statistic value
- calculate the width of each bar, based on the value of the width variable
- draw a frame around the axis area using a color determined by the current style

You can use statement options to change the type of chart, to display specific statistics, and to modify the appearance of the chart. You can also specify an additional variable to subgroup your data, which divides the bars into segments and displays a legend to identify the segments.

In addition, you can make the following changes with global statements:

- use the LEGEND statement to modify the legend
- use the TITLE and FOOTNOTE statements to add titles and footnotes to the chart
- use the PATTERN statement to create PATTERN definitions that define the color and type of area fill for patterns used in graphs.
Examples: GAREABAR Procedure

Example 1: Generating an Area Bar Chart

Features: VBAR Statement
Sample library member: GABSUMVR

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

This area bar chart reveals three geographic sites (Lima, NY, Rome) along the horizontal axis. The width of each bar represents the sum of the salespersons assigned to each site. The height of each bar represents the sum of the sales for each site. The chart shows that NY had the greatest sales, as well as the greatest number of salespersons.

Program

filename odsout ".*;

data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
  datalines;
  Lima 1 4043.97 4
  NY 1 8225.26 12
  Rome 1 3543.97 6
  Lima 2 3723.44 5
  NY 2 8595.07 18
Rome  2 5558.29   10
Lima  3 4437.96    8
NY    3 9847.91   24
Rome 3 6789.85   14
Lima  4 6065.57   10
NY    4 11388.51  26
Rome 4 8509.08   16
ods _all_ close;
goptions reset=all dev=actximg;
ods html path=odsout file="areabarchartsum.html";
proc gareabar data=totals;
  vbar site*salespersons /
    sumvar=sales;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

Program Description

Create a file reference for the ODS output The current working directory is specified in this example.
filename odsout ".";

Create the WORK.TOTALS data set.

data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
datalines;
Lima  1 4043.97    4
NY    1 8225.26   12
Rome 1 3543.97    6
Lima  2 3723.44    5
NY    2 8595.07   18
Rome 2 5558.29   10
Lima  3 4437.96    8
NY    3 9847.91   24
Rome 3 6789.85   14
Lima 4 6065.57   10
NY    4 11388.51  26
Rome 4 8509.08   16
;

Close the currently open ODS destinations.
ods _all_ close;

Set the graphics environment. Specify the ACTXIMG device.
goptions reset=all dev=actximg;
Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```
ods html path=odsout file="areabarchartsum.html";
```

Run PROC GAREABAR with VBAR statement. The VBAR statement creates a vertical bar for each value of SITE. The SALESPERSONS variable sets the width of the bars. The SUMVAR=SALES option controls the height of each of the bars.

```
proc gareabar data=totals;
  vbar site*salespersons /
    sumvar=sales;
run;
quit;
```

Close ODS HTML.

```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

```
ods html; /* Not required in SAS Studio */
```

### Example 2: Generating an Area Bar Chart with a Numeric Chart Variable

**Features:**
- VBAR statement options
  - SUMVAR=
  - WSTAT=

**Sample library member:**
- GABNUMVR

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

This chart displays a numeric chart variable, QUARTER, representing the four quarters of an unspecified year. The GAREABAR procedure treats all values of a numeric chart variables as discrete, unless the CONTINUOUS option is used. GAREABAR does not calculate midpoints.

The total sales for each quarter of the year are represented by the height of each bar along the vertical axis. The width of each bar along the horizontal axis indicates the percentage of salespersons during each quarter. The chart shows the correlation between the number of salespersons, and the total sales.
Program

filename odsout ".";

data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
  datalines;
  Lima 1 4043.97 4
  NY 1 8225.26 12
  Rome 1 3543.97 6
  Lima 2 3723.44 5
  NY 2 8595.07 18
  Rome 2 5558.29 10
  Lima 3 4437.96 8
  NY 3 9847.91 24
  Rome 3 6789.85 14
  Lima 4 6065.57 10
  NY 4 11388.51 26
  Rome 4 8509.08 16
;
  ods _all_ close;
  goptions reset=all dev=actximg;
  ods html path=odsout file="areabarchartnum.html";
  proc gareabar data=totals;
    vbar quarter*salespersons/
      sumvar=sales
      wstat=pct;
  run;
  quit;
  ods html close;

  ods html; /* Not required in SAS Studio */
**Program Description**

**Create a file reference for the ODS output** The current working directory is specified in this example.

```plaintext
   filename odsout ".";
```

**Create the WORK.TOTALS data set.**

```plaintext
data totals;
   input Site $ Quarter Sales Salespersons;
   format Sales dollar12.2;
   datalines;
   Lima   1  4043.97   4
   NY     1  8225.26   12
   Rome   1  3543.97   6
   Lima   2  3723.44   5
   NY     2  8595.07   18
   Rome   2  5558.29  10
   Lima   3  4437.96   8
   NY     3  9847.91  24
   Rome   3  6789.85  14
   Lima   4  6065.57  10
   NY     4 11388.51  26
   Rome   4  8509.08  16
;
```

**Close the currently open ODS destinations.**

```plaintext
   ods _all_ close;
```

**Set the graphics environment.** Specify the ACTXIMG device.

```plaintext
   goptions reset=all dev=actximg;
```

**Open the ODS HTML destination.** The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```plaintext
   ods html path=odsout file="areabarchartnum.html";
```

**Run PROC GAREABAR with VBAR statement.** The VBAR statement creates a vertical bar for each value of QUARTER. The SALESPERSONS variable sets the width of the bars. The SUMVAR=SALES option controls the height of each of the bars. The WSTAT=PCT option displays the width statistic as a percentage (the number of salespersons as a percentage of the whole).

```plaintext
   proc gareabar data=totals;
      vbar quarter*salespersons/
         sumvar=sales
         wstat=pct;
   run;
   quit;
```

**Close ODS HTML.**

```plaintext
   ods html close;
```
Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).

```sas
ods html; /* Not required in SAS Studio */
```

**Example 3: Generating an Area Bar Chart with Subgroups**

**Features:**
- HBAR statement options
  - `SUBGROUP=`
  - `SUMVAR=`
  - `RSTAT=`
  - `WSTAT=`

**Sample library member:** GABSUBGR

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

This example uses the `SUBGROUP=` option to display the same statistics as displayed by Examples 1 and 2. Similar to Example 1, this example shows the total sales for each of the three geographic sites. The relative thickness of each bar represents the number of salespersons at each site.

The addition of subgroups to this chart shows the relative percentage of sales for each quarter. This chart demonstrates that all of the sites had most of their sales posted in the fourth quarter.

---

**Program**

```sas
filename odsout ".";

data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
```
Example 3: Generating an Area Bar Chart with Subgroups

```sas
proc gareabar data=totals;
  hbar site*salespersons /
    sumvar=sales
    subgroup=quarter
    wstat=PCT;
run;
quit;
```

**Program Description**

Create a file reference for the ODS output The current working directory is specified in this example.

```sas
filename odsout ".";
```

Create the WORK.TOTALS data set.

```sas
data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
  datalines;
  Lima 1 4043.97  4
  NY    1 8225.26 12
  Rome 1 3543.97  6
  Lima 2 3723.44  5
  NY    2 8595.07 18
  Rome 2 5558.29 10
  Lima 3 4437.96  8
  NY    3 9847.91 24
  Rome 3 6789.85 14
  Lima 4 6065.57 10
  NY    4 11388.51 26
  Rome 4  8509.08 16
;```

ods _all_ close;

goptions reset=all dev=actximg;

ods html path=odsout file="areabarchartsub.html";

proc gareabar data=totals;
  hbar site*salespersons /
    sumvar=sales
    subgroup=quarter
    wstat=PCT;
run;
quit;
```

ods html close;

ods html; /* Not required in SAS Studio */
Close the currently open ODS destinations.
ods _all_ close;

Set the graphics environment. Specify the ACTXIMG device.
goptions reset=all dev=actximg;

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.
ods html path=odsout file="areabarchartsub.html";

Run PROC GAREABAR with an HBAR statement. The HBAR statement creates a horizontal bar for each value of SITE. The SALESPERSONS variable sets the width of each bar. The SUMVAR=SALES option sets the length of each bar. The WSTAT=PERCENT option displays the width statistic as a percentage, which enables you to compare the distribution of salespersons for each quarter. The SUBGROUP=QUARTER option and the RSTAT=SUM option are reflected in the statistics that are displayed as absolute numbers along the horizontal bar.

proc gareabar data=totals;
  hbar site*salespersons /
       sumvar=sales
       subgroup=quarter
       wstat=PCT;
  run;
quit;

Close ODS HTML.
ods html close;

Open ODS HTML. Open an ODS destination for subsequent programs (not required in SAS Studio).
ods html; /* Not required in SAS Studio */

Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT= option to Calculate Statistics as Percentages

Features:
  HBAR statement options
    SUBGROUP=
    RSTAT=
    WSTAT=

Sample library member:
  GABWSTAT

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

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.

Example 4: Area Bar Chart with Subgroups; Using the `RSTAT=` option and the `WSTAT=` option to Calculate Statistics as Percentages

Program

```
filename odsout ".";
data totals;
input Site $ Quarter Sales Salespersons;
format Sales dollar12.2;
datalines;
Lima  1  4041.97   4
NY    1  8225.26  12
Rome  1  3543.97   6
Lima  2  3723.44   5
NY    2  8595.07  18
Rome  2  5558.29  10
Lima  3  4437.96   8
NY    3  9847.91  24
Rome  3  6789.85  14
Lima  4  6065.57  10
NY    4 11388.51  26
Rome  4  8509.08  16
;
ods _all_ close;
goptions reset=all dev=actximg;
ods html path=odsout file="areabarchartwstat.html";
proc gareabar data=totals;
   hbar site*salespersons /
      sumvar=sales
      subgroup=quarter
      rstat=PCT
      wstat=PCT;
run;
```
Program Description

Create a file reference for the ODS output The current working directory is specified in this example.

```sas
filename odsout ".";
```

Create the WORK.TOTALS data set.

```sas
data totals;
  input Site $ Quarter Sales Salespersons;
  format Sales dollar12.2;
  datalines;
  Lima 1 4043.97 4
  NY 1 8225.26 12
  Rome 1 3543.97 6
  Lima 2 3723.44 5
  NY 2 8595.07 18
  Rome 2 5558.29 10
  Lima 3 4437.96 8
  NY 3 9847.91 24
  Rome 3 6789.85 14
  Lima 4 6065.57 10
  NY 4 11388.51 26
  Rome 4 8509.08 16
;```

Close the currently open ODS destinations.

```sas
ods _all_ close;
```

Set the graphics environment. Specify the ACTXIMG device.

```sas
goptions reset=all dev=actximg;
```

Open the ODS HTML destination. The PATH= option specifies the file reference for the ODS output that was created previously. The FILE= option specifies a name for the HTML output file.

```sas
ods html path=odsout file="areabarchartwstat.html";
```

Run PROC GAREABAR with an HBAR statement. The HBAR statement creates a horizontal bar for each value of SITE. The SALESPERSONS variable sets the width of each bar. The SUMVAR=SALES option sets the length of each bar. The WSTAT=PERCENT option displays the width statistic as a percentage. (The percentage of salespersons is shown by the relative thickness of each bar along the vertical axis.) The SUBGROUP=QUARTER option and the RSTAT=PCT option, request that the sales for each quarter are displayed as a percentage on the horizontal axis.

```sas
proc gareabar data=totals;
  hbar site*salespersons /
    sumvar=sales
    subgroup=quarter
```

```
Chapter 33 • GAREABAR Procedure
```
Example 4: Area Bar Chart with Subgroups; Using the RSTAT= option and the WSTAT=
option to Calculate Statistics as Percentages

```
rstat=PCT
  wstat=PCT;
  run;
  quit;
```

Close ODS HTML.
```
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs (not required in
SAS Studio).
```
ods html; /* Not required in SAS Studio */
```
Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs

Overview: GBARLINE Procedure

About the GBARLINE Procedure

The GBARLINE procedure produces bar-line charts. Bar-line charts are vertical bar charts with one or more plot overlays. These charts graphically represent the value of a statistic calculated for one or more variables in an input SAS data set. The charted variables can be either numeric or character.

The procedure calculates these statistics:

- sum
- mean
- frequency or cumulative frequency
- percentage or cumulative percentage

Use the GBARLINE procedure to do the following tasks:

- display and compare exact and relative magnitudes
- examine the contribution of parts to the whole
- analyze where data are out of balance
- display a long series of data, and show trends and patterns

In conjunction with the SYMBOL statement, the GBARLINE procedure can produce needle plot overlays, and overlay plots with stepped interpolation.

Note: PROC GBARLINE is not supported by Java.

Chart Terminology

Bar-line charts display the magnitude of data with bars, each of which represents a category of data (midpoint). The height of the bars represents the value of the bar statistic for the corresponding midpoint.

Figure 34.1 on page 817 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.
About Bar-Line Charts

The GBARLINE procedure produces a bar chart based on the values of a chart variable and an optional response variable (SUMVAR= option). The computed statistic can be set with the TYPE= option. Each line plot uses the same chart variable and has an optional response variable (SUMVAR= option). A computed statistic can be set with the TYPE= option.

Figure 34.2 on page 818 illustrates the parts of a bar-line chart.
Bar-line charts have three axes:

- a midpoint axis that shows the categories of data, based on the chart variable
- a left response axis that displays the scale of values for the bar statistic (based on the response variable, if specified)
- a right response axis that displays the scale of values for the line statistic (based on the response variable, if specified)

The response axes are divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks. Each axis is labeled with the variable name or label. The right response axis is scaled to accommodate all the line variable response values when multiple PLOT statements are present.
About the Chart Variable

The *chart variable* is the variable in the input data set whose value determines the categories of data represented by the bar and lines. The chart variable generates the midpoints to which each observation in the data set contributes.

A character chart variable is always discrete.

About Midpoints

*Definition: Midpoints*

*Midpoints* are the values of the chart variable that identify categories of data. By default, midpoints are selected or calculated by the procedure. The way the procedure handles the midpoints depends on whether the values of the chart variable are character, discrete numeric, or continuous numeric.

*Character Values*

A character chart variable generates a midpoint for each unique value of the variable. In the following example, the chart variable CITY contains the names of three different cities, and each city is a midpoint, resulting in three midpoints for the chart:

*Figure 34.3  Character Midpoints*

![Character Midpoints Diagram](image)

By default, character midpoints are arranged in alphabetic order. If a character variable has an associated format, then the values are arranged in order of the formatted values.

*Continuous Numeric Values*

A continuous numeric variable generates midpoints that represent ranges of values. By default, the GBARLINE procedure determines the number of uniform ranges (LEVELS), calculates the number of observations in each range, and then computes the $\text{TYPE}=$ 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 34.4 Continuous Numeric Midpoints

By default, midpoints of ranges are arranged in ascending order.

**Selecting and Ordering Midpoints**

For character or discrete numeric values, you can use the MIDPOINTS= option to rearrange the midpoints or to exclude midpoints from the chart. For example, to change the default alphabetic order of the midpoints in Figure 34.3 on page 819, specify the following midpoints:

```plaintext
midpoints="Tokyo" "Denver" "Seattle"
```

To exclude the midpoint for Denver, specify the following midpoints:

```plaintext
midpoints="Tokyo" "Seattle"
```

In this case, values excluded by the option are not included in the calculation of the chart statistic.

You can order or select discrete numeric midpoint values just as you do character values, but you omit the quotation marks when specifying numeric values.

For continuous numeric variables, use the LEVELS= or MIDPOINTS= option to change the number of midpoints. These options can also control the range of values each midpoint represents or change the order of the midpoints. To control the range of values each midpoint represents, use the MIDPOINTS= option to specify the midpoint value of each range. For example, to select the ranges 20–29, 30–39, and 40–49, specify the following values:

```plaintext
midpoints=25 35 45;
```

Alternatively, to select the number of midpoints that you want and let the procedure calculate the ranges and midpoints, use the LEVELS= option.

You can also use formats to control the ranges of continuous numeric variables, but in that case the values are no longer continuous but become discrete.

*Note:* You cannot use the MIDPOINTS= option to exclude continuous numeric values from the chart. This is because values below or above the ranges specified by the option are automatically included in the first and last midpoints. To exclude continuous numeric values from a chart, use a WHERE statement in a DATA step or the WHERE= data set option.
About Response Variables

Response variables can be specified for either the bar chart or any line plot with the SUMVAR= option.

For example:

```
BAR age / DISCRETE SUMVAR=weight; PLOT / SUMVAR=height;
```

When you specify a response variable, the only statistics available are SUM or MEAN. SUM is the default. To change the statistic, you specify the TYPE= option (for example, TYPE=MEAN).

If you do not specify a response variable, a summary statistic for the chart variable is computed. By default it is FREQ (frequency). You can use the TYPE= option to indicate another statistic: PERCENT, CFREQ (cumulative frequency) or CPERCENT (cumulative percent).

For more information about these statistics, see “About Chart Statistics” on page 821. See also the PLOT statement option descriptions of “SUMVAR=plot-variable” on page 858 and “TYPE=statistic” on page 858.

About Chart Statistics

**Definition: Chart Statistics**

The chart statistics are the statistical values calculated for the chart variable or the response variable. When there is no response variable, the GBARLINE procedure calculates one of four possible statistics with the default being FREQ. When there is a response variable, one of two possible statistics is computed with the default being SUM. You can specify the chart statistic with the TYPE= option for both the bar chart and any line plot. For the bar chart, the default statistic is frequency. For the plot variable, the default statistic is sum.

The examples given in the descriptions of these statistics in the following topics assume a data set with two variables, CITY and SALES. The values of CITY are Denver, Seattle, and Tokyo. There are 21 observations: seven for Denver, nine for Seattle, and five for Tokyo.

**Frequency**

The frequency statistic is the total number of observations in the data set for each midpoint. For example, seven observations of the bar variable, CITY, contain the value Denver, so the frequency for the Denver midpoint is 7.
Cumulative Frequency

The cumulative frequency statistic adds the frequency for the current midpoint to the frequency of all of the preceding midpoints. For example, the frequency for the Denver midpoint is 7, and the frequency for the next midpoint, Seattle, is 9. Therefore, the cumulative frequency for Seattle is 16 and the cumulative frequency for Tokyo is 21.

Percentage

The percentage statistic is calculated by dividing the frequency for each midpoint by the total frequency count for all midpoints in the chart or group and multiplying it by 100. For example, the frequency count for the Denver midpoint is 7 and the total frequency count for the chart is 21, so the percentage statistic for Denver is 33.3%.

Cumulative Percentage

The cumulative percentage statistic adds the percentage for the current midpoint to the percentage for all of the preceding midpoints in the chart or group. For example, the percentage for the Denver midpoint is 33.3, and the percentage for the next midpoint, Seattle, is 42.9, so the cumulative percentage for Seattle is 76.2.

Sum

The sum statistic is the total of the values, for each midpoint, for the variable specified by the SUMVAR= option. For example, if you specify SUMVAR=SALES and the values of the SALES variable for the seven Denver observations are 8734, 982, 1504, 3207, 4502, 624, and 918, the sum statistic for the Denver midpoint is 20,471.

You must use the SUMVAR= option to specify the variable for which you want the sum statistic.

Mean

The mean statistic is the average of the values, for each midpoint, for the variable specified by the SUMVAR= option. For example, if TYPE=MEAN and SUMVAR=SALES, the mean statistic for the Denver midpoint is 2924.42.

You must use the SUMVAR= option to specify the variable for which you want the mean statistic.

Calculating Weighted Statistics

By default, each observation is counted only once in the calculation of a chart statistic. To calculate weighted statistics in which an observation can be counted more than once, use the FREQ= option. This option identifies a variable whose values are used as a multiplier for the observation in the calculation of the statistic. If the value of the FREQ= variable is missing, zero, or negative, then the observation is excluded from the calculation.

If you use the SUMVAR= option, then the SUMVAR= variable value for an observation is multiplied by the FREQ= variable value for the observation. The product of this calculation determines the chart statistic.
For example, to use a variable called COUNT to produce weighted statistics, assign FREQ=COUNT. If you also assign the variable HEIGHT to the SUMVAR= option, then the following table shows how the values of COUNT and HEIGHT would affect the statistic calculation:

<table>
<thead>
<tr>
<th>Value of COUNT</th>
<th>Value of HEIGHT</th>
<th>Number of times the observation is used</th>
<th>Value used for HEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>5</td>
<td>325</td>
</tr>
<tr>
<td>.</td>
<td>63</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>-3</td>
<td>60</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

By default, the percentage and cumulative percentage statistics are calculated based on the frequency. Suppose you want to graph a percentage or cumulative percentage based on a sum. You can use the FREQ= option to specify a variable to use for the sum calculation and then specify PCT as the statistic, as shown in this example:

freq=COUNT type=pct;

Because the variable that is specified by the FREQ= option determines the number of times an observation is counted, the value of COUNT is the equivalent of the sum statistic.

See also the descriptions of the options “TYPE=statistic” on page 847, “SUMVAR=numeric-variable” on page 846, and “FREQ=numeric-variable” on page 838.

Note: The FREQ= option is not supported by ActiveX or Java.

---

**Missing Values**

By default, the GBARLINE procedure ignores missing midpoint values for the chart variable. If you specify the MISSING option, then missing values are treated as a valid midpoint and are included on the axis. Missing values for the subgroup variables are always treated as valid subgroups.

When the value of the variable that is specified in the FREQ= option is missing, zero, or negative, the observation is excluded from the calculation of the chart statistic.

When the value of the variable specified in the SUMVAR= option is missing, the observation is excluded from the calculation of the chart statistic.

If all of the values for a response variable are missing for the bar chart, a midpoint is drawn, but no bar appears above it. For a line plot, no marker is drawn and the line connects the adjacent markers.
Plot Variable Values Out of Range

Exclude data values from a plot overlay by restricting the range of axis values with the RAXIS= options or with the ORDER= option in an AXIS statement. When an observation contains a value outside of the specified axis range, the GBARLINE procedure excludes the observation from the plot and issues a message to the log.

If you specify interpolation with a SYMBOL definition, then the values outside the axis range are excluded from interpolation calculations by default. As a result, this can change interpolated values for the plot overlay.

To specify that values outside of the axis range are included in the interpolation calculations, use the MODE= option in a SYMBOL statement. When MODE=INCLUDE, values that fall outside of the axis range are included in interpolation calculations but are excluded from the plot. The default (MODE=EXCLUDE) omits observations that are outside of the axis range from interpolation calculations. See the “SYMBOL Statement” on page 412 for details.

Controlling Patterns, Outlines, Colors, and Images

How to Control Patterns, Outlines, Colors, and Images

Default patterns, colors, outlines, and, in some cases, images, are defined by the current style, whether that style is the default GSTYLE or one that you specify with the ODS statement. You can turn off styles by specifying the NOGSTYLE system option. Alternatively, you can override individual aspects of a graph's appearance by specifying PATTERN statements, SYMBOL statements, graphics options, and procedure options.

The following sections summarize pattern behavior for the GBARLINE procedure. For more information, see the “PATTERN Statement” on page 398 and the “SYMBOL Statement” on page 412.

Default Patterns, Symbols, Lines, Colors, and Outlines

The default pattern that the GBARLINE procedure uses is a solid fill. The default colors are determined by the current style and the device.

By default, the system option GSTYLE is in effect. As a result, the procedure generates output using the style's default bar fill colors, plot line colors, widths, symbols, patterns, and outline colors. Specifically, the GBARLINE procedure uses the default values when you do not specify any of the following:

- *any* PATTERN statements
- the CPATTERNS= graphics options
- the COLORS= graphics options
- the COUTLINE= option in the BAR statement
- *any* SYMBOLS statements
If you do not specify any of these statements or options, then the GBARLINE procedure performs the following operations:

- selects the first default fill pattern, which is always solid, and rotates it through the list of colors available in the current style, generating one solid pattern for each color. When the solid patterns are exhausted, the procedure selects the next default subgroup bar pattern (empty) and rotates it through the appropriate set of colors. It continues in this fashion until all of the required patterns have been assigned.

If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, then the procedure uses all the colors in the list to generate the patterns.

- uses the style's outline color to outline every patterned area.

- uses the style's default symbol for the initial PLOT statement points. The procedure uses the second default symbol for the next PLOT statement, and the third default symbol for the next PLOT statement. The procedure continues orderly through the set of symbols belonging to that style until all the PLOT statements have been satisfied.

- connects all the plot symbols with a solid line.

If you specify the NOGSTYLE system option, the fill pattern is solid and the color comes from the device's color list. The GBARLINE procedure uses a solid fill for the bars that it rotates once through the device's default color list, skipping the foreground color. (Typically, the foreground color is the first color in the device's color list.) Suppose no SYMBOL or PATTERN statements are in effect and the COLORS= option is not used in the GOPTIONS statement. In this case the plot line colors begin with the next color from the same color list used to color the bars. By doing this, the procedure prevents the plot line from being the same color as a bar fill. Specifically, GBARLINE performs the following operations:

- selects the first default fill, which is always solid, and rotates it through the color list, generating one solid pattern for each color. If the first color in the device's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.

- uses the foreground color to outline every patterned area.

- selects the next default pattern fill (if it needs additional patterns), and rotates that pattern through the color list, skipping the foreground color as before. The procedure continues in this fashion until it has generated enough patterns for the chart.

- uses the device's default color to outline every patterned area.

- selects the next color in the list after the last bar color and uses it to draw the first PLOT statement symbol and connecting line.

- rotates through the color list for any subsequent PLOT statements.

If the procedure needs additional patterns, PROC GBARLINE selects the next default pattern fill (empty) and rotates it through the color list, skipping the foreground color as before. The procedure continues in this fashion until it has generated enough patterns for the chart.

You might change or override the default behavior for patterns if you specify a color list with the COLORS= option in a GOPTIONS statement. For example, if the COLORS= list contains more than one color, then the procedure rotates the default solid pattern through that list. It uses every color, even if the foreground color is black (or white). The default outline color remains the foreground color or the color specified by the current style.
For a description of the COLORS= or other graphics options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

**User-Defined Patterns, Colors, Lines, Symbols, and Outlines**

To override the default patterns and select fills and colors for the bars, use the PATTERN statement. Only solid and empty bar patterns are valid; all other pattern fills are ignored. For a complete description of all bar patterns, see “VALUE=bar/block-pattern” on page 400.

When you use PATTERN statements, the procedure uses the specified patterns until all of the PATTERN definitions that they generate have been used. Then, if more patterns are required, the procedure returns to the default pattern rotation. To change the outline color of any pattern, whether the pattern is default or user-defined, use the COUTLINE= option in the BAR statement that generates the chart. (See “COUTLINE=bar-outline-color | SAME” on page 834.) 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 412. 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 336.

You can use the IBACK= graphics option to specify image files that fill the background area. For additional information, including a list of recognized image file types, see “Image File Types Supported by SAS/GRAPH” on page 331 and “Displaying an Image in a Graph Background” on page 333.

**Controlling When Bar Patterns Change**

The PATTERNID= option controls when the pattern changes. By default, all of the bars are the same pattern. If you specify PATTERNID=MIDPOINT, then the pattern changes every time the midpoint value changes.

Instead of changing the pattern for each midpoint, you can change the pattern for each BY group by changing the value of the PATTERNID= option. See “PATTERNID=BY | MIDPOINT | SUBGROUP” on page 844 for details.

**Controlling Axis Color**

By default, axis elements use the first color in the color list or the colors that are specified by AXIS statement color options. However, BAR statement options can also control the color of the axis lines, text, and frame.

<table>
<thead>
<tr>
<th>Table 34.2 Controlling Axis Color</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Axis Element.</strong></td>
</tr>
<tr>
<td>the axis text</td>
</tr>
</tbody>
</table>
Axis Element. | Specify Color with this Option
---|---
the axis lines | CAXIS=
the area within the frame | CFRAME=

Syntax: GBARLINE Procedure

**Restrictions:** This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135. Not supported by JAVA and JAVAIMG

**Requirement:** One BAR statement

**Global statements:** AXIS , FOOTNOTE, GOPTIONS, LEGEND, PATTERN, SYMBOL, TITLE

**Note:** The procedure can include the SAS/GRA PH statement BY on page 370, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRA PH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

```sas
PROC GBARLINE DATA=input-data-set
    <ANNOTATE=Annotate-data-set>
    <IMAGEMAP=output-data-set>
    <UNIFORMAXES>
    BAR bar-variable / option(s);
    <PLOT -1 / option(s)> <PLOT-2 / option(s)> ...;
```

**PROC GBARLINE Statement**

Identifies the data set containing the chart and response variables. Can specify an annotate data set.

**Restriction:** Not supported by JAVA and JAVAIMG

**Requirement:** An input data set is required.

**Syntax**

```sas
PROC GBARLINE DATA=input-data-set
    <ANNOTATE=Annotate-data-set>
    <IMAGEMAP=output-data-set>
    <UNIFORMAXES>;
```
Required Argument

**DATA=** *input-data-set*

specifies the SAS data set that contains the variable or variables to chart. If not specified, an input data set must have been created before issuing the PROC GBARLINE statement.

Default: The GBARLINE procedure uses the most recently created SAS data set.

See

- “The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*
- “About Data Set Options” in *SAS Data Set Options: Reference*
- “About the Chart Variable” on page 819

Optional Arguments

PROC GBARLINE statement options affect all graphs produced by the procedure.

**ANNOTATE=** *Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GBARLINE procedure. To annotate individual graphs, use the ANNOTATE= option in the BAR statement.

Alias: ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635

**IMAGEMAP=** *output-data-set*

creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See

- “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
- “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198

**UNIFORMAXES**

generates identical scales automatically on both the bar and plot axes. Use this option to perform axes scaling automatically, instead of generating the data values on the axes with the ORDER= option in the AXIS statement.

Interaction: The UNIFORMAXES option overrides an ORDER= option when both are specified.

Notes: The UNIFORMAXES option might automatically scale the bar axis beyond the range specified by the FORMAT statement.

When one axis range is much larger than the other, the resulting bars or plots in the graph appear flattened.

Invoking different statistical functions between the bar and plot statements when using this option might yield unexpected results. For example, specifying a cumulative percentage in a BAR statement and a
cumulative frequency in a PLOT statement, in conjunction with UNIFORMAXES, might generate an abnormal graph.

BAR Statement

Creates vertical bar charts in which the height of the bars represents the value of the bar statistic for each category of data.

Restriction: Not supported by JAVA and JAVA IMG

Requirement: One bar variable is required.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE

Supports: Drill-down functionality

Syntax

BAR chart-variable [/ option(s)];

Summary of Optional Arguments

Appearance options

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GBARLINE procedure.

CAUTOREF=reference-line-color
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option.

CAXIS=axis-color
specifies a color for the response and midpoint axis lines and for the default axis area frame.

CERROR=error-bar-color
specifies the color of error bars.

CFRAME=background-color
specifies the color with which to fill the axis area.

COUTLINE=bar-outline-color | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color.

CREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies colors for reference lines.

CTEXT=text-color
specifies a color for all text on the axes and legend.

FRAME | NOFRAME
specifies whether the axis area frame is drawn.

LAUTOREF=reference-line-type
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option.

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the plot part of the graph.
LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)  
specifies line types for reference lines.

NOLEGEND  
suppresses the legend generated by the LEGEND= option.

PATTERNID=BY | MIDPOINT | SUBGROUP  
specifies how fill patterns are assigned.

SPACE=bar-spacing  
specifies the amount of space between individual bars along the midpoint axis.

WAUTOREF=reference-line-width  
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option.

WIDTH=bar-width  
specifies the width of the bars.

WOUTLINE=bar-outline-width  
specifies the width of the bar outline in pixels.

WREF=reference-line-width | (reference-line-width) | (reference-line-width-list)  
specifies line widths for reference lines.

Axes options

ASCENDING  
arranges the bars in ascending order of the value of the bar statistic.

AUTOREF  
draws a reference line at each major tick mark on the bar (left) response axis.

AXIS=AXIS<1 …99>  
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

CLIPREF  
clips the reference lines at the bars.

DESCENDING  
arranges the bars in descending order of the value of the chart statistic.

MAXIS=AXIS<1 …99>  
assigns the specified AXIS definition to the midpoint axis.

MINOR=number-of-minor-ticks  
specifies the number of minor tick marks between each major tick mark on the bar response axis.

NOAXIS  
suppresses the left BAR response axis and displays the midpoint and right PLOT axes.

NOBASEREF  
suppresses the zero reference line when the SUM or MEAN bar statistic has negative values.

NOZERO  
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

RANGE  
displays on the axis of the chart the range of numeric values represented by each bar.

RAXIS=value-list | AXIS<1 …99>  
AXIS=value-list | AXIS<1 …99>
specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis.

\texttt{REF=value | (value) | (value-list)}

draws reference lines at the specified points on the chart response axis.

**Catalog entry description options**

\texttt{DESCRIPTION="description"}

specifies a description of the output.

\texttt{NAME="name"}

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

**Midpoint options**

\texttt{DISCRETE}

treats a numeric chart variable as a discrete variable rather than as a continuous variable.

\texttt{LEVELS=number-of-midpoints | ALL}

specifies the number of midpoints to be graphed for the chart variable.

\texttt{MIDPOINTS=OLD}

generates default midpoints

\texttt{MIDPOINTS=value-list}

specifies the midpoint values for the bars

\texttt{MISSING}

accepts a missing value as a valid midpoint for the chart variable.

\texttt{SUBGROUP=subgroup-variable}

divides the bars into segments according to the values of \texttt{subgroup-variable}.

**ODS options**

\texttt{HTML_LEGEND=variable}

identifies the variable in the input data set whose values create links or data tips or both.

\texttt{HTML=variable}

identifies the variable in the input data set whose values create links or data tips or both.

\texttt{URL=character-variable}

specifies a character variable whose values are URLs.

**Statistic options**

\texttt{CFREQ}

displays the cumulative frequency statistic above the bars.

\texttt{CLM=confidence-level}

specifies the confidence intervals to use when drawing error bars.

\texttt{CNEWCENT}

displays the cumulative percentage statistic above the bars.

\texttt{ERRORBAR=BARS | BOTH | TOP}

draws confidence intervals dependent on the statistic type.

\texttt{FREQ}

displays the frequency statistic above the bars.

\texttt{FREQ=numeric-variable}

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.
INSIDE=statistic
displays the values of the specified statistic inside the bars.

MEAN
displays the mean statistic above the bars.

OUTSIDE=statistic
displays the values of the specified statistic above the bars.

PERCENT
displays the percentages of observations having a given value for the bar variable above the bars.

STATFMT=format-specification
overrides the GBARLINE default format of the displayed statistical value.

SUM
displays the sum statistic above the bars.

SUMVAR=numeric-variable
specifies a numeric variable for sum or mean calculations.

TYPE=statistic
specifies the chart statistic.

Required Argument

chart-variable
specifies the variable that defines the categories of data to chart. The variable must be in the input data set.

See “About the Chart Variable” on page 819

Optional Arguments

Options in the BAR statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313. For details about specifying images, see Chapter 23, “Adding Images to SAS/GRAPH Output,” on page 331. For a complete description of the graphics options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GBARLINE procedure. To annotate individual graphs, use the ANNOTATE= option in the BAR statement.

Alias ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635

ASCENDING
arranges the bars in ascending order of the value of the bar statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. ASCENDING reorders the bars from shortest to longest. The ordering is left to right.

ASCENDING overrides any midpoint order specified in the MIDPOINTS= option or specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

AUTOREF
draws a reference line at each major tick mark on the bar (left) response axis. To draw reference lines at specific points on the response axis, use the REF= option.
By default, reference lines are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.

**AXIS=AXIS<1 ...99>**
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

See “AXIS Statement” on page 345

**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 LATOREF= 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.

displays the cumulative frequency statistic above the bars. A maximum of two statistics can be printed if the INSIDE= option is also used.

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction The CFREQ option is ignored if the FREQ option is specified.
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.

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.

CREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. The CAUTOREF= option overrides the CREF= reference line color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color only to the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (color1 color2 ...colorN) or separated by commas (color1, color2 ..., colorN). If you do not specify the CREF= option, the GBARLINE
procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE system option is specified. To specify line types for these reference lines, use the LREF= option.

Alias CR=

Style reference LineStyle attribute of the GraphGridLines element

**CTEXT=text-color**
specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GBARLINE procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the BAR statement.
2. the color specified by the CTEXT= option in the BAR statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, if the NOGSTYLE system option is specified, then black for the ActiveX device and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, then the CTEXT= color overrides the AXIS statement's COLOR= specification. This means that the CTEXT= color is used for axis labels and values. The AXIS statement's COLOR= color is still used for all other axis elements, such as tick marks.

Alias CT=

**Note** If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

**DESCENDING**
arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. DESCENDING reorders the bars from longest to shortest. The ordering is left to right.

**Interaction** DESCENDING overrides any midpoint order that is specified with the MIDPOINTS= option or that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

**DESCRIPTION=description**
specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
• the description and the properties for the output in the Results window.
• the description and properties for the catalog entry in the SAS Explorer.
• the Description field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default BAR AND LINE PLOT OF chart-variable

DISCRETE

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GBARLINE procedure creates a separate midpoint and, hence, a separate bar for each unique value of the chart variable. If the chart variable has a format associated with it, then each formatted value is treated as a midpoint.

Interactions

The LEVELS=number-of-midpoints option is ignored when you use the DISCRETE option.

The MIDPOINTS= option overrides the DISCRETE option.

The ORDER= option in an AXIS statement that is assigned to the midpoint axis can rearrange or exclude discrete midpoint values.

ERRORBAR=BARS | BOTH | TOP

draws confidence intervals for either of the following:

• the mean of the SUMVAR= variable for each midpoint if you specify TYPE=MEAN

• the percentage of observations assigned to each midpoint if you specify TYPE=PCT with no SUMVAR= option.

The ERRORBAR= option cannot be used with values of the TYPE= option other than MEAN or PCT. Valid values for ERRORBAR= are as follows:

BARS
draws error bars as bars half the width of the main bars.

BOTH
draws error bars as two ticks joined by a line (default).

TOP
draws the error bar as a tick for the upper confidence limit that is joined to the top of the bar by a line.

By default, ERRORBAR= uses a confidence level of 95%. You can specify different confidence levels with the CLM= option.

When you use ERRORBAR= with TYPE=PCT, the confidence interval is based on a normal approximation. Let TOTAL be the total number of observations, and PCT be the percentage assigned to a given midpoint. The standard error of the percentage is approximated as follows:

\[ \text{APSTDERR} = 100 \times \sqrt{\left(\frac{\text{PCT}/100}{\text{TOTAL}}\right) \times \left(1-\left(\frac{\text{PCT}/100}{\text{TOTAL}}\right)\right)}; \]
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:

\[ UCLP = PCT + APSTDERR \times PROBIT\left( 1-(1-LEVEL/100)/2 \right) \]

The lower confidence limit for the percentage is computed as follows:

\[ LCLP = PCT - APSTDERR \times PROBIT\left( 1-(1-LEVEL/100)/2 \right) \]

When you use ERRORBAR= with TYPE=MEAN, the sum variable must have at least two nonmissing values for each midpoint. Let N be the number of observations assigned to a midpoint. Let, MEAN be the mean of those observations and STD be the standard deviation of the observations. The standard error of the mean is computed as follows:

\[ STDERR = STD / \sqrt{N}; \]

Let LEVEL be the confidence level specified using the CLM= option, with a default value of 95. The upper confidence limit for the mean is computed as follows:

\[ UCLM = MEAN + STDERR \times TINV\left( 1-(1-LEVEL/100)/2, N-1 \right) \]

The lower confidence limit for the mean is computed as follows:

\[ LCLM = MEAN - STDERR \times TINV\left( 1-(1-LEVEL/100)/2, N-1 \right) \]

Suppose you want the error bars to represent a given number, C, of standard errors instead of a confidence interval, and the number of observations assigned to each midpoint is the same. You can then find the appropriate value for the CLM= option by running a DATA step. For example, suppose you want error bars that represent one standard error (C=1) with a sample size of N. You can run the following DATA step to compute the appropriate value for the CLM= option and assign that value to a macro variable &LEVEL:

```plaintext
data null; c = 1; n = 10;
level = 100 * (1 - 2 * (1 - probt( c, n-1)));
put all;
call symput("level",put(level,best12.));
run;
```

Then, when you run the GBARLINE procedure, you can specify CLM=&LEVEL.

Note that this method does not work precisely if different midpoints have different numbers of observations. However, choosing an average value for N can yield sufficiently accurate results for graphical purposes if the sample sizes are large or do not vary much.

### FRAME | NOFRAME

specifies whether the axis area frame is drawn. The default is FRAME, which draws a frame around the axis area. Specifying NOFRAME removes the axis area frame, including any background color or image. To remove one or more axis elements, use either the AXIS statement or the NOAXIS option.

The color of the frame or backplane outline is the color of the midpoint axis, which is determined by the CAXIS= option.

<table>
<thead>
<tr>
<th>Alias</th>
<th>FR</th>
<th>NOFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>The NOFRAME option overrides the CFRA... graphics option.</td>
<td></td>
</tr>
</tbody>
</table>

### FREQ

displays the frequency statistic above the bars. Non-integer values are rounded down to the nearest integer. A maximum of two statistics can be printed using the
INSIDE= option for the second. This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction
This option overrides the CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

See
“About Chart Statistics” on page 821
“Displaying Statistics in Bar-Line Charts” on page 849

FREQ=numeric-variable
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that is specified by the value of numeric-variable for that observation. If the value of numeric-variable is missing, zero, or negative, then the observation is not used in the statistic calculation. Non-integer values of numeric-variable are truncated to integers. The FREQ= option is valid with all chart statistics.

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are affected by applying a format to numeric-variable.

Restriction
Not supported by Java and ActiveX

See
“Calculating Weighted Statistics” on page 822

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction
Not supported by Java

See
“Data Tips for Web Presentations” on page 191
“Adding Links and Enhancements with the URL=, HTML=, and HTMLLEGEND= Options” on page 192

Example
“Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 865

HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction
Not supported by ActiveX
INSIDE=statistic

displays the values of the specified statistic inside the bars. Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM
- MEAN

Interaction To display statistics with INSIDE=SUM or INSIDE=MEAN, you must also specify the SUMVAR= option.

See “About Chart Statistics” on page 821

“Displaying Statistics in Bar-Line Charts” on page 849

LAUTOREF=reference-line-type

specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style. If the NOGSTYLE system option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CAUTOREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LEGEND=LEGEND<1 ...99>

assigns the specified LEGEND definition to the plot part of the graph. The LEGEND= option is ignored if the specified LEGEND definition is not in effect. When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option creates a legend by default. In this case, only one bar is represented in the legend.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable: bar city / subgroup=city;

You can specify both a BAR and PLOT legend on the same graph. If you specify the same location with the LEGEND= option in the BAR statement and the POSITION= option in the LEGEND statement, a single combined legend is drawn. However, ActiveX output displays separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options. See the “LEGEND Statement” on page 377 for more information.

Restriction Partially supported by ActiveX

See “SUBGROUP=subgroup-variable” on page 846

“LEGEND Statement” on page 377
**LEVELS=number-of-midpoints | ALL**

specifies the number of midpoints to be graphed for the chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. If your data contains a large number of unique midpoint values (more than 200), then you can use the XPIXELS and YPIXELS GOPTIONS. This enables the device driver to render a larger (and more readable) graph. The LEVELS=number-of-midpoints option is ignored if any of the following is true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

**LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)**

specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=reference-line-type for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and is delimited by spaces (type1 type2 ...typeN). If you do not specify the LREF= option, the GBARLINE procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. If the NOGSTYLE system option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

**MAXIS=AXIS<1 ...99>**

assigns the specified AXIS definition to the midpoint axis. The MAXIS= option is ignored if the specified AXIS definition does not exist.

See  “AXIS Statement” on page 345

“About Midpoints” on page 819

**MEAN**

displays the mean statistic above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

**Restriction**  This option is ignored if the bars are too narrow to avoid overlapping values.

**Interactions**  This option is ignored if the FREQ, CFREQ, PERCENT, CPERCENT, or SUM option is specified.

The MEAN option is ignored unless you also use the SUMVAR= option.
MIDPOINTS=\textit{value-list}  
 specifies the midpoint values for the bars. The way you specify \textit{value-list} depends on the type of the bar variable.

- For numeric chart variables, \textit{value-list} is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - \texttt{n < ...n>}
  - \texttt{n TO n <BY increment>}
  - \texttt{n< ...n> TO n <BY increment> <n < ...n> >}

If a numeric chart variable has an associated format, the specified values must be the \textit{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 \textit{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 \textit{value-list} specifies a unique numeric value.

- For character bar variables, \textit{value-list} is a list of unique character values enclosed in quotation marks and separated by blanks:
  - \texttt{'value-1' < ...'value-n>'}

If a character variable has an associated format, the specified values must be the \textit{formatted} values.

You are able to specify up to 256 characters for each variable value.

For a complete description of \textit{value-list}, see “ORDER=(\textit{value-list})” on page 353 in the AXIS statement.

If the \textit{value-list} for either type of variable specifies so many midpoints that the axis values overwrite each other, then the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, this problem can be corrected by adjusting the size of the text with the HTEXT= graphics option. Or correct the problem by increasing the number of cells that are displayed in your graphics. Do this by using the HPOS= and VPOS= graphics options.

The ORDER= option in the AXIS statement overrides the order specified in the MIDPOINTS= option. The BAR statement options ASCENDING and DESCENDING also override both the MIDPOINTS= and ORDER= options in the AXIS statement.

\textbf{Note} Any character bar variable value with a long text string has the potential to cause a scaling issue and might produce a wider bar chart than desired.
See “About Midpoints” on page 819

**MIDPOINTS=OLD**

**Restriction**
The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

**MINOR=number-of-minor-ticks**
specifies the number of minor tick marks between each major tick mark on the bar response axis.

**Interaction**
The MINOR= option in a bar chart statement overrides the MINOR= option in an AXIS definition assigned to the response axis with the RAXIS= option.

**MISSING**
accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored.

**NAME="name"**
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name1*, *name2*, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name,
that index number is appended to the output filename. See “About Filename Indexing” on page 119.

**Note:** Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

**Default** GBARLIN

**NOAXIS** suppresses the left BAR response axis and displays the midpoint and right PLOT axes. The axis lines, axis labels, axis values, and all major and minor tick marks are suppressed on the left axis. If you specify an axis definition with the MAXIS= or RAXIS= options, then the axes are generated as defined in the AXIS statement. However, all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH=, and OFFSET= are used.

To remove only selected axis elements such as lines, values, or labels, use specific AXIS statement options. 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 CFRAMES= option. To remove the axis frame, use the NOFRAME option in the procedure.

**NOBASEREF** suppresses the zero reference line when the SUM or MEAN bar statistic has negative values.

**NOLEGEND** suppresses the legend generated by the LEGEND= option.

**NOZERO** suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

**Note** There might be occasions when you assign bar label names to each bar with the VALUE= option in an AXIS statement and a bar is omitted from your graph. In this case the label names might be inadvertently shifted and assigned to the wrong bar.

**OUTSIDE=statistic** displays the values of the specified statistic above the bars. Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- SUM
- MEAN
Interaction To display statistics with OUTSIDE=SUM or OUTSIDE=MEAN, you must also specify the SUMVAR= option. A second statistic can be displayed by also using the INSIDE= option.

See “About Chart Statistics” on page 821
“Displaying Statistics in Bar-Line Charts” on page 849

PATTERNID=BY | MIDPOINT | SUBGROUP specifies how fill patterns are assigned. By default, all of the bars are the same color. Values for PATTERNID= are as follows:

BY changes patterns each time the value of the BY variable changes. All bars use the same pattern if the GBARLINE procedure does not include a BY statement.

MIDPOINT changes patterns every time the midpoint value changes.

SUBGROUP changes patterns every time the value of the subgroup variable changes. The bars must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all bars have the same pattern.

PERCENT displays the percentages of observations having a given value for the bar variable above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

Alias PCT

Restriction This option is ignored if the bars are too narrow to avoid overlapping values.

Interaction The PERCENT option is ignored if the FREQ or CFREQ option is specified.

See “About Chart Statistics” on page 821
“Displaying Statistics in Bar-Line Charts” on page 849

RANGE displays on the axis of the chart the range of numeric values represented by each bar. In the graphics output:

- The starting value of each range is indicated with the less-than symbol (<).
- The ending value is indicated with the greater-than-or-equal-to symbol (>=).
- The default midpoint axis label is “chart-variable RANGE.”

The RANGE option has no effect on axes that represent character data. By default, the values shown on the axis are determined by the value of the “MIDPOINTS=value-list” on page 841.

Interaction If specified, the option “DISCRETE” on page 836 overrides the RANGE option.

During the creation of vertical bar charts, when options RANGE and LEVELS= are both used, RANGE displays the endpoints of the range for each vertical bar. It does this rather than displaying just the
numerical midpoint of the range as would be displayed with the LEVELS= option alone. If necessary, RANGE automatically angles or rotates the values and they might be less readable.

If the RANGE, LEVELS=, and AXIS VALUE= options are all specified, the RANGE option is ignored and the LEVELS= option displays the numeric midpoint of each vertical bar.

**RAXIS=**<br>
**AXIS=value-list | AXIS<1 …99>**<br>

specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis. See “**MIDPOINTS=value-list**” on page 841 for a description of value-list. By default, the GBARLINE procedure scales the response axis automatically and provides an appropriate number of tick marks. The left response axis applies to the BAR statement when a PLOT statement is used. Otherwise, both the left and right axes apply to the BAR statement.

You can specify negative values, but negative values are reasonable only when TYPE=SUM or TYPE=MEAN and one or more of the sums or means are less than zero. Frequency and percentage values are never less than zero.

For lists of values, a separate major tick mark is created for each individual value. A warning message is written to the SAS log if the values are not evenly spaced.

If the values represented by the bars are larger than the highest tick mark value, then the bars are truncated at the highest tick mark.

See “**AXIS Statement**” on page 345

**REF=value | (value) | (value-list)**<br>

draws reference lines at the specified points on the chart response axis.

See “**MIDPOINTS=value-list**” on page 841 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**<br>
specifies the amount of space between individual bars along the midpoint axis. Bar-spacing can be any nonnegative number, including decimal values. Units are character cells. By default, the GBARLINE procedure calculates spacing based on the size of the axis area and the width of the bars on the chart. Use SPACE=0 to leave no space between adjacent bars.

Restriction The SPACE= option is ignored if its value results in a chart that is too large to fit in the space available for the midpoint axis. As a result, a warning message is issued in the log.

**STATFMT=format-specification**<br>
overrides the GBARLINE default format of the displayed statistical value. The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.
If you specify an option of TYPE=mean, an INSIDE=pct option, and an OUTSIDE=sum option, a STATFMT=f8.1 option applies only to the calculated TYPE=mean statistical value. For example:

```plaintext
proc gbarline;
  bar mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
    statefmt=f8.1;
run;
```

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the STATFMT= option.

If you change the previous example to specify an OUTSIDE=mean option, then STATFMT=f8.1 applies to the OUTSIDE=mean option and the TYPE=mean option. The statistical types match.

The STATFMT= option does not control the format of the response axis tick marks.

**Alias** SFMT=, SFORMAT=, STATFORMAT=

**SUBGROUP=** subgroup-variable

divides the bars into segments according to the values of subgroup-variable. Subgroup-variable can be either character or numeric and is always treated as a discrete variable. The SUBGROUP= option creates a separate segment within each bar for every unique value of the subgroup variable for that midpoint.

When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option automatically creates a legend by default. In this case, only one bar is represented in the legend. To assign a LEGEND definition, use the LEGEND= option.

See “LEGEND Statement” on page 377

Example “Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 865

**SUM**

displays the sum statistic above the bars. A maximum of two statistics can be printed using the INSIDE= option for the second.

**Restriction** This option is ignored if the bars are too narrow to avoid overlapping values.

**Interactions** This option is ignored if the FREQ, CFREQ, PERCENT, or CPERCENT option is specified.

The SUM option is ignored unless you also use the SUMVAR= option.

See “About Chart Statistics” on page 821

“Displaying Statistics in Bar-Line Charts” on page 849

**SUMVAR=** numeric-variable

specifies a numeric variable for sum or mean calculations. The GBARLINE procedure calculates the sum or, if requested, the mean of numeric-variable for each midpoint. The resulting statistics are represented by the length of the bars along the response axis, and they are displayed at major tick marks.
When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Example  “Example 1: Producing a Basic Bar-Line Chart” on page 861

**TYPE=statistic**
specifies the chart statistic.

- If the SUMVAR= option is not used, statistic can be one of the following:
  - FREQ
    - frequency (default)
  - CFREQ
    - cumulative frequency
  - PERCENT
    - percentage
  - CPERCENT
    - cumulative percentage

- If the SUMVAR= option is used, statistic can be one of the following:
  - SUM
    - sum (default)
  - MEAN
    - mean

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also “Calculating Weighted Statistics” on page 822.

See  “About Chart Statistics” on page 821 for a complete description of statistic types

**URL=character-variable**
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction  This option affects graphics output that is created through the ODS HTML destination only

Interaction  If you specify both the HTML= and URL= options, then the URL= option is ignored

See  “Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

**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=\textit{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 \textit{bar-width} are character cells. The value for \textit{bar-width} must be greater than zero, but it does not have to be an integer, for example: \texttt{bar site / width=1.5;}

\textbf{Restrictions} Sometimes the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis. In this case the procedure issues a warning in the SAS log and ignores the WIDTH= specification.

If the specified width is too narrow, the procedure might display the midpoint values vertically.

\textbf{WOUTLINE=\textit{bar-outline-width}}

specifies the width of the bar outline in pixels. WOUTLINE= affects both the slice and subgroup outlines.

\textbf{Style reference} LineThickness attribute of the GraphOutLines element

\textbf{WREF=}\texttt{(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=\textit{reference-line-width} for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list requires parentheses and widths separated by spaces \texttt{(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.

\textbf{Style reference} LineThickness attribute of the GraphReference element

\section*{Details}

\textbf{Description}

The BAR statement specifies the variable that defines the categories of data to chart. This statement automatically performs the following operations:

- determines the midpoints
- calculates the chart statistic for each midpoint (the default is FREQ)
- scales the response axis and the bars according to the statistic value
- determines bar width and spacing
- assigns patterns to the bars (the default bar pattern is SOLID)
- draws a frame around the axis area using the color defined by the current style or the first color in the color list if the NOGSTYLE system option is specified.

You can use statement options to do the following:

- select or order the midpoints (bars)
- control the tick marks on the response axis
• change the type of chart statistic
• display specific statistics
• modify the appearance of the chart

You can also specify additional variables by which to subgroup or sum the data.

Bar charts support subgroups, which subdivide the bars into segments based on the values of a subgroup variable.

In addition, you can do the following actions:
• use global statements to add a legend, modify the axes, and change the bar patterns. See Chapter 24, “SAS/GRAPH Statements,” on page 343 for more information.
• add titles and footnotes to the chart. See “TITLE, FOOTNOTE, and NOTE Statements” on page 447 for more information.
• use an Annotate data set to enhance the chart. See Chapter 27, “Using Annotate Data Sets,” on page 635 for more information.
• display an image in the background of the chart. See “IBACK” on page 576 for more information.
• display images in the bars of the chart. See the option “IMAGE=fileref | "external-file"” on page 399 for the PATTERN statement.

The Chart Statistic and the Response Axis
In bar-line charts, the scale of values of the chart statistic is displayed on the left response axis. By default, the response axis is divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks unless a log axis has been requested. For sum and mean statistics, the major tick marks are labeled with values of the SUMVAR= variable (formatted if the variable has an associated format). The response axis is also labeled with the statistic type.

Specifying Logarithmic Axes
Logarithmic axes can be specified with the AXIS statement. See Chapter 24, “SAS/GRAPH Statements,” on page 343 for a complete description.

Displaying Statistics in Bar-Line Charts
Statistic values on bar-line charts are not printed by default, so you must explicitly request a statistic with the FREQ, CFREQ, PERCENT, CPERCENT, SUM, MEAN, INSIDE=, or OUTSIDE= option.

For graphs generated with the ActiveX device, you can display one statistic for each bar. For graphs generated with other devices, you can display up to two statistics for each bar. Statistics can be displayed either above the bars or inside the bars.

To specify a statistic that you want to display above the bars, specify the statistic option (FREQ, CFREQ, PERCENT, CPERCENT, SUM, or MEAN) or specify OUTSIDE=statistic. To specify a statistic that you want to display inside the bars, specify INSIDE=statistic.

For graphs generated with the ActiveX device, the OUTSIDE= option overrides INSIDE=, and INSIDE= overrides the FREQ, CFREQ, PERCENT, CPERCENT, SUM, and MEAN options. For graphs generated with other devices, the individual statistic options override the OUTSIDE= option.
If more than one statistic option is specified, only the highest priority statistic is displayed. The priority order, from highest to lowest, is as follows:

1. FREQ
2. CFREQ
3. PERCENT
4. CPERCENT
5. SUM
6. MEAN

The bars must be wide enough to accommodate the text. You can adjust the width of the bars with the WIDTH= option. To control the font and size of the text, use the HTEXT= and FTEXT= graphics options.

**Ordering and Selecting Midpoints**

To rearrange character or discrete numeric midpoint values or to select ranges for numeric values, use the MIDPOINTS= option. Changing the number of midpoints for numeric variables changes the range of values for individual midpoints. However, it does not change the range of values for the chart as a whole. For details, see “About Midpoints” on page 819.

Like the MIDPOINTS= option, the ORDER= option in the AXIS statement can rearrange the order of the midpoints or suppress the display of discrete numeric or character values. However, the ORDER= option cannot calculate the midpoints for a continuous numeric variable, nor can it exclude values from the calculations. For details, see the option description for “ ORDER=(value-list) | DESCENDING” on page 383.

---

**PLOT Statement**

Creates one or more plot overlays on top of the bar-line chart.

**Restriction:** Not supported by JAVA

**Requirement:** This option affects graphics output that is created through the ODS HTML destination only. If specified, a PLOT statement or statements must be specified after the BAR statement.

**Global statements:** AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE

**Supports:** Data tips and drill-down functionality

---

**Syntax**

PLOT <options(s)>;

   <PLOT –1< options(s)>> <;PLOT–2< options(s)> …>;

---

**Summary of Optional Arguments**

**Appearance options**

CAUTOREF=reference-line-color
specifies the color of reference lines drawn at major tick marks, as
determined by the AUTOREF option.

CAXIS=axis-color
specifies a color for the tick marks and for the axis area frame on the plot
(right) response axis.

CREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies colors for reference lines.

CTEXT=text-color
specifies a color for all text on the plot response axis and legend.

LAUTOREF=reference-line-type
specifies the line type for reference lines at major tick marks, as determined
by the AUTOREF option.

LEGEND=LEGEND<1 ...99>
generates a legend and assigns the specified LEGEND definition to the
legend.

LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines.

NOLINE
suppresses the line connecting the PLOT symbols

NOMARKER
suppresses drawing the marker symbol.

WAUTOREF=reference-line-width
specifies the line width for reference lines at major tick marks, as determined
by the AUTOREF option.

WREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines.

Axes options

ASCENDING
joins the plot points in ascending order of the value of the plot statistic.

AUTOREF
draws a reference line at each major tick mark on the plot (right) response
axis.

AXIS=AXIS<1 ...99>
specifies values for the major tick marks or assigns specified AXIS
definitions to the axis.

CLIPREF
clips the reference lines at the bars.

DESCENDING
joins the plot points in descending order of the value of the plot statistic.

MINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major
tick mark on the PLOT response axis.

NOAXIS
suppresses the right PLOT response axis and displays the midpoint and left
BAR axes.

RAXIS=value-list | 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.

REF=value | (value) | (value-list)
draws reference lines at the specified points using the chart response axis.
ODS options

- HTML_LEGEND=variable
  identifies the variable in the input data set whose values create links or data tips or both.
- HTML=variable
  identifies the variable in the input data set whose values create links or data tips or both.
- URL=character-variable
  specifies a character variable whose values are URLs.

Statistic options

- CFREQ
  displays the cumulative frequency statistic above each plot data point.
- CPERCENT
  displays the cumulative percentage statistic above each plot data point.
- FREQ
  displays the frequency statistic above each plot data point.
- FREQ=numeric-variable
  specifies a variable whose values weight the contribution of each observation in the computation of the plot statistic.
- MEAN
  displays the mean statistic above each plot data point.
- PERCENT
  displays the percentages of observations that have a given value for the plot variable above each plot data point.
- STATFMT=format-specification
  overrides the GBARLINE default format of the displayed statistical value.
- SUM
  displays the sum statistic above each plot data point.
- SUMVAR=plot-variable
  specifies the variable to plot.
- TYPE=statistic
  specifies the plot statistic.

Optional Arguments

You can specify as many options as you want and list them in any order.

- ASCENDING
  joins the plot points in ascending order of the value of the plot statistic. By default, plot points are connected from left to right.
- AUTOREF
  draws a reference line at each major tick mark on the plot (right) response axis. To draw reference lines at specific points on the response axis, use the REF= option. By default, reference lines are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.
- AXIS=AXIS<1 ...99>
  specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

See “RAXIS=value-list | AXIS<1 ...99> AXIS=value-list | AXIS<1 ...99>” on page 857
CAUTOREF=reference-line-color
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE system option is specified. To specify a line type for these reference lines, use the LAUTOREF= option.

Style reference  Color attribute of the GraphGridLines element

CAXIS=axis-color
specifies a color for the tick marks and for the axis area frame on the plot (right) response axis. If you omit the CAXIS option, the default color is the color defined by the default style or is the first color in the color list.

CFREQ
displays the cumulative frequency statistic above each plot data point. If symbols are used, it appears above each symbol. CFREQ is ignored if the PERCENT or CPERCENT option is specified.

See  “About Chart Statistics” on page 821

CLIPREF
clips the reference lines at the bars. Using this option makes the reference lines appear to be behind the bars.

CPERCENT
displays the cumulative percentage statistic above each plot data point. If symbols are used, it appears above each symbol. CPERCENT overrides the FREQ, CFREQ, and PERCENT options.

Alias  CPCT

See  “About Chart Statistics” on page 821

CREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. The CAUTOREF= option overrides the CREF= reference line color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color only to the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (color1 color2 ...colorN) or separated by commas (color1, color2 ..., colorN). If you do not specify the CREF= option, the GBARLINE procedure uses the color specified by the CAXIS= option. If neither option is specified, then the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified. To specify line types for these reference lines, use the LREF= option.

Alias  CRF=

Style reference LineStyle attribute of the GraphGridLines element

CTEXT=text-color
specifies a color for all text on the plot response axis and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GBARLINE procedure looks for the text color in the following order:
1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the PLOT statement.

2. the color specified by the CTEXT= option in the PLOT statement

3. the color specified by the CTEXT= option in a GOPTIONS statement.

4. the color specified in the current style. However, if the NOGSTYLE system option is specified, then black for the ActiveX device and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, then the CTEXT= color overrides the AXIS statement's COLOR= specification. This means that the CTEXT= color is used for axis labels and values. The COLOR= color is still used for all other axis elements, such as tick marks.

**Alias**
CT=

**Style reference**
GraphLabelText, GraphValueText

**Note**
If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

**DESCENDING**
joins the plot points in descending order of the value of the plot statistic. By default, plot points are connected from left to right.

**FREQ**
displays the frequency statistic above each plot data point. If symbols are used, it appears above each symbol. Non-integer values are rounded down to the nearest integer.

**Restriction**
FREQ is ignored if the CFREQ, PERCENT, or CPERCENT option is specified.

**See**
“About Chart Statistics” on page 821

**FREQ=numeric-variable**
specifies a variable whose values weight the contribution of each observation in the computation of the plot statistic. Each observation is counted the number of times that is specified by the value of numeric-variable for that observation. If the value of numeric-variable is missing, zero, or negative, then the observation is not used in the statistic calculation. Non-integer values of numeric-variable are truncated to integers. The FREQ= option is valid with all plot statistics.

You cannot use TYPE=PERCENT, TYPE=CPERCENT, TYPE=FREQ, or TYPE=CFREQ with the SUMVAR= option. You must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are not affected by applying a format to numeric-variable.

**Restriction**
Not supported by ActiveX
HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction Not supported by Java or ActiveX

See “Data Tips for Web Presentations” on page 191
“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

Example “Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 865

HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

LAUTOREF=reference-line-type
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CAUTOREF= option.

LEGEND=LEGEND<1 …99>
generates a legend and assigns the specified LEGEND definition to the legend. The LEGEND= option is ignored if the specified LEGEND definition is not in effect. When you specify the LEGEND option, the BAR statement generates a legend even if the SUBGROUP= option is not specified. This output differs from the output generated by the GCHART procedure where the SUBGROUP option automatically creates a legend by default. In this case, only one bar is represented in the legend.

You can specify both a BAR and PLOT legend on the same graph. If you specify the same location with the LEGEND= option in the BAR statement and the POSITION= option in the LEGEND statement, a single combined legend is drawn. However, ActiveX output displays separate but adjacent legends.

The ActiveX device does not support all LEGEND statement options.

Restriction Not supported by Java. Partially supported by ActiveX.
Note Only one PLOT statement can contain a LEGEND= reference. If you request a PLOT legend, then all of the PLOT lines are displayed in the legend.

See “LEGEND Statement” on page 377

Example “Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs” on page 865

LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)

specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=reference-line-type for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and types separated by spaces (type1 type2 ...typeN). If you do not specify the LREF= option, the GBARLINE procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

Alias LR=

Style reference GraphReference

Restriction Not supported by Java

MEAN displays the mean statistic above each plot data point. If symbols are used, it appears above each symbol. MEAN is ignored unless you also use the SUMVAR= option.

See “About Chart Statistics” on page 821

MINOR=number-of-minor-ticks

specifies the number of minor tick marks that are drawn between each major tick mark on the PLOT response axis. Minor tick marks are not labeled. The MINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

NOAXIS suppresses the right PLOT response axis and displays the midpoint and left BAR axes. The axis lines, axis labels, axis values, and all major and minor tick marks are suppressed on the right axis. If you specify an axis definition with the MAXIS= or RAXIS= options, then the axes are generated as defined in the AXIS statement. However, all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH=, and OFFSET= are still used.

To remove only selected axis elements such as lines, values, or labels, use specific AXIS statement options.

NOAXIS does not suppress either the default frame or an axis area fill requested by the CFRAME= option. To remove the axis frame, use the NOFRAME option in the procedure.
Alias  NOAXES

**NOLINE**
suppresses the line connecting the PLOT symbols. This is true regardless of what is specified in the SYMBOL statement.

**NOMARKER**
suppresses drawing the marker symbol. This is true regardless of what is specified in the SYMBOL statement.

**PERCENT**
displays the percentages of observations that have a given value for the plot variable above each plot data point. If symbols are used, it appears above each symbol.

Alias  PCT

Restriction  The PERCENT option is ignored if the CPERCENT option is specified.

See  “About Chart Statistics” on page 821

**RAXIS=value-list | AXIS=1 ...99>**
specifies the major tick mark values for the PLOT (right) response axis or assigns an AXIS definition. The way you specify *value-list* depends on the type of variable:

- **For numeric variables**, *value-list* is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - \( n < \ldots n \)
  - \( n \) TO \( n \) \(<\)BY \( \text{increment}\)\( >\)
  - \( n < \ldots n \) TO \( n \) \(<\)BY \( \text{increment}\)\( ><n < \ldots n >\)
  
  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 < \ldots 'SAS-value'^i\>
  - \('SAS-value'^i \) TO \('SAS-value'^i)<\)BY \( \text{interval}\)

Restriction  Any response values that exceed the highest tick mark value are not plotted.

Note  The overlay plot line connects only the visible plot response values.

**REF=value | (value) | (value-list)**
draws reference lines at the specified points using the chart response axis. See “**MIDPOINTS=value-list**” on page 841 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.
The `STATFMT=` option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.

If you specify an option of TYPE=mean, an INSIDE=pct option, and an OUTSIDE=sum option, a STATFMT=f8.1 option applies only to the calculated TYPE=mean statistical value. For example:

```plaintext
proc gbarline;
plot mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
     statefmt=f8.1;
run;
```

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the STATFMT= option.

If you change the previous example to specify an OUTSIDE=mean option, then STATFMT=f.8.1 applies to the OUTSIDE=mean option and the TYPE=mean option. The statistical types match.

The STATFMT= option does not control the format of the response axis tick marks.

**Alias**  SFMT=, SFORMAT=, STATFORMAT=

**SUM**

displays the sum statistic above each plot data point. If symbols are used, it appears above each symbol.

**Restrictions**
The SUM option is ignored unless you also use the SUMVAR= option.

This option is ignored if the MEAN option is specified.

**See**

“About Chart Statistics” on page 821

**SUMVAR=plot-variable**
specifies the variable to plot. Plot-variable, if specified, must be numeric. The GBARLINE procedure calculates the sum or, if requested, the mean of plot-variable for each midpoint.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

**See**

“About Response Variables” on page 821

**Example**

“Example 1: Producing a Basic Bar-Line Chart” on page 861

**TYPE=statistic**
specifies the plot statistic.

- If the SUMVAR= option is not used, statistic can be one of the following:

  - FREQ frequency (default)
  - CFREQ cumulative frequency
PERCENT
  percentage
CPERCENT
  cumulative percentage

- If SUMVAR= is used, statistic can be one of the following:
  SUM
    sum (default)
  MEAN
    mean

Because you cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT with SUMVAR=, you must use the FREQ= option to calculate percentages or frequencies based on a sum.

See  “About Chart Statistics” on page 821
     “Calculating Weighted Statistics” on page 822

URL=character-variable
  specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions  This option affects graphics output that is created through the ODS HTML destination only.
  Not supported by GIF, PNG, and ACTIVEX devices.

Interaction  If you specify both the HTML= and URL= options, then the URL= option is ignored.

See  “Overview of Enhancing Web Presentations” on page 188
     “Example: GIF Output with Drill-Down Links” on page 163

WAUTOREF=reference-line-width
  specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CAUTOREF= option.

Style reference  LineThickness attribute of the GraphGridLines element

WREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
  specifies line widths for reference lines. Line widths are specified as whole numbers. Specifying a line width without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the WAUTOREF= option overrides WREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list is (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the CREF= option.
Details

Using the PLOT Statement
PLOT statements are optional, but if specified, they must follow the BAR statement. If
you do not specify any PLOT statements, GBARLINE generates only a bar chart and
duplicates the chart response axis (left axis) as the right response axis.

To specify a variable to plot, use the SUMVAR= option. If you do not specify a plot
variable, GBARLINE uses the chart variable as the plot variable. For more information,
see “About Response Variables” on page 821 and the option description for
“SUMVAR=plot-variable” on page 858.

Description
The PLOT statement specifies one plot request. You can use multiple PLOT statements
to generate multiple plots. The PLOT statement automatically does the following:

• scales the plot response (right) axis to include the maximum and minimum data
  values
• plots data points within the axis and connects them from left to right
• labels the plot response axis and displays each major tick mark value

You can use statement options to specify a plot variable, manipulate the plot response
axis, modify the appearance of your graph, and describe catalog entries. You can use
SYMBOL definitions to modify plot symbols for the data points, suppress the joining of
data points, or specify other types of interpolation. For more information, see the
“SYMBOL Statement” on page 412.

In addition, you can use global statements to add a legend, modify the axis, or add titles,
footnotes, and notes to the plot.

About SYMBOL Definitions

How SAS/GRAPH Generates and Assigns SYMBOL Definitions
SYMBOL statements control the appearance of plot symbols and lines. They can specify
the following attributes:

• the shape, size, and color of the plot symbols that mark the data points
• the plot line style, color, and width
• an interpolation method (either JOIN, NEEDLE, STEP, or NONE) for plotting data
• how missing values are treated in interpolation calculations

SYMBOL definitions are assigned either by default by the GBARLINE procedure or
explicitly with a plot request.

If no SYMBOL definition is currently in effect, the GBARLINE procedure produces a
join interpolation using the default plot symbol. For multiple PLOT statements where no
SYMBOL statements were specified, the procedure rotates through the default symbols
for the current device.

See the “SYMBOL Statement” on page 412 for a complete discussion of its features.
**About Interpolation Methods**
You can produce plot overlays such as step plot overlays by specifying interpolation methods with the SYMBOL statement. For PROC GBARLINE, you can use the SYMBOL statement to do the following tasks:

- connect data points with straight lines (JOIN)
- produce overlay plots with unconnected data points (NONE)
- use a step function to connect the data points (STEP)

For bar-line charts, points on the plot overlays are automatically connected by default, which is equivalent to specifying the JOIN interpolation method.

The “SYMBOL Statement” on page 412 describes the JOIN, STEP, and NONE interpolation methods.

---

**Examples: GBARLINE Procedure**

**Example 1: Producing a Basic Bar-Line Chart**

| Features: | BAR and PLOT statement options  |
| Sample library member: | GBLSTOCK |
| Note: | The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com. |

This example produces a basic bar-line chart showing the volume and closing price for each of five days of trading activity on the New York Stock Exchange. The vertical bars indicate the volume using the left (chart) response axis, and the line plot shows the closing price. This graph uses the statistical style.
Program

goptions reset=all border;

title1 "NYSE Closing Price and Volume - 2002";

data nyse;
    format Day date7.;
    format High Low Close comma12.;
    format Volume comma12.;
    input Day date7. High Low Close Volume;

datalines;
01AUG07 10478.76 10346.24 10426.91 1908809
02AUG07 11042.92 10298.44 10274.65 1807543
05AUG07 10498.22 10400.31 10456.43 1500656
06AUG07 10694.47 10636.32 10762.98 1498403
07AUG07 10801.12 10695.13 10759.48 1695602
run;

proc gbarline data=nyse;
    bar day / discrete sumvar=volume space=4;
    plot / sumvar=close;
run;
quit;

Program Description

Set the graphics environment. Some graphics options might override style attributes, so if you are using a style, specify the minimum graphic options needed by your graph.

goptions reset=all border;
Define the title.

```
title1 "NYSE Closing Price and Volume - 2002";
```

Create the data set NYSE. NYSE contains one observation for each of five workdays. Each observation includes the date, closing price, and volume.

```
data nyse;
    format Day date7.;
    format High Low Close comma12.;
    format Volume comma12.;
    input Day date7. High Low Close Volume;

datalines;
  01AUG07  10478.76  10346.24  10426.91  1908809
  02AUG07  11042.92  10298.44  10274.65  1807543
  05AUG07  10498.22  10400.31  10456.43  1500656
  06AUG07  10694.47  10636.32  10762.98  1498403
  07AUG07  10801.12  10695.13  10759.48  1695602
run;
```

Produce the bar-line chart. The SUMVAR= option in the BAR statement specifies the variable whose values determine the height of the bars. The SUMVAR= option in the PLOT statement specifies the variable whose values are used to calculate the overlay plot.

```
proc gbarline data=nyse;
    bar day / discrete sumvar=volume space=4;
    plot / sumvar=close;
run;
quit;
```

Example 2: Calculating Weighted Statistics

**Features:**
- BAR statement options
  - AXIS=
  - SUMVAR=
- PLOT statement options
  - AXIS=
  - FREQ=
  - SUMVAR=

**Other features:**
- AXIS statement

**Data set:** SASHELP.FAILURE

**Sample library member:** GBLWTSTA

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

This example uses the FREQ= option to calculate weighted statistics for the line plot. During the manufacture of a metal-oxide semiconductor (MOS) capacitor, various defects and their frequencies were recorded.
Program

goptions reset=all border;

data myfail;
  set sashelp.failure;
  label Cause='Defect';
  select (Cause);
    when ("Contamination")  Cost=3.5;
    when ("Metallization")  Cost=10;
    when ("Oxide")          Cost=10.5;
    when ("Corrosion")      Cost=4.5;
    when ("Doping")         Cost=3.6;
    when ("Silicon")        Cost=5.4;
    otherwise               Cost=1.0;
  end;
run;

title1 "The Cost of Defects";
axis1 label="Defect Count";
axis2 label="Total Cost";
proc gbarline data=myfail;
  bar Cause / sumvar=Count axis=axis1;
  plot / sumvar=Count freq=cost axis=axis2;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

This graph uses the data set entitled FAILURE found in the SASHELP library. Each observation of the FAILURE data set contains the type of manufacturing defect and a count of how many times it occurred. Define data that associates a cost with each defect. Assign a label of 'Defect' in place of the variable name in the data set.
data myfail;
  set sashelp.failure;
  label Cause='Defect';
  select (Cause);
    when ("Contamination")  Cost=3.5;
    when ("Metallization")  Cost=10;
    when ("Oxide")          Cost=10.5;
    when ("Corrosion")      Cost=4.5;
    when ("Doping")         Cost=3.6;
    when ("Silicon")        Cost=5.4;
    otherwise                Cost=1.0;
  end;
run;

Define the title.

  title1 "The Cost of Defects";

Define the labels for the axes.

  axis1 label=("Defect Count");
  axis2 label=("Total Cost");

Produce the bar-line chart. The SUMVAR= option in the BAR statement specifies the variable that determines the height of the bars. The SUMVAR= option in the PLOT statement specifies the plot variable, in this case the Cause of Failure, or Defect. GBARLINE multiplies the value of the FREQ= variable (cost assigned) by the value of the COUNT variable, and uses the result to determine the plot points.

  proc gbarline data=myfail;
    bar Cause / sumvar=Count axis=axis1;
    plot / sumvar=Count freq=cost axis=axis2;
  run;
  quit;

Example 3: Specifying Subgroups, Multiple Plots, Data Tips, and Drill-Down URLs

<table>
<thead>
<tr>
<th>Features:</th>
<th>BAR statement options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DISCRETE</td>
</tr>
<tr>
<td></td>
<td>HTML=</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
</tr>
<tr>
<td></td>
<td>MAXIS=</td>
</tr>
<tr>
<td></td>
<td>RAXIS=</td>
</tr>
<tr>
<td></td>
<td>SUBGROUP=</td>
</tr>
<tr>
<td></td>
<td>SUMVAR=</td>
</tr>
<tr>
<td>PLOT statement options</td>
<td>AXIS=</td>
</tr>
<tr>
<td></td>
<td>HTML=</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
</tr>
<tr>
<td>Multiple PLOT statements</td>
<td>SUMVAR=</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other features:</th>
<th>ODS statement option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STYLE=</td>
</tr>
</tbody>
</table>
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.
Program

filename odsout ".";
goptions reset=all border;
ods _all_ close;
ods html path=odsout file="electricpowerUS.html" style=analysis gtitle
nogfootnote;
title1 "US Electric Power - Revenue and Generation Sources";
footnote1 j=1 italic
  link="http://www.eia.doe.gov/cneaf/electricity/epa/epat7p3.html"
  "Link to Bar Data: USEIA Energy Customer Sectors";
footnote2 j=1 italic
  link="http://www.eia.doe.gov/cneaf/electricity/epa/epat1p1.html"
  "Link to Line Data: USEIA Energy Generation Sources";
axis1 label=(j=c "Revenue" j=c "(billions)") minor=none; /* left */
axis2 label=(j=c "Power" j=c "(GWh)") minor=none; /* right */
axis3 label=none; /* bottom */
/* Bar legend */
legend1 position=(middle right outside) across=1
  label=(position=(top ) j=1 "Customer Sector");
/* Line plot legend */
legend2 position=(bottom right outside) across=1 repeat=1
  label=(position=(top) j=1 "Generation Source") ;
symbol1 c=black value=circle;
symbol2 value=dot;
proc gbarline data=sashelp.electric;
  bar year / discrete sumvar=Revenue subgroup=Customer
    raxis=axis1 maxaxis=axis3 legend=legend1
    html=revtip name="US_Electric_Power"
    des="Chart of US Electricity Generation Sources and Consumers";
    plot / sumvar=AllPower html=alltip legend=legend2 axis=axis2;
    plot / sumvar=Coal html=coaltip;
    plot / sumvar=Nuclear html=nuketip;
    plot / sumvar=NaturalGas html=gastip;
    plot / sumvar=Hydro html=hydrotip;
    plot / sumvar=Other html=othertip;
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */

Program Description

Create a file reference for the output. The current working directory is specified in this example.

filename odsout ".";
Set the graphics environment.

goptions reset=all border;

Close the currently open ODS destinations.

ods _all_ close;

Set the HTML options. The PATH= option specifies the file reference that was defined previously for the output. The FILE= option specifies a name for the HTML output file. The GTITLE option causes the title to be rendered as part of the graph image instead of being created by the HTML code as text. Alternatively, the NOGFOOTNOTE option causes the footnote to be created by the HTML file as text instead of being rendered as an image with the rest of the graph. Notice that, as a result, the TITLE appears within the graph frame, but the footnotes appear outside the frame.

ods html path=odsout file="electricpowerUS.html" style=analysis gtitle nogfootnote;

Define the title and footnotes. The LINK= option in the FOOTNOTE statement defines drill-down URLs for the source of the information.

title1 "US Electric Power - Revenue and Generation Sources*;

footnote1 j=italic link="http://www.eia.doe.gov/cneaf/electricity/epa/epat7p3.html" "Link to Bar Data: USEIA Energy Customer Sectors*;

footnote2 j=italic link="http://www.eia.doe.gov/cneaf/electricity/epa/epat1p1.html" "Link to Line Data: USEIA Energy Generation Sources" ;

Define the labels for the axes. The AXIS1 statement defines the axis properties for the bar response (left) axis. The AXIS2 statement defines the properties for the plot response (right) axis. The AXIS3 statement is used to suppress the default label on the midpoint axis.

axis1 label=(j=c "Revenue" j=c "(billions)") minor=none; /* left */
axis2 label=(j=c "Power" j=c "Generated" j=c "(GWh)") minor=none; /* right */
axis3 label=none; /* bottom */

Specify options for the bar and plot legends. Use different LEGEND statements and position the legends in different places for the bar chart and the overlay plots. This causes GBARLINE to produce two separate legends instead of combining the legends into one.

/* Bar legend */
legend1 position=(middle right outside) across=1
    label=(position=(top) j=1 "Customer Sector");

/* Line plot legend */
legend2 position=(bottom right outside) across=1 repeat=1
    label=(position=(top) j=1 "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.

```sas
proc gbarline data=sashelp.electric;
   bar year / discrete sumvar=Revenue subgroup=Customer
      raxis=axis1 maxis=axis3 legend=legend1
      html=revtip name="US_Electric_Power"
      des="Chart of US Electricity Generation Sources and Consumers";

   plot / sumvar=AllPower   html=alltip legend=legend2 axis=axis2;
   plot / sumvar=Coal       html=coaltip;
   plot / sumvar=Nuclear    html=nuketip;
   plot / sumvar=NaturalGas html=gastip;
   plot / sumvar=Hydro      html=hydrotip;
   plot / sumvar=Other      html=othertip;
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```sas
ods html close;
```

Open ODS HTML; Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```
Chapter 35
GCHART Procedure

Overview: GCHART Procedure ........................................... 872
About the GCHART Procedure ........................................ 872
About Block Charts ..................................................... 872
About Bar Charts ....................................................... 873
About Pie, Detail Pie, and Donut Charts ......................... 875
About Star Charts ....................................................... 876

Chart Terminology ....................................................... 877

Understanding Chart Variables ...................................... 879
About Chart Variables ................................................ 879
Missing Values ........................................................ 880

Understanding Midpoints .............................................. 880
About Midpoints ....................................................... 880
Character Values ....................................................... 880
Discrete Numeric Values ............................................ 881
Continuous Numeric Values ........................................ 881
Selecting and Ordering Midpoints .................................. 882

About Chart Statistics .................................................. 883
Definition: Chart Statistics .......................................... 883
Frequency ............................................................. 883
Cumulative Frequency .............................................. 883
Percentage ............................................................ 883
Cumulative Percentage ............................................ 883
Sum ........................................................................ 884
Mean ...................................................................... 884
Calculating Weighted Statistics ..................................... 884

About Patterns .............................................................. 885
Pattern Categories ..................................................... 885
Default Patterns and Outlines ................................. 885
User-Defined Patterns, Outlines, and Images ............ 886

Syntax: GCHART Procedure .............................................. 886
PROC GCHART Statement ............................................ 887
BLOCK Statement ...................................................... 888
HBAR, HBAR3D, VBAR, and VBAR3D Statement .... 901
PIE, PIE3D, and DONUT Statement ......................... 936
STAR Statement ....................................................... 960

Examples: GCHART Procedure ....................................... 975
Example 1: Specifying the Sum Statistic in a Block Chart .... 975
Example 2: Grouping and Subgrouping a Block Chart .... 977
Overview: GCHART Procedure

About the GCHART Procedure

The GCHART procedure produces six types of charts: block charts, horizontal and vertical bar charts, pie and donut charts, and star charts. These charts graphically represent the value of a statistic calculated for one or more variables in an input SAS data set. The charted variables can be either numeric or character.

The procedure calculates these statistics:
• frequency or cumulative frequency counts
• percentages or cumulative percentages
• sums
• means

Use the GCHART procedure to do the following tasks:
• display and compare exact and relative magnitudes
• examine the contribution of parts to the whole
• analyze where data are out of balance

About Block Charts

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

Figure 35.1 on page 873 shows a simple block chart of total sales for three manufacturing sites. Each site is a midpoint and occupies one square. The name of the site (the midpoint value) is printed below the square. By default, midpoint values are arranged in ascending order from left to right. The label below the midpoint grid names the chart variable.

Sales for the site (the chart statistic) are represented by the height of the block; sales amount (the formatted statistic value) is printed below the block. The heading above the blocks describes the type of statistic, in this case SUM.
Figure 35.1 Block Chart (GCHBKSUM)

The program for this chart is in “Example 1: Specifying the Sum Statistic in a Block Chart” on page 975. For more information about producing block charts, see the “BLOCK Statement” on page 888.

About Bar Charts

Horizontal and vertical bar charts display the magnitude of data with bars, each of which represents a category of data (midpoint). The length (or height) of the bars represents the value of the chart statistic for the corresponding midpoint. Both horizontal and vertical bar charts can be either two-dimensional or three-dimensional shapes, depending on which statement you choose.

Figure 35.2 on page 874 shows a simple two-dimensional, horizontal bar chart of total sales for three manufacturing sites. Each site is a midpoint and is displayed as a bar. The name of the site (the midpoint value) is printed on the midpoint axis beside the bar. By default, midpoint values are arranged in ascending alphabetical or numeric order from top to bottom of the chart and labeled with the name or label of the chart variable.

The chart statistics, in this case total sales for each site, are represented by the length of the bars. The response axis displays the scale of values for the chart statistic. The table of statistics to the right of the bars displays the statistic for each bar. Both a column in the table and the response axis are labeled with the name of the summary variable and the type of statistic.
Figure 35.2 Horizontal Bar Chart (GCHBRSUM (a))

The program for this chart is “Example 3: Specifying the Sum Statistic in Bar Charts” on page 979.

Figure 35.3 on page 874 shows the same data presented as a three-dimensional, vertical bar chart. The two types of bar charts have essentially the same characteristics except for where they display statistical values. By default, horizontal bar charts display a table of statistic values to the right of the bars. You can specify that vertical bar charts display the statistic value above or inside each bar.

Figure 35.3 Vertical (Three-Dimensional) Bar Chart (GCHBRSUM (b))
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 35.4 on page 875 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 35.4  Pie Chart (GCHPISUM (a))

Figure 35.5 on page 876 shows the three-dimensional version of the same pie chart. This version features the exploded slice.
Figure 35.5  Three-Dimensional Pie Chart (GCHPISUM (b))

Figure 35.6 on page 876 shows a detail pie chart generated from the same data.

Figure 35.6  Detail Pie Chart (GCHDTPIE)

The programs for these charts are in “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991 and “Example 13: Creating a Detail Pie Chart” on page 1004. For more information about producing pie or donut charts, see “PIE, PIE3D, and DONUT Statement” on page 936.

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 35.7 on page 877 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 35.7  Star Chart (GCHSTSUM)

The program for this chart is “Example 11: Specifying the Sum Statistic in a Star Chart” on page 999. For more information about producing star charts, see “STAR Statement” on page 960. For an alternative way of producing similar types of charts, see Chapter 42, “GRADAR Procedure,” on page 1243.

Chart Terminology

The GCHART procedure produces charts based on the values of a chart variable. These values are represented by a set of midpoints. The chart itself displays information about the chart variable in the form of chart statistics.

Figure 35.8 on page 878 and Figure 35.9 on page 879 illustrate these terms as well as other terms used with the GCHART procedure.
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.
Pie charts show statistics based on values of a variable called the chart variable. Generally, the values of the chart variable are represented by the slices in the chart. Beside each pie slice a number (or character string) appears that identifies the value or range of values assigned to that slice by the GCHART procedure. This number (or character string) is known as the midpoint for that slice. The statistic value for each midpoint is displayed beneath the midpoint. Each pie slice represents a different value of a given variable (the chart variable). Because the pie chart forms a circle of 360 degrees, each slice represents a percentage of degrees of the circle. The number of degrees created by each slice represents the statistic value for the midpoint.

---

**Understanding Chart Variables**

**About Chart Variables**

The **chart variable** is the variable in the input data set whose values determine the categories of data represented by the bars, blocks, slices, or spines. The chart variable generates the midpoints to which each observation in the data set contributes.

The chart variable can be either character or numeric. Character chart variables contain character values, which are always discrete. Numeric chart variables fall into two categories: discrete and continuous.

*Note:* You can apply a format that converts multiple values or a range of values to a single formatted value. In this case the GCHART procedure produces a single midpoint for that single formatted value.
• *Discrete variables* contain a finite number of specific numeric values that are to be represented on the chart. For example, a variable that contains years, such as 1984 or 2001, is a discrete variable.

• *Continuous variables* contain a range of numeric values that are to be represented on the chart. For example, a variable of temperature data that contains real values between 0 and 212 is a continuous variable.

Numeric chart variables are always treated as continuous variables unless the DISCRETE option is used in the action statement, or, unless a format is used to group ranges of values. In most cases it is a good idea to specify the DISCRETE option when using date values.

**Missing Values**

By default, the GCHART procedure ignores missing midpoint values for the chart variable. If you specify the MISSING option, then missing values are treated as a valid midpoint and are included on the chart. Missing values for the group and subgroup variables are always treated as valid groups and subgroups.

When the value of the variable that is specified in the FREQ= option is missing, 0, or negative, the observation is excluded from the calculation of the chart statistic.

When the value of the variable specified in the SUMVAR= option is missing, the observation is excluded from the calculation of the chart statistic.

---

**Understanding Midpoints**

**About Midpoints**

*Midpoints* are the values of the chart variable that identify categories of data. By default, midpoints are selected or calculated by the procedure. The way the procedure handles the midpoints depends on whether the values of the chart variable are character, discrete numeric, or continuous numeric.

**Character Values**

A character chart variable generates a midpoint for each unique value of the variable. For example, if the chart variable CITY contains the names of three different cities, each city is a midpoint, resulting in three midpoints for the chart:
(In pie charts, midpoint values that compose a small percentage of the total for the chart might be placed in the OTHER slice and do not produce a separate midpoint.)

By default, character midpoints are arranged in alphabetic order. If a character variable has an associated format, the values are arranged in order of the formatted values.

**Discrete Numeric Values**

A numeric chart variable used with the DISCRETE option generates a midpoint for each unique value of the chart variable. For example, the numeric variable YEAR used with the DISCRETE option produces one midpoint for each year:

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 35.12  Continuous Numeric Midpoints*

![Continuous Numeric Midpoints](image)

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 35.10 on page 881, specify the following:

```
    midpoints="Tokyo" "Denver" "Seattle"
```

To exclude the midpoint for Denver, specify the following:

```
    midpoints="Tokyo" "Seattle"
```

In this case, values excluded by the option are not included in the calculation of the chart statistic.

You can order or select discrete numeric midpoint values just as you do character values, but you omit the quotation marks when specifying numeric values.

For continuous numeric variables, use the LEVELS= or MIDPOINTS= option to

- change the number of midpoints
- control the range of values each midpoint represents
- change the order of the midpoints.

To control the range of values each midpoint represents, use the MIDPOINTS= option to specify the median value of each range. For example, to select the ranges 20–29, 30–39, and 40–49, specify the following:

```
    midpoints=25 35 45
```

Alternatively, to select the number of midpoints that you want and let the procedure calculate the ranges and medians, use the LEVELS= option.

After you specify the number of midpoints that you want, the range for each midpoint is calculated automatically, using the algorithm in Terrell and Scott (1985).

You can also use formats to control the ranges of continuous numeric variables, but in that case the values are no longer continuous but discrete.
Note: You cannot use the MIDPOINTS= option to exclude continuous numeric values from the chart. Values below or above the ranges specified by the option are automatically included in the first and last midpoints, respectively. To exclude continuous numeric values from a chart, use a WHERE statement in a DATA step or the WHERE= data set option.

See also the description of the LEVELS= and MIDPOINTS= options for the appropriate statement.

About Chart Statistics

**Definition: Chart Statistics**

The chart statistic is the statistical value calculated for the chart variable and represented by each block, bar, or slice. The GCHART procedure calculates six chart statistics; the default statistic is frequency.

The examples given in the descriptions of these statistics assume a data set with two variables, CITY and SALES. The values of CITY are Denver, Seattle, and Tokyo. There are 21 observations: seven for Denver, nine for Seattle, and five for Tokyo.

**Frequency**

The frequency statistic is the total number of observations in the data set for each midpoint. For example, seven observations of the chart variable, CITY, contain the value Denver, so the frequency for the Denver midpoint is 7.

**Cumulative Frequency**

The cumulative frequency statistic adds the frequency for the current midpoint to the frequency of all of the preceding midpoints. For example, the frequency for the Denver midpoint is 7, and the frequency for the next midpoint, Seattle, is 9, so the cumulative frequency for Seattle is 16.

You cannot request cumulative frequency with the DONUT, PIE, PIE3D, or STAR statements.

**Percentage**

The percentage statistic is calculated by dividing the frequency for each midpoint by the total frequency count for all midpoints in the chart or group and multiplying it by 100. For example, the frequency count for the Denver midpoint is 7 and the total frequency count for the chart is 21, so the percentage statistic for Denver is 33.3%.

**Cumulative Percentage**

The cumulative percentage statistic adds the percentage for the current midpoint to the percentage for all of the preceding midpoints in the chart or group. For example, the percentage for the Denver midpoint is 33.3, and the percentage for the next midpoint, Seattle, is 42.9, so the cumulative percentage for Seattle is 76.2.
You cannot request cumulative percentage with the DONUT, PIE, PIE3D, or STAR statements.

**Sum**

The sum statistic is the total of the values for the SUMVAR= variable for each midpoint. For example, if you specify SUMVAR=SALES, and the values of the SALES variable for the seven Denver observations are 8734, 982, 1504, 3207, 4502, 624, and 918, then the sum statistic for the Denver midpoint is 20,471.

You must use the SUMVAR= option to specify the variable for which you want the sum statistic.

**Mean**

The mean statistic is the average of the values for the SUMVAR= variable for each midpoint. For example, if TYPE=MEAN and SUMVAR=SALES, the mean statistic for the Denver midpoint is 2924.42.

You must use the SUMVAR= option to specify the variable for which you want the mean statistic.

**Calculating Weighted Statistics**

By default, each observation is counted only once in the calculation of the chart statistic. To calculate weighted statistics in which an observation can be counted more than once, use the FREQ= option. This option identifies a variable whose values are used as a multiplier for the observation in the calculation of the statistic. If the value of the FREQ= variable is missing, 0, or negative, the observation is excluded from the calculation.

If you use the SUMVAR= option, then the SUMVAR= variable value for an observation is multiplied by the FREQ= variable value for that observation when calculating the chart statistic.

For example, to use a variable called COUNT to produce weighted statistics, assign FREQ=COUNT. If you also assign the variable HEIGHT to the SUMVAR= option, then the following table shows how the values of COUNT and HEIGHT would affect the statistic calculation:

<table>
<thead>
<tr>
<th>Value of COUNT</th>
<th>Value of HEIGHT</th>
<th>Number of Times the Observation is Used</th>
<th>Value Used for HEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>5</td>
<td>325</td>
</tr>
<tr>
<td>.</td>
<td>63</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>-3</td>
<td>60</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

By default, the percentage and cumulative percentage statistics are calculated based on the frequency. If you want to chart a percentage or cumulative percentage based on a sum, you can use the FREQ= option to specify a variable to use for the “sum” calculation. You can also specify the PCT statistic, as shown in this example:
Because the variable that is used by the FREQ= option determines the number of times an observation is counted, the value of COUNT is the equivalent of the sum statistic. See also the descriptions of the TYPE=, SUMVAR=, and FREQ= options for the action statements.

About Patterns

**Pattern Categories**

When a chart needs one or more patterns, the procedure uses either one of the following:
- default patterns and outlines that are automatically generated by SAS/GRAPH
- patterns, colors, outlines, and images that are defined by PATTERN statements, graphics options, and procedure options

The following sections summarize pattern behavior for the GCHART procedure. For more information, see the “PATTERN Statement” on page 398.

**Default Patterns and Outlines**

The GCHART procedure uses default patterns and outlines when you do not do the following:
- specify any PATTERN statements
- use the CPATTERN= graphics option
- use the COLORS= graphics options
- use the COUTLINE= option in the action statement

The default patterns, colors, and outlines are generated from the current style. If all of the above conditions are true, and the GSTYLE option is in effect, then the GCHART procedure does the following:
- selects the default fill, which is always solid, and rotates it through the color list of the current style, generating one solid pattern for each color. If the first color in the style's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.
- uses the style outline color to outline every patterned area.

If all of the above conditions are true, and the NOGSTYLE option is specified, then the GCHART procedure does the following:
- selects the first default fill, which is always solid, and rotates it through the device's color list, generating one solid pattern for each color. If the first color in the device's color list is black (or white), the procedure skips that color and begins generating patterns with the next color.
- uses the foreground color to outline every patterned area.
- if the procedure needs additional patterns, GCHART selects the next default pattern fill that is appropriate to the type of chart. GCHART then rotates the pattern fill
through the color on the list, skipping the foreground color as before. The procedure continues in this manner until it has generated enough patterns for the chart.

Changing any of the above conditions changes or overrides the default behavior:

- If you specify a color list with the COLORS= option in a GOPTIONS statement. If the list contains more than one color, the procedure produces a solid pattern through that list, using every color, even if the foreground color is black (or white). The default outline color remains the style outline color.

- If you specify either COLORS=(one-color) or the CPATTERN= graphics option, the default fill pattern changes from solid to the list of appropriate hatch patterns. The procedure uses the specified color to generate one pattern definition for each hatch pattern in the list. The default outline color remains the style outline color. (The Java and ActiveX devices do not support hatch patterns.)

For a description of these graphics options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

**User-Defined Patterns, Outlines, and Images**

You can use PATTERN statements to specify patterns, including color or fill type or both. You can also specify images to fill the bars of two-dimensional bar charts. For complete information about all patterns, see the “PATTERN Statement” on page 398. See also the section on controlling patterns and colors for each chart type.

When you use PATTERN statements, the procedure uses the specified patterns until all of the PATTERN definitions that they generate have been used. Then, if more patterns are required, it returns to the default pattern rotation.

To change the outline color of any pattern, whether it is a default or user-defined pattern, use the COUTLINE= option in the action statement that generates the chart.

The GCHART procedure uses the device colors for outlines in block, bar, pie, and star charts regardless of the ODS Style specified. Similarly, the color of the midpoint grid lines in particular charts depend on the device specified. For example, when you use ODS Style HighContrast with its black background, change from a black or dark-colored outline or axis color to a more visible color. Use the appropriate options available in the action statement that generates the chart. The COUTLINE= option changes the outline color of a block, bar, pie, or star chart. The CAXIS= option changes the axis color in a block or bar chart.

Two-dimensional bar charts created with the HBAR and VBAR statements can use the PATTERN statement to fill specified bars with specified images. For details, see the option description for “IMAGE=fileref | external-file” on page 399. Other means of including images in charts include adding background images to bar charts. The IBACK= option specifies an image file that fills the entire area behind the graph. The IFRAME= option specifies an image file that fills the area within the axes of the graph.

For additional information, including a list of recognized image file types, see “Image File Types Supported by SAS/GRAPH” on page 331.

---

**Syntax: GCHART Procedure**

**Restriction:** This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.
**PROC GCHART Statement**

Identifies the data set containing the chart variables. Can specify annotation and an output catalog.

**Requirement:** An input data set is required. If not specified, an input data set must have been created before issuing the PROC GCHART statement.

**Syntax**

```
PROC GCHART DATA=input-data-set
  <ANNOTATE=Annotate-data-set>
  <GOUT=<libref>:output-catalog>
  <IMAGEMAP=output-data-set>
  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)>;
```

**Required Argument**

**DATA=input-data-set**

specifies the SAS data set that contains the variable or variables to chart. If not specified, an input data set must have been created before issuing the PROC GCHART statement.

**Default** By default, the procedure uses the most recently created SAS data set.

**See** “The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*
Optional Arguments
PROC GCHART statement options affect all graphs produced by the procedure.

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See Chapter 27, “Using Annotate Data Sets,” on page 635

GOUT=<libref:><output-catalog
specifies the SAS catalog in which to save the graphics output that is produced by the GCHART procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

Restriction Not supported by Java or ActiveX

See “Specifying the Catalog Name and Entry Name for Your GRSEGs” on page 120

IMAGEMAP=output-data-set
creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192 and “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198

BLOCK Statement
Creates block charts in which the height of the blocks represents the value of the chart statistic for each category of data.

Requirement: At least one chart variable is required.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Supports: Drill-down functionality
Syntax

BLOCK chart-variable(s) <option(s)>;

Summary of Optional Arguments

Appearance options

- **ANNOTATE=Annotate-data-set**
  specifies a data set to annotate all graphs that are produced by the GCHART procedure.

- **BLOCKMAX=max-value**
  specifies the chart statistic value of the tallest block on the chart.

- **CAXIS=grid-color**
  specifies the color for the midpoint grid.

- **COUTLINE=block-outline-color | SAME**
  outlines all blocks or all block segments and legend values in the subgroup legend (if it appears) using the specified color.

- **CTEXT=text-color**
  specifies a color for all text on the grid and legend.

- **LEGEND=LEGEND<1 ..99>**
  assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

- **NOHEADING**
  suppresses the heading describing the type of statistic.

- **NOLEGEND**
  suppresses the legend automatically generated by the SUBGROUP= option.

- **PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP**
  specifies how fill patterns are assigned.

- **WOUTLINE=block-outline-width**
  specifies the width of the block outline in pixels.

Catalog entry description options

- **DESCRIPTION="description"**
  specifies a description of the output.

- **NAME="name"**
  specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Midpoint options

- **DISCRETE**
  treats a numeric chart variable as a discrete variable rather than as a continuous variable.

- **GROUP=group-variable**
  organizes the data according to the values of group-variable.

- **LEVELS=number-of-midpoints | ALL**
  specifies the number of midpoints to be graphed for the chart variable.

- **MIDPOINTS=OLD**

- **MIDPOINTS=value-list**
  specifies the midpoint values for the blocks.
MISSING
accepts a missing value as a valid midpoint for the chart variable.

SUBGROUP= subgroup-variable
divides the blocks into segments according to the values of subgroup-variable.

ODS options

HTML_LEGEND= variable
identifies the variable in the input data set whose values create links or data
tips or both.

HTML= variable
identifies the variable in the input data set whose values create links or data
tips or both.

URL= character-variable
specifies a character variable whose values are URLs.

Statistic options

FREQ= numeric-variable
specifies a variable whose values weight the contribution of each observation
in the computation of the chart statistic.

G100
calculates the percentage and cumulative percentage statistics separately for
each group.

STATFMT= format-specification
overrides the GCHART default format of the displayed statistical value.

SUMVAR= numeric-variable
specifies a numeric variable for sum or mean calculations.

TYPE= statistic
specifies the chart statistic.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to chart. Each chart
variable draws a separate chart. All variables must be in the input data set. Separate
multiple chart variables with blanks. The values of a chart variable used with the
BLOCK statement have a maximum length of 13.

See "Understanding Chart Variables" on page 879

Optional Arguments

Options in a BLOCK statement affect all graphs produced by that statement. You can
specify as many options as you want and list them in any order. For details about
specifying colors, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page
313. For a complete description of the graphics options, see Chapter 25, “Graphics
Options and Device Parameters Dictionary,” on page 515.

ANNOTATE= Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GCHART
procedure. To annotate individual graphs, use the ANNOTATE= option in the action
statement.

Alias ANNO=
Note  Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.

See  Chapter 27, “Using Annotate Data Sets,” on page 635

**BLOCKMAX=max-value**

specifies the chart statistic value of the tallest block on the chart. This option lets you produce a series of block charts using the same scale. All blocks are rescaled using max-value as if it were the maximum value on the chart.

Restriction  Not supported by Java or ActiveX

**CAXIS=grid-color**

specifies the color for the midpoint grid. By default, the midpoint grid uses the color of the current device. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

Style reference  Color attribute of the GraphAxisLines element

Note  Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

**COUTLINE=block-outline-color | SAME**

outlines all blocks or all block segments and legend values in the subgroup legend (if it appears) using the specified color. SAME specifies that the outline color of a block or a block segment or a legend value is the same as the interior pattern color.

The default outline color depends on the use of the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current device.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias  CO=

Style reference  Color attribute of the GraphOutlines element

Restriction  Partially supported by Java and ActiveX

Note  Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See  “Controlling Block Chart Patterns and Colors” on page 900

“About Patterns” on page 885

Example  “Example 2: Grouping and Subgrouping a Block Chart” on page 977

**CTEXT=text-color**

specifies a color for all text on the grid and legend. The GCHART procedure looks for the text color in the following order:
1. colors specified for labels and values on an assigned LEGEND statement, which
   override the CTEXT= option specified in the BLOCK statement.
2. the color specified by the CTEXT= option in the BLOCK statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, the default color when the
   NOGSTYLE option is specified is black for the Java and ActiveX devices and
   the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL=
    color is used for legend labels.

Alias          CT=
Style reference Color attribute of the GraphValueText and the GraphLabelText
elements

Note
If you use a BY statement in the procedure, the color of the BY
variable labels is controlled by the CBY= option in the
GOPTIONS statement

DESCRIPTION="description"
specifies a description of the output. The maximum length for description is 256
characters. The description does not appear in the output. The descriptive text is
shown in each of the following:

• the chart description for web output (depending on the device driver). See “Chart
  Descriptions for Web Presentations” on page 189 for more information
• the Table of Contents that is generated when you use the CONTENTS= option in
  an ODS HTML statement, assuming that the output is generated while the
  contents page is open
• the description and the properties for the output in the Results window
• the description and properties for the catalog entry in the SAS Explorer
• the Description field of the PROC GREPLAY window

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution
options, which work as they do when used on statements such as TITLE,
FOOTNOTE, and NOTE. Refer to “Substituting BY Line Values in a Text String” on
page 959. The 256-character limit applies before the substitution takes place for
these options. Thus, if in the SAS program the description text exceeds 256
characters, it is truncated to 256 characters, and then the substitution is performed.

Alias          DES=
Default        BLOCK CHART OF chart-variable
Style reference Color attributes of the GraphValueText and the GraphLabelText
elements

DISCRETE
treats a numeric chart variable as a discrete variable rather than as a continuous
variable. The GCHART procedure creates a separate midpoint and, hence, a separate
grid square and block for each unique value of the chart variable. If the chart variable
has a format associated with it, then each formatted value is treated as a midpoint.
The LEVELS=number-of-midpoints option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.

**FREQ=numeric-variable**

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times specified by the value of numeric-variable for that observation. If the value of numeric-variable is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of numeric-variable are truncated to integers.

FREQ= is valid with all chart statistics.

You cannot use the PERCENT, CPERCENT, FREQ, or CFREQ statistics with the SUMVAR= option. As a result, you must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum.

The statistics are not affected by applying a format to numeric-variable.

See “Calculating Weighted Statistics” on page 884

**G100**

calculates the percentage and cumulative percentage statistics separately for each group. When you use G100, the individual percentages reflect the contribution of the midpoint to the group and total 100% for each group. G100 is ignored unless you also use the GROUP= option.

By default, the individual percentages reflect the contribution of the midpoint to the entire chart and total 100% for the entire chart.

**GROUP=group-variable**

organizes the data according to the values of group-variable. Group-variable can be either character or numeric and is always treated as a discrete variable. The group variable can have up to 12 different values.

GROUP= produces a group grid that contains a separate row of blocks for each unique value of the group variable. Each row contains a square for each midpoint. The groups are arranged from front to back in ascending order of the group variable values. These values are printed to the left of each row; the group variable name or label is printed above the list of group values.

By default, each group includes all midpoints, even if no observations for the group fall within the midpoint range. Missing values for group-variable are treated as a valid group.

Example “Example 2: Grouping and Subgrouping a Block Chart” on page 977

**HTML=variable**

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction  Not supported by Java or ActiveX
See  “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

LEGEND=LEGEND<1 ...99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does not generate a legend. LEGEND= is ignored if the following is true:

• SUBGROUP= is not used.
• The specified LEGEND definition is not in effect.
• The NOLEGEND option is used.
• The PATTERNID= option is set to any value other than SUBGROUP. That is, the value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable:

block city / subgroup=city;

The Java and ActiveX devices do not support all LEGEND statement options. See the “LEGEND Statement” on page 377 for more information.

Restriction  Partially supported by Java and ActiveX
See  “SUBGROUP= subgroup-variable” on page 898
“LEGEND Statement” on page 377
Example  “Example 2: Grouping and Subgrouping a Block Chart” on page 977

LEVELS=number-of-midpoints | ALL
specifies the number of midpoints to be graphed for the chart variable. After you specify the number of midpoints that you want, the range for each midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). When you specify LEVELS=ALL, up to 25 unique numeric or character midpoint values are graphed. LEVELS=ALL is ignored if more than 25 values are specified. The LEVELS=number-of-midpoints option is ignored if any of the following is true:

• The chart variable is character type.
• The DISCRETE option is used.
• The MIDPOINTS= option is used.

MIDPOINTS=value-list
specifies the midpoint values for the blocks. The way you specify value-list depends on the type of variable:
For numeric chart variables, value-list is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:

- \( n < ... n \)
- \( n \) TO \( n < \text{BY increment} > \)
- \( n < ... n \) TO \( n < \text{BY increment} > < n < ... n > \)

If a numeric variable has an associated format, the specified values must be the unformatted values.

If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

- The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
- The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
- All other values in value-list specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

If you include the DISCRETE option, then each value in value-list specifies a unique numeric value.

For character chart variables, value-list is a list of unique character values enclosed in quotation marks and separated by blanks:

'value-1' < ...'value-n'>

If a character variable has an associated format, the specified values must be the formatted values.

For a complete description of value-list, see the “value-list” on page 353 in the AXIS statement.

If value-list for either type of variable specifies so many midpoints that the axis values overwrite each other, then the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, you can correct crowded values by increasing the number of cells in your graphics is displayed using the HPOS= and VPOS= graphics options.

See “Understanding Midpoints” on page 880

Example “Example 2: Grouping and Subgrouping a Block Chart” on page 977

MIDPOINTS=OLD

generates default midpoints using the Nelder algorithm (Applied Statistics 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

MISSING

accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored. Missing values are always valid for the group and subgroup variables.

NAME="name"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.
The following applies to name:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAHP appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

  **Note:** Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

**Default**  

GCHART

**NOHEADING**

suppresses the heading describing the type of statistic. For the Java and ActiveX devices, NOHEADING is the default. For other devices, by default the heading is printed at the top of each block chart.

**Restriction**  

Not supported by Java or ActiveX

**Example**  

“Example 2: Grouping and Subgrouping a Block Chart” on page 977

**NOLEGEND**

suppresses the legend automatically generated by the SUBGROUP= option. NOLEGEND is ignored if the SUBGROUP= option is not used.
PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned. By default, PATTERNID=SUBGROUP. Values for PATTERNID= are as follows:

BY
changes patterns each time the value of the BY variable changes. All blocks use the same pattern if the GCHART procedure does not include a BY statement.

GROUP
changes patterns every time the value of the group variable changes. All blocks in each group (row) use the same pattern, but a different pattern is used for each group.

MIDPOINT
changes patterns every time the midpoint value changes. If you use the GROUP= option, the respective midpoint patterns are repeated for each group.

SUBGROUP
changes patterns every time the value of the subgroup variable changes. The blocks must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all block faces have the same pattern.

Note
If you use the SUBGROUP= option and specify a PATTERNID= value other than SUBGROUP, the block segments use the same pattern and are indistinguishable.

See
“Controlling Block Chart Patterns and Colors” on page 900

Example
“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

STATFMT=format-specification
overrides the GCHART default format of the displayed statistical value. The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.

If you specify an option of TYPE=MEAN, an INSIDE=PCT option, and an OUTSIDE=SUM option, a STATFMT=f8.1 option applies only to the calculated TYPE=MEAN statistical value. For example:

```
proc gchart;
  block mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
    statfmt=f8.1;
run;
```

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the STATFMT= option.

If you change the previous example to specify an OUTSIDE=MEAN option, then STATFMT=f8.1 applies to the OUTSIDE=MEAN option and the TYPE=MEAN option. The statistical types match.

The STATFMT= option does not control the format of the response axis tick marks.

Alias
SFMT=, SFORMAT=, STATFORMAT=
**SUBGROUP=**\(\text{subgroup-variable}\)
divides the blocks into segments according to the values of **subgroup-variable**. **Subgroup-variable** can be either character or numeric and is always treated as a discrete variable. **SUBGROUP=** creates a separate segment within each block for every unique value of the subgroup variable for that midpoint.

If **PATTERNID=****SUBGROUP** (the default setting), each segment is filled with a different pattern, and a legend providing a key to the patterns is automatically generated. If the value of **PATTERNID=** is anything other than **SUBGROUP**, the segments are all the same color, the legend is suppressed, and the subgrouping effect is lost.

By default the legend appears at the bottom of the chart. To modify the legend, assign a **LEGEND** definition with the **LEGEND=** option. To suppress the legend, specify **NOLEGEND**.

See “**LEGEND Statement**” on page 377

**Example** “**Example 2: Grouping and Subgrouping a Block Chart**” on page 977

**SUMVAR=**\(\text{numeric-variable}\)
specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of **numeric-variable** for each midpoint. The resulting statistics are represented by the height of the blocks in each square. The values of a numeric variable used with the **BLOCK** statement have a maximum length of 8.

When you use the **SUMVAR=** option, the **TYPE=** option value must be either **SUM** or **MEAN**. With **SUMVAR=**, the default is **TYPE=SUM**.

**Example** “**Example 1: Specifying the Sum Statistic in a Block Chart**” on page 975

**TYPE=**\(\text{statistic}\)
specifies the chart statistic.

- If the **SUMVAR=** option is not used, **statistic** can be one of the following:
  - **FREQ**
    - frequency (the default)
  - **CFREQ**
    - cumulative frequency
  - **PERCENT** PCT
    - percentage
  - **CPERCENT** CPCT
    - cumulative percentage

- If **SUMVAR=** is used, **statistic** can be either:
  - **SUM**
    - sum (the default)
  - **MEAN**
    - mean

You cannot specify the statistics **PERCENT**, **CPERCENT**, **FREQ**, or **CFREQ** in conjunction with the **SUMVAR=** option. As a result, you must use **FREQ=** to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also “**Calculating Weighted Statistics**” on page 884.
If you specify TYPE=MEAN and use the SUBGROUP= option, the height of the block represents the mean for the entire midpoint. The subgroup segments are proportional to the subgroup's contribution to the sum for the block.

See  “About Chart Statistics” on page 883

Example  “Example 2: Grouping and Subgrouping a Block Chart” on page 977

URL=character-variable specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction  This option affects graphics output that is created through the ODS HTML destination only.

Interaction  If you specify both the HTML= and URL= options, then the URL= option is ignored

See  “Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

WOUTLINE=block-outline-width specifies the width of the block outline in pixels.

Style reference  LineThickness attribute of the GraphOutlines element

Restriction  Not supported by Java

Details

Description

The BLOCK statement specifies the variable or variables that define the categories of data to chart. This statement automatically does the following actions:

• determines the midpoints.
• calculates the chart statistic for each midpoint (the default is FREQ).
• scales the blocks according to the statistic value.
• assigns patterns and colors to the block faces and the grid. The default block pattern is solid.

You can use statement options to select or order the midpoints (blocks), to change the type of chart statistic, and to modify the appearance of the chart. You can also specify additional variables by which to group, subgroup, or sum the data.

Block charts enable grouping, which organizes the blocks into rows based on the values of a group variable. Block charts also enable subgrouping, which subdivides the blocks into segments based on the values of a subgroup variable.

In addition, you can use global statements to modify the block patterns and the legend, as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.

Note:  If you get a message that the chart is too large to display on your terminal or printer, try one or both of the following:
reduce the size of the character cells defined for the output device by specifying larger values for the HPOS= and VPOS= graphics options

- decrease the size of the chart text with the HTEXT= graphics option

See “The Graphics Output and Device Display Areas” on page 70 for details.

**Controlling Block Chart Patterns and Colors**

**Default Patterns and Outlines**

Each block in a block chart is filled with a pattern, but only the front faces of the blocks display the patterns. Because the system option, GSTYLE, is in effect by default, the procedure uses the style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the bars with bar or block patterns, beginning with the default fill, SOLID, and uses each color in the color list available in the default style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.

- outlines blocks and block segments using the color defined by the style.

- colors the midpoint grid with the color of the current style.

See “About Patterns” on page 885 for more information about how the GCHART procedure assigns default patterns and outlines.

**User-Defined Patterns**

To override the default patterns and select fills and colors for the blocks or block segments, use the PATTERN statement. Only bar or block patterns are valid; all other pattern fills are ignored.

For a complete description of all bar or block patterns, see the description of the PATTERN statement option “VALUE=bar/block-pattern” on page 400.

Whenever you use PATTERN statements, the default pattern outline color is that of the current style. Only when the EMPTY pattern is used does the pattern change to SAME. That is, the outline color is the same as the fill color. To specify the outline color, use the “COUTLINE=block-outline-color | SAME” on page 891.

**When Patterns Change**

The PATTERNID= option controls when the pattern changes. By default, PATTERNID=SUBGROUP. Therefore, when you use the SUBGROUP= option to subdivide the blocks, the pattern automatically changes each time the subgroup value changes. Each subdivision of the block displays a different pattern. As a result, the number of values for the SUBGROUP= variable determines the number of block patterns on the chart. If you do not subdivide the blocks, all blocks use the same pattern.

Instead of changing the pattern for each subgroup, you can change the pattern for each midpoint, each group, or each BY group, by changing the value of the PATTERNID= option. See the “About Patterns” on page 885 for details.

**Grid Color**

By default, grid elements use the color specified in the current style. To change the grid color, use the CAXIS= option. To change the grid text color, use the CTEXT= option.
**Controlling Block Chart Text**
To control the font and size of text on the chart, use the graphic options “FTEXT” on page 555 and “HTEXT” on page 575.

Because block charts do not use AXIS statements, you must use a LABEL statement instead to suppress the label for the midpoint variable. See “Example 2: Grouping and Subgrouping a Block Chart” on page 977.

**Displaying Negative or Zero Values**
The relative block heights in the chart represent the scaled value of the chart statistic value for the midpoint. The statistic can have a value of 0 or, in the case of sum and mean, a negative value. In these cases the base of the block is drawn in the square for the corresponding midpoint. Figure 35.13 on page 901 shows an example of a chart with 0 and negative statistic values.

![Figure 35.13 Block Chart with 0 and Negative Statistic Values](image)

**HBAR, HBAR3D, VBAR, and VBAR3D Statement**
Create horizontal or vertical bar charts in which the length or height of the bars represents the value of the chart statistic for each category of data.

- **Requirement:** At least one chart variable is required.
- **Global statements:** AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE
- **Supports:** Drill-down functionality

**Syntax**
```
HBAR | HBAR3D | VBAR | VBAR3D chart-variable(s) </option(s>);
```
Summary of Optional Arguments

Appearance options

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GCHART procedure.

CAUTOREF=reference-line-color
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option.

CAXIS=axis-color
specifies a color for the response and midpoint axis lines and for the default axis area frame.

CERROR=error-bar-color
specifies the color of error bars in bar charts.

CFRAME=background-color
specifies the color with which to fill the axis area in two-dimensional bar charts or the floor, back wall, and side wall of three-dimensional bar charts.

CGROUPREF (HBAR and VBAR only)
specifies the color of the reference line dividing each group of bars produced with the HBAR and VBAR statements.

CLM=confidence-level
specifies the confidence intervals to use when drawing error bars.

CMIDPOINTREF (HBAR and VBAR only)
specifies the color of the midpoint reference lines between bars produced with the HBAR and VBAR statements.

COUTLINE=bar-outline-color | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it appears) using the specified color.

CREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies colors for reference lines.

CTEXT=text-color
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME | NOFRAME
specifies whether a line is drawn around the axis area of a two-dimensional bar chart or a back wall is drawn in a three-dimensional bar chart.

GROUPREF (HBAR and VBAR only)
draws a reference line dividing each group of bars produced by the HBAR and VBAR statements.

GSPACE=group-spacing
specifies the amount of extra space between groups of bars.

IFRAME=fileref | 'external-file'
identifies the image file that you want to apply to a two-dimensional chart's axis area or a three-dimensional chart's back wall.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of an image, or stretch a single image.

LAUTOREF=reference-line-type
specifies the line type for reference lines at major tick marks, as determined by the AUTOREF option.

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

LGROUPREF=reference-line-type
specifies line types for reference lines dividing each group of bars produced by the HBAR and VBAR statements.

LMIDPOINTREF=reference-line-type
specifies line types for midpoint reference lines between bars produced by HBAR and VBAR statements.

LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines.

MIDPOINTREF (HBAR and VBAR only)
draws a reference line at the midpoint between each bar produced by the HBAR and VBAR statements.

NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option.

NOSYMBOL
suppresses the legend that is automatically generated by the SUBGROUP= option.

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned.

SHAPE=three-dimensional-bar-shape (HBAR3D and VBAR3D only)
specifies the shape of the bars in charts that are produced with the HBAR3D and VBAR3D statements.

SPACE=bar-spacing
specifies the amount of space between individual bars or between the bars within each group if you also use the GROUP= option.

SUBOUTSIDE=statistic (VBAR and VBAR3D only)
displays a list of the subgroup values of the specified statistic above the bars.

WAUTOREF=reference-line-width
specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option.

WGROUPREF=reference-line-width
specifies the line width for reference lines between groups of bars produced by the HBAR and VBAR statements.

WIDTH=bar-width
specifies the width of the bars.

WMIDPOINTREF=reference-line-width
specifies the line width for midpoint reference lines between bars produced by the HBAR and VBAR statements.

WOUTLINE=bar-outline-width
specifies the width of the outline in pixels.

WREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines.

Axis options

ASCENDING
arranges the bars in ascending order of the value of the chart statistic.

AUTOREF
draws a reference line at each major tick mark on the response axis.

AXIS=AXIS<1 …99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

**CLIPREF**
clips the reference lines at the bars.

**DESCENDING**
aranges the bars in descending order of the value of the chart statistic.

**FRONTREF**
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

**GAXIS= AXIS<1 ...99>**
assigns the specified AXIS definition to the group axis.

**MAXIS= AXIS<1 ...99>**
assigns the specified AXIS definition to the midpoint axis.

**MINOR= number-of-minor-ticks**
specifies the number of minor tick marks between each major tick mark on the response axis.

**NOAXIS**
suppresses all axes, including axis lines, axis labels, axis values, and all major and minor tick marks.

**NOBASEREF**
suppresses the zero reference line when the SUM or MEAN chart statistic has negative values.

**NOZERO**
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar.

**RANGE**
displays on the axis of the chart the range of numeric values represented by each bar.

**RAXIS= value-list | AXIS<1 ...99>**
**AXIS= value-list | AXIS<1 ...99>**
specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis.

**REF= value | (value) | (value-list)**
draws reference lines at the specified points on the response axis.

### Catalog entry description options

**DESCRIPTION= "description"**
specifies a description of the output.

**NAME= "name"**
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

### Midpoint options

**DISCRETE**
treats a numeric chart variable as a discrete variable rather than as a continuous variable.

**GROUP= group-variable**
organizes the data according to values of group-variable.

**LEVELS= number-of-midpoints | ALL**
specifies the number of midpoints to be graphed for a chart variable.

**MIDPOINTS= OLD**

MIDPOINTS=value-list
   specifies the midpoint values for the bars.

MISSING
   accepts a missing value as a valid midpoint for the chart variable.

SUBGROUP=subgroup-variable
   divides the bars into segments according to the values of subgroup-variable.

ODS options

HTML_LEGEND=variable
   identifies the variable in the input data set whose values create links or data tips or both.

HTML=variable
   identifies the variable in the input data set whose values create links or data tips or both.

URL=character-variable
   specifies a character variable whose values are URLs.

Statistic options

CFREQ
   displays the cumulative frequency statistic in the table of statistics and above vertical bars.

CFREQLABEL='column-label' | NONE (HBAR and HBAR3D only)
   specifies the text of the column label for the CFREQ statistic in the table of statistics.

CPERCENT
   displays the cumulative percentage statistic in the table of statistics and above vertical bars.

CPERCENTLABEL='column-label' | NONE (HBAR and HBAR3D only)
   specifies the text of the column label for the CPERCENT statistic in the table of statistics.

ERRORBAR=BARS | BOTH | TOP
   draws confidence levels.

FREQ
   displays the frequency statistic in the table of statistics and above vertical bars.

FREQ=numeric-variable
   specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

FREQLABEL='column-label' | NONE (HBAR and HBAR3D only)
   specifies the text of the column label for the FREQ statistic in the table of statistics.

G100
   calculates the percentage and cumulative percentage statistics separately for each group.

INSIDE=statistic
   displays the values of the specified statistic inside the bars.

MEAN
   displays the mean statistic in the table of statistics and above vertical bars.

MEANLABEL='column-label' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the MEAN statistic in the table of statistics.

NOSTATS (HBAR and HBAR3D only)
suppresses the table of statistics.

OUTSIDE=statistic
displays the values of the specified statistic above the bars.

PERCENT
prints the percentages of observations having a given value for the chart variable in the table of statistics and above vertical bars.

PERCENTLABEL='column-label' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the PERCENT statistic in the table of statistics.

PERCENTSUM
calculates a percent of the sum variable for horizontal or vertical bar charts.

STATFMT= format-specification
overrides the GCHART default format of the displayed statistical value.

SUM
displays the sum statistic in the table of statistics and above vertical bars.

SUMLABEL='column-label' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the SUM statistic in the table of statistics.

SUMVAR=numeric-variable
specifies a numeric variable for sum or mean calculations.

TYPE=statistic
specifies the chart statistic.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Multiple chart variables must be separated with blanks.

See “Understanding Chart Variables” on page 879

Optional Arguments

Options in an HBAR, HBAR3D, VBAR, or VBAR3D statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313. For details about specifying images, see Chapter 23, “Adding Images to SAS/GRAPH Output,” on page 331. For a complete description of the graphics options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

Note Annotate coordinate systems 1, 2, 7, and 8 (data system coordinates) are not valid with block, pie, donut, or star charts.
ASCENDING
arranges the bars in ascending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. The ASCENDING option reorders the bars from shortest to longest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right.

If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The ASCENDING option overrides any midpoint order specified with the MIDPOINTS= option or specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

AUTOREF
draws a reference line at each major tick mark on the response axis. To draw reference lines at specific points on the response axis, use the REF= option.

By default, reference lines in two-dimensional bar charts are drawn in front of the bars. To draw reference lines behind the bars, use the CLIPREF option.

By default, reference lines in three-dimensional bar charts are drawn on the back plane of the axis. To draw reference lines in front of the bars, use the FRONTREF option.

Example “Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

AXIS=AXIS<1 ...99>
specifies values for the major tick marks or assigns specified AXIS definitions to the axis.

CAUTOREF=reference-line-color
specifies the color of reference lines drawn at major tick marks, as determined by the AUTOREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified. To specify a line type for these reference lines, use the LAUTOREF= option.

Style reference Color attribute of the GraphGridLines element

CAXIS=axis-color
specifies a color for the response and midpoint axis lines and for the default axis area frame. If you omit the CAXIS= option, PROC GCHART searches for a color specification in this order:

1. the COLOR= option in AXIS definitions
2. the color specified in the current device. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

This option also specifies the default color for all reference lines.
**Style reference**

Color attribute of the GraphAxisLines attribute

**Note**

Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

**CERROR=error-bar-color**

specifies the color of error bars in bar charts. The default is the color of the response axis, which is controlled by the CAXIS= option.

**Style reference**

Color attribute of the GraphError element

**CFRAME=background-color**

specifies the color with which to fill the axis area in two-dimensional bar charts or the floor, back wall, and side wall of three-dimensional bar charts. The axis area color does not affect the frame color, which is always the same as the midpoint axis line color and controlled by the CAXIS= option. By default, the axis area in two-dimensional bar charts is not filled.

**Alias**

CFR=

**Style reference**

Color attribute of the GraphWalls element

**Restriction**

CFRAME= is overridden by the NOFRAME and IFRAME= options. If the IFRAME= option is in effect, an image appears in the axis area instead of the color.

**Note**

If the color of the background, the bar, and the outline are the same, you might not be able to distinguish the bars.

**See**

“FRAME | NOFRAME” on page 913

“IFRAME=fileref | ’external-file’” on page 917

**Example**

“Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

**CFREQ**

displays the cumulative frequency statistic in the table of statistics and above vertical bars. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ option is specified.

**Restriction**

The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

**Interaction**

Specifying the CFREQ option when creating an HBAR chart with an ActiveX device causes the cumulative frequencies to display at the end of each bar instead of the frequencies.

**See**

“About Chart Statistics” on page 883

“Displaying Statistics in Horizontal Bar Charts” on page 933

“Displaying Statistics in Vertical Bar Charts” on page 934
CFREQLABEL='column-label' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the CFREQ statistic in the table of statistics. 
Column-label can be up to 32 characters long, but a single line of the label can be no 
more than 24 characters. By default, a label with more than one word breaks as close 
to the center of the line as possible. A double space in the string forces a line break. 
To suppress the label, specify CFREQLABEL=NONE.

Restriction Not supported by Java or ActiveX

CGROUPREF (HBAR and VBAR only)
specifies the color of the reference line dividing each group of bars produced with 
the HBAR and VBAR statements. To draw reference lines between specific groups 
of bars produced with the HBAR and VBAR statements, use the GROUPREF= option.

CLIPREF clips the reference lines at the bars. This makes the reference lines appear to be 
behind the bars. Because the CLIPREF option is the default for three-dimensional 
bar charts, it affects only two-dimensional charts.

Example “Example 5: Controlling Midpoints and Statistics in a Horizontal Bar 
Chart” on page 985

CLM=confidence-level
specifies the confidence intervals to use when drawing error bars. Values for 
confidence-level must be greater than or equal to 50 and strictly less than 100. The 
default is 95. See “ERRORBAR=BARS | BOTH | TOP” on page 912 for details 
about how error bars are computed and drawn.

CMIDPOINTREF (HBAR and VBAR only)
specifies the color of the midpoint reference lines between bars produced with the 
HBAR and VBAR statements. To draw the reference lines between specific 
midpoints of bars produced with the HBAR and VBAR statements, use the 
MIDPOINTREF= option.

COUTLINE=bar-outline-color | SAME
outlines all bars or bar segments and legend values in the subgroup legend (if it 
appears) using the specified color. SAME specifies that the outline color of a bar or a 
bar segment or a legend value is the same as the interior pattern color.

The default outline color depends on the use of the PATTERN statement:

• If you do not specify a PATTERN statement, the default outline color is the color 
of the current device.

• If you specify the NOGSTYLE option and no PATTERN statement, the default 
outline color is black for the Java or ActiveX devices. Otherwise, the default 
outline color is the foreground color. If you specify an EMPTY PATTERN 
statement, then the default outline color is the same as the fill color.

Alias CO=

Style reference Color attribute of the GraphOutlines element

Note Use this option to specify a contrasting color when you use an ODS 
Style with a dark or black background, such as HighContrast.

See “Controlling Bar Chart Patterns, Colors, and Images” on page 935
“About Patterns” on page 885

Examples

“Example 3: Specifying the Sum Statistic in Bar Charts” on page 979

“Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

**CPERCENT**

displays the cumulative percentage statistic in the table of statistics and above vertical bars. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, or PERCENT option is specified.

**Alias**

CPCT=

**Restriction**

The table of statistic values is not printed when creating an HBAR chart with an ActiveX device

**Interaction**

Specifying the CPERCENT option when creating an HBAR chart with an ActiveX device causes the cumulative percentages to display at the end of each bar instead of the frequencies.

**See**

“About Chart Statistics” on page 883

“Displaying Statistics in Horizontal Bar Charts” on page 933

“Displaying Statistics in Vertical Bar Charts” on page 934

**CREF=** reference-line-color | (reference-line-color) | (reference-line-color-list)

specifies colors for reference lines. Specifying a single color without parentheses applies that color to all reference lines, including lines drawn with the AUTOREF and REF= options. Note that the CAUTOREF= option overrides the CREF= reference color for reference lines drawn with the AUTOREF option. Specifying a single color in parentheses applies that color to only the first reference line drawn with the REF= option. Specifying a reference color list applies colors in sequence to successive lines drawn with the REF= option. The syntax of the color list requires parentheses and color names separated by spaces (color1 color2 ...colorN) or separated by commas (color1, color2, ..., colorN). If you do not specify the CREF= option, the GCHART procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified. To specify line types for these reference lines, use the LREF= option.

**Alias**

CR=

**Style reference**

LineStyle attribute of the GraphGridLines element

**CPERCENTLABEL=** "column-label" | NONE (HBAR and HBAR3D only)

specifies the text of the column label for the CPERCENT statistic in the table of statistics. Column-label can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify CPERCENTLABEL=NONE.
Restriction Not supported by Java or ActiveX

CTEXT=text-color specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the BAR statement.
2. the color specified by the CTEXT= option in the BAR statement.
3. the color specified by the CTEXT= option in the GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. In this instance the COLOR= color is still used for all other axis colors, such as tick marks.

Alias CT=
Style reference Color attributes of the GraphValueText and the GraphLabelText elements
Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

DESCENDING arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. The DESCENDING option reorders the bars from longest to shortest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right. If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The DESCENDING option overrides any midpoint order that is specified with the MIDPOINTS= option. The DESCENDING option also overrides any midpoint order that is specified in the ORDER= option in an AXIS statement assigned to the midpoint axis.

DESCRIPTION="description" specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
the description and the properties for the output in the Results window.
• the description and properties for the catalog entry in the SAS Explorer.
• the Description field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=
Default BAR CHART OF chart-variable
Example “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**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=number-of-midpoints option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option. The ORDER= option in an AXIS statement that is assigned to the midpoint axis can rearrange or exclude discrete midpoint values.

Example “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**ERRORBAR=BARS | BOTH | TOP**
draws confidence intervals for either of the following:
• the mean of the SUMVAR= variable for each midpoint if you specify TYPE=MEAN
• the percentage of observations assigned to each midpoint if you specify TYPE=PCT with no SUMVAR= option

The ERRORBAR= option cannot be used with values of the TYPE= option other than MEAN or PCT. Valid values for the ERRORBAR= option are:

BARS
draws error bars as bars half the width of the main bars.

BOTH
draws error bars as two ticks joined by a line (default).

TOP
draws the error bar as a tick for the upper confidence limit that is joined to the top of the bar by a line.

By default, the ERRORBAR= option uses a confidence level of 95%. You can specify different confidence levels with the CLM= option.

When you use the ERRORBAR= option with TYPE=PCT, the confidence interval is based on a normal approximation. Let TOTAL be the total number of observations,
and PCT be the percentage assigned to a given midpoint. The standard error of the percentage is approximated as follows:

\[
\text{APSTDERR} = 100 \times \sqrt{\left(\frac{\text{PCT}}{100}\right) \times \left(1 - \frac{\text{PCT}}{100}\right) / \text{TOTAL}};
\]

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:

\[
\text{UCLP} = \text{PCT} + \text{APSTDERR} \times \text{PROBIT} \left(1 - \frac{1 - \text{LEVEL}/100}{2}\right);
\]

The lower confidence limit for the percentage is computed as follows:

\[
\text{LCLP} = \text{PCT} - \text{APSTDERR} \times \text{PROBIT} \left(1 - \frac{1 - \text{LEVEL}/100}{2}\right);
\]

When you use the ERRORBAR= option with TYPE=MEAN, the sum variable must have at least two nonmissing values for each midpoint. Let N be the number of observations assigned to a midpoint, MEAN be the mean of those observations, and STD be the standard deviation of the observations. The standard error of the mean is computed as follows:

\[
\text{STDERR} = \frac{\text{STD}}{\sqrt{\text{N}}};
\]

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:

\[
\text{UCLM} = \text{MEAN} + \text{STDERR} \times \text{TINV} \left(\frac{1 - \text{LEVEL}/100}{2}, \text{N}-1\right);
\]

The lower confidence limit for the mean is computed as

\[
\text{LCLM} = \text{MEAN} - \text{STDERR} \times \text{TINV} \left(\frac{1 - \text{LEVEL}/100}{2}, \text{N}-1\right);
\]

In some instances, you want the error bars to represent a given number C of standard errors instead of a confidence interval. If the number of observations assigned to each midpoint is the same, then you can find the appropriate value for the CLM= option by running a DATA step. For example, suppose you want error bars that represent one standard error (C=1) with a sample size of N. Run the following DATA step to compute the appropriate value for the CLM= option and assign that value to a macro variable &LEVEL:

```plaintext
data _null_;  
c = 1; n = 10;  
level = 100 * (1 - 2 * (1 - probt( c, n-1)));  
put _all_;  
call symput("level",put(level,best12.));  
run;
```

Then when you run the GCHART procedure, you can specify CLM=&LEVEL.

Note that this trick does not work precisely if different midpoints have different numbers of observations. However, choosing an average value for N can yield sufficiently accurate results for graphical purposes if the sample sizes are large or do not vary much.

If you want to create a chart with a legend and error bars, use the SUBGROUP= option with the ERRORBAR= option, and set the SUBGROUP= option to the midpoint variable.

**FRAME | NOFRAME**

specifies whether a line is drawn around the axis area of a two-dimensional bar chart or a back wall is drawn in a three-dimensional bar chart. The default is FRAME. Specifying the NOFRAME option removes the axis area frame from two-dimensional charts, including any background color or image. For three-dimensional charts, NOFRAME removes the back wall color or image, and leaves the vertical and horizontal axis planes and axes. To remove these planes, use the NOPLANE option.
in the AXIS statement. To remove one or more axis elements, use either the AXIS statement or the NOAXIS option.

The color of the frame is the color of the midpoint axis, which is determined by the CAXIS= option.

**Alias**  
FR | NOFR

**Restriction**  
The NOFRAME option overrides the CFRAMEx and IFRAmax options and the goption “IBACK” on page 576.

**See**  
“CFRAMEx=background-color” on page 908

“IFRAmax=fileref | 'external-file’” on page 917

**Examples**  
“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

**FREQ**  
displays the frequency statistic in the table of statistics and above vertical bars. Non-integer values are rounded down to the nearest integer. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values. This option overrides the CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

**See**  
“About Chart Statistics” on page 883

“Displaying Statistics in Horizontal Bar Charts” on page 933

“Displaying Statistics in Vertical Bar Charts” on page 934

**Example**  
“Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

**FREQLABEL='column-label' | NONE (HBAR and HBAR3D only)**
specifies the text of the column label for the FREQ statistic in the table of statistics. column-label can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify FREQLABEL=NONE.

**Restriction**  
Not supported by Java or ActiveX

**Examples**  
“Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

**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`.

<table>
<thead>
<tr>
<th>Restriction</th>
<th>The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>When specifying the ACTIVEX device with an HBAR chart, the frequencies are displayed at the end of each bar by default.</td>
</tr>
<tr>
<td>See</td>
<td>“Calculating Weighted Statistics” on page 884</td>
</tr>
</tbody>
</table>

**FRONTREF**

specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars. By default, reference lines in three-dimensional bar charts are drawn on the back plane of the axis.

**G100**

calculates the percentage and cumulative percentage statistics separately for each group. When you use the G100 option, the individual percentages reflect the contribution of the midpoint to the group and total 100% for each group. The G100 option is ignored unless you also use the GROUP= option.

By default, the individual percentages reflect the contribution of the midpoint to the entire chart and total 100% for the entire chart.

**GAXIS=AXIS<1 ...99>**

assigns the specified AXIS definition to the group axis. (A group axis is created when you use the GROUP= option.) You can use the AXIS definition to modify the order of the groups, the text of the labels, and appearance of the axis. The GAXIS= option is ignored if the specified AXIS definition does not exist.

The AXIS statement options `MAJOR=` and `MINOR=` are ignored in AXIS definitions assigned to the group axis because the axis does not use tick marks. A warning message is written to the SAS log if these options appear in the AXIS definition.

The Java and ActiveX devices do not support all AXIS statement options. See the “AXIS Statement” on page 345 for more information.

To remove groups from the chart, use the `ORDER=` option in the AXIS statement.

To suppress the brackets drawn around the values on the group axis in vertical bar charts, use the `NOBRACKETS` option in the AXIS statement.

**Restriction** Partially supported by Java and ActiveX

**See**

“AXIS Statement” on page 345

**Example**

“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**GROUP=group-variable**

organizes the data according to values of `group-variable`. `Group-variable` can be either character or numeric and is always treated as a discrete variable.
The GROUP= option produces a separate group of bars for each unique value of the group variable. Missing values for group-variable are treated as a valid group. The groups are arranged in ascending order of the group variable values.

By default, each group includes all midpoints, even if no observations for the group fall within the midpoint range, meaning that no bar is drawn at the midpoint. Use the NOZERO option to suppress midpoints with no observations.

The GROUP= option also produces a group axis that lists the values that distinguish the groups. The group axis has no axis line but displays up to 256 characters of the group variable name or label. To modify the group axis, assign an AXIS definition with the GAXIS= option.

In horizontal bar charts, the group axis is to the left of the midpoint axis. The groups are arranged from top to bottom, starting with the lowest value at the top.

In vertical bar charts, the group axis is below the midpoint axis and the groups are arranged from left to right starting with the lowest value at the left. The group label in a vertical bar chart can be narrower than all the bars in the group. In this case brackets are added to the label to emphasize which bars belong in each group. Group brackets are not displayed if the space between the group values is less than one and one-half character cells. Use the NOBRACKETS option in the AXIS statement to suppress the group brackets.

Example “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

GROUPREF (HBAR and VBAR only)
draws a reference line dividing each group of bars produced by the HBAR and VBAR statements. To draw reference lines at specific points on the response axis, use the REF= option.

GSPACE=group-spacing
specifies the amount of extra space between groups of bars. Group-space can be any nonnegative number. Units are character cells. Use GSPACE=0 to leave no extra space between adjacent groups of bars. In this case, the same space appears between groups of bars as between the bars in the same group.

The GSPACE= option is ignored unless you also use the GROUP= option. By default, the GCHART procedure calculates group spacing based on the size of the axis area and the number of bars in the chart.

Example “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
**HTML_LEGEND=variable**

Identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

*Restriction* Not supported by Java or ActiveX

*See* "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" on page 192

**IFRAME=fileref | 'external-file'**

Identifies the image file that you want to apply to a two-dimensional chart's axis area or a three-dimensional chart's back wall. *fileref* must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement. *external-file* must specify the complete filename of the image file that you want to use. The format of *external-file* varies across operating environments.

*Restriction* Not supported by Java

*Interactions* This option is overridden by the NOIMAGEPRINT goption. For information about the NOIMAGEPRINT goption, see “IMAGEPRINT” on page 577.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

*See* the goption “IBACK” on page 576 to fill the background surrounding two- and three-dimensional bar charts

“IMAGESTYLE=TILE | FIT” on page 917

“Displaying Images on Data Elements” on page 336

“CFRAME=background-color” on page 908

**IMAGESTYLE=TILE | FIT**

Specifies whether to use multiple instances of an image, or stretch a single image. Use multiple instances of an image to fill the axis area or back wall (TILE). You can choose to stretch a single instance of an image to fill the axis area or back wall (FIT). The TILE value is the default.

*Restriction* Not supported by Java

*See* “IFRAME=fileref | 'external-file’” on page 917

**INSIDE=statistic**

Displays the values of the specified statistic inside the bars. For the Java and ActiveX devices, this option is valid for both horizontal and vertical bar charts. However, Java and ActiveX devices do not support INSIDE=PERCENTSUM. For other devices, this option is valid only for vertical bar charts. For subgrouped bar charts generated with the Java and ActiveX devices, you can display only one statistic for each bar. These devices do not create both inside and outside bar labels. For graphs generated with the Java and ActiveX devices, the OUTSIDE= option overrides the INSIDE= option.
Statistic can be one of the following:

- FREQ
- CFREQ
- CPERCENT | CPCT
- MEAN
- PERCENT | PCT
- PERCENTSUM | PCTSUM
- SUM

If the bars are subgrouped, only the following statistics are valid:

- FREQ
- PERCENT | PCT
- SUBPCT
- PERCENTSUM | PCTSUM
- SUM

With subgroups, PERCENT displays the percent contribution of each subgroup to the midpoint value of the bar, based on frequency. The PERCENT values for each subgroup total the percent contribution of the bar to the whole. For example, if the percent contribution of the whole bar is 60%, the PERCENT statistic for all the subgroups in that bar are 60% total. To calculate PERCENT based on the SUMVAR= variable, use the FREQ= and TYPE= options. For details, see “Calculating Weighted Statistics” on page 884.

SUBPCT displays the percent contribution of each subgroup to the total bar. The SUBPCT values for each subgroup total the percent contribution to the whole bar. Because of rounding, the total of the percents might not equal 100.

See

- “About Chart Statistics” on page 883
- “Displaying Statistics in Horizontal Bar Charts” on page 933
- “Displaying Statistics in Vertical Bar Charts” on page 934

Examples

- “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982
- “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**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>

generates the specified LEGEND definition to the legend generated by the
SUBGROUP= option. The LEGEND= option itself does not generate a legend.

LEGEND= is ignored if any of the following is true:

- The SUBGROUP= option is not used.
- The specified LEGEND definition is not in effect.
- The NOLEGEND option is used.
- The PATTERNID= option is set to any value other than SUBGROUP. That is, the
  value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the
chart variable as the subgroup variable:

```
hbar city / subgroup=city;
```

Restriction: The Java and ActiveX devices do not support all LEGEND statement
options.

See: “LEGEND Statement” on page 377
    “SUBGROUP= subgroup-variable” on page 928

Example: “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on
page 982

LEVELS= number-of-midpoints | ALL

specifies the number of midpoints to be graphed for a chart variable. After you
specify the number of midpoints that you want, the range for each midpoint is
calculated automatically, using the algorithm in Terrell and Scott (1985).

If you specify LEVELS=ALL, then all unique numeric or character midpoint values
are graphed. You might have instances where the data contains a large number of
unique midpoint values (more than 200). In this case you can use the XPIXELS and
YPIXELS GOPTIONS to enable the device driver to render a larger (and more
readable) graph.

The LEVELS=number-of-midpoints option is ignored if any of the following are
true:

- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

LGROUPREF=reference-line-type

specifies line types for reference lines dividing each group of bars produced by the
HBAR and VBAR statements. Line types are specified as whole numbers from 1 to
46, with 1 representing a solid line and the other values representing dashed lines.
The default line type is retrieved from the current style, or if the NOGSTYLE option
is specified, the default value is 1, which draws a solid line. To specify a color for
these reference lines, use the CGROUPREF= option. To specify a width for these
reference lines, use the WGROUPREF= option.
LMIDPOINTREF=reference-line-type
specifies line types for midpoint reference lines between bars produced by HBAR and VBAR statements. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify a color for these reference lines, use the CMIDPOINTREF= option. To specify a width for these reference lines, use the WMIDPOINTREF= option.

Style reference LineStyle attribute of the GraphGridLines element

LREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. Specifying a line type without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the LAUTOREF= option overrides LREF=reference-line-type for reference lines drawn with the AUTOREF option. Specifying a single line type in parentheses applies that line type to the first reference line drawn with the REF= option. Specifying a line type list applies line types in sequence to successive reference lines drawn with the REF= option. The syntax of the line-type list requires parentheses and line types separated by spaces (type1 type2 ...typeN). If you do not specify the LREF= option, the GCHART procedure uses the type specified by the AXIS statement's STYLE= option. If neither option is specified, the default line type is retrieved from the current style. However, if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CREF= option.

Alias LR=

Style reference LineStyle attribute of the GraphGridLines element

Restriction Not supported by Java

MAXIS=AXIS<1 …99>
assigns the specified AXIS definition to the midpoint axis. The MAXIS= option is ignored if the specified AXIS definition does not exist.

Restriction The Java and ActiveX devices do not support all AXIS statement options.

See “AXIS Statement” on page 345
“Understanding Midpoints” on page 880

Example “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

MEAN
displays the mean statistic in the table of statistics and above vertical bars. By default, the column heading in the table includes the name of the variable for which the mean is calculated. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, PERCENT, CPERCENT, or SUM option is specified. The MEAN option is ignored unless you also use the SUMVAR= option.
Restriction

The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction

Specifying the MEAN option when creating an HBAR chart with an ActiveX device causes the means to display at the end of each bar instead of the frequencies.

See

“About Chart Statistics” on page 883

“Displaying Statistics in Horizontal Bar Charts” on page 933

“Displaying Statistics in Vertical Bar Charts” on page 934

MEANLABEL='column-label' | NONE (HBAR and HBAR3D only)

specifies the text of the column label for the MEAN statistic in the table of statistics. column-label can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify MEANLABEL=NONE.

Restriction

Not supported by Java and ActiveX

Example

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

MIDPOINTREF (HBAR and VBAR only)

draws a reference line at the midpoint between each bar produced by the HBAR and VBAR statements. To draw reference lines at specific points on the response axis, use the REF= option.

MIDPOINTS=value-list

specifies the midpoint values for the bars. The way you specify value-list depends on the type of the chart variable.

• For numeric chart variables, value-list is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  • n < ...n>
  • n TO n <BY increment>
  • n< ...n> TO n <BY increment> <n < ...n>>

If a numeric variable has an associated format, the specified values must be the unformatted values.

If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:

• The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
• The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
• All other values in value-list specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

If you include the DISCRETE option, then each value in value-list specifies a unique numeric value.
• For character chart variables, value-list is a list of unique character values enclosed in quotation marks and separated by blanks:
  'value-1' < ...'value-n'>

If a character variable has an associated format, the specified values must be the formatted values.

You are able to specify up to 256 characters for each variable value.

For a complete description of value-list, see “ORDER=(value-list)” on page 353 in the AXIS statement.

If the value-list for either type of variable specifies so many midpoints that the axis values overwrite each other, the values might be unreadable. In this case the procedure writes a warning to the SAS log. On many devices, this problem can be corrected by adjusting the size of the text with the HTEXT= graphics option. You can also correct the problem by increasing the number of cells in your graphics display. Do this by using the HPOS= and VPOS= graphics options.

The ORDER= option in the AXIS statement overrides the order specified in the MIDPOINTS= option. The bar chart statement options ASCENDING and DESCENDING also override both the MIDPOINTS= and ORDER= options in the AXIS statement.

Note Any character bar variable value with a long text string has the potential to cause a scaling issue and might produce a wider bar chart than desired.

See “Understanding Midpoints” on page 880

Example “Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

MIDPOINTS=OLD

generates default midpoints using the Nelder algorithm (Applied Statistics 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

MINOR=number-of-minor-ticks

specifies the number of minor tick marks between each major tick mark on the response axis. The MINOR= option in a bar chart statement overrides the number of minor tick marks specified in the MINOR= option in an AXIS definition. The AXIS definition is assigned to the response axis with the RAXIS= option.

MISSING

accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored. Missing values are always valid for group and subgroup variables.

NAME="name"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

• The name can be up to 256 characters in length.
• Special characters in the name are converted to underscores.
• For the GRSEG entry name:
  • The name is truncated to eight characters.
• The first character is always represented in uppercase, and all other characters are represented in lowercase.
• If the name begins with a number, an underscore is prepended to the name.
• If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

For the graphics output filename:
• The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
• All characters are represented in lowercase.
• If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
• If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
• If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.
• The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GCHART

Example “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

NOAXIS

suppresses all axes, including axis lines, axis labels, axis values, and all major and minor tick marks. If you specify an axis definition with the GAXIS, MAXIS=, or RAXIS= options, then the axes are generated as defined in the AXIS statement. But in this case all lines, labels, values, and tick marks are suppressed. Therefore, AXIS statement options such as ORDER=, LENGTH, and OFFSET= are still used.

To remove only selected axis elements such as lines, values or labels, use specific AXIS statement options.

The NOAXIS option does not suppress either the default frame or an axis area fill requested by the CFRAMES= option. To remove the frame around the two-dimensional axis area or the three-dimensional back wall, use the NOFRAME option in the procedure. To remove the horizontal or vertical axis planes, use the NOPLANE option in the AXIS statement.

NOBASEREF

suppresses the zero reference line when the SUM or MEAN chart statistic has negative values.
NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option.

Alias NOSYMBOL

Interaction
The NOLEGEND option is ignored if the SUBGROUP= option is not used.

NOSTATS (HBAR and HBAR3D only)
suppresses the table of statistics. The NOSTATS option suppresses both the default statistics and specific statistics requested by the FREQ, CFREQ, PERCENT, CPERCENT, SUM, and MEAN options.

Restriction
Not supported by Java

NOSYMBOL
suppresses the legend that is automatically generated by the SUBGROUP= option.

Alias NOLEGEND

Interaction
The NOSYMBOL option is ignored if the SUBGROUP= option is not used.

NOZERO
suppresses any midpoints for which there are no corresponding values of the chart variable and, hence, no bar. The NOZERO option is typically used with the GROUP= option. This combination suppresses midpoints when not all values of the chart variable are present for every group or if the chart statistic for the bar is 0.

Note
There are instances when a bar is omitted and bar labels are specified with the VALUE= option in an AXIS statement. This might result in the labels being shifted and not displaying with the correct bar.

Example
“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

OUTSIDE=statistic
displays the values of the specified statistic above the bars.

For example, OUTSIDE=PERCENT displays the percentages of observations having a given value for the chart variable. For the Java and ActiveX devices, this option is valid for both horizontal and vertical bar charts. However, a restriction exists when specifying OUTSIDE=PERCENTSUM. Although it is supported by the ACTXIMG device it is not supported by the JAVA, ACTIVEX, or JAVAIMG devices. For other devices, this OUTSIDE= option is valid only for vertical bar charts. For subgrouped bar charts generated with the JAVA and ACTIVEX devices, you can display only one statistic for each bar. These devices do not create both inside and outside bar labels. For graphs generated with the JAVA and ACTIVEX devices, the OUTSIDE= option overrides the INSIDE= option.

Statistic can be one of the following:

- FREQ
- CFREQ
- PERCENT | PCT
- CPERCENT | CPCT
- PERCENTSUM | PCTSUM
• SUM
• MEAN

See “About Chart Statistics” on page 883 for definitions of each of these statistics.

“Displaying Statistics in Horizontal Bar Charts” on page 933

“Displaying Statistics in Vertical Bar Charts” on page 934

Examples “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP
specifies how fill patterns are assigned. By default, PATTERNID=SUBGROUP.
Values for PATTERNID= are as follows:

BY
changes patterns each time the value of the BY variable changes. All bars use the same pattern if the GCHART procedure does not include a BY statement.

GROUP
changes patterns every time the value of the group variable changes. All bars in each group use the same pattern, but a different pattern is used for each group.

MIDPOINT
changes patterns every time the midpoint value changes. If you use the GROUP= option, the respective midpoint patterns are repeated for each group.

SUBGROUP
changes patterns every time the value of the subgroup variable changes. The bars must be subdivided by the SUBGROUP= option for the SUBGROUP value to have an effect. Without the SUBGROUP= option, all bars have the same pattern.

Note If you use the SUBGROUP= option and specify a PATTERNID= value other than SUBGROUP, the bar segments use the same pattern and are indistinguishable.

See “Controlling Bar Chart Patterns, Colors, and Images” on page 935

Examples “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

PERCENT
prints the percentages of observations having a given value for the chart variable in the table of statistics and above vertical bars. If a chart variable is not specified, percentages are calculated from the weighted contribution of each observation (frequency). Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ or CFREQ option is specified.

Alias PCT=
Restriction  The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the PERCENT option when creating an HBAR chart with an ActiveX device causes the percentages to display at the end of each bar instead of the frequencies.

See “About Chart Statistics” on page 883
“Displaying Statistics in Horizontal Bar Charts” on page 933
“Displaying Statistics in Vertical Bar Charts” on page 934

PERCENTLABEL=’column-label’ | NONE  (HBAR and HBAR3D only)

specifies the text of the column label for the PERCENT statistic in the table of statistics. column-label can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify PERCENTLABEL=NONE.

Restriction  Not supported by Java and ActiveX

PERCENTSUM  calculates a percent of the sum variable for horizontal or vertical bar charts. The PERCENTSUM option is ignored if the SUMVAR= option is not specified.

Alias PCTSUM

Restriction  When PERCENTSUM is the statistic specified by the INSIDE= option, it is not supported by any Java or ActiveX device. The specification of OUTSIDE=PERCENTSUM is supported by the ACTXIMG device but is not supported by the JAVA, ActiveX, or JAVAIMG devices.

See “About Chart Statistics” on page 883
“Displaying Statistics in Horizontal Bar Charts” on page 933
“Displaying Statistics in Vertical Bar Charts” on page 934

RANGE  displays on the axis of the chart the range of numeric values represented by each bar. In the graphics output:

- The starting value of each range is indicated with the less-than symbol (<).
- The ending value is indicated with the greater-than-or-equal-to symbol (>=).
- The default midpoint axis label is “chart-variable RANGE.”

The RANGE option has no effect on axes that represent character data. By default, the values shown on the axis are determined by the value of the “MIDPOINTS=value-list” on page 921.

Interaction  If specified, the “DISCRETE” option overrides the RANGE option.

During the creation of vertical bar charts, when options RANGE and LEVELS= are both used, RANGE displays the endpoints of the range for each vertical bar. It displays this rather than just the numerical midpoint of the range as would be displayed with the LEVELS=
option alone. If necessary, RANGE automatically angles or rotates the values and they might be less readable.

If the RANGE, LEVELS=, and AXIS VALUE= options are all specified, the RANGE option is ignored and the LEVELS= option displays the numeric midpoint of each vertical bar.

\[ \text{RAXIS= value-list | AXIS<1 ...99>} \]
\[ \text{AXIS= value-list | AXIS<1 ...99>} \]

specifies values for the major tick mark divisions on the response axis or assigns the specified AXIS definition to the axis. See the “MIDPOINTS= value-list” on page 921 for a description of value-list. By default, the GCHART procedure scales the response axis automatically and provides an appropriate number of tick marks.

You can specify negative values. But note that negative values are reasonable only when TYPE=SUM or TYPE=MEAN and one or more of the sums or means are less than 0. Frequency and percentage values are never less than 0.

For lists of values, a separate major tick mark is created for each individual value. A warning message is written to the SAS log if the values are not evenly spaced.

If the values represented by the bars are larger than the highest tick mark value, the bars are truncated at the highest tick mark.

If you use a BY statement with the PROC GCHART statement, then the same response axes are produced for each BY group when RAXIS= value-list is used. The same is true if there is an ORDER= list in the AXIS statement assigned to the response axis.

Restriction

The Java and ActiveX devices do not support all AXIS statement options.

See

the “AXIS Statement” on page 345 for more information.

Examples

“Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

\[ \text{REF= value | (value) | (value-list)} \]

draws reference lines at the specified points on the response axis.

See the “MIDPOINTS= value-list” on page 921 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.

\[ \text{SHAPE= three-dimensional-bar-shape (HBAR3D and VBAR3D only)} \]

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
Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**SPACE=** `bar-spacing`

Specifies the amount of space between individual bars or between the bars within each group if you also use the `GROUP=` option. `Bar-spacing` can be any nonnegative number, including decimal values. Units are character cells. You cannot specify a unit of measure, such as inches or percent. By default, the GCHART procedure calculates spacing based on the size of the axis area and the width of the bars on the chart. Use `SPACE=0` to leave no space between adjacent bars.

The `SPACE=` option is ignored if the following is true:

- You specify the `WIDTH=` option and are using the Java or ActiveX devices.
- The specified spacing requests a chart that is too large to fit in the space available for the midpoint axis. In this case, a warning message is issued.

Examples

- “Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982
- “Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204

**STATFMT=** `format-specification`

Overrides the GCHART default format of the displayed statistical value. The `STATFMT=` option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.

If you specify an option of TYPE=MEAN, an INSIDE=PCT option, and an OUTSIDE=SUM option, a `STATFMT=f8.1` option applies only to the calculated TYPE=MEAN statistical value. For example:

```plaintext
proc gchart;
  vbar mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
        statfmt=f8.1;
run;
```

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the `STATFMT=` option.

If you change the previous example to specify an OUTSIDE=MEAN option, then `STATFMT=f8.1` applies to the OUTSIDE=MEAN option and the TYPE=MEAN option. The statistical types match.

The `STATFMT=` option does not control the format of the response axis tick marks.

**Alias** SFMT=, SFORMAT=, STATFORMAT=

**SUBGROUP=** `subgroup-variable`

Divides the bars into segments according to the values of `subgroup-variable`. `Subgroup-variable` can be either character or numeric and is always treated as a
discrete variable. SUBGROUP= creates a separate segment within each bar for every unique value of the subgroup variable for that midpoint.

If PATTERNID=SUBGROUP (the default setting), each segment is filled with a different pattern and a legend that provides a key to the patterns is automatically generated. If the value of PATTERNID= is anything other than SUBGROUP, the segments are all the same color, the legend is suppressed, and the subgrouping effect is lost.

By default the legend appears at the bottom of the chart. To modify the legend, assign a LEGEND definition with the LEGEND= option. To suppress the legend, specify NOLEGEND.

See “LEGEND Statement” on page 377

Examples

“Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982
“Example 2: Creating a Drill-Down HTML Presentation for the Web” on page 204
“Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart” on page 985

SUBOUTSIDE=statistic (VBAR and VBAR3D only)
displays a list of the subgroup values of the specified statistic above the bars. The values are listed vertically above each bar in the order in which they occur on the bar from the top down. This option is valid for VBAR and VBAR3D charts only. The Java and ActiveX devices do not support this option. For all of the other devices:

• SUBOUTSIDE= overrides OUTSIDE=.
• If you specify both the INSIDE= and SUBOUTSIDE= options, use the same statistic for both. If you specify a different statistic for the SUBOUTSIDE= option, SAS/GRAPH uses the INSIDE= statistic for SUBOUTSIDE= instead.

Statistic can be one of the following:

• FREQ
• CFREQ
• PERCENT | PCT
• CPERCENT | CPCT
• SUM

Restriction
Not supported by Java and ActiveX

See “About Chart Statistics” on page 883
“Displaying Statistics in Vertical Bar Charts” on page 934

SUM displays the sum statistic in the table of statistics and above vertical bars. By default, the column heading in the table includes the name of the variable for which the sum is calculated. Default statistics are suppressed when you request specific statistics. For vertical bar charts, this option is ignored if the bars are too narrow to avoid overlapping values or if the FREQ, CFREQ, PERCENT, or CPERCENT option is specified. SUM is ignored unless you also use the SUMVAR= option.
Restriction The table of statistic values is not printed when creating an HBAR chart with an ActiveX device.

Interaction Specifying the SUM option when creating an HBAR chart with an ActiveX device causes the sums to display at the end of each bar instead of the frequencies.

See “About Chart Statistics” on page 883
“Displaying Statistics in Horizontal Bar Charts” on page 933
“Displaying Statistics in Vertical Bar Charts” on page 934

SUMLABEL='column-label' | NONE (HBAR and HBAR3D only)
specifies the text of the column label for the SUM statistic in the table of statistics. Column-label can be up to 32 characters long, but a single line of the label can be no more than 24 characters. By default, a label with more than one word breaks as close to the center of the line as possible. A double space in the string forces a line break. To suppress the label, specify SUMLABEL=NONE.

Restriction Not supported by Java and ActiveX

SUMVAR=numeric-variable
specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of numeric-variable for each midpoint. The resulting statistics are represented by the length of the bars along the response axis, and they are displayed at major tick marks.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Examples “Example 3: Specifying the Sum Statistic in Bar Charts” on page 979
“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

TYPE=statistic
specifies the chart statistic.

- If the SUMVAR= option is not used, statistic can be one of the following:
  FREQ
  frequency (the default)
  CFREQ
  cumulative frequency
  PERCENT PCT
  percentage
  CPERCENT CPCT
  cumulative percentage

- If the SUMVAR= option is used, statistic can be either of the following:
  SUM
  sum (the default)
  MEAN
  mean
You cannot use TYPE=FREQ, TYPE=CFREQ, TYPE=PERCENT, or TYPE=CPERCENT with the SUMVAR= option. Therefore, you must use the FREQ= option to calculate percentages, cumulative percentages, frequencies, or cumulative frequencies based on a sum. See also “Calculating Weighted Statistics” on page 884.

If you specify TYPE=MEAN and use the SUBGROUP= option, the height or length of the bar represents the mean for the entire midpoint. The subgroup segments are proportional to the subgroup's contribution to the sum for the bar. See also the option description for “SUBGROUP=subgroup-variable” on page 928.

See “About Chart Statistics” on page 883 for a complete description of statistic types.

**Example**

“Example 6: Generating Error Bars in a Horizontal Bar Chart” on page 989

**URL=character-variable**

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

**Restriction**

This option affects graphics output that is created through the ODS HTML destination only.

**Interaction**

If you specify both the HTML= and URL= options, then the URL= option is ignored.

See “Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

**WAUTOREF=reference-line-width**

specifies the line width for reference lines at major tick marks, as determined by the AUTOREF option. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CAUTOREF= option.

**Style reference**

LineThickness attribute of the GraphGridLines element

**WGROUPREF=reference-line-width**

specifies the line width for reference lines between groups of bars produced by the HBAR and VBAR statements. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CGROUPREF= option. To specify a line type for these reference lines, use the LGROUPREF= option.

**Style reference**

LineThickness attribute of the GraphGridLines element

**WMIDPOINTREF=reference-line-width**

specifies the line width for midpoint reference lines between bars produced by the HBAR and VBAR statements. Line widths are specified as whole numbers. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the CMIDPOINTREF= option. To specify a line type for these reference lines, use the LMIDPOINTREF= option.
WIDTH=bar-width

specifies the width of the bars. By default, the GCHART procedure selects a bar width that accommodates the midpoint values displayed on the midpoint axis using a hardware font and a height of one cell. Units for bar-width are character cells. The value for bar-width must be greater than 0, but it does not have to be an integer, for example:

vbar site / width=1.5;

Sometimes the requested bar width results in a chart that is too large to fit in the space available for the midpoint axis. In this case the procedure issues a warning in the log and ignores the WIDTH= option. If the specified width is too narrow, the procedure displays the midpoint values vertically.

Example

“Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart” on page 982

WOUTLINE=bar-outline-width

specifies the width of the outline in pixels. The WOUTLINE= option affects both the bar and subgroup outlines.

Style reference LineThickness attribute of the GraphOutlines element

Restriction Not supported by Java

WREF=reference-line-width | (reference-line-width) | (reference-line-width-list)

specifies line widths for reference lines. Line widths are specified as whole numbers. Specifying a line width without parentheses applies that type to all reference lines drawn with the AUTOREF and REF= options. Note that the WAUTOREF= option overrides WREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the REF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the REF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the CREF= option.

Style reference LineThickness attribute of the GraphReference element

Restriction Not supported by Java

Details

Description

The HBAR, HBAR3D, VBAR, and VBAR3D statements specify the variable or variables that define the categories of data to chart. These statements automatically do the following:

- determine the midpoints.
- calculate the chart statistic for each midpoint (the default is FREQ).
- scale the response axis and the bars according to the statistic value.
- determine bar width and spacing.
• assign patterns to the bars; the default bar or block pattern is SOLID.
• draw a frame around the axis area using a color determined by the current style. Or, if the NOGSTYLE option is specified, use the first color in the device's color list.

You can use statement options to do the following:
• select or order the midpoints (bars)
• control the tick marks on the response axis
• change the type of chart statistic
• display specific statistics
• modify the appearance of the chart.

You can also specify additional variables by which to group, subgroup, or sum the data.

All bar charts allow grouping, which uses an additional category to organize the bars into groups, and subgrouping, which divides the bars into segments.

In addition, you can do the following:
• use global statements to modify the axes (including requesting a logarithmic axis), the bar patterns, and the legend. See Chapter 24, “SAS/GRAPH Statements,” on page 343 for more information.
• add titles, footnotes, and notes to the chart. See the “TITLE, FOOTNOTE, and NOTE Statements” on page 447 for more information.
• use an Annotate data set to enhance the chart. See Chapter 27, “Using Annotate Data Sets,” on page 635 for more information.
• display an image in the axis area or back wall. See the option description for “IFRAME=fileref | 'external-file'” on page 917.
• display images in the bars of an HBAR or VBAR chart, and with a device of ACTIVEX or ACTXIMG, in the bars of an HBAR3D or VBAR3D chart. See the option description for “IMAGE=fileref | "external-file"” on page 399 in the PATTERN statement.

The Chart Statistic and the Response Axis

Where Chart Statistic Values Are Displayed
In bar charts, the scale of values of the chart statistic is displayed on the response axis. By default, the response axis is divided into evenly spaced intervals identified with major tick marks that are labeled with the corresponding statistic value. Minor tick marks are evenly distributed between the major tick marks unless a log axis has been requested. For sum and mean statistics, the major tick marks are labeled with values of the SUMVAR= variable (formatted if the variable has an associated format). The response axis is also labeled with the statistic type.

Specifying Logarithmic Axes
Logarithmic axes can be specified with the AXIS statement. See the “AXIS Statement” on page 345 for a complete discussion.

Displaying Statistics in Horizontal Bar Charts

Details
For graphs generated with the Java and ActiveX devices, default statistics are not generated, but you can display one statistic at the end of each bar. To specify the statistic,
specify the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN option.

For graphs generated with other devices, the HBAR and HBAR3D statements print a table of statistic values to the right of the bars. When the value of TYPE= is FREQ, CFREQ, PERCENT, or CPERCENT, the frequency, cumulative frequency, percentage, and cumulative percentage statistics are printed next to the bars by default. When TYPE=SUM, the frequency and sum statistic values are printed by default. When TYPE=MEAN, the frequency and mean statistic values are printed by default. However, if you use the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN options to request specific statistics, the default statistics are not printed.

For sum and mean, the name of the SUMVAR= variable is added to the heading for the column of values.

**Specifying the Table of Statistics**

You can use the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, and MEAN options to select only certain statistics. Without the SUMVAR= option, only the frequency, cumulative frequency, percentage, and cumulative percentage statistics can be printed. With the SUMVAR= option, all statistics, including the sum and mean, can be printed. You can suppress all statistics with the NOSTATS option.

To change the column labels for any statistic in the table, use one or more of the statistic column label options: FREQLABEL=, CFREQLABEL=, PERCENTLABEL=, CPERCENTLABEL=, SUMLABEL=, and MEANLABEL=.

To control the font and size of the text in the table of statistics, use the graphics options “FTEXT” on page 555 and “HTEXT” on page 575.

**Displaying Statistics in Vertical Bar Charts**

Statistic values on vertical bar charts are not printed by default. You must explicitly request a statistic with the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, MEAN, INSIDE=, OUTSIDE=, or SUBOUTSIDE= option.

For graphs generated with the Java and ActiveX devices, you can display one statistic for each bar. For graphs generated with other devices, you can display up to two statistics. Statistics can be displayed either above the bars or inside the bars.

To specify a statistic that you want to display above the bars, specify the statistic option (FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, or MEAN) or specify OUTSIDE=statistic. When subgrouping is used, to display a list of the subgroup statistics above each bar, specify SUBOUTSIDE=statistic. To specify a statistic that you want to display inside the bars, specify INSIDE=statistic.

For graphs generated with the Java and ActiveX devices, the OUTSIDE= option overrides INSIDE=, and INSIDE= overrides the FREQ, CFREQ, PERCENT, CPERCENT, PERCENTSUM, SUM, and MEAN options. For graphs generated with other devices, the individual statistic options and the SUBOUTSIDE= option override the OUTSIDE= option.

If more than one statistic option is specified, only the highest priority statistic is displayed. The priority order, from highest to lowest, is as follows:

1. FREQ  
2. CFREQ  
3. PERCENT  
4. CPERCENT  
5. PERCENTSUM
6. SUM
7. MEAN

The bars must be wide enough to accommodate the text. You can adjust the width of the bars with the WIDTH= option. To control the font and size of the text, use the graphics options “FTEXT” on page 555 and “HTEXT” on page 575.

**Ordering and Selecting Midpoints**
To rearrange character or discrete numeric midpoint values or to select ranges for numeric values, use the MIDPOINTS= option. Changing the number of midpoints for numeric variables can change the range of values for individual midpoints. Note that it does not change the range of values for the chart as a whole. For details see “Understanding Midpoints” on page 880.

Like the MIDPOINTS= option, the ORDER= option in the AXIS statement can rearrange the order of the midpoints or suppress the display of discrete numeric or character values. However, the ORDER= option cannot calculate the midpoints for a continuous numeric variable, or exclude values from the calculations. For details, see the description of the “value-list” on page 353 in the AXIS statement.

**Controlling Bar Chart Patterns, Colors, and Images**

**Default Patterns and Outlines**
Each bar in a bar chart is filled with a pattern. Because the system option, GSTYLE, is in effect by default, the procedure uses the style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the bars with bar patterns, beginning with the default fill, SOLID, and rotates it through the list of colors available in the default style. When these colors are exhausted, the procedure a slightly modified version of the original color list. It continues in this fashion until each of the chart variables have been assigned a unique pattern.

If you use the default style colors and the first color in the list is either black or white, then the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.

- outlines bars and bar segments using the color defined by the style.

See “About Patterns” on page 885 for more information about how the GCHART procedure assigns default patterns and outlines.

**User-Defined Patterns**
To override the default patterns and select fills and colors for the bars or bar segments, use the PATTERN statement. Only bar or block patterns are valid; all other pattern fills are ignored. For a complete description of all bar or block patterns, see “VALUE=bar/block-pattern” on page 400 in the PATTERN statement.

Whenever you use PATTERN statements, the default pattern outline color is that of the current style. Only when the EMPTY pattern is used does the pattern change to SAME. That is, the outline color is the same as the fill color. To specify the outline color, use “COUTLINE=bar-outline-color | SAME” on page 909.

**When Patterns Change**
The PATTERNID= option controls when the pattern changes. By default, PATTERNID=SUBGROUP. Therefore, when you use the SUBGROUP= option to
subdivide the bars, the pattern automatically changes each time the subgroup value changes. Also, each subdivision of the bar displays a different pattern. As a result, the number of values for the SUBGROUP= variable determines the number of bar patterns on the chart. If you do not subdivide the bars, all bars use the same pattern.

Instead of changing the pattern for each subgroup, you can change the pattern for each midpoint, each group, or each BY group by changing the value of PATTERNID=. See the description for option “PATTERNID=BY | GROUP | MIDPOINT | SUBGROUP” on page 925 for details.

Axis Color
By default, axis elements use the colors specified in the current style or the colors that are specified by AXIS statement color options. However, axis statement options can also control the color of the axis lines, text, and frame.

<table>
<thead>
<tr>
<th>To change the color of ...</th>
<th>Use this option ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>the axis text</td>
<td>CTEXT=</td>
</tr>
<tr>
<td>the axis lines</td>
<td>CAXIS=</td>
</tr>
<tr>
<td>the area within the frame</td>
<td>CFRAME=</td>
</tr>
</tbody>
</table>

Adding Images to Bar Charts
You can apply images to the bars and to the axis area of two-dimensional bar charts developed with the HBAR and VBAR statements. In three-dimensional bar charts, you can apply images to the back wall, and to the bars when using certain interactive devices. For details, see Chapter 23, “Adding Images to SAS/GRAPH Output,” on page 331.

PIE, PIE3D, and DONUT Statement
create pie or donut charts. The size of a pie slice represents the value of the chart statistic for that category of data in relation to the total chart statistic for all categories.

<table>
<thead>
<tr>
<th>Requirement:</th>
<th>At least one chart variable is required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>FOOTNOTE, LEGEND, PATTERN, TITLE</td>
</tr>
<tr>
<td>statements:</td>
<td></td>
</tr>
<tr>
<td>Supports:</td>
<td>Drill-down functionality</td>
</tr>
</tbody>
</table>

Syntax
PIE | PIE3D | DONUT chart-variable(s) </ option(s)>;

Summary of Optional Arguments
Appearance options

- **ANNOTATE=Annotate-data-set**
  specifies a data set to annotate all graphs that are produced by the GCHART procedure.
- **CFILL=fill-color**
specifies one color for all patterns in the chart, regardless of whether the fill is solid or hatch.

**COUTLINE=** *slice-outline-color | SAME*
outlines all slices, rings (subgroups), and legend values (if a legend appears) in the specified color.

**DETAIL_RADIUS=** *percent (PIE and DONUT only)*
determines the size of the detail pie.

**EXPLODE=** *value-list | ALL*
pulls the specified slices slightly out from the rest of the pie for added emphasis.

**FILL=** *SOLID | X*
specifies the fill pattern for all slices in the chart.

**INVISIBLE=** *value-list*
makes the specified slices invisible, as if they had been removed from the pie.

**NOHEADING**
suppresses the heading that is normally printed at the top of each page or display of output for all devices except Java and ActiveX.

**RADIUS=** *value*
specifies the radius of the pie and donut in their respective charts.

**WOUTLINE=** *slice-outline-width*
specifies the width of the outline in pixels.

**Catalog entry description options**

**DESCRIPTION=** "*description"*
specifies a description of the output.

**NAME=** "*name"*
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

**Detail pie options**

**DETAIL_THRESHOLD=** *percent (PIE and DONUT only)*
determines whether a detail slice is included in the inner pie.

**DETAIL=** *variable (PIE and DONUT only)*
produces an inner pie overlay whose slices show the major components that comprise the outer pie's slice.

**Detail pie slice-labeling options**

**DETAIL_PERCENT=** *BEST | NONE (PIE and DONUT only)*
specifies the algorithm to use for displaying the percentage values for the detail pie slices.

**DETAIL_SLICE=** *BEST | NONE (PIE and DONUT only)*
specifies the algorithm to use for displaying the detail variable labels for the inner pie slices.

**DETAIL_VALUE=** *BEST | NONE (PIE and DONUT only)*
specifies the algorithm to use for displaying the data values for the detail pie slices.

**Donut-labeling options**

**DONUTPCT=** *percent (DONUT only)*
specifies the size of the donut hole in percent of the radius of the whole chart.

**LABEL=** (text argument(s)) (DONUT only)
defines the text that is displayed in the donut hole.
Grouping and subgrouping options

**ACROSS=number-of-columns**

draws *number-of-columns* pies across the procedure output area.

**DOWN=number-of-rows**

draws *number-of-rows* pies vertically in the procedure output area.

**GROUP=group-variable**

organizes the data according to values of *group-variable* and produces a separate pie (or donut) chart for each unique value of *group-variable*.

**NOGROUPHEADING**

suppresses the headings that are normally printed above each pie when you use the GROUP= option.

**SUBGROUP=subgroup-variable**

divides the chart into concentric rings according to the values of *subgroup-variable*.

Midpoint options

**DISCRETE**

treats a numeric chart variable as a discrete variable rather than as a continuous variable.

**LEVELS=number-of-midpoints | ALL**

specifies the number of midpoints to be graphed for a chart variable.

**MIDPOINTS=OLD**


**MIDPOINTS=value-list**

specifies the midpoint values for the slices.

**MISSING**

accepts a missing value as a valid midpoint for the chart variable.

**OTHER=percent-of-total**

collects all midpoints with chart statistic values less than or equal to *percent-of-total* into a generic midpoint named OTHER.

ODS options

**HTML_LEGEND=variable**

identifies the variable in the input data set whose values create links or data tips or both.

**HTML=variable**

identifies the variable in the input data set whose values create links or data tips or both.

**URL=character-variable**

specifies a character variable whose values are URLs.

Slice-labeling options

**CTEXT=text-color**

specifies a color for all text on the axes and legend.

**LEGEND | LEGEND=LEGEND<1 ...99>**

generates a legend for the slice names (midpoint values) instead of printing them beside the slices.

**MATCHCOLOR**

uses the slice pattern color for all slice labels.

**NOLEgend**
suppresses the legend that is automatically generated by the SUBGROUP= option.

OTHERCOLOR=color
specifies the color to use for the OTHER slice.

OTHERLABEL='text-string'
specifies a text string up to 16 characters for the label for the OTHER slice.

PERCENT=ARROW | INSIDE | NONE | OUTSIDE
prints the percentage represented by each slice using the specified labeling method.

PLABEL=(text-option(s))
modifies the font, height, and color of the pie slice labels.

PPERCENT=(text-option(s))
modifies the font, height, and color of the displayed percentages associated with the pie slice labels.

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice.

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Slice-ordering options

ANGLE=degrees
starts the first slice at the specified angle.

ASCENDING
arranges the slices in ascending order of the value of the chart statistic.

CLOCKWISE
draws the slices clockwise starting at the twelve o'clock position.

DESCENDING
arranges the slices in descending order of the value of the chart statistic.

JSTYLE
arranges the midpoints in descending order of the statistic value and draws the slices clockwise starting at the twelve o'clock position.

Statistic options

FREQ=numeric-variable
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

STATFMT=format-specification
overrides the GCHART default format of the displayed statistical value.

SUMVAR=numeric-variable
specifies a numeric variable for sum or mean calculations.

TYPE=statistic
specifies the chart statistic.

Required Argument

chart-variable(s)
specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Separate multiple chart variables with blanks.

See "Understanding Chart Variables” on page 879
Optional Arguments

Options in a PIE, PIE3D, or DONUT statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details about specifying colors, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313. For a complete description of the graphics options, see Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

**ACROSS=number-of-columns**

draws number-of-columns pies across the procedure output area. ACROSS is ignored unless you also use the GROUP= option.

If number-of-columns calls for more pies than fit horizontally in the graphics output area, no pies are drawn and an error message is written to the SAS log.

If you also use the DOWN= option, the pies are drawn in left-to-right and top-to-bottom order.

**ANGLE=degrees**

starts the first slice at the specified angle. A value of 0 for degrees corresponds to the three o'clock position. Degrees can be either positive or negative. Positive values move the starting position in the counterclockwise direction; negative values move the starting position clockwise. By default, ANGLE=0. Successive slices are drawn counterclockwise from the starting slice.

**ANNOTATE=Annotate-data-set**

specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

**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, it is possible that no pattern is specified in the pattern statement or with the FILL= option. In this case the procedure starts with the default solid fill and then, beginning with P2N0, uses each default pie hatch pattern with the specified color.

For the outline color, the procedure uses the default color, which is retrieved from the current style. However, if the NOGSTYLE option is specified, the procedure uses the first color in the device's color list. Use the COUTLINE= option to specify a
different outline color. The CFILL= option overrides any other pattern color specification and controls the color of all slices.

**Style reference**  
Color attribute of the GraphData1 element

**See**  
“Controlling Bar Chart Patterns, Colors, and Images” on page 935

“About Patterns” on page 885

**Example**  
“Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996

**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*

draws the slices clockwise starting at the twelve o'clock position. Although this position implies ANGLE=90, you can use the ANGLE= option to specify a different starting angle.

**COUTLINE=** *slice-outline-color | SAME*

outlines all slices, rings (subgroups), and legend values (if a legend appears) in the specified color. SAME specifies that the outline color of a slice or a slice segment or a legend value is the same as the interior pattern color.

The default outline color depends in the PATTERN statement:

- If you do not specify a PATTERN statement, the default outline color is the color of the current device.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

**Alias**  
**CO=**

**Style reference**  
Color attribute of the GraphOutlines element

**Note**

Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

**See**  
“Controlling Slice Patterns and Colors” on page 958

“About Patterns” on page 885

**Examples**

“Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991

“Example 8: Subgrouping a Donut or Pie Chart” on page 994

**CTEXT=** *text-color*
specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure looks for the text color in the following order:

1. the colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified on either the PIE or DONUT statement.
2. the color specified by the CTEXT= option in the PIE or DONUT statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

<table>
<thead>
<tr>
<th>Alias</th>
<th>CT=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style reference</td>
<td>Color attributes of the GraphValueText and the GraphLabelText elements</td>
</tr>
<tr>
<td>Note</td>
<td>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.</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 8: Subgrouping a Donut or Pie Chart” on page 994</td>
</tr>
</tbody>
</table>

**DESCENDING**

arranges the slices in descending order of the value of the chart statistic. By default, slices are arranged in ascending order of alphabetical or numeric midpoint value, without regard to size or summary statistic. DESCENDING reorders the slices from largest to smallest. The OTHER slice is still last, regardless of its size.

If you also use the GROUP= option, the reordering is performed separately for each group, so the order of midpoint values might be different for each pie or donut.

DESCENDING overrides any midpoint order that is specified with the MIDPOINTS= option.

**DESCRIPTION=**"**description**"

specifies a description of the output. The maximum length for **description** is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the **Description** field of the PROC GREPLAY window.

The **description** can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the **description** text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

| Alias | DES= |
Based on the chart type, either PIE CHART or DONUT CHART OF chart-variable

**DETAIL=variable (PIE and DONUT only)**
produces an inner pie overlay whose slices show the major components that comprise the outer pie's slice. **Variable** is the variable whose values are used to construct the detail pie. Detail pie slices are always displayed in descending order. If you specify the DETAIL= option and either GROUP= or SUBGROUP=, then the DETAIL= option is ignored.

**DETAIL_PERCENT=BEST | NONE (PIE and DONUT only)**
specifies the algorithm to use for displaying the percentage values for the detail pie slices. NONE turns off the display of the percentage values.

**DETAIL_RADIUS=percent (PIE and DONUT only)**
determines the size of the detail pie. **Percent** specifies the percent of the outer pie radius to use as the detail pie radius. The valid range is 25 to 90. The default is 75.

**DETAIL_SLICE=BEST | NONE (PIE and DONUT only)**
specifies the algorithm to use for displaying the detail variable labels for the inner pie slices. NONE turns off the display of the detail variable labels.

**DETAIL_THRESHOLD=percent (PIE and DONUT only)**
determines whether a detail slice is included in the inner pie. Any detail slice comprising **percent** or more percent of the whole pie is included. The valid range for **percent** is 0 to 75. The default is 4.

**DETAIL_VALUE=BEST | NONE (PIE and DONUT only)**
specifies the algorithm to use for displaying the data values for the detail pie slices. NONE turns off the display of the data values.

**DISCRETE**
treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a separate slice for each unique value of the chart variable. If the chart variable has a format associated with it, each formatted value is treated as a midpoint.

The **LEVELS=number-of-midpoints** option is ignored when you use the DISCRETE option. The **MIDPOINTS=** option overrides the DISCRETE option.

**DONUTPCT=percent (DONUT only)**
specifies the size of the donut hole in percent of the radius of the whole chart. Values of **percent** range from 0 to 99. By default, DONUTPCT=25.

Example “Example 8: Subgrouping a Donut or Pie Chart” on page 994

**DOWN=number-of-rows**
draws **number-of-rows** pies vertically in the procedure output area. The DOWN= option is ignored unless you also use the GROUP= option.

If **number-of-rows** calls for more pies than fit vertically in the graphics area of the output device, no pies are drawn and an error message is written to the SAS log.

If you also use the ACROSS= option, the pies are drawn in left-to-right and top-to-bottom order.

**EXPLODE=value-list | ALL**
pulls the specified slices slightly out from the rest of the pie for added emphasis. **Value-list** is the list of midpoint values for the slices to be exploded. For a description of **value-list**, see “**MIDPOINTS=value-list**” on page 948.
The values in the value list must match the existing midpoints exactly, including the case of character midpoints. Any values in the list that do not correspond to existing midpoints are ignored.

Using EXPLODE=ALL pulls all of the slices outward from the center of the pie. When you use the EXPLODE= option, the radius is reduced to allow room for exploded slices.

Restriction  EXPLODE=ALL is not supported by Java and ActiveX for the slice labeled ‘Other’.

Interaction  When used with subgroups, the EXPLODE= option is supported only by the ActiveX and Java devices.

Example  “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991

FILL=SOLID | X

specifies the fill pattern for all slices in the chart:

SOLID  
rotates a solid fill through the color list of the current style as many times as necessary. SOLID is the default.

Alias  S

X  
rotates a single hatch pattern through the list of colors defined in the current style. If the NOGSTYLE option is specified, it rotates the hatch pattern through the device color list as many times as necessary. If you do not specify the colors= goption, the fill skips the first color in the color list.

By default, the outline color is the color defined by the current style. If the NOGSTYLE option is specified, the outline color is the first color in the device's color list. If PATTERN statements are used to specify colors, the slice outline color matches the slice fill color.

Style reference  Color attribute of the GraphData1 element

Restriction  Partially supported by Java and ActiveX

Interactions  FILL= overrides any pattern that is specified in PATTERN statements.

If any PATTERN statements have been defined, the colors in the PATTERN definitions are used, in order, before the default style color rotation.

See  “Controlling Bar Chart Patterns, Colors, and Images” on page 935

“PATTERN Statement” on page 398

FREQ=numeric-variable

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times specified by the value of numeric-variable for that observation. If the value of numeric-variable is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of numeric-variable are truncated to integers.
FREQ= is valid with all chart statistics.

Because you cannot use TYPE=PERCENT or TYPE=FREQ with the SUMVAR= option, you must use the FREQ= option to calculate percentages and frequencies based on a sum.

The statistics are not affected by applying a format to numeric-variable.

See “Calculating Weighted Statistics” on page 884

GROUP=group-variable

organizes the data according to values of group-variable and produces a separate pie (or donut) chart for each unique value of group-variable. Group-variable can be either character or numeric and is always treated as a discrete variable. Missing values for group-variable are treated as a valid group. By default, each group includes only those midpoints with nonzero chart statistic values.

By default, the charts are produced in ascending order of group variable value and each is drawn on a separate page or display. Therefore, the effect of the GROUP= option is essentially the same as using a BY statement. The exception is that the GROUP= option causes the midpoints with the same value to use the same color and fill pattern. To place more than one pie on a page or display, use the ACROSS= or DOWN= options, or both.

See “BY Statement” on page 370

Example “Example 10: Grouping and Arranging Pie Charts” on page 998

HTML=variable

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

HTML_LEGEND=variable

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restrictions Not supported by Java and ActiveX

If either subgroups or the DETAIL= option are specified, then the HTML_LEGEND= option is ignored.

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
INVISIBLE=value-list
makes the specified slices invisible, as if they had been removed from the pie. Labels are not printed for invisible slices. Value-list is the list of midpoint values for the invisible slices. See the “MIDPOINTS=value-list” on page 948 for a description of value-list.

The values in the value list must match the existing midpoints exactly, including the case of character midpoints. Any values in the list that do not correspond to existing midpoints are ignored.

JSTYLE
arranges the midpoints in descending order of the statistic value and draws the slices clockwise starting at the twelve o'clock position. The JSTYLE option has the same effect as specifying both the DESCENDING and CLOCKWISE options.

LABEL=(text argument(s)) (DONUT only)
defines the text that is displayed in the donut hole. Text-argument(s) defines the text or the appearance of the label, or both. Text-argument(s) can be one or more of the following:

'text-string'
provides the text of the label. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVALn | #BYVAL(BY-variable-name)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. Variable-name is not case sensitive.

Examples
“Example 7: Using BY-group Processing to Generate a Series of Charts” on page 488
“Example 8: Combining Graphs and Reports in a Web Page” on page 495

#BYVARn | #BYVAR(BY-variable-name)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of n indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. Variable-name is not case sensitive.
Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See “Substituting BY Line Values in a Text String” on page 959

text-description-suboption modifies a characteristic such as the font, color, or size of the text string(s) that follows it. Text-description-suboption can be

- ANGLE=degrees
- COLOR=color
- FONT=font
- HEIGHT=text-height <units>
- JUSTIFY=LEFT | CENTER | RIGHT
- ROTATE=degrees

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Restrictions The Java and ActiveX devices do not support all of the suboptions. See “Text Description Suboptions for Donut” on page 954 for a complete description.

#BYVAL or #BYVAR substitution in a text string is not available in the DATA Step Graphics Interface or in the Annotate facility. This is because BY lines are not created in a DATA step.

Example “Example 8: Subgrouping a Donut or Pie Chart” on page 994

LEGEND | LEGEND=LEGEND<1 ...99>
generates a legend for the slice names (midpoint values) instead of printing them beside the slices. The legend displays each slice name and its associated pattern. This option also suppresses the display of the chart statistic values. To display the chart statistics, use the VALUE= option.

If you use the SUBGROUP= option, the legend is automatically generated. However, because patterning is always by midpoint, the legend still describes the midpoint values, not the subgroups.

Specifying LEGEND=LEGENDn assigns the specified LEGEND statement to the legend.

Restriction The Java and ActiveX devices do not support all LEGEND statement options. See the “LEGEND Statement” on page 377 for more information.

Note If you request a legend and the slices use hatch patterns, the patterns in the slices are oriented to be visually equivalent to the legend.

See “LEGEND Statement” on page 377

Example “Example 8: Subgrouping a Donut or Pie Chart” on page 994
LEVELS=\textit{number-of-midpoints} | ALL

specifies the number of midpoints to be graphed for a chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985).

If you specify \texttt{LEVELS=ALL}, then all unique numeric or character midpoint values are graphed. You might have instances where the data contains a large number of unique midpoint values (more than 200). In this case you can use the XPIXELS and YPIXELS GOPTIONS to enable the device driver to render a larger (and more readable) graph.

The \texttt{LEVELS=} option is ignored if any of the following is true:

- The chart variable is character type.
- The \texttt{MIDPOINTS=} option is used.

\textbf{MATCHCOLOR}

uses the slice pattern color for all slice labels. \texttt{MATCHCOLOR} overrides the color that is specified in the \texttt{CTEXT=} option.

\textbf{MIDPOINTS=}\textit{value-list}

specifies the midpoint values for the slices. The way you specify \textit{value-list} depends on the type of variable:

- For numeric chart variables, \textit{value-list} is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - \texttt{n < ...n>}
  - \texttt{n TO n <BY increment>}
  - \texttt{<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 \texttt{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 \textit{value-list} specify the median of a range of values, and the GCHART procedure calculates the midpoint values.

If you include the \texttt{DISCRETE} option, then each value in \textit{value-list} specifies a unique numeric value.

- For character chart variables, \textit{value-list} is a list of unique character values enclosed in quotation marks and separated by blanks:
  - \texttt{'value-1' < ...'value-n'>}

If a character variable has an associated format, the specified values must be the formatted values.

You are able to specify up to 256 characters for each variable value.

For a complete description of \textit{value-list}, see the “\textit{value-list}” on page 353 in the AXIS statement.
Midpoints that represent small percentages are collected into a generic midpoint named OTHER. See the descriptions of “OTHER=percent-of-total” on page 950 and “OTHERLABEL='text-string’” on page 951 for more information.

See “Understanding Midpoints” on page 880

Example “Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996

**MIDPOINTS=OLD**

generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976). The MIDPOINTS=OLD option is ignored unless the chart variable is numeric.

**MISSING**

accepts a missing value as a valid midpoint for the chart variable. By default, observations with a missing value are ignored. Missing values are always valid for the group and subgroup variable.

**NAME="name"**

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

**For the GRSEG entry name:**

- The name is truncated to eight characters.
- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, **SAS/GRAPH** appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

**For the graphics output filename:**

- The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
- If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

**Note:** Prior to **SAS 9.4M2**, if the name begins with a number, an underscore is prepended to the filename.
The maximum allowable filename length is device-specific. If the length of
the name exceeds the maximum for the graphics device, an error results and
no graphics output file is generated.

Default GCHART

NOGROUPHEADING
suppresses the headings that are normally printed above each pie when you use the
GROUP= option.

NOHEADING
suppresses the heading that is normally printed at the top of each page or display of
output for all devices except Java and ActiveX. For the Java and ActiveX devices,
NOHEADING is the default.

Restriction Not supported by Java and ActiveX

Example “Example 8: Subgrouping a Donut or Pie Chart” on page 994

NOLEGEND
suppresses the legend that is automatically generated by the SUBGROUP= option.
NOLEGEND is ignored if the SUBGROUP= option is not used.

OTHER=percent-of-total
collects all midpoints with chart statistic values less than or equal to percent-of-total
into a generic midpoint named OTHER. The value of percent-of-total can be 0 to
100; the default value is 4. Therefore, any slice that represents 4% or less of the total
is put in the OTHER category.

The OTHER slice is the last slice in the pie, regardless of the order of the slices. (In
other words, it is the slice immediately before the starting slice.)

If only one midpoint falls into the OTHER category, its slice is displayed in its
normal position in the pie and retains its original label. For example, suppose a pie
has these slices and percent values: Coal 35%, Gas 15%, Hydro 5%, and Oil 45%. If
you specify OTHER=5, Hydro remains the third slice instead of becoming the last
slice.

Note If you specify a small value for percent-of-total, the GCHART procedure
might not be able to label all of the small slices.

OTHERCOLOR=color
specifies the color to use for the OTHER slice. If you omit the OTHERCOLOR= option, GCHART searches for a color specification in this order:

1. the CFILL= option
2. the COLOR= option in a PATTERN statement
3. the COLOR= option in a GOPTIONS statement
4. the color of the current style, or, the first color in the device's color list if the
NOGSTYLE option is specified

Style reference Color attribute of the GraphData1 to GraphDataN element,
depending on the number of slices in the pie

See For more information, see “Controlling Slice Patterns and Colors”
on page 958.
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 956. By default, PERCENT=NONE (percentage is not displayed).

Whether the slice percent is displayed with or without decimal places, depends on
the range of values across the chart. The only way to control the appearance of these
values is to calculate the percentage with a DATA step. You can also calculate the
percentage with a statistical procedure and use the resulting data set as input to the
GCHART procedure. Assign the variable that contains the calculated percentages to
the SUMVAR= option.

Examples
“Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996
“Example 10: Grouping and Arranging Pie Charts” on page 998

PLABEL=(text-option(s))
modifies the font, height, and color of the pie slice labels. Text-options can be one or
more of the following:

• COLOR=color
• FONT=font
• HEIGHT=text-height <units>

Specify as many options as you want, but enclose them all in one set of parentheses,
even if you specify only one option.

Style reference
Font Attributes of the GraphValueText element

Restriction
The Java and ActiveX devices do not support all of the text options

Interaction
If there is no PPERCENT= option specification, the PLABEL= option controls the attributes for both the pie slice label and its
associated percentage.

See
“Text Description Suboptions for Donut” on page 954 for a complete description

“PPERCENT=(text-option(s))” on page 951

PPERCENT=(text-option(s))
modifies the font, height, and color of the displayed percentages associated with the
pie slice labels. This option enables you to differentiate the attributes of the
percentage from the attributes of its associated text label. For example, you can
specify a particular font and an orange color for the percentage with the
PPERCENT= option, and a different font and a blue color for the label with the
PLABEL= option:

ppercent=(font='Albany AMT' h=1.3 color=orange);
plabel=(font='Arial' h=1 color=blue);

Text-options can be one or more of the following:

• COLOR=color
Specify as many options as you want, but enclose them all in one set of parentheses, even if you specify only one option.

Default
When PPERCENT is not specified, the percent displayed on the graph will share the same attributes as those of the pie slice label.

Restriction
The Java and ActiveX devices do not support all of the text options

See
“Text Description Suboptions for Donut” on page 954 for a complete description

RADIUS=value
specifies the radius of the pie and donut in their respective charts. Value is the pie radius in character cells.

SLICE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice name (midpoint value) for each slice.

Default
SLICE=OUTSIDE (the name is outside of the slice)

See
“Selecting and Positioning Slice Labels” on page 956 for a description of the option values

Examples
“Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996
“Example 10: Grouping and Arranging Pie Charts” on page 998

STATFMT=format-specification
overrides the GCHART default format of the displayed statistical value. The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.

If you specify an option of TYPE=MEAN, an INSIDE=PCT option, and an OUTSIDE=SUM option, a STATFMT=f8.1 option applies only to the calculated TYPE=MEAN statistical value. For example:

```proc gchart;
pie mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
   statfmt=f8.1;
run;
```

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the STATFMT= option.

If you change the previous example to specify an OUTSIDE=MEAN option, then STATFMT=f8.1 applies to the OUTSIDE=MEAN option and the TYPE=MEAN option. The statistical types match.

The STATFMT= option does not control the format of the response axis tick marks.

Alias
SFMT=, SFORMAT=, STATFORMAT=
**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. There might be an inner subgroup that contains a value that is not in the outer subgroup. In this case the new slice for that value is placed either last or just before the "other" slice, if one is present. That slice order is continued for any remaining subgroups.

Each ring is labeled with its subgroup value; labels are placed to the right of the chart. If the GROUP= option is also used and if all groups contain the same subgroups, then only the first (upper left) chart on each page is labeled. If any group differs in the number of subgroups that it contains, then all charts are labeled.

By default the subgroups are outlined in the foreground color. To specify an outline color, use the COUTLINE= option.

The SUBGROUP= option automatically generates a legend for the midpoint values (not the subgroup values) and suppresses display of the chart statistic. By default the legend appears at the bottom of the chart. To modify the legend, assign a LEGEND definition. To suppress the legend, specify NOLEGEND. To display the chart statistic, use the VALUE= option.

**Interaction**

If EXPLODE is also used, it is ignored.

**See**

“Controlling Bar Chart Patterns, Colors, and Images” on page 935

“LEGEND Statement” on page 377

**Examples**

“Example 8: Subgrouping a Donut or Pie Chart” on page 994

“Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996

---

**SUMVAR=**numeric-variable

specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of *numeric-variable* for each midpoint. The resulting statistics are represented by the size of the slice and displayed beside of each slice.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

**Example**

“Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991

---

**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 “About Chart Statistics” on page 883
“Calculating Weighted Statistics” on page 884

URL=character-variable
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See “Overview of Enhancing Web Presentations” on page 188
“Example: GIF Output with Drill-Down Links” on page 163

VALUE=ARROW | INSIDE | NONE | OUTSIDE
controls the position and style of the slice value (chart statistic) for each slice.

Default VALUE=OUTSIDE (the value is outside the slice)

See “Selecting and Positioning Slice Labels” on page 956 for a description of the option values

Example “Example 9: Ordering and Labeling Slices in a Pie Chart” on page 996

WOUTLINE=|slice-outline-width|
specifies the width of the outline in pixels. WOUTLINE= affects both the slice outlines and the subgroup outlines.

Style reference LineThickness attribute of the GraphOutlines element

Restriction Not supported by Java and ActiveX

Text Description Suboptions for Donut
The LABEL= option in the DONUT statement and both the PLABEL= and PPERCENT= options in the PIE statement use text description suboptions to change the attributes of the text string or strings that follow.

ANGLE=degrees
specifies the angle at which the baseline of the text string or strings is rotated with respect to the horizontal. A positive value for degrees moves the baseline
counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal).

Valid in DONUT

Alias A=

Restriction Not supported by Java

COLOR=color

specifies the color for the text string or strings. The COLOR= suboption stays in effect until another COLOR= specification is encountered. If you omit the COLOR= option, the LABEL= option uses the color defined by the current style.

Valid in DONUT, PIE, PIE3D

Alias C=

Restriction Not supported by Java

Interaction COLOR= suboption ignores the CTEXT= graphics option

See Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313 for details about specifying color

FONT=font

specifies the font for the text string or strings. If you omit the FONT= option, the LABEL= option uses the font that is specified by the FTEXT= graphics option. If no font is specified, it uses the default hardware font, NONE.

Valid in DONUT, PIE, PIE3D

Alias F=

Restriction The Java and ActiveX devices do not support all fonts

See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about specifying font

“FTEXT” on page 555

HEIGHT=text-height <units>

specifies the height of the text string or strings. Text-height is the number of units. If you omit the HEIGHT= option, the LABEL= option uses the height that is specified by the HTEXT= graphics option. If no text height is specified and if the default text height is too large for the donut hole, the size of the label is reduced to fit. Units can be CELLS | CM | IN | PCT | PT. If you omit units, then the HEIGHT= option uses the unit that is specified by the GUNIT= graphics option, or the default unit, CELLS.

Valid in DONUT, PIE, PIE3D

Alias H=

Restriction Not supported by Java and ActiveX

JUSTIFY=LEFT | CENTER | RIGHT

specifies the alignment of the text string or strings. By default, JUSTIFY=CENTER.
Alias
J=

Restriction
Not supported by Java and ActiveX

ROTATE=
specifies the angle at which each character is rotated with respect to the baseline of the text string. The angle is measured from the current text baseline angle specified by the ANGLE= suboption. A positive value for degrees rotates the character counterclockwise; a negative value rotates it clockwise. By default, ROTATE=0 (parallel to the baseline).

Valid in
DONUT

Restriction
Not supported by Java

Details

Description
The PIE, PIE3D, and DONUT statements specify the variable or variables that define the categories of data to chart. These statements automatically do the following:

• determine the midpoints.
• calculate the chart statistic for each midpoint (the default is FREQ).
• scale each slice to represent its chart statistic. No slice is drawn if the chart statistic for the midpoint is 0.
• order the slices by midpoint value in ascending order starting at the three o'clock position and proceeding counterclockwise around the pie.
• print the slice name (midpoint value) and slice value (chart statistic) beside each slice.
• assign patterns and colors to the slices. The default pie pattern is PSOLID.

You can use statement options to select or order the midpoints (slices), to change the type of chart statistic, and to modify the appearance of the chart. This includes changing the content and position of the slice labels, and the patterns used by the slices. You can also specify additional variables by which to group, subgroup, or sum the data. Statement options can also produce special effects, such as exploded or invisible slices.

Donut and pie charts allow grouping and subgrouping. Grouping creates two or more separate pie or donut charts that are displayed in rows or columns on one graph. Subgrouping creates a separate ring of slices within the circle for each value of the subgroup variable. The concentric rings of the subgrouped pie or donut chart make it easy to compare slice values between subgroups.

In addition, you can use global statements to modify patterns and legends, as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.

Selecting and Positioning Slice Labels
By default, each slice is labeled with its midpoint value (slice name) and its chart statistic value (slice value), which are printed outside of the slice. You can control where and how these labels are displayed with the SLICE= and VALUE= options, respectively. In addition, each slice can display the percentage its midpoint contributes to the total chart statistic (slice percent). Use the PERCENT= option to request slice percent.
The SLICE=, VALUE=, and PERCENT= options use the same values:

**ARROW**
places the text outside the slice and connects the text to the slice with a line. This labeling method reduces the radius of the pie. The arrow uses the color that is specified by the CTEXT= option in the PIE, PIE3D, or DONUT statement. If the CTEXT= option is omitted, the arrow uses the color defined by the current style.

**INSIDE**
places the text inside the slice. The label overlays the slice fill patterns. This labeling method increases the radius of the pie.

**NONE**
suppresses the text.

**OUTSIDE**
places the text outside of the slice.

Figure 35.14 on page 957 illustrates these values.

Figure 35.14  Slice Labeling Methods

![Slice Labeling Methods](image)

The SLICE= and VALUE= options are dependent on each other. If you specify only VALUE= or only SLICE=, the other option automatically uses the same labeling method. The PERCENT= option is independent of these two.

Be careful about the combinations that you specify. For example, if you specify PERCENT=ARROW and VALUE=OUTSIDE, the line that connects the percentage information to each slice might overlay the statistic value.

If your pie has many slices, the labels might overlap, particularly if there are several small slices together. You can correct the overlapping labels by using any of the following options:

- the HTEXT= graphics option to decrease the size of the labels.
- the GRSEG Graphics Editor to adjust the labels by moving or resizing the text.
- the ANGLE= option to change the orientation of the pie.
- the MIDPOINTS= option to rearrange slices so that small slices are not together.
- the OTHER= option to group more midpoints into the OTHER category.
the HPOS= and VPOS= graphics options to increase the number of cells in your display. See “The Graphics Output and Device Display Areas” on page 70 for details.

Controlling Slice Patterns and Colors

How to Control Slice Patterns and Colors
Pie and donut charts are always patterned by midpoint. Even when you specify subgrouping, the patterning method does not change from midpoint to subgroup.

Default Patterns and Outlines
Each slice in a pie or donut chart is filled with a pattern. Because the system option GSTYLE is in effect by default, the procedure uses the current style's default patterns and outlines when producing output. By default, the procedure does the following:

- fills the slices with pie patterns, beginning with the default fill, PSOLID, and rotates it through the list of colors available in the current style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

Note: PIE3D always uses solid patterns.

If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all the colors in the list to generate the patterns.

- outlines slices and subgroup segments using the color defined by the style. To change the outline color, use the COUTLINE= option.

See “About Patterns” on page 885 for more information about how the GCHART procedure assigns default patterns and outlines.

Controlling Patterns
You can control slice patterns and their outlines in several ways.

- To select a different fill for the slices, such as empty or hatched, you can do the following:
  - request a single hatched fill pattern for all slices by specifying the FILL=X option on the PIE or DONUT statement. The pattern specified by FILL=X uses the colors in the color list as many times as needed to generate all of the patterns that are required by the chart. If you specify a single color with either CFILL= or the graphics option, CPATTERN=, all slices use the same color as well as the same pattern.
  - specify a pattern with the VALUE= option in the PATTERN statement. Only pie patterns are valid; all other pattern specifications are ignored. For a complete description of all pie patterns, see “VALUE=pie/star-pattern” on page 403 in the PATTERN statement.

If no color options are specified, the procedure rotates each specified fill once through the list of colors available in the current style. Otherwise, the PATTERN statement generates one pattern definition for the specified pattern and color. When all of the specified patterns are exhausted, the procedure starts rotating through the default pie patterns, beginning with PSOLID.

- To select colors for the slices, you can do the following:
• specify a single pattern color with the CFILL= option, or with the CPATTERN= graphics option, or with a COLORS= list of one color. For the PIE and DONUT statements, CFILL= starts with the default solid color and uses the foreground color for outlines. In contrast, using the CPATTERN= graphics option or a COLORS= list of one color skips the solid pattern. Beginning with P2N0, these options use each pie hatch pattern with the specified color, and use the fill color for the outline color.

• specify only the COLOR= option in one or more PATTERN statements. In this case, the procedure creates a solid pattern for each specified color. When it runs out of PATTERN statements, it returns to the default patterns, beginning with PSOLID, and rotates them each through the color list. Whenever you specify a PATTERN statement, the default outline color is SAME.

• To define specific patterns and colors for the slices, use PATTERN statements and specify both the VALUE= and COLOR= options. You might provide fewer PATTERN definitions than the chart requires. In this case the GCHART procedure uses the default pattern rotation for the slices that are drawn after all of the defined patterns are exhausted.

See “About Patterns” on page 885 for more information about how the GCHART procedure uses patterns and outlines. See the “PATTERN Statement” on page 398 for a description of default pie patterns.

Modifying the Statistic Heading and the Group Heading
By default, the procedure prints a heading at the top of each pie (or donut) chart that indicates the type of statistic charted and the name of the chart variable. For an example see Figure 35.4 on page 875. You can suppress this heading with the NOHEADING option.

When you use the GROUP= option, a heading is printed above each pie indicating the name of the group variable and its value for the particular pie (for example, SITE=Paris). You can suppress these headings with the NOGROUPHEADING option. You can also suppress the variable name SITE= so that only the value Paris remains. To do this, use a LABEL statement and assign a null value to the variable name, for example:

label site='00'x;

The AXIS statement cannot be used by the PIE, PIE3D, and DONUT statements. Instead, you should use the graphics options “FTEXT” on page 555 and “HTEXT” on page 575 to control the font and height of text on the chart. Increasing the value of the HTEXT= graphics option decreases the size of the pie if any slice labels are positioned outside.

Substituting BY Line Values in a Text String
The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.
If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

• The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.

• There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the “BY Statement” on page 370.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

---

**STAR Statement**

Creates star charts in which the length of the spines represents the value of the chart statistic for each category of data or midpoint.

**Restriction:** Not supported by Java and ActiveX

**Requirement:** At least one chart variable is required.

**Global statements:** FOOTNOTE, PATTERN, TITLE

**Supports:** Drill-down functionality (slices only)

---

**Syntax**

```
STAR chart-variable(s) </ option(s)>;
```

**Summary of Optional Arguments**

**Appearance options**

```
ANGLE=degrees
```

starts the first slice at the specified angle.

```
ANNOTATE=Annotate-data-set
```

specifies a data set to annotate all graphs that are produced by the GCHART procedure.

```
ASCENDING
```

arranges the bars in ascending order of the value of the chart statistic.

```
CFILL=fill-color
```

specifies one color for all slices in the chart, regardless of whether the fill is solid or hatch.

```
COUTLINE=star-outline-color | SAME
```

specifies the color for the circle that surrounds the star chart and for the slice outlines or spines.

```
DESCENDING
```

arranges the bars in descending order of the value of the chart statistic.

**FILL=SOLID | X**
specifies the fill pattern for all slices in the star chart.

**LEGEND=LEGEND<1 ...99>**
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option.

**NOCONNECT**
draws only star spines without connecting lines.

**NOLEGEND**
suppresses the legend automatically generated by the SUBGROUP= option.

**STARMAX=max-value**
scales the chart so that the outside (or edge) of the circle represents the value that is specified by max-value.

**STARMIN=min-value**
scales the chart so that the center of the circle represents the value that is specified by min-value.

**WOUTLINE=slice-outline-width**
specifies the width of the outline in pixels.

**Catalog entry description options**
**DESCRIPTION="description"**
specifies a description of the output.

**NAME="name"**
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

**Grouping options**
**ACROSS=number-of-columns**
draws number-of-columns stars across the procedure output area.

**DOWN=number-of-rows**
draws number-of-rows stars vertically in the procedure output area.

**GROUP=variable**
organizes the data according to values of group-variable and produces a separate star chart for each unique value of group-variable.

**Labeling options**
**CTEXT=text-color**
specifies a color for all text on the axes and legend.

**MATCHCOLOR**
uses the slice pattern color for all slice labels.

**NOGROUPHEADING**
suppresses the headings normally printed above each star when you use the GROUP= option.

**NOHEADING**
suppresses the heading normally printed at the top of each page or display of star chart output.

**PERCENT=ARROW | INSIDE | NONE | OUTSIDE**
prints the percentage represented by each slice using the specified labeling method.

**SLICE=ARROW | INSIDE | NONE | OUTSIDE**
controls the position and style of the slice name (midpoint value) for each slice.

**VALUE=**ARROW | INSIDE | NONE | OUTSIDE

decides the position and style of the slice value (chart statistic) for each slice.

**Midpoint options**

**DISCRETE**

treats a numeric chart variable as a discrete variable rather than as a continuous variable.

**LEVELS=**number-of-midpoints | ALL

specifies the number of midpoints to be graphed for a chart variable.

**MIDPOINTS=**OLD


**MIDPOINTS=**value-list

specifies the midpoint values for the slices.

**MISSING**

accepts a missing value as a valid midpoint for the chart variable.

**ODS options**

**HTML_LEGEND=**variable

identifies the variable in the input data set whose values create links or data tips or both.

**HTML=**variable

identifies the variable in the input data set whose values create links or data tips or both.

**URL=**character-variable

specifies a character variable whose values are URLs.

**Statistic options**

**FREQ=**numeric-variable

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

**STATFMT=**format-specification

overrides the GCHART default format of the displayed statistical value.

**SUMVAR=**numeric-variable

specifies a numeric variable for sum or mean calculations.

**TYPE=**statistic

specifies the chart statistic.

**Required Argument**

**chart-variable(s)**

specifies one or more variables that define the categories of data to chart. Each chart variable draws a separate chart. All variables must be in the input data set. Separate multiple chart variables with blanks.

See “Understanding Chart Variables” on page 879

**Optional Arguments**

Options in a STAR statement affect all of the graphs that are produced by that statement. You can specify as many options as you want and list them in any order. For details
about specifying colors, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313.

**ACROSS=** *number-of-columns*

draws *number-of-columns* stars across the procedure output area. The ACROSS= option is ignored unless you also use the GROUP= option. If *number-of-columns* calls for more stars than fit horizontally in the graphics area of the output device, no stars are drawn and an error message is written to the SAS log.

If you also use the DOWN= option, the star charts are drawn in left-to-right and top-to-bottom order.

**ANGLE=** *degrees*

starts the first slice at the specified angle. A value of 0 for *degrees* corresponds to the three o'clock position. *Degrees* can be either positive or negative. Positive values move the starting position counterclockwise; negative values move the starting position clockwise.

If the star chart uses spines instead of slices, *degrees* specifies the angle of the position halfway between the first spine and the last spine.

By default, ANGLE=0, which places the first spine or the center of the first slice of the star at the 0-degree position. Successive star spines or slices are drawn counterclockwise from the starting position.

**ANNOTATE=** *Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GCHART procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

**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. For the PIE and DONUT statements, it is possible that no pattern is specified in the pattern statement or with the FILL= option. In this case the procedure starts with the default solid fill and then, beginning with P2N0, uses each default pie hatch pattern with the specified color. For the outline color, the procedure uses the default color, which is retrieved from the current style. However, if the NOGSTYLE option is specified, the procedure uses the first color in the device's color list. Use the
COUTLINE= option to specify a different outline color. The CFILL= option overrides any other pattern color specification and controls the color of all slices.

Style reference  
Color attribute of the GraphData1 element

COUTLINE=star-outline-color | SAME

specifies the color for the circle that surrounds the star chart and for the slice outlines or spines. SAME specifies that the outline color of a slice is the same as the interior pattern color. Specifying COUTLINE=SAME affects only slice outlines and has no effect on the color of the circle.

The default circle and outline color are both specified in the current device. However, if the NOGSTYLE option is specified, then the default circle color is the first color in the device's color list (the foreground color). In addition, the default slice outline color is determined as follows:

- If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

Alias
CO=

Style reference  
Color attribute of the GraphOutlines element

Note
Use this option to specify a contrasting color when you use an ODS Style with a dark or black background, such as HighContrast.

See
“Selecting Patterns for the Star Charts” on page 973

“About Patterns” on page 885

Example
“Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1001

CTEXT=text-color

specifies a color for all text on the axes and legend. This includes axis labels, tick mark values, legend labels, and legend value descriptions. The GCHART procedure looks for the text color in the following order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the STAR statement.
2. the color specified by the CTEXT= option in the STAR statement.
3. the color specified by the CTEXT= option in a GOPTIONS statement.
4. the color specified in the current style. However, the default color when the NOGSTYLE option is specified is black for the Java and ActiveX devices and the first color in the color list for all other devices.

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR=...
 specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

<table>
<thead>
<tr>
<th>Alias</th>
<th>CT=</th>
</tr>
</thead>
</table>

**Style reference**

Color attributes for the GraphLabelText and the GraphValueText elements

**Note**

If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

**DESCENDING**

arranges the bars in descending order of the value of the chart statistic. By default, bars are arranged in ascending order of midpoint value, without regard to the lengths of the bars. DESCENDING reorders the bars from longest to shortest. In horizontal bar charts the ordering is top to bottom; in vertical bar charts the ordering is left to right. If you also use the GROUP= option, the reordering is performed separately for each group, so the order of the midpoints might be different for each group.

The DESCENDING option overrides any midpoint order that is specified either with the MIDPOINTS= option or the ORDER= option in an AXIS statement assigned to the midpoint axis.

**DESCRIPTION=**"*description*

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the Description field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the *description* text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

<table>
<thead>
<tr>
<th>Alias</th>
<th>DES=</th>
</tr>
</thead>
</table>

**Default**

STAR CHART OF *chart-variable*

**DISCRETE**

treats a numeric chart variable as a discrete variable rather than as a continuous variable. The GCHART procedure creates a separate midpoint and, hence, a separate star slice for each unique value of the chart variable. If the variable has a format associated with it, each format value is treated as a separate value.

The LEVELS=number-of-midpoints option is ignored when you use the DISCRETE option. The MIDPOINTS= option overrides the DISCRETE option.
**DOWN=**\(\text{number-of-rows}\)

draws \(\text{number-of-rows}\) 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=**\(\text{SOLID | X}\)

specifies the fill pattern for all slices in the star chart:

- **SOLID**
  
  rotates a solid fill through the list of colors available in the default style as many times as necessary. SOLID is the default.

  **Alias**  
  
  S

- **X**
  
  rotates a single hatch pattern through the list of colors defined in the current style. If the NOGSTYLE option is specified, it rotates the hatch pattern through the device color list as many times as necessary. If you do not specify the **colors=** goption, the fill skips the first color in the color list.

**FILL=** overrides any pattern that is specified in PATTERN statements.

By default, the outline color is the color defined by the current style. If the NOGSTYLE option is specified, the outline color is the first color in the device's color list. If PATTERN statements are used to specify colors, the slice outline color matches the slice fill color.

If any PATTERN statements have been defined, the colors in the PATTERN definitions are used, in order, before the default style color rotation.

**FREQ=**\(\text{numeric-variable}\)

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that are specified by the value of **numeric-variable** for that observation. If the value of **numeric-variable** is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of **numeric-variable** are truncated to integers.

The **FREQ=** option is valid with all chart statistics.

Because you cannot use **TYPE=PERCENT** or **TYPE=FREQ** with the **SUMVAR=** option, you must use **FREQ=** to calculate percentages and frequencies based on a sum.

The statistics are not affected by applying a format to **numeric-variable**.

**See**  

“Calculating Weighted Statistics” on page 884
GROUP=variable
organizes the data according to values of group-variable and produces a separate star chart for each unique value of group-variable. Group-variable can be either character or numeric and is always treated as a discrete variable. Missing values for group-variable are treated as a valid group.

By default, the charts are produced in ascending order of group variable value and each is drawn on a separate page or display. Therefore, the effect of GROUP= is essentially the same as using a BY statement. The exception is that GROUP= causes the midpoints with the same value to use the same color and fill pattern. To place more than one star chart on a page or display, use the ACROSS= or DOWN= options, or both.

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction
Not supported by Java or ActiveX

See
“Overview of Enhancing Web Presentations” on page 188

HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction
Only star charts with slices support drill-down functionality

See
“Overview of Enhancing Web Presentations” on page 188

LEGEND=LEGEND<1 …99>
assigns the specified LEGEND definition to the legend generated by the SUBGROUP= option. The LEGEND= option itself does not generate a legend.

The LEGEND= option is ignored if any of the following are true:

• The SUBGROUP= option is not used.
• The specified LEGEND definition is not in effect.
• The NOLEGEND option is used.
• The PATTERNID= option is set to any value other than SUBGROUP. That is, the value of PATTERNID= is BY or GROUP or MIDPOINT.

To create a legend based on the chart midpoints instead of the subgroups, use the chart variable as the subgroup variable:

block city / subgroup=city;

Restriction
The Java and ActiveX devices do not support all LEGEND statement options.
LEVELS=number-of-midpoints | ALL
specifies number of midpoints to be graphed for a chart variable. After you specify the number of midpoints that you want, the range for each numeric midpoint is calculated automatically using the algorithm described in Terrell and Scott (1985). If you specify LEVELS=ALL, then all unique numeric or character midpoint values are graphed. If your data contains a large number of unique midpoint values (more than 200), then you can use the XPIXELS and YPIXELS GOPTIONS. This enables the device driver to render a larger (and more readable) graph. The LEVELS=number-of-midpoints option is ignored if any of the following are true:
- The chart variable is character type.
- The DISCRETE option is used.
- The MIDPOINTS= option is used.

MATCHCOLOR
uses the slice pattern color for all slice labels. MATCHCOLOR overrides the color that is specified in the CTEXT= option. If the chart uses spines instead of slices, the spine color is used for the slice label and value text.

MIDPOINTS=value-list
specifies the midpoint values for the slices. The way you specify value-list depends on the type of variable:
- For numeric chart variables, value-list is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - n < ...n>
  - n TO n <BY increment>
  - n < ...n> TO n <BY increment> <n < ...n>>
If a numeric variable has an associated format, the specified values must be the unformatted values.
If you omit the DISCRETE option, then numeric values are treated as continuous, which means that the following is true by default:
- The lowest midpoint consolidates all data points from negative infinity to the median of the first two midpoints.
- The highest midpoint consolidates all data points from the median of the last two midpoints up to infinity.
- All other values in value-list specify the median of a range of values, and the GCHART procedure calculates the midpoint values.
If you include the DISCRETE option, each value in value-list specifies a unique numeric value.
- For character chart variables, value-list is a list of unique character values enclosed in quotation marks and separated by blanks:
  ’value-1’ < ... ’value-n’>
If a character variable has an associated format, the specified values must be the formatted values.

For a complete description of value-list, see the “value-list” on page 353 in the AXIS statement.

See “Understanding Midpoints” on page 880

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=

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

• The name can be up to 256 characters in length.

• Special characters in the name are converted to underscores.

• For the GRSEG entry name:
  • The name is truncated to eight characters.
  • The first character is always represented in uppercase, and all other characters are represented in lowercase.
  • If the name begins with a number, an underscore is prepended to the name.
  • If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

• For the graphics output filename:
  • The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  • All characters are represented in lowercase.
  • If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  • If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  • If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.
The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default: GCHART

NOCONNECT

draws only star spines without connecting lines. By default, the spines are connected to form slices.

Example: “Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1001

NOGROUPHEADING

suppresses the headings normally printed above each star when you use the GROUP= option.

NOHEADING

suppresses the heading normally printed at the top of each page or display of star chart output.

Example: “Example 12: Charting a Discrete Numeric Variable in a Star Chart” on page 1001

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 973. 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 973. By default, SLICE=OUTSIDE (the name is outside the slice).

STARMAX=max-value

scales the chart so that the outside (or edge) of the circle represents the value that is specified by max-value. By default, the value for STARMAX= is the maximum chart statistic value.

STARMIN=min-value

scales the chart so that the center of the circle represents the value that is specified by min-value. By default, STARMIN=0. If the chart statistic has negative values, by default the value for the STARMIN= option is the minimum chart statistic value.

STATFMT=format-specification

overrides the GCHART default format of the displayed statistical value. The STATFMT= option associates a specified format with a calculated statistical value such as that specified with the frequency (FREQ=) option or TYPE= option. Use this option to change the default format that might contain decimal points, percentages, or commas.

If you specify an option of TYPE=MEAN, an INSIDE=PCT option, and an OUTSIDE=SUM option, a STATFMT=f8.1 option applies only to the calculated TYPE=MEAN statistical value. For example:

```plaintext
proc gchart;
star mid / discrete width=3 sumvar=varname, type=mean inside=pct outside=sum
```
statfmt=f8.1;
run;

In this case the INSIDE= and OUTSIDE= option values display their default values. They are unaffected by the STATFMT= option.

If you change the previous example to specify an OUTSIDE=MEAN option, then STATFMT=f8.1 applies to the OUTSIDE=MEAN option and the TYPE=MEAN option. The statistical types match.

The STATFMT= option does not control the format of the response axis tick marks.

Alias SFMT=, SFORMAT=, STATFORMAT=

SUMVAR=numeric-variable
specifies a numeric variable for sum or mean calculations. The GCHART procedure calculates the sum or, if requested, the mean of the value of numeric-variable for each midpoint. The resulting statistics are represented by the size of the slice and displayed beside each slice.

When you use the SUMVAR= option, the TYPE= option must be either SUM or MEAN. With the SUMVAR= option, the default is TYPE=SUM.

Example “Example 11: Specifying the Sum Statistic in a Star Chart” on page 999

TYPE=statistic
specifies the chart statistic.

• If the SUMVAR= option is not used, statistic can be one of the following:
  
  FREQ
  frequency (the default)
  
  PERCENT PCT
  percentage
  
  If the SUMVAR= option is used, statistic can be one of the following:
  
  SUM
  sum (the default)
  
  MEAN
  mean

Because you cannot use TYPE=FREQ or TYPE=PERCENT with the SUMVAR= option, you must use FREQ= to calculate percentages or frequencies based on a sum.

See “About Chart Statistics” on page 883

“Calculating Weighted Statistics” on page 884

URL=character-variable
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only.

Not supported by Java or ActiveX

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored
See “Overview of Enhancing Web Presentations” on page 188

See “Example: GIF Output with Drill-Down Links” on page 163

---

**VALUE=ARROW | INSIDE | NONE | OUTSIDE**

controls the position and style of the slice value (chart statistic) for each slice.

**Default** VALUE=OUTSIDE (the value is outside of the slice)

See “Selecting and Positioning Spine and Slice Labels” on page 973 for a description of the option values

**WOUTLINE=slice-outline-width**

specifies the width of the outline in pixels. The WOUTLINE= option affects the slice outlines.

**Style reference** LineThickness attribute of the GraphOutlines element

---

**Details**

**Description**

The STAR statement specifies the variable or variables that define the categories of data to chart. This statement automatically does the following:

- determines the midpoints.
- calculates the chart statistic for each midpoint (the default is FREQ).
- scales each spine or slice to represent the chart statistic. Slices or spines are drawn for all midpoints where the value of the chart statistic is greater than the value that is specified in the STARMIN= option.
- arranges the spines or slice counterclockwise around the star in ascending order of midpoint value, starting at the three o'clock position.
- prints the midpoint value and chart statistic beside each spine or slice.
- assigns patterns to the slices.

If all the data to be charted with the STAR statement are positive, the center of the star represents 0 and the outside circle represents the maximum value. If negative values are calculated for the chart statistic, the center represents the minimum value in the data. You can specify other values for the center and outside of the circle with the STARMIN= and STARMAX= options.

You can also use statement options to

- select or order the midpoints
- change the type of chart statistic
- modify the appearance of the chart. This includes the content and position of the spine or slice labels, and patterns that fill the slice.

You can specify additional variables by which to group or sum the data.

Star charts allow grouping, which creates two or more separate charts that are displayed in rows or columns on one graph.

In addition, you can use global statements to modify patterns as well as add titles, footnotes, and notes to the chart. You can also use an Annotate data set to enhance the chart.
Selecting and Positioning Spine and Slice Labels

By default, each spine or slice is labeled with its midpoint value and its chart statistic value, which are printed outside of the circle. You can control where and how these labels are displayed with the SLICE= and VALUE= options, respectively. In addition, each spine can display the percentage that its midpoint contributes to the total chart statistic (spine percent). Use the PERCENT= option to request spine percent.

The SLICE=, VALUE=, and PERCENT= options use the same values:

ARROW
places the text outside of the star circle and connects the text to the circle with a line. The line points to the spine or the center of the slice. The arrow uses the color that is specified by the CTEXT= option in the STAR statement. If you omit the CTEXT= option, the arrow uses the color defined by the current style.

INSIDE
places the text inside the star circle.

NONE
suppresses the text.

OUTSIDE
places the text outside the star circle.

Figure 35.14 on page 957 illustrates these values.

The SLICE= and VALUE= options are dependent on each other. If you specify only VALUE= or only SLICE=, the other option automatically uses the same labeling method. The PERCENT= option is independent of these two.

Be careful about the combinations that you specify. For example, if you specify PERCENT=ARROW and VALUE=OUTSIDE, the line that connects the percentage information to each spine might overlay the statistic value.

Selecting Patterns for the Star Charts

Determining Pattern Selection

Star charts are always patterned by midpoint.

Default Patterns and Outlines

Each slice in a star chart is filled with a pattern. Because the system option GSTYLE is in effect by default, the procedure uses the current style’s default patterns and outlines when producing output. By default, the procedure does the following:

- fills the slices with star patterns, beginning with the default fill, PSOLID, and rotates it through the list of colors available in the default style. When these colors are exhausted, the procedure rotates through a slightly modified version of the original list of colors. It continues in this fashion until all of the chart variables have been assigned a unique pattern.

  If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, the procedure uses all of the colors in the list to generate the patterns.

- outlines slices using the color defined by the style. To change the outline color, use the COUTLINE= option.

See “About Patterns” on page 885 for more information about how the GCHART procedure assigns default patterns and outlines.
Controlling Patterns

You can control slice patterns and their outlines in several ways.

- To select a different fill for the slices, such as empty or hatched, you can do the following:

  - request a single hatched fill pattern for all slices by specifying the FILL=X option in the STAR statement. The pattern that is specified by FILL=X rotates through the list of colors available in the current style. This happens as many times as needed to generate all the patterns required by the chart. If you specify a single color with either CFILL= or the graphics option, CPATTERN=, all slices use the same color as well as the same pattern.
  
  - specify a pattern with the VALUE= option in the PATTERN statement. Only star patterns are valid; all other pattern specifications are ignored. For a complete description of all star patterns, see the option description for “VALUE=pie/star-pattern ” on page 403 in the PATTERN statement

    If no color options are specified, the procedure rotates each specified fill once through the list of colors available in the current style. Otherwise, the PATTERN statement generates one pattern definition for the specified pattern and color. When all of the specified patterns are exhausted, the procedure starts rotating through the default star patterns, beginning with PSOLID.

- To select colors for the slices, you can do the following:

  - specify a single pattern color with the CFILL= option, or with the CPATTERN= graphics option, or with a COLORS= list of one color. For the PIE and DONUT statements, CFILL= starts with the default solid color and uses the foreground color for outlines. In contrast, using the CPATTERN= graphics option or a COLORS= list of one color skips the solid pattern. Beginning with P2N0, these options use each pie hatch pattern with the specified color, and use the fill color for the outline color.
  
  - specify only the COLOR= option in one or more PATTERN statements. In this case, the procedure creates a solid pattern for each specified color. When it runs out of PATTERN statements, it returns to the default patterns, beginning with PSOLID, and rotates them each through the list of colors available in the current style. Whenever you specify a PATTERN statement, the default outline color is SAME.

  - To define specific patterns and colors for the slices, use PATTERN statements and specify both the VALUE= and COLOR= options. If you provide fewer PATTERN definitions than the chart requires, the GCHART procedure uses the default pattern rotation for the slices that are drawn after all defined patterns are exhausted.

See “About Patterns” on page 885 for more information about how the GCHART procedure uses patterns and outlines. See the “PATTERN Statement” on page 398 for a description of default star patterns.

Modifying the Statistic Heading and the Group Heading

By default, the procedure prints a heading at the top of each chart indicating the type of statistic charted and the name of the chart variable. For an example see Figure 35.4 on page 875. You can suppress this heading with the NOHEADING option.

When you use the GROUP= option, a heading is printed above each star indicating the name of the group variable and its value for the particular star. An example is SITE=Paris. You can suppress these headings with the NOGROUPHEADING option. You can also suppress the variable name SITE= so that only the value Paris remains. To
do this, use a LABEL statement and assign a null value to the variable name, as shown in this example:

```
label site="00"x;
```

The AXIS statement cannot be used by the STAR statement. Instead, you should use the “FTEXT” on page 555 and “HTEXT” on page 575 graphics options to control the font and height of text on the chart. Increasing the value of HTEXT= decreases the size of the star if any slice labels are positioned outside.

---

**Examples: GCHART Procedure**

**Example 1: Specifying the Sum Statistic in a Block Chart**

**Features:**
- BLOCK statement option SUMVAR=

**Other features:**
- FORMAT statement and the GOPTIONS statement option BORDER

**Sample library member:**
- GCHBKSUM

**Note:**
- The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

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.

---

**Total Sales**

<table>
<thead>
<tr>
<th>Site</th>
<th>Total Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>$11,649</td>
</tr>
<tr>
<td>Paris</td>
<td>$12,458</td>
</tr>
<tr>
<td>Sydney</td>
<td>$8,920</td>
</tr>
</tbody>
</table>
Program

goptions reset=all border;

data totals;
    length dept $ 7 site $ 8;
    input dept site quarter sales;
    datalines;
    Parts Sydney 1 7043.97
    Parts Atlanta 1 8225.26
    Parts Paris 1 5543.97
    Tools Sydney 4 1775.74
    Tools Atlanta 4 3424.19
    Tools Paris 4 6914.25
    ;
    title "Total Sales";
    proc gchart data=totals;
        format sales dollar8.;
        block site / sumvar=sales;
        run;
    quit;

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

goptions reset=all border;

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

data totals;
    length dept $ 7 site $ 8;
    input dept site quarter sales;
    datalines;
    Parts Sydney 1 7043.97
    Parts Atlanta 1 8225.26
    Parts Paris 1 5543.97
    Tools Sydney 4 1775.74
    Tools Atlanta 4 3424.19
    Tools Paris 4 6914.25
    ;

Define the title. This title appears on the chart.

title "Total Sales";

Produce the block chart. The BLOCK statement produces a block chart. SUMVAR= calculates the sum of SALES for each value of the chart variable SITE. With SUMVAR= the default statistic is SUM. The variable SALES is assigned a dollar format.

proc gchart data=totals;
    format sales dollar8.;
    block site / sumvar=sales;
    run;
    quit;
Example 2: Grouping and Subgrouping a Block Chart

Features: BLOCK statement options GROUP=, LEGEND=, MIDPOINTS=, NOHEADING, SUBGROUP=, and TYPE=

Other features: GOPTION statement option BORDER, LABEL statement, LEGEND statement, and default pattern rotation

Sample library member: GCHBKGRP

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

This example shows average quarterly sales for each department at two of the three manufacturing sites in the TOTALS data set; it excludes the Paris site from the chart.

The program groups the chart data (sites) by department, and subgroups department sales data by quarter. Each site is a midpoint. Because the sites are grouped by department, each midpoint has a separate square for each department and the height of the block represents total sales for that department.

The blocks are subgrouped to show how quarterly sales contribute to total sales; each segment represents sales for a quarter. A legend explaining the subgroup patterns appears below the midpoint grid.

The subgroups use four default patterns and colors that are retrieved from the current style. The patterns are created by rotating the default fill, solid, through the color list that is defined in the current style.

Program

goptions reset=all border;

data totals;

Average Sales by Department

Program

goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 3043.97
Parts Sydney 3 5142.63
Parts Atlanta 1 5225.26
Parts Atlanta 2 3529.06
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Repairs Sydney 2 5543.97
Repairs Atlanta 1 3788.93
Repairs Atlanta 2 4492.89
Repairs Atlanta 3 3914.25
;
title "Average Sales by Department";
legend1 cborder=black
label=('Quarter:')
position=(bottom right outside)
mode=protect
across=1;
proc gchart data=totals;
format quarter roman.;
format sales dollar8.;
label site="00"x dept="00"x;
block site / sumvar=sales
type=mean
midpoints="Sydney" "Atlanta"
group=dept
subgroup=quarter
legend=legend1
noheading;
run;
quit;

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

```
goptions reset=all border;
```

Create data set TOTALS. TOTALS contains quarterly sales data for two of the three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 3043.97
Parts Sydney 3 5142.63
Parts Atlanta 1 5225.26
Parts Atlanta 2 3529.06
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
```
Repairs Sydney 2 5543.97
Repairs Atlanta 1 3788.93
Repairs Atlanta 2 4492.89
Repairs Atlanta 3 3914.25
;

Define the title. This title appears on the chart.

title "Average Sales by Department";

Define legend characteristics. LABEL= assigns new text to the legend label. CBORDER= draws a black frame around the legend.

legend1 cborder=black
label=("Quarter:")
position=(bottom right outside)
mode=protect
across=1;

Produce the block chart. The LABEL statement suppresses the midpoint and group labels by assigning a null hexadecimal string to each variable name.

proc gchart data=totals;
format quarter roman.;
format sales dollar8.;
label site="00"x dept="00"x;

The BLOCK statement produces the block chart. The TYPE= option specifies the chart statistic as the mean value of the summary variable SALES for each site. The MIDPOINTS= option selects the two sites and the order in which they appear. The GROUP= option creates a separate row of blocks for each different value of DEPT. The SUBGROUP= option divides each block into separate segments for the four quarters. The LEGEND= option assigns the LEGEND1 statement to the graph. NOHEADING suppresses the default heading that would otherwise appear above the chart.

block site / sumvar=sales
type=mean
midpoints="Sydney" "Atlanta"
group=dept
subgroup=quarter
legend=legend1
noheading;
run;
quit;

Example 3: Specifying the Sum Statistic in Bar Charts

Features: HBAR statement option SUMVAR= and VBAR3D statement option SUMVAR=
Other features: FORMAT statement, GOPTIONS statement option BORDER, and RUN-group processing
Sample library member: GCHBRSUM
Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
This example produces two bar charts that show the total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site.

In the first horizontal bar chart shown, the summary statistics are printed by default to the right of the bars and display the formatted values of SALES.

The output also shows the frame that is drawn by default around the axis area.

The second bar chart is a three-dimensional vertical bar chart, shown in the following output. Vertical bar charts do not generate a table of statistics and by default do not print any chart statistics.
Program

goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;  
Parts Sydney 1 7043.97  
Parts Atlanta 1 8225.26  
Parts Paris 1 5543.97  
Tools Sydney 4 1775.74  
Tools Atlanta 4 3424.19  
Tools Paris 4 6914.25  
;

title1 "Total Sales";

proc gchart data=totals;
format sales dollar8.;
hbar site / sumvar=sales;
run;

vbar3d site / sumvar=sales;
run;
quit;

Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement
draws a black border around the graph.

goptions reset=all border;

Create data set TOTALS. TOTALS contains quarterly sales data for three
manufacturing sites for one year. Sales figures are broken down by department.

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;  
Parts Sydney 1 7043.97  
Parts Atlanta 1 8225.26  
Parts Paris 1 5543.97  
Tools Sydney 4 1775.74  
Tools Atlanta 4 3424.19  
Tools Paris 4 6914.25  
;

Define the title. This title appears on the chart.

title1 "Total Sales";

Produce the horizontal bar chart. The HBAR statement produces a two-dimensional
bar chart. SUMVAR= calculates the sum of SALES for each value of the chart variable
SITE. The default statistic for SUMVAR= is SUM. The variable SALES is assigned a
dollar format. The embedded RUN statement is required to end this first RUN-Group
and honor the action statement and other SAS statements. It signals that the procedure is
to remain active.
proc gchart data=totals;
format sales dollar8.;
  hbar site / sumvar=sales;
run;

Produce the vertical bar chart. Because the procedure supports RUN-group processing, you do not have to repeat the PROC GCHART statement to generate the second chart. The VBAR3D statement produces a three-dimensional vertical bar chart.

vbar3d site / sumvar=sales;
run;
quit;

Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart

Features:  VBAR statement options CFRAME=, INSIDE=SUBPCT, LEGEND=, MAXIS=, OUTSIDE=SUM, RAXIS=, SPACE=, SUBGROUP=, and WIDTH=

Other features:  AXIS statement, FORMAT statement, GOPTIONS statement option BORDER, and LEGEND statement

Sample library member:  GCHBRGRP

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

This example subgroups by department the three-dimensional vertical bar chart of total sales for each site that is shown in “Example 3: Specifying the Sum Statistic in Bar Charts” on page 979. In addition to subdividing the bars to show the amount of sales for each department for each site, the chart displays statistics both inside and outside of the bars. OUTSIDE=SUM prints the total sales for the site above each bar. INSIDE=SUBPCT prints the percent each department contributed to the total sales for its site inside each subgroup segment.
Program

```sas
options reset=all cback=white;

data totals;
    length dept $ 7 site $ 8;
    input dept site quarter sales;
    datalines;
    Parts Sydney 1 7043.97
    Parts Atlanta 1 8225.26
    Tools Paris 4 1775.74
    Tools Atlanta 4 3424.19
    Repairs Sydney 2 5543.97
    Repairs Paris 3 6914.25
    ;
    title1 "Total Sales by Site";
    axis1 label=none offset=(10,8);
    axis2 label=none order=(0 to 20000 by 5000) minor=none offset=(,0);
    legend1 label=none shape=bar(.15in,.15in) cborder=black;
    proc gchart data=totals;
    format sales dollar8.;
    vbar3d site / sumvar=sales subgroup=dept inside=subpct
        outside=sum
        width=9
        space=7
        maxis=axis1
        raxis=axis2
        cframe=white
        autoref cref=gray
        legend=legend1;
    run;
```

Example 4: Subgrouping a Three-Dimensional Vertical Bar Chart
Program Description

Set the graphics environment. The CBACK option in the GOPTIONS statement changes the graph background color to white.

```
options reset=all cback=white;
```

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

```
data totals;
  length dept $ 7 site $ 8;
  input dept site quarter sales;
  datalines;
    Parts Sydney 1 7043.97
    Parts Atlanta 1 8225.26
    Tools Paris 4 1775.74
    Tools Atlanta 4 3424.19
    Repairs Sydney 2 5543.97
    Repairs Paris 3 6914.25
  ;
```

Define the title. This title appears on the chart.

```
title1 "Total Sales by Site";
```

Modify the midpoint axis. The LABEL= option suppresses the axis label. The OFFSET= option moves the first bar from the beginning of the axis line and moves the last bar from the end of the axis line.

```
axis1 label=none offset=(10,8);
```

Modify the response axis. The LABEL= option suppresses the axis label. The ORDER= option specifies the major tick values for the response axis. The MINOR= option suppresses the minor tick marks. The OFFSET= option moves the top tick mark to the end of the axis line.

```
axis2 label=none order=(0 to 20000 by 5000) minor=none offset=(,0);
```

Modify the legend. The LABEL= option suppresses the legend label. The SHAPE= option defines the size of the legend values. The CBORDER= option draws a black frame around the legend.

```
legend1 label=none shape=bar(.15in,.15in) cborder=black;
```

Produce the vertical bar chart. The SUBGROUP= option creates a separate bar segment for each department. The INSIDE= option prints the subgroup percent statistic inside each bar segment. The OUTSIDE= option prints the sum statistic above each bar. The WIDTH= option makes the bars wide enough to display the statistics. The SPACE= option controls the space between the bars. The MAXIS= option assigns the AXIS1 statement to the midpoint axis. The RAXIS= option assigns the AXIS2 statement to the response axis. The CFRAFME= option specifies the color for the three-dimensional planes. The AUTOREF option draws a gray reference line at each major tick mark on the response axis. The LEGEND= option assigns the LEGEND1 statement to the subgroup legend.
proc gchart data=totals;
format sales dollar8.;
vbar3d site / sumvar=sales subgroup=dept inside=subpct
outside=sum
width=9
space=7
maxis=axis1
raxis=axis2
cframe=white
autoref cref=gray
legend=legend1;
run;
quit;

Example 5: Controlling Midpoints and Statistics in a Horizontal Bar Chart

Features: HBAR statement options AUTOREF, COUTLINE=, CLIPREF, and SUBGROUP=;
and HBAR3D statement options FREQ, FREQLABEL=, and MIDPOINTS=

Other features: GOPTIONS statement option BORDER, AXIS statement, LEGEND statement, and
RUN-group processing

Sample library member: GCHBRMID

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

This example uses the FITNESS data set to produce a horizontal bar chart that shows the
number of people in each age group in a fitness program.

It charts the numeric variable AGE with the frequency statistic. Because the values of
AGE are continuous, the procedure automatically divides the ages into ranges and
displays the midpoint of each age range. The frequency statistic calculates the number of
observations in each range. The chart statistic defaults to FREQ because the SUMVAR=
and TYPE= options are omitted. The table of statistics displays all the statistic values.

The second part of this example modifies the midpoint axis and the table of statistics,
and uses RUN-group processing to produce the following chart. This part of the program
specifies the midpoint value for each bar and requests only the FREQ statistic for the
table.
Program

goptions reset=all border;

data fitness;
  input age sex $ heart exer aero;
  datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
;

title1 "Fitness Program Participants";

axis1 label=("Number of People")
  minor=(number=1)
  offset=(0,0);
Program Description

Set the graphics environment. The BORDER option in the GOPTIONS statement draws a black border around the graph.

goptions reset=all border;

Create the data set FITNESS. FITNESS contains age and sex of participants, as well as the number of times they exercise each week and their resting heart rate and aerobic power.

data fitness;
    input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2
Define the title. This title appears on the chart.

```sas
title1 "Fitness Program Participants";
```

Modify the response axis. The OFFSET= option moves the first and last tick marks to the ends of the axis line. The ORDER= option places major tick marks on the response axis from 1 to 14.

```sas
axis1 label="Number of People"
   minor=(number=1)
   offset=(0,0);
```

Modify the legend. The VALUE= option specifies the text that describes the values.

```sas
legend1 label=none
   value="Women" "Men";
```

Produce the first horizontal bar chart. Because neither the MIDPOINTS= option nor the DISCRETE option is used, the procedure automatically selects the midpoints. The SUBGROUP= option divides the bars according to the values of SEX and automatically generates a legend. The AUTOREF option adds reference lines to the chart at each major tick mark. The CLIPREF option positions the reference lines behind the bars. The WREF= option sets the reference line width to 1 pixel. The CREF= option sets the reference line color to green. The LREF= option sets the reference line type to 4 (dashed line). The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals that the procedure is to remain active.

```sas
proc gchart data=fitness;
   hbar age / subgroup=sex
      legend=legend1
         autoref
         clipref
         raxis=axis1
         wref=1
         cref=green
         lref=4
   ;
run;
```

Modify the response axis for the second chart. The ORDER= option places major tick marks on the response axis at intervals of 1.

```sas
axis1 order=(0 to 4 by 1)
   label="Number of People"
   minor=(number=1)
   offset=(0,0);
```

Modify the midpoint axis label for the second chart.

```sas
axis2 label="Age * j=r "Group"];```

Produce the second horizontal bar chart with modified midpoints. The MIDPOINTS= option specifies the middle value of the range of values represented by each bar. The FREQ option requests that only the frequency statistic appears in the table. The FREQLABEL= option specifies the text for the column heading in the table of statistics.
Example 6: Generating Error Bars in a Horizontal Bar Chart

Features: HBAR statement options CLM=, ERRORBAR=, FREQLABEL=, MEANLABEL=, NOFRAME, SUMVAR=, and TYPE=

HBAR statement options:

Other features: GOPTIONS statement option BORDER and AXIS statement

Sample library member: GCHERRBR

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

This example uses the FITNESS data set to chart the mean heart rate for each age group with error bars showing the confidence limits for the average. The response axis label describes the confidence limit for the error bars. To make the error bars easier to read, the program suppresses the frame that the procedure draws around the axis area. Descriptive column head labels in the table of statistics replace the statistic names that appear by default.

Program

goptions reset=all border;
data fitness;

```sas
hbar3d age / midpoints=(30 40 50)
    freq freqlabel="Total in Group"
    subgroup=sex
    autoref
    maxis=axis2
    raxis=axis1
    legend=legend1
    coutline=black
    clipref
    wref=1
    cref=green
    lref=4
;
run;
quit;
```
```
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2;

title1 "Average Resting Heart Rate by Age";
axis1 label=('Heart Rate' j=c "Error Bar Confidence Limits: 95%")
minor=(number=1);
axis2 label=('Age' j=r "Group");
proc gchart data=fitness;
hbar age / type=mean
sumvar=heart
frelab="Number in Group"
menlab="Mean Heart Rate"
errobar=bars
cerror=orange
clm=95
midpoints=(30 40 50)
ra=x=axis1
ma=x=axis2
noframe;
run;
quit;
```

**Program Description**

**Set the graphics environment.** The BORDER option in the GOPTIONS statement draws a black border around the graph.
```
goptions reset=all border;
```

**Create the data set FITNESS.** FITNESS contains age and sex of participants, as well as the number of times they exercise each week and their resting heart rate and aerobic power.
```
data fitness;
input age sex $ heart exer aero;
datalines;
28 M 86 2 36.6
41 M 76 3 26.7
30 M 78 2 33.8
29 M 54 3 44.8
48 F 66 2 28.9
36 F 66 2 33.2;
```

**Define the title.** This title appears on the chart.
```
title1 "Average Resting Heart Rate by Age";
```
Modify the axis labels. AXIS1 is assigned to the response axis and AXIS2 is assigned to the midpoint axis.

```plaintext
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.

```plaintext
proc gchart data=fitness;
   hbar age / type=mean
       sumvar=heart
       freqlabel="Number in Group"
       meanlabel="Mean Heart Rate"
       errorbar=bars
cerror=orange
c1m=95
   midpoints=(30 40 50)
   raxis=axis1
   maxis=axis2
   noframe;
run;
quit;
```

---

**Example 7: Specifying the Sum Statistic for a Pie Chart**

**Features:** PIE statement option SUMVAR=, and PIE3D statement options EXPLODE= and SUMVAR=

**Other features:** G_OPTIONS statement option BORDER, FORMAT statement, and RUN-group processing

**Sample library member:** GCHPISUM

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

This example produces two pie charts that show total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. It represents the statistics as slices of the pie. By default, the midpoint value and the summary statistic are printed beside each slice.

The pie slices use the default pattern fill, which is solid. Each slice displays a different color because, by default, pie charts are patterned by midpoint.
The second pie chart is a three-dimensional pie chart with an exploded slice, as shown in the following output.

Program

```r
goptions reset=all border;
data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
```
Tools Paris 4 6914.25
;
title "Total Sales"
proc gchart data=totals;
format sales dollar8.;
pie site / sumvar=sales;
run;
pie3d site / sumvar=sales explode="Paris"
run;
quit;

Program Description

**Set the graphics environment.** The BORDER option in the GOPTIONS statement draws a black border around the graph.

goptions reset=all border;

**Create data set TOTALS.** TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;

**Define the title.** This title appears on the chart.

title "Total Sales"

**Produce the first pie chart.** The pie statement produces a two dimensional pie chart. The SUMVAR= option calculates the sum of SALES for each value of the chart variable SITE. The default statistic for the SUMVAR= option is SUM. The variable SALES is assigned a dollar format. The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals that the procedure is to remain active.

proc gchart data=totals;
format sales dollar8.;
pie site / sumvar=sales;
run;

**Produce the second pie chart.** The PIE3D statement produces a three-dimensional pie chart. The EXPLODE= option separates the slice for PARIS from the rest of the pie.

pie3d site / sumvar=sales explode="Paris"
run;
Example 8: Subgrouping a Donut or Pie Chart

Features: DONUT statement options DONUTPCT=, LABEL=, LEGEND=, NOHEADING, and SUBGROUP=

Other features: GOPTIONS statement option BORDER and LEGEND statement

Sample library member: GCHSBGRP

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

This example produces a donut chart that is similar to the pie chart in “Example 7: Specifying the Sum Statistic for a Pie Chart” on page 991. The similarity is that each slice represents total sales for a site and each slice is a different color. However, in this donut chart the sites are subgrouped by department, so that each department is represented as a concentric ring with slices.

Subgrouping suppresses the chart statistic and the midpoint labels. Instead, it automatically labels the rings with the subgroup values and generates a legend that shows how the patterns are associated with the midpoint values. Subgrouping a pie chart produces the same results but without the hole in the center.

To make the donut chart as large as possible, the program suppresses the default heading and moves the legend into the space at the left of the chart.

Program

goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;

Program Description

**Set the graphics environment.** The BORDER option in the GOPTIONS statement draws a black border around the graph.

```sas
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.

```sas
data totals;
  length dept $ 7 site $ 8;
  input dept site quarter sales;
  datalines;
  Parts Sydney 1 7043.97
  Parts Atlanta 1 8225.26
  Parts Paris 1 5543.97
  Tools Sydney 4 1775.74
  Tools Atlanta 4 3424.19
  Tools Paris 4 6914.25
  ;
```

**Define the title.** This title appears on the chart.

```sas
title "Sales by Site and Department";
```

**Modify the subgroup legend.** The LABEL= option suppresses the legend label. The POSITION=, the OFFSET=, and the ACROSS= options arrange the legend entries in a column to the left of the pie chart.

```sas
legend1 label=none
position=(middle left)
offset=(5,)
across=1;
```
Produce the donut chart. The SUBGROUP= option divides the donut into rings. Each ring represents a value of the subgroup variable, DEPT. The DONUTPCT= option controls the size of the donut hole, which contains the text specified by the LABEL= option. The NOHEADING option suppresses the default heading that contains the name of the chart variable and the type of statistic. The LEGEND= option assigns the LEGEND1 statement to the chart.

```
proc gchart data=totals;
format sales dollar8.;
donut site / sumvar=sales
  subgroup=dept
donutpct=30
  label=(*All" justify=center "Quarters")
noheading
legend=legend1;
run;
quit;
```

Example 9: Ordering and Labeling Slices in a Pie Chart

Features: PIE statement options MIDPOINTS=, PERCENT=ARROW, SLICE=ARROW, and VALUE=NONE

Other features: GOPTIONS Statement option BORDER

Sample library member: GCHLABEL

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

This example produces a pie chart of the types of vehicles produced worldwide. The labeled slices represent the percent of total production for each source. Instead of the sum statistic, each slice displays the percent each midpoint contributes to the whole pie. Arrows connect the midpoint labels to the slices. The slices are arranged by the MIDPOINTS= option so that similar types of vehicles are shown next to each other in the pie chart.
Program

goptions reset=all border;

title "Types of Vehicles Produced Worldwide";

proc gchart data=sashelp.cars;
pie type / other=0
midpoints="Truck" "SUV" "Sedan" "Wagon" "Sports" "Hybrid"
value=none
percent=arrow
slice=arrow
noheading;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the title. This title appears on the chart.

title "Types of Vehicles Produced Worldwide";

Produce the pie chart. This graph uses the data set entitled CARS found in the SASHELP library. OTHER=0 specifies that all midpoints, no matter how small, display a slice. The MIDPOINTS= option assigns the order of the slices. Each slice displays the percent contribution to total production and the slice name. VALUE=NONE suppresses the chart statistic. Both the SLICE= and PERCENT= options specify the ARROW labeling method to point to the slice, but only one arrow line is displayed.

proc gchart data=sashelp.cars;
pie type / other=0
midpoints="Truck" "SUV" "Sedan" "Wagon" "Sports" "Hybrid"
value=none
Example 10: Grouping and Arranging Pie Charts

Features: PIE statement options ACROSS=, CLOCKWISE, GROUP=, OTHER=, PERCENT=OUTSIDE, and SLICE=OUTSIDE

Other features: GOPTIONS statement option BORDER

Sample library member: GCHPIGRP

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

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.

Types of Trucks Produced Worldwide

Program

goptions reset=all border;
title "Types of Trucks Produced Worldwide";
proc gchart data=sashelp.cars(where=(type="SUV" or type="Truck"));
Program Description

Set the graphics environment.

goptions reset=all border;

Define the title. This title appears on the chart.

title "Types of Trucks Produced Worldwide";

Produce the pie charts. This graph uses the data set entitled CARS found in the SASHELP library. The GROUP= option creates a separate pie for each model. In combination with the GROUP= option, the ACROSS= option draws two charts across one page. The OTHER= option collects all the midpoints with statistic values less than or equal to 5% of the total into one slice. CLOCKWISE begins drawing the slices at the 12 o’clock position in alphabetic order of the midpoint. The PERCENT=OUTSIDE and SLICE=OUTSIDE display the labels outside the slices.

proc gchart data=sashelp.cars(where=(type="SUV" or type="Truck"));
  pie make / group=type
    across=2
    other=5 otherlabel="Other Makes"
    clockwise value=none
    slice=outside percent=outside;
run;
quit;

Example 11: Specifying the Sum Statistic in a Star Chart

Features: STAR statement option SUMVAR=

Other features: FORMAT statement

Sample library member: GCHSTSUM

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

This example produces a star chart of total sales for three sites by charting the values of the character variable SITE and calculating the sum of the variable SALES for each site. It represents the statistics as slices of the star. The center of the circle represents 0 and the edge of the circle represents the largest value, in this case Paris sales. By default, the spines are joined and filled with a solid pattern to form slices. In addition, the midpoint value and the formatted values of the sales statistics are printed beside each slice.
Program

goptions reset=all border;

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Tools Sydney 4 1775.74
Tools Atlanta 4 3424.19
Tools Paris 4 6914.25
;
title "Total Sales";
proc gchart data=totals;
format sales dollar8.;
star site / sumvar=sales;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create data set TOTALS. TOTALS contains quarterly sales data for three manufacturing sites for one year. Sales figures are broken down by department.

data totals;
length dept $ 7 site $ 8;
input dept site quarter sales;
datalines;
Parts Sydney 1 7043.97
Parts Atlanta 1 8225.26
Parts Paris 1 5543.97
Define the title. This title appears on the chart.

```sas
   title "Total Sales";
```

Produce the star chart. The SUMVAR= option calculates the sum of SALES for each value of the chart variable SITE. Because the TYPE= option is omitted, the default statistic is sum. The FORMAT statement assigns a format to the variable SALES.

```sas
   proc gchart data=totals;
   format sales dollar8.;
   star site / sumvar=sales;
   run;
   quit;
```

---

**Example 12: Charting a Discrete Numeric Variable in a Star Chart**

<table>
<thead>
<tr>
<th>Features:</th>
<th>STAR statement options DISCRETE, FILL=, NOCONNECT, NOHEADING, and SUMVAR=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other features:</td>
<td>GOPTIONS statement option BORDER</td>
</tr>
<tr>
<td>Sample library member:</td>
<td>GCHDSCRT</td>
</tr>
<tr>
<td>Note:</td>
<td>The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.</td>
</tr>
</tbody>
</table>

This example produces two star charts that show the total number of parts that were rejected each month for a year. The STAR statement uses the DISCRETE option so that each unique value of the numeric variable DATE is a separate midpoint and has a separate spine. Each slice displays the formatted midpoint value and the chart statistic. Specifying FILL=S rotates the solid pattern through all the colors in the style's list of colors as many times as necessary to provide patterns for all the slices.

The second chart uses the NOCONNECT option so that the chart uses spines instead of slices.
Program

goptions reset=all border;

data rejects;
  informat date date9.;
  input site $ date badparts;
  datalines;
  Sydney 01JAN1997  8
  Sydney 01FEB1997 11
  Sydney 28JUN1997 13
  Sydney 31OCT1997  6
  Paris 11APR1997 12
  Paris 04MAY1997 12
  Paris 30AUG1997 14
  Paris 01DEC1997  7
  Atlanta 15MAR1997  7
  Atlanta 18JUL1997 12
  Atlanta 03SEP1997 10
  Atlanta 12NOV1997  9
  ;

  title "Rejected Parts";
Program Description

**Set the graphics environment.** The BORDER option in the GOPTIONS statement draws a black border around the graph.

```sas
options 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.

```sas
data rejects;
informat date date9.;
input site $ date badparts;
datalines;
Sydney 01JAN1997 8
Sydney 01FEB1997 11
Sydney 28JUN1997 13
Sydney 31OCT1997 6
Paris 11APR1997 12
Paris 04MAY1997 12
Paris 30AUG1997 14
Paris 01DEC1997 7
Atlanta 15MAR1997 7
Atlanta 18JUL1997 12
Atlanta 03SEP1997 10
Atlanta 12NOV1997 9
;
```

**Define the title.** This title appears on the chart.

```sas
title "Rejected Parts";
```

**Produce the first star chart.** The DISCRETE option must be specified because the numeric chart variable, DATE is assigned the WORDDATE3. Using FILL=S fills all the slices with solid patterns. The embedded RUN statement is required to end this first RUN-Group and honor the action statement and other SAS statements. It signals the procedure to remain active.

```sas
proc gchart data=rejects;
format date worddate3.;
star date / discrete
sumvar=badparts
noheading
fill=s;
run;
star date / discrete
sumvar=badparts
noheading
noconnect;
run;
quit;
```
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.

```sas
star date / discrete
sumvar=badparts
noheading
noconnect;
run;
```

Example 13: Creating a Detail Pie Chart

**Features:**
- PIE statement options DETAIL=, DETAIL_PERCENT=, DETAIL_SLICE=, DETAIL_THRESHOLD=, DETAIL_VALUE=, and LEGEND

**Other features:**
- GOPTIONS statement option BORDER

**Sample library member:**
- GCHDTPIE

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

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.

**Types of Vehicles Produced Worldwide (Details)**
Program

goptions reset=all border;
title "Types of Vehicles Produced Worldwide (Details)";

proc gchart data=sashelp.cars;
    pie type / detail=drivetrain
detail_percent=best
detail_value=none
detail_slice=best
detail_threshold=2
    legend
    ;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the title. This title appears on the chart.

title "Types of Vehicles Produced Worldwide (Details)";

Produce the detail pie chart. This graph uses the data set entitled CARS found in the SASHELP library. The DETAIL= option produces an inner pie overlay showing the percentage that each DRIVETRAIN contributes toward each type of vehicle. The DETAIL_PERCENT= option and the DETAIL_SLICE= option control the positioning of the detail slice labels. The DETAIL_VALUE= option turns off the display of the number of DRIVETRAINS for each detail slice. The DETAIL_THRESHOLD= option shows all detail slices that contribute more than 2% of the entire pie. The LEGEND option displays a legend for the slice names and their midpoint values, instead of printing them beside the slices.

proc gchart data=sashelp.cars;
    pie type / detail=drivetrain
detail_percent=best
detail_value=none
detail_slice=best
detail_threshold=2
    legend
    ;
run;
quit;

References


Overview: GCONTOUR Procedure

The GCONTOUR procedure enables you to generate two-dimensional plots representing three-dimensional relationships. For example, the following Figure 36.1 on page 1008 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.
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

**CONTOUR Plot Terms**
Input Data

The GCONTOUR procedure requires three numeric variables to produce a plot. The input data set forms a rectangular grid from the values of $x$ and $y$. The $z$ variable is plotted on the grid as the third dimension. Only one value of $z$ is required for each $(x,y)$ grid location. If multiple observations have the same $z$ value for any $(x,y)$ combination, only the last observation is plotted.

Data Ranges

PROC GCONTOUR produces a rectangular grid with axes scaled to include the minimum data values and maximum data values of $x$ and $y$. Each axis is labeled with the variable name or label. The contour lines represent the levels of magnitude by grouping the common values of the $z$ variable. The level of each contour line is displayed in the legend. The legend label is the $z$ variable's name or label.
Missing Values

PROC GCONTOUR requires data values for at least 50% of the z variable, for each unique combination of (x,y). The INCOMPLETE option can be used to override this requirement. The G3GRID procedure can also be used to create data for missing values. (See Chapter 47, “G3GRID Procedure,” on page 1417).

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 47, “G3GRID Procedure,” on page 1417. For an interpolation example see “Example 1: Using the Default Interpolation Method” on page 1428.

Syntax: GCONTOUR Procedure

Restriction: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

Requirement: At least one PLOT statement is required.

Global statements: AXIS , FOOTNOTE, GOPTIONS, LEGEND, PATTERN, SYMBOL, TITLE

Note: The procedure can include the SAS/GRAPH statement BY on page 370, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

PROC GCONTOUR <DATA=input-data-set>
<ANNOTATE=Annotate-data-set>
<GOUT=<libref.>output-catalog>
<INCOMPLETE>;
   PLOT y*x=z <options>;

PROC GCONTOUR Statement

Identifies the data set that contains the plot variables. Can also specify an annotate data set, an output catalog, and the incomplete option.

Requirement: An input data set is required.
Syntax

PROC GCONTOUR <DATA= input-data-set>
<ANNOTATE=Annotate-data-set>
<GOUT=<libref.;output-catalog>
<INCOMPLETE>;

Optional Arguments

PROC GCONTOUR statement options affect all graphs produced by the procedure.

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs produced by the GCONTOUR procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

 DATA= 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.

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.

PLOT Statement

Creates contour plots using the values of three numeric variables from the input data set as the source of the contour coordinates.

Requirement: A plot request is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, LEGEND, NOTE, PATTERN, SYMBOL, TITLE
Syntax

PLOT y*x=z <option(s)>;

Summary of Optional Arguments

Appearance options

ANNOTATE= Annotate-data-set
  specifies an Annotate data set to enhance the charts produced by the PLOT statement.

CAXIS=axis-color
  specifies a color for axis lines, axis tick marks, and the frame around the plot.

CFRAME=background-color
  fills the axis area with the specified color.

COUTLINE=outline-color
  specifies a color for outlining filled areas.

CTEXT=text-color
  specifies a color for the axis labels, axis tick mark values, legend labels, and legend value descriptions.

GRID
  draws reference lines at all major tick marks on both axes.

LEGEND=LEGEND<1...99>
  assigns legend characteristics from the corresponding legend definition to the plot's legend.

NOAXIS
  specifies that a plot have no axis values, axis labels, or axis tick marks.

NOFRAME
  suppresses the frame that is drawn around the plot area.

NOLEGEND
  suppresses the legend that describes the plot by displaying the z variable name or label, the legend values, and legend value descriptions.

Catalog entry description options

DESCRIPTION="description"
  specifies a description of the output.

NAME="name"
  specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Contour options

CLEVELS=color(s)
  specifies a color or list of colors for the contour levels.

JOIN
  combines adjacent grid cells with the same pattern to form a single pattern area.

LEVELS=value-list
  specifies up to 100 values for the z variable.

LJOIN
  displays filled contour areas with contour lines.

LLEVELS=line-type-list
  lists line types for plot contour lines.
NLEVELS=number-of-levels
  specifies the number of contour levels to plot.

PATTERN
  specifies that the plot contour levels are represented by rectangles filled with patterns.

SMOOTH
  produces smooth gradient areas between levels.

**Horizontal axis options**

AUTOHREF
  displays reference lines originating at the major tick marks on the horizontal axis.

CHREF=reference-line-color | (reference-line-color) | (reference-line-color list)
  specifies a color or colors for reference lines drawn with the HREF= option and the AUTOHREF option.

HAXIS=AXIS<1...99>
  assigns axis characteristics from the corresponding axis definition to the horizontal x axis.

HMINOR=number-of-minor-tick marks
  specifies the number of minor tick marks to draw between each major tick mark on the horizontal x axis.

HREF=value | (value) | (value-list)
  displays up to 100 reference lines originating on the horizontal x axis at specified values within the x axis range.

HREVERSE
  specifies that the order of the values on the horizontal x axis be reversed.

LAUTOHREF=reference-line-type
  specifies a line type for reference lines specified by the AUTOHREF option.

LHREF=reference-line-type | (reference-line-type) | (reference-line-type list)
  specifies line types for reference lines originating on the horizontal axis.

WAUTOHREF=reference-line-width
  specifies a line width for reference lines specified by the AUTOHREF option.

WHREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
  specifies a line width for reference lines specified by the HREF= option.

XTICKNUM=number-of-major-tick-marks
  specifies the number of major tick marks on the horizontal x axis.

**Labeling option**

AUTOLABEL | AUTOLABEL=(autolabel_suboptions)
  labels the contour lines.

**Vertical axis options**

AUTOVREF
  displays reference lines originating at the major tick marks on the vertical axis.

CAUTOHREF=reference-line-color
  specifies a color for all the reference lines displayed by the AUTOHREF option.

CAUTOVREF=reference-line-color
  specifies a color for all the reference lines displayed by the AUTOVREF option.
CVREF=reference-line-color | (reference-line-color) | (reference-line-color list)
specifies a color or colors for reference lines drawn with the VREF= option
and the AUTOVREF option.

LAUTOVREF=reference-line-type
specifies a line type for reference lines specified by the AUTOVREF option.

LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines originating on the vertical axis.

VAXIS=AXIS<1...99>
assigns axis characteristics from the corresponding axis definition to the
vertical y axis.

VMINOR=number-of-minor-tick-marks
specifies the number of minor tick marks located between each major tick
mark on the vertical y axis.

VREF=value | (value) | (value-list)
displays up to 100 reference lines originating on the vertical y axis at
specified values within the y axis range.

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

WAUTOVREF=reference-line-width
specifies a line width for reference lines specified by the AUTOVREF option.

WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies a line width for reference lines specified by the VREF= option.

YTICKNUM=number-of-major-tick-marks
specifies the number of major tick marks on the vertical y axis.

**Required Argument**

\[ y*x=z \]
specifies three numeric variables from the input data set:

- \[ y \]
is the variable that is plotted on the vertical axis.

- \[ x \]
is the variable that is plotted on the horizontal axis.

- \[ z \]
is the variable that is plotted as contour lines.

**Optional Arguments**

Options in a PLOT statement affect all graphs that are produced by that statement. You
can specify as many options as you want and list them in any order. If you use a BY
statement on the procedure, the options in each PLOT statement affect all graphs
produced by that BY statement.

**ANNOTATE= Annotate-data-set**
specifies an Annotate data set to enhance the charts produced by the PLOT
statement.

<table>
<thead>
<tr>
<th>Alias</th>
<th>ANNO=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>Partially supported by Java and ActiveX</td>
</tr>
<tr>
<td>See</td>
<td>Chapter 32, “GANNO Procedure,” on page 779</td>
</tr>
</tbody>
</table>
AUTOHREF
displays reference lines originating at the major tick marks on the horizontal axis.

Restriction
Not supported by Java

AUTOLABEL | AUTOLABEL=(autolabel_suboptions)
labels the contour lines. Autolabel suboptions enable you to control the appearance of these labels.

The label for each contour line is the value of the z variable for that contour level. The labels are displayed in BEST format. The FORMAT statement enables you to change the display format.

You can also use the SYMBOL statement to control the appearance and text of contour labels and lines.

When AUTOLABEL is used with the LLEVELS= option, LLEVELS is ignored.
When AUTOLABEL is used with the CLEVELS= option, AUTOLABEL is ignored.

Restriction
Not supported by Java and ActiveX

See
“AUTOLABEL= Suboptions” on page 1025

Example
“Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033

AUTOVREF
displays reference lines originating at the major tick marks on the vertical axis.

Restriction
Not supported by Java

CAUTOHREF=reference-line-color
specifies a color for all the reference lines displayed by the AUTOHREF option. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction
Not supported by Java

CAUTOVREF=reference-line-color
specifies a color for all the reference lines displayed by the AUTOVREF option. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified

Restriction
Not supported by Java

CAXIS=axis-color
specifies a color for axis lines, axis tick marks, and the frame around the plot. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction
Partially supported by Java
CFRAME=background-color
fills the axis area with the specified color. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CFR=

CHREF=reference-line-color | (reference-line-color) | (reference-line-color list)
specifies a color or colors for reference lines drawn with the HREF= option and the AUTOHREF option. Colors are specified as follows:
• specifying a single color without parentheses applies that color to all reference lines drawn with the HREF= option and the AUTOHREF option
• specifying a single color within parentheses applies the color to the first reference line drawn with the HREF= option only
• specifying a list of colors within parentheses applies the colors sequentially to the reference lines drawn with the HREF= option only

The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CH=

Restriction Not supported by Java

CLEVELS=color(s)
specifies a color or list of colors for the contour levels. GCONTOUR substitutes user-defined colors in the ODS style. If more colors are needed, GCONTOUR uses the next color in the ODS style until all lines have an associated color. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Restriction Not supported by Java and partially supported by ActiveX

COUTLINE=outline-color
specifies a color for outlining filled areas. This option is ignored unless the PATTERN option is also used. COUTLINE=SAME creates a plot with outlines that are the same color as the adjacent fill color.

Restriction Not supported by Java and ActiveX

Note The outline color is the only distinction between empty patterns. Use of this option makes the patterns look the same when VALUE=EMPTY in PATTERN definitions.

Example “Example 4: Using Patterns and Joins” on page 1038

CTEXT=text-color
specifies a color for the axis labels, axis tick mark values, legend labels, and legend value descriptions. GCONTOUR uses the first color that it finds in the following list:
• colors specified for labels and values on assigned AXIS and LEGEND statements.
• the color specified by the CTEXT= option in the PLOT statement.
• the color specified by the CTEXT= option in a GOPTIONS statement.
• the color specified in the current style, or if the NOGSTYLE system option is specified, the default color is the first color in the color list for each device.
The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification. In this case CTEXT= is used for axis labels and values. Meanwhile, the COLOR= color is still used for all other axis colors, such as tick marks.

Note: If you use a BY statement in the procedure, the color of the BY variables' labels is controlled by the CBY= option in the GOPTIONS statement.

Example: “Example 4: Using Patterns and Joins” on page 1038

CVREF=reference-line-color | (reference-line-color) | (reference-line-color list) specifies a color or colors for reference lines drawn with the VREF= option and the AUTOVREF option. Colors are specified as follows:

- specifying a single color without parentheses applies that color to all reference lines drawn with the VREF= option and the AUTOVREF option
- specifying a single color within parentheses applies the color to the first reference line drawn with the VREF= option only
- specifying a list of colors within parentheses applies the colors sequentially to the reference lines drawn with the VREF= option only

Alias: CV=

Restriction: Not supported by Java

DESCRIPTION="description" specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open
- the description and the properties for the output in the Results window
- the description and properties for the catalog entry in the SAS Explorer
- the Description field of the PROC GREPLAY window

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias: DES=

Default: PLOT OF y*x=z, where y*x=z is the request specified in the PLOT statement.
GRID
draws reference lines at all major tick marks on both axes. This option produces the
same result as would be achieved with specifying both the AUTOHREF and
AUTOVREF options.

Restriction  Not supported by Java

HAXIS=AXIS<1...99>
assigns axis characteristics from the corresponding axis definition to the horizontal x
axis. If the AXIS statement specifies the REFLABEL= option, labels are applied in
sequence to all reference lines generated with the HREF= option.

Restriction  Partially supported by Java and ActiveX

See  “AXIS Statement” on page 345

Example  “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis,
Modifying the Legend” on page 1033

HMINOR=number-of-minor-tick marks
specifies the number of minor tick marks to draw between each major tick mark on
the horizontal x axis.

Alias  HM=

Interaction  The HMINOR= option overrides the MINOR= option in an AXIS
definition assigned to the horizontal axis

Example  “Example 3: Specifying Contour Levels” on page 1036

HREF=value | (value) | (value-list)
displays up to 100 reference lines originating on the horizontal x axis at specified
values within the x axis range. Any values specified beyond the axis range are not
drawn, and a warning is issued to the log. To specify labels for this option, use the
HAXIS= option. The value-list can be an explicit list of values, a starting value and
an ending value with an interval increment, or a combination of both forms:

•  \[ n \ldots n \]

•  \[ n \text{ TO } n < \text{BY increment} > \]

•  \[ n < \ldots n \text{ TO } n < \text{BY increment} > n < \ldots n \]

Restriction  Not supported by Java

HREVERSE
specifies that the order of the values on the horizontal x axis be reversed.

Restriction  Not supported by Java

JOIN
combines adjacent grid cells with the same pattern to form a single pattern area.

Restriction  This option is ignored unless the PATTERN option is also used.

Note  Java and ActiveX support the JOIN option without the pattern option.
LAUTOHREF=reference-line-type
specifies a line type for reference lines specified by the AUTOHREF option. The reference-line-type value is any integer from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Default 1 (solid)

Restriction Not supported by Java

See “Specifying Line Types” on page 442 for available line types

LAUTOVREF=reference-line-type
specifies a line type for reference lines specified by the AUTOVREF option. The reference-line-type value is any integer from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Restriction Not supported by Java

See “Specifying Line Types” on page 442 for available line types

LEGEND=LEGEND<1...99>
assigns legend characteristics from the corresponding legend definition to the plot's legend. To suppress the legend, use the NOLEGEND option. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect.

Restriction Partially supported by Java (always displayed on the right side of plot) and ActiveX

Interaction If you use the SHAPE= option in a LEGEND statement, the value LINE is valid. If you use the PATTERN option, SHAPE=BAR is also valid.

See “LEGEND Statement” on page 377

Example “Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend” on page 1033

LEVELS=value-list
specifies up to 100 values for the z variable. Because GCONTOUR uses the z variable to calculate plot contour levels, you can use the LEVELS= option to change the number of contour levels plotted.

The value-list can be an explicit list of values, a starting value and an ending value with an interval increment, or a combination of both forms:

- \( n <... n > \).
- \( n \) TO \( n \) <BY increment >.
- \( n <... n > \) TO \( n \) <BY increment > \( n <... n > \).

If a variable has an associated format, the specified values must be the unformatted values.

The contour lines on the plot represent the intersection of a plane formed by the \( (x,y) \) variables, and the surface that is formed by the values of the \( z \) variable.

LHREF=reference-line-type | (reference-line-type) | (reference-line-type list)
specifies line types for reference lines originating on the horizontal axis. The reference-line-type value is any integer from 1 to 46. A value of 1 specifies a solid
line; values 2 through 46 specify dashed lines. When using this option, the following is true:

- specifying a single line type without parentheses applies that line type to all reference lines drawn with the HREF= option and the AUTOHREF option
- specifying a single line type within parentheses applies the line type to the first reference line drawn with the HREF= option only
- specifying a list of line types within parentheses applies the line types sequentially to the reference lines drawn with the HREF= option only
- the LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option

Alias LH=

Restriction Not supported by Java and partially supported by ActiveX

See “Specifying Line Types” on page 442 for available line types

LJOIN

displays filled contour areas with contour lines.

Restriction Supported by Java and ActiveX only

LLEVELS=\textit{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 \textit{line-type-list} are integers from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines.

Default 1 (solid)

Restriction Not supported by Java and partially supported by ActiveX

See “Specifying Line Types” on page 442 for the line types represented by each number.

Example “Example 3: Specifying Contour Levels” on page 1036

LVREF=\textit{reference-line-type} | (\textit{reference-line-type}) | (\textit{reference-line-type-list})

specifies line types for reference lines originating on the vertical axis. Valid values for \textit{line-type-list} are integers from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. When using this option, the following is true:

- specifying a single line type without parentheses applies that line type to all reference lines drawn with the VREF= option and the AUTOVREF option
- specifying a single line type within parentheses applies the line type to the first reference line drawn with the VREF= option only
- specifying a list of line types within parentheses applies the line types sequentially to the reference lines drawn with the VREF= option only
- the LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option

Alias LV=
Restriction: Partially supported by Java and ActiveX

See “Specifying Line Types” on page 442 for the line types represented by each number.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAF appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

*Note:* Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default: GCONTOUR

Example: “Example 3: Specifying Contour Levels” on page 1036

NLEVELS=number-of-levels
specifies the number of contour levels to plot.

Valid values are integers from 1 to 100.
There are 2 algorithms used to compute the contour levels, where L represents an array of levels.

If the value of NLEVELS= is less than 7, then the algorithm is as follows:

\[
D = \frac{(Z_{\text{max}} - Z_{\text{min}})}{\text{NLEVELS}} \\
de = 0.5 \times D \\
L[0] = Z_{\text{min}} + d, \quad L[i] = L[i-1] + D
\]

In this algorithm, each level is the midpoint of a number of ranges equal to the value of the NLEVELS= option. These ranges exactly cover the range of the \( z \) variable.

If the value of NLEVELS= is greater than or equal to 7, then the algorithm is as follows:

\[
e = 0.05 \times \frac{(100 - \text{NLEVELS})}{93} \\
d = (Z_{\text{max}} - Z_{\text{min}}) \times e \\
D = \frac{(Z_{\text{max}} - Z_{\text{min}} - 2d)}{(\text{NLEVELS} - 1)} \\
L[0] = Z_{\text{min}} + d, \quad L[i] = L[i-1] + D
\]

Default 20 for Java, 6 for ActiveX, 7 for all other devices

Restriction Partially supported by Java and ActiveX

Example “Example 3: Specifying Contour Levels” on page 1036

NOAXIS specifies that a plot have no axis values, axis labels, or axis tick marks. The frame is displayed around the plot unless you use the NOFRAME option.

Alias NOAXES

Restriction Partially supported by Java

NOFRAME suppresses the frame that is drawn around the plot area.

Restriction Not supported by Java

NOLEGEND suppresses the legend that describes the plot by displaying the \( z \) variable name or label, the legend values, and legend value descriptions.

Default LEGEND

PATTERN specifies that the plot contour levels are represented by rectangles filled with patterns. The pattern for each rectangle is determined by calculating the mean of the values of the \( z \) variable for the four corners of the rectangle. The pattern is then assigned for the level closest to the mean.

To explicitly define patterns, use PATTERN definitions for map or plot patterns.

See “PATTERN Statement” on page 398

Example “Example 4: Using Patterns and Joins” on page 1038

SMOOTH produces smooth gradient areas between levels.
Restriction  Supported by Java and ActiveX only

**VAXIS=AXIS<1...99>**

Assigns axis characteristics from the corresponding axis definition to the vertical \( y \) axis. If the AXIS statement specifies the REFLABEL= option, labels are applied in sequence to all reference lines generated with the VREF= option.

Restriction  Partially supported by Java and ActiveX

See  “AXIS Statement” on page 345

Example  “Example 3: Specifying Contour Levels” on page 1036

**VMINOR=number-of-minor-tick marks**

Specifies the number of minor tick marks located between each major tick mark on the vertical \( y \) axis. Value labels are not displayed for minor tick marks.

Alias  VM=

Interaction  The VMINOR= option overrides the MINOR= option in an AXIS definition that is assigned to the vertical axis.

Example  “Example 3: Specifying Contour Levels” on page 1036

**VREF=value | (value) | (value-list)**

Displays up to 100 reference lines originating on the vertical \( y \) axis at specified values within the \( y \) axis range. Any values specified beyond the axis range are not drawn, and a warning is issued to the log. To specify labels for these reference lines, use the VAXIS= option. The value-list can be an explicit list of values, a starting value and an ending value with an interval increment, or a combination of both forms:

- \( n <... n > \).
- \( n \) TO \( n <BY increment > \).
- \( n <... n > \) TO \( n <BY increment > n <... n > \).

Restriction  Not supported by Java

**VREVERSE**

Specifies that the order of the values on the vertical axis be reversed.

Restriction  Not supported by Java

**WAUTOHREF=reference-line-width**

Specifies a line width for reference lines specified by the AUTOHREF option. The reference-line-width can be any number greater than zero, and does not need to be an integer. If you specify a value that is less than zero, an error message is written to the SAS log.

Default  Current style setting, 1 if NOGSTYLE

Restriction  Not supported by Java and ActiveX

**WAUTOVREF=reference-line-width**

Specifies a line width for reference lines specified by the AUTOVREF option. The reference-line-width can be any number greater than zero, and does not need to be an
integer. If you specify a value that is less than zero, an error message is written to the SAS log.

Default: Current style setting, 1 if NOGSTYLE

Restriction: Not supported by Java and ActiveX

**WHREF=** *(reference-line-width) | (reference-line-width-list)*
specifies a line width for reference lines specified by the HREF= option. The *(reference-line-width)* can be any integer that is greater than zero. If you specify a value that is less than zero or is not an integer value, an error message is written to the SAS log. Specifying a line width without parentheses applies that type to all reference lines drawn with the WAUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=*(reference-line-width)* for reference lines drawn with the WAUTOHREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the HREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HREF= option. The syntax of the line-width list requires parentheses and the line widths separated by spaces *(width1 width2 ...widthN)*. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.). To specify colors for these reference lines, use the CHREF= option.

Default: Current style setting, 1 if NOGSTYLE

Restriction: Not supported by Java and ActiveX

**WVREF=** *(reference-line-width) | (reference-line-width-list)*
specifies a line width for reference lines specified by the VREF= option. The *(reference-line-width)* can be any integer that is greater than zero. If you specify a value that is less than zero or is not an integer value, an error message is written to the SAS log. Specifying a line width without parentheses applies that type to all reference lines drawn with the WAUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=*(reference-line-width)* for reference lines drawn with the WAUTOVREF option. Specifying a single line width in parentheses applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and the line widths separated by spaces *(width1 width2 ...widthN)*. The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.). To specify colors for these reference lines, use the CVREF= option.

Default: Current style setting, 1 if NOGSTYLE

Restriction: Not supported by Java and ActiveX

**XTICKNUM=** *(number-of-major-tick-marks)*
specifies the number of major tick marks on the horizontal \(x\) axis. The value of *(number-of-major-tick-marks)* must be greater than or equal to 2. The MAJOR= and ORDER= options in an AXIS definition that is assigned to the horizontal \(x\) axis overrides the XTICKNUM= option.

**YTICKNUM=** *(number-of-major-tick-marks)*
specifies the number of major tick marks on the vertical \(y\) axis. The value of *(number-of-major-tick-marks)* must be greater than or equal to 2. The MAJOR= and ORDER= options in an AXIS definition that is assigned to the vertical \(y\) axis overrides the YTICKNUM= option.
**AUTOLABEL= Suboptions**

The AUTOLABEL= option accepts the following autolabel suboptions:

- **CHECK=checking-factor | NONE**
  
  specifies a collision checking factor that controls collisions between contour label text and other contour lines or other labels. Values can be integers from 0 to 100, inclusive, where 0 provides minimal collision checking and 100 provides maximal collision checking. Fractional values are permitted.

  CHECK=NONE suppresses contour label collision checking and can lessen the time needed to compute the contour plot.

  Default 75

- **MAXHIDE=amount <units>**

  specifies the maximum amount of the contour line that can be hidden by contour labels. The value of amount must be greater than zero.

  Valid units are CELLS (horizontal character cell positions), CM (centimeters), IN (inches), or PCT (percentage of the width of the graphics output area). If you omit units, a unit specification is searched for in this order:

  1. the GUNIT= option in a GOPTIONS statement
  2. the default unit, CELLS

  For units that you specify as PCT or CELLS, the MAXHIDE= suboption calculates the amount of contour line that can be hidden based on the width of the graphics output area. For example, suppose you specify MAXHIDE=50 PCT and the graphics output area is 9 inches wide. Then the maximum amount of the contour line that can be hidden by labels is 4.5 inches.

  This option maintains data integrity. It provides a check for overly small increments in the STEP= option in the SYMBOL statement. Moreover, the MAXHIDE= option can prevent small contours from being significantly hidden even when the value of the STEP= option is sufficiently large.

  Default MAXHIDE=100

- **REVEAL**

  specifies that the contour lines are visible through the label text as dashed lines. Line style 33 is used. This option provides a simple way to see all portions of labeled contours, and can be used to inspect the label positions with respect to the contour lines. The REVEAL option is primarily used for debugging. Occasionally, single-character contour labels can be placed off center from the clipped portion of the contour line when the contour line is irregular or jagged.

- **TOLANGLE=angle**

  specifies the maximum angle (the tolerance angle) between any two adjacent characters of a contour label. The value of angle must be between 1 and 85 degrees. To force contour labels to fall on very smooth sections, specify a small tolerance angle.

  Default 30
Details

Description
The PLOT statement specifies the three variables to plot. It can also control the contour levels, label the plot lines, and modify the axes as well as the general appearance of the graph. Only one plot request can be specified in a PLOT statement.

To specify multiple plots for a single PROC GCONTOUR statement, use multiple PLOT statements. If a global statement is specified more than once, the last occurrence is applied to all PLOT statements in that PROC step.

The PLOT statement does the following actions:
• plots the contour lines as levels using the values of the $z$ variable
• scales the axes to include the minimum data values and the maximum values of $x$ and $y$
• labels the $x$ and $y$ axes
• generates a labeled legend representing the values of the $z$ variable's contour levels

Global statements enable you to modify the axes, the legend, the contour lines and contour line labels, the fill patterns and pattern colors for contour areas. You can also add titles, footnotes, and notes to the plot. You can use an Annotate data set and set GOPTIONS to enhance the appearance of the plot. Moreover, you can filter your data with a WHERE clause. You can format your data values and add labels to the variables. You can generate multiple plots with a BY statement or multiple plot statements.

Selecting Contour Levels
The LEVEL= option enables you to customize your plot, by specifying values for the contour levels.

The LEVELS= option represents the $z$ variable values as a third dimension, using uniquely colored contour lines.

Figure 36.3 Selecting Contour Levels
Using the PATTERN option with the LEVELS= option generates a plot with contour levels that are displayed as solid filled rectangles. The rectangles are formed by points in the \((x, y)\) grid. The contour pattern of a rectangle, or grid cell, is determined by average value of the \(z\) variable for the four corners of the rectangle. The grid cell is assigned the pattern for the level closest to the calculated mean. For example, suppose you specified contour levels of 0, 5, and 10, and the plot contains a grid cell with a mean of 100. The grid cell is assigned the pattern for the nearest level: 10. A grid cell with a mean of 7.6 is also assigned the pattern for the 10 level. The same data used with the following PLOT statement in the GCONTOUR procedure produces a similar contour plot:

```
plot y*x=z / levels=-7.5 to 7.5 by 2.5/
    pattern;
run;
```

The following contour plot with the PATTERN option uses the same data and contour levels as “Example 4: Using Patterns and Joins” on page 1038. Contour plots using the same contour levels can present your data differently, if one plot uses a pattern and the other does not. However, the contour pattern boundaries in the following contour plot do not correspond to the contour lines shown in the example.

![Contour Plot with the PATTERN Option](image)

Using the data to create a surface plot with the G3D procedure, the contour lines in a GCONTOUR procedure plot represent the intersection of the plane and the surface.

For example, suppose that you use the G3D procedure, and your data produces a surface plot like the one shown below.

The contour lines represent the intersection of the surface lines with the plane parallel to the plane formed by the variables \(x\) and \(y\). The intersections are located at the \(z\) values of \(-7.5, -5.0, -2.5\), and so on.
Specifying Axis Order
You can use AXIS statements to modify the text and appearance of plot axes. You can then assign the axes to the contour plot with the PLOT statement's HAXIS= and VAXIS= options. If the AXIS statement uses an ORDER= option, there are special considerations for using that AXIS definition with the GCONTOUR procedure.

A list of variable values that are specified with the AXIS statement's ORDER= option must contain numbers listed in ascending or descending order. These numbers are treated as a continuous data range for an axis. Thus, for a contour line or pattern to span the entire specified range, it is not necessary for the maximum and minimum values of the list to match exactly with the maximum and minimum data values of the corresponding x or y variable. For example, suppose that you assign this AXIS definition to the horizontal (x) axis:

```
axis1 order=-2.5 to 2.5 by .5
```

Suppose also that the horizontal axis variable has these values: –5, –4, –3, –2, –1, 0, 1, 2, 3, 4, 5. Depending on the data, contours could extend through the full range of the ORDER= list rather than from –2 to 2. –2 to 2 are the actual values of the variable assigned to the horizontal (x) axis. In this case, values are interpolated for the x variable at any point where the y variable intersects the minimum axis value (–2.5) or the maximum axis value (2.5). Data values that are outside of the axis range (in this case, –5, –4, –3, 3, 4, and 5) are clipped from the plot.

When ORDER= lists cause data clipping, internal plotting grids are modified according to these rules:

- If an ORDER= list causes data clipping on a single axis, linear interpolation generates the z values of the starting or ending column, or both columns of the plotting grid. For example, in the previous example, the value of z is interpolated for –2.5 and 2.5 on the horizontal (x) axis.

- If ORDER= lists cause data clipping on both axes, the response variable values of the new corners are derived by fitting the new x, y location on a plane. This plane is formed by three of the original four points of the corresponding grid square.
When assigning the following AXIS definition to a plot of the same data, the contour levels on the plot do not extend beyond the range of the data:

```
axis1 order=-10 to 10 by 1;
```

**Figure 36.6**  The ORDER= Option, Values Match the Range of the Data Values

**Figure 36.7**  The ORDER= Option, Values Clip the Range of Data Values
Modifying Contour Lines and Labels with the SYMBOL Statement

Controlling Line and Label Attributes
When you use the AUTOLABEL option, the LLEVELS= and CLEVELS= options are ignored, and contour-line and label attributes are controlled by the SYMBOL statement. Defaults are used if not enough SYMBOL statements are specified to match the number of contour levels.

If a SYMBOL statement does not include a color option, the SYMBOL statement can be applied to more than one contour level. In this case, the SYMBOL statement is used once with every color in the color list and generates more than one SYMBOL definition. See the “SYMBOL Statement” on page 412 for details.

Table 36.1 on page 1030 describes how SYMBOL statement options affect contour plot lines and labels.

Table 36.1  The Effect of SYMBOL Statement Options on Contour Lines and Labels

<table>
<thead>
<tr>
<th>SYMBOL Statement Option</th>
<th>Contour Line or Label Element Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE=line-type</td>
<td>Contour line style</td>
</tr>
<tr>
<td>WIDTH=n</td>
<td>Contour line thickness</td>
</tr>
<tr>
<td>CI=line-color or COLOR=color</td>
<td>Contour line color</td>
</tr>
<tr>
<td>FONT=font</td>
<td>Contour label font</td>
</tr>
<tr>
<td>HEIGHT=height</td>
<td>Contour label height</td>
</tr>
<tr>
<td>CV=color or COLOR=color</td>
<td>Contour label color</td>
</tr>
</tbody>
</table>
**Example 1: Simple Contour Plot**

**Features:**
- PLOT statement

**Data set:**
- SASHELP.LAKE

**Sample library member:**
- GCTLAKE

**Note:**
- The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This simple contour plot displays the various depths of a lake. The dimensions of the lake are plotted on the x and y axes. The z variable is plotted as the third dimension, as levels represented by contour lines. The contour line levels are displayed and labeled in the legend.
Program

goptions reset=all border;

title "Lake Data";
footnote j=r "GCTLAKE";

proc gcontour data=sashelp.lake;
    plot length*width=depth;
run;
quit;

Program Description

Set the graphics environment. Draw a BORDER around the graphics output area.

goptions reset=all border;

Define titles and footnotes. Add TITLE content. Add FOOTNOTE content and placement.

title "Lake Data";
footnote j=r "GCTLAKE";

Generate contour plot. Generate a simple contour plot using SASHELP.LAKE. Use one PLOT statement to define the grid lines and the contour lines.

proc gcontour data=sashelp.lake;
    plot length*width=depth;
run;
quit;
Example 2: Labeling Contour Lines, Modifying the Horizontal Axis, Modifying the Legend

Features:
- PLOT statement options
  - AUTOLABEL=
  - HAXIS=
  - LEGEND=
  - NAME=

Other features:
- AXIS statement options
  - MINOR=
  - ORDER=
- LEGEND statement options
  - ACROSS=
  - LABEL=
  - POSITION=
- SYMBOL statement options
  - COLOR=
  - FONT=
  - HEIGHT=
  - VALUE=

Data set: SASHELP.LAKE
Sample library member: GCTLABEL

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

This example modifies “Example 1: Simple Contour Plot” on page 1031 to label contour levels with the AUTOLABEL option. The SYMBOL statement used with the AUTOLABEL option enables you to customize the attributes of the contour lines and labels.

In this example, SYMBOL1 and SYMBOL7 assign text labels, text fonts, text height, and text and line color for the first and seventh contour level lines. SYMBOL2–SYMBOL6 define the text height, and the text and line color for contour level lines 2–6.

Moreover, the LEGEND statement attributes are modified to reposition the legend closer to the data.
Program

goptions reset=all border;

title1 "Lake Data";
footnote1 j=r "GCTLABEL";
symbol1 value="DEEP"
  color=red
  font="Arial Rounded MT Bold"
  height=.6;
symbol2 color=green
  height=.45;
symbol3 color=blue
  height=.45;
symbol4 color=orange
  height=.45;
symbol5 color=purple
  height=.45;
symbol6 color=magenta
  height=.45;
symbol7 value="Shallow"
  color=navy
  font="Arial Rounded MT Bold"
  height=.7;

legend1 position=(right middle)
  label=(position=top)
  across=1;

axis1 order=(0 to 10 by 2) minor=none;

proc gcontour data=sashelp.lake;
  plot length*width=depth/
    autolabel=(check=none)
    haxis=axis1
    legend=legend1
    name="GCTLABEL";
run;
quit;

Program Description

**Set the graphics environment.** Draw a BORDER around the graphics output area.

```plaintext
goptions reset=all border;
```

**Define title and footnote.** Add TITLE content. Add FOOTNOTE content and placement.

```plaintext
title1 "Lake Data";
footnote1 j=r "GCTLABEL";
```

**Define symbol statements.** SYMBOL statements define the characteristics of the lines and labels for the contour lines. Each SYMBOL statement is associated with an individual contour level starting with the first level displayed in the LEGEND.

```plaintext
symbol1 values="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 values="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.

```plaintext
legend1 position=(right middle)
   label=(position=top)
   across=1;
```

**Define axis characteristics.** The ORDER= option specifies the order in which the data values appear on the axis; the ORDER= values are displayed as the major tick marks values; MINOR=NONE suppresses all minor tick marks.

```plaintext
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.

```sas
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

<table>
<thead>
<tr>
<th>Features</th>
<th>PLOT statement options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HMINOR=</td>
</tr>
<tr>
<td></td>
<td>LLEVELS=</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
</tr>
<tr>
<td></td>
<td>NLEVELS=</td>
</tr>
<tr>
<td></td>
<td>VAXIS=</td>
</tr>
<tr>
<td></td>
<td>VMINOR=</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AXIS statement option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER=</td>
</tr>
</tbody>
</table>

Data set: POLLEN

Sample library member: GCTNLVEL

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

This contour plot shows the amount of pollen in the air for five work days (x-axis) in a four-week series (y-axis). The PLOT statement uses the NLEVELS= option to specify the number of contour levels to plot for the z variable. The NLEVELS= option enables you to specify up to 100 levels in your plot.
Program

```
goptions reset=all border;

data pollen;
  input Week Workdays Pollen @@;
  datalines;
1 1 50 1 2 96 1 3 28 1 4 94 1 5 124 2 1 204 2 2 153 2 3 43 2 4 21 2 5 60
3 1 37 3 2 23 3 3 57 3 4 21 3 5 65 4 1 8 4 2 144 4 3 22 4 4 141 4 5 95
;  
run;

title1 "The Amount of Pollen Particles in a Cubic Meter of Air";
footnote1 j=r "GCTNLVEL";
axis1 order=(1 to 4 by 1);
proc gcontour data=pollen;
  plot week*workdays=pollen/
    hminor=0
    llevels= 2 20 21 33 25 41
    name="GCTNLVEL"
    nlevels=6
    vaxis=axis1
    vminor=0;
  run;
quit;
```

Program Description

**Set the graphics environment.** Draw a border around the graphics output area.
```
goptions reset=all border;
```

**Create data set.** Create the data set.
```
data pollen;
  input Week Workdays Pollen @@;
```
Add a title and a footnote. Add TITLE content. Add FOOTNOTE content and placement:

```plaintext
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:

```plaintext
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.

```plaintext
proc gcontour data=pollen;
plot week*workdays=pollen/
hminor=0
llevels= 2 20 21 33 25 41
name="GCTNLVEL"
nlevels=6
vaxis=axis1
vminor=0;
run;
quit;
```

Example 4: Using Patterns and Joins

**Features:**
- PLOT statement options
  - COOUTLINE=
  - CTEXT=
  - HAXIS=
  - JOIN
  - LEGEND=
  - PATTERN
  - VAXIS=

**Other features:**
- AXIS statement options
  - COLOR=
  - LABEL=
  - VALUE=
  - WIDTH=
- LEGEND statement

**Data set:** SWIRL
Sample library member: GCTPATJ

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

This example demonstrates the differences between using lines and patterns to represent contour levels. The first PLOT statement generates a plot with lines representing contour levels.

Output 36.1  Line Contour Levels
The second PLOT statement specifies the PATTERN option to fill and color contour levels. Additional PLOT statement options outline filled areas in gray and specify green text for all text on the axes and in the legend.

**Output 36.2  Pattern Contour Levels**

The third PLOT statement uses the JOIN option to combine adjacent grid cells with the same pattern to form a single pattern area. Additional options enhance the plot by modifying the axes and framing the legend.

**Output 36.3  Contour Plot with Joined Cells**

Program

```plaintext
options reset=all border;
data swirl;
```
do x= -5 to 5 by 0.25;
do y= -5 to 5 by 0.25;
    if x+y=0 then z=0;
    else z=(x*y)*((x*x-y*y)/(x*x+y*y));
output;
end;
end;
run;
title1 "Line Contour Levels";
footnote1 j=r "GCTPATR1";
proc gcontour data=swirl;
    plot y*x=z;
run;
quit;
title1 "Pattern Contour Levels";
footnote j=r "GCTPATR2";
proc gcontour data=swirl;
    plot y*x=z /
        ctext=green
        coutline=gray
        pattern;
run;
quit;
title "Contour Plot with Joined Cells";
footnote j=r "GCTPATR3";
axis1 label=none
    value=("-5" '' "0" '' "5")
    color=red
    width=3;
axis2 label=none
    value=("-5" '' "0" '' "5")
    color=red
    width=3;
legend1 frame;
proc gcontour data=swirl;
    plot y*x=z /
        haxis=axis1
        join
        legend=legend1
        pattern
        vaxis=axis2;
run;
quit;

Program Description

Set the graphics environment. Draw a border around the graphics output area.

goptions reset=all border;
Create the data set. The data set SWIRL is generated data that produces a symmetric contour pattern, which is useful for illustrating the pattern option.

```sas
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;
runc
```

Define the title and the footnote. Add TITLE content. Add FOOTNOTE content, and placement.

```sas
title1 "Line Contour Levels";
footnote1 j=r "GCTPATR1";
```

Generate the first contour plot. Generate a simple contour plot.

```sas
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.

```sas
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.

```sas
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.

```sas
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.

```sas
axis1 label=none
  value=(-5 ' ' "0" ' ' "5")
  color=red
  width=3;
axis2 label=none
```
Define the legend characteristics. Add a frame around the legend.

```sql
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.

```sql
proc gcontour data=swirl;
plot y*x=z /
    haxis=axis1
    join
    legend=legend1
    pattern
    vaxis=axis2;
run;
quit;
```

References

Chapter 37
GDEVICE Procedure

Overview: GDEVICE Procedure .................................................. 1046

Device Catalogs ................................................................. 1046
  The DEVICES Catalogs .................................................... 1046
  The Current Catalog ...................................................... 1046
  Search Order of Device Catalogs ....................................... 1047

Ways to Use the GDEVICE Procedure ................................. 1047
  Procedure Windows versus Program Code ........................ 1047
  Running the GDEVICE Procedure in a Windowing Environment 1048
  Running the GDEVICE Procedure in Program Mode ............. 1048

Syntax: GDEVICE Procedure ................................................ 1049
  PROC GDEVICE Statement ............................................... 1049
  ADD Statement .................................................................. 1050
  COPY Statement .................................................................. 1051
  DELETE Statement ............................................................ 1052
  FS Statement ..................................................................... 1052
  LIST Statement ................................................................... 1052
  MODIFY Statement ............................................................ 1053
  QUIT Statement .................................................................. 1054
  RENAME Statement ........................................................... 1054

Using the GDEVICE Windows ........................................... 1055
  About the GDEVICE Windows .......................................... 1055
  GDEVICE Window Commands .......................................... 1055
  DIRECTORY Window ....................................................... 1056
  Detail Window ................................................................. 1056
  Parameters Window ........................................................ 1057
  Geolors Window .............................................................. 1057
  Chartype Window ............................................................ 1058
  Colormap Window ........................................................... 1058
  Metagraphics Window ...................................................... 1059
  Gprolog Window ............................................................. 1059
  Gepilog Window .............................................................. 1059
  Gstart Window ................................................................. 1059
  Gend Window ................................................................. 1059
  Host File Options Window .............................................. 1060
  Host Commands Window ................................................ 1060

Creating or Modifying Device Entries ............................ 1060
  Write Access, Personal Catalogs, and Device Entry Names .... 1060
  Creating a New Device Entry ........................................... 1061
  Modifying an Existing Device Entry ................................. 1062
Overview: GDEVICE Procedure

The GDEVICE procedure is a tool for examining and changing the parameters of the graphics device driver catalog entries used with SAS/GRAPH software. With the GDEVICE procedure, you can use either the GDEVICE windows or GDEVICE procedure statements to do the following:

- list the device entries stored in any DEVICES catalog
- view the parameters for any device entry
- create and modify new device entries
- copy, modify, rename, or delete existing device entries

See Chapter 9, “Using Graphics Devices,” on page 79 for a discussion of device drivers and device entries, as well as directions for selecting device drivers, and changing the settings of device parameters.

For a complete list of device entries supplied by SAS that are supported by your operating environment, see the SASHELP.DEVICES catalog.

Device Catalogs

The DEVICES Catalogs

Device entries are stored in SAS catalogs that are named libref:DEVICES. Device entries for your operating environment that SAS supplies with SAS/GRAPH software are stored in the SASHELP.DEVICES.

Custom device entries are typically stored in a catalog named GDEVICEn.DEVICES (where n can be any number from 0 to 9). However, device entries that have been created or modified by a system administrator specifically for your site also might be stored in SASHELP.DEVICES. (On multi-user systems, the administrator is usually the person who has Write access to the SASHELP.DEVICES catalog.)

The Current Catalog

When the GDEVICE procedure determines which catalog it should use, it searches for the catalog in the following order:

1. the catalog name specified in the CATALOG= option in the PROC GDEVICE statement.
2. the catalog associated with the GDEVICE0 libref, if the libref has been assigned.
3. the catalog SAS supplies, SASHELP.DEVICES. (SASHELP.DEVICES is usually write-protected and is opened in Browse mode.)

The first catalog SAS encounters is the current catalog.
To specify the current catalog by:

- use the CATALOG= option in the PROC GDEVICE statement (this is required to open a driver entry in Update mode)
- assign the GDEVICE0 libref to the appropriate catalog

**Search Order of Device Catalogs**

SAS/GRAPH searches only librefs starting with GDEVICE0 through GDEVICE9. The libraries must contain a catalog named DEVICES for SAS/GRAPH to search for the device entries for any driver. If you have personal device catalogs in more than one SAS library, you must assign librefs in the sequence GDEVICE0, GDEVICE1, GDEVICE2, and so on.

If the libref GDEVICE0 has been assigned to a SAS library, SAS/GRAPH looks in that library for a catalog named DEVICES. If the GDEVICE0.DEVICES catalog exists, it is checked for the specified device entry. If the device entry is not there, SAS/GRAPH looks next for a library with the libref GDEVICE1 and for a catalog named DEVICES in that library. The search is repeated for the sequence of librefs through GDEVICE9.

The search terminates if either of the following are true:

- Any of the GDEVICE libraries do not contain a DEVICES catalog.
- The librefs do not follow the numeric sequence of GDEVICE0, GDEVICE1, GDEVICE2, and so on, in that order.

If SAS/GRAPH terminates the search for either reason, or it does not find the specified device entry, SAS/GRAPH searches the SASHELP.DEVICES catalog. If the specified device entry is not found in the SASHELP.DEVICES catalog, an error message is written to the log.

**Note:** As stated above, the search for entries terminates if there is a break in the sequence. For example, the catalog GDEVICE1.DEVICES is not checked if the libref GDEVICE0 is undefined or if GDEVICE0 does not contain a catalog named DEVICES.

**Ways to Use the GDEVICE Procedure**

**Procedure Windows versus Program Code**

There are two ways to use the GDEVICE procedure:

- browse or edit the fields in the GDEVICE procedure windows (windowing mode). See “Running the GDEVICE Procedure in a Windowing Environment” on page 1048.
- submit GDEVICE procedure statements in a SAS program (program mode). See “Running the GDEVICE Procedure in Program Mode” on page 1048.

If you run SAS in a windowing environment, you can use either the GDEVICE procedure windows or the GDEVICE procedure statements. In a windowing environment, the GDEVICE procedure automatically opens the GDEVICE procedure windows.
If you run SAS in line mode or batch mode, you can use only GDEVICE procedure statements. In a non-windowing environment, the GDEVICE procedure automatically uses line mode.

Both methods provide identical functionality and enable you to display or modify device parameters or create new device entries.

Running the GDEVICE Procedure in a Windowing Environment

In a windowing environment, open the GDEVICE windows by submitting the PROC GDEVICE statement without the NOFS option:

```
proc gdevice;
```

This opens the DIRECTORY window in Browse mode. This window lists all of the device entries in the current catalog. (See “The Current Catalog” on page 1046.)

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 1055.

In a windowing environment, you can switch between the GDEVICE windows and program statements while you are running the procedure. See the “FS Statement” on page 1052 and the “NOFS” on page 1050 option in “PROC GDEVICE Statement” on page 1049.

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 1046.) 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.
Syntax: GDEVICE Procedure

**Requirement:**
Statements other than the PROC GDEVICE statement can be used only in a non-windowing or batch environment. In these environments, at least one statement is required to give GDEVICE an action to perform. In a windowing environment, the PROC GDEVICE statement is required. In program mode, at least one statement is required.

**Note:**
You must have Write access to the device catalog in order to modify, add, or delete entries.

*PROC GDEVICE Statement*

Starts the procedure and determines whether it is running in windowing mode or program mode. Can identify a device catalog and specify how that catalog is opened.

**Syntax**

```
PROC GDEVICE <CATALOG=<libref:>SAS-catalog> <BROWSE> <NOFS>;
    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);
    RENAME device-entry NEWNAME=entry-name;
    QUIT | END | STOP;
```

**Optional Arguments**

Options used in the PROC GDEVICE statement affect how you use the procedure. They specify how to open the catalog.

**BROWSE**

opens a catalog in Browse mode. You cannot modify a catalog when you open it with
the BROWSE option. If you are running in program mode when you use BROWSE,
you can use only the FS, LIST, QUIT, END, or STOP statements.


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 1047.

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.

Restriction: Not valid in Browse mode.
Requirement: You must have Write access to the device catalog in order to add entries, and use
CATALOG= in the PROC GDEVICE statement.

Syntax
ADD new-device-entry required-parameters <options>;

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
are all of the following parameters:

- MODULE = driver-module
- XMAX = width <IN | CM>
- YMAX = height <IN | CM>
- XPIXELS = width-in-pixels
- YPIXELS = height-in-pixels

Also, you must specify one or both of these parameter pairs:

- LCOLS = landscape-columns and LROWS = landscape-rows
- PCOLS = portrait-columns and PROWS = portrait-rows

All of the required parameters for the ADD statement correspond to device
parameters of the same name. Refer to Chapter 25, “Graphics Options and Device
Parameters Dictionary,” on page 515 for a description of each parameter.

Options
All optional parameters for the ADD statement correspond to device parameters of the
same name that are documented in Chapter 25, “Graphics Options and Device
Parameters Dictionary,” on page 515.

Note: Each entry in the dictionary states whether the entry (that is, the option or
parameter) can be used by the GOPTIONS statement, the GDEVICE procedure, or
any of the GDEVICE windows. Any entry that says it can be used by the GDEVICE
procedure or a GDEVICE window is a device parameter. You can specify any
parameter that can be specified with the GDEVICE procedure.
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 specify at least one color for the COLORS= option.

Details

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.

Restriction: Not valid in Browse mode.
Requirement: You must have Write access to the catalog where the device entry is being copied.
Tip: If you copy and modify a device entry that is a Universal Printer shortcut device, then you also need to create a Universal Printer that has the same name. See “Device Categories and Modifying Default Output Attributes” on page 87 and “Configuring Universal Printing Using the Windowing Environment” in SAS Language Reference: Concepts for more information.
See: “Creating or Modifying Device Entries” on page 1060
“Example: Creating a Custom Device Entry with Program Statements” on page 1062

Syntax

COPY device-entry where;

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.

where

must be one or both of the following:

FROM= <libref> .SAS catalog

names the catalog from which to copy device-entry.

NEWNAME=new-device-entry

specifies a name for the copy of the device entry that is placed in the current catalog. New-device-entry must be a valid name for a SAS catalog entry and cannot already exist in the current catalog.

If you copy device entries across catalogs and you do not specify a new name, the GDEVICE procedure uses the original name for the new device entry.
DELETE Statement

Deletes the device entry from the current catalog.

Restriction: Not valid in Browse mode.

Requirement: You must have Write access to the current catalog to delete a device entry. You must specify the CATALOG= option in the PROC GDEVICE statement to have Write access to the current catalog.

Note: A device entry cannot be restored once it has been deleted. Depending on the environment in which you are using the GDEVICE procedure, you might be asked to verify that you really want to delete the entry.

Syntax

DELETE device-entry;

Required Argument

device-entry
specifies the one-level name of device entry to delete.

Restriction The entry must exist in the current catalog.

FS Statement

Switches from program mode to the GDEVICE windows.

Requirement: You must be running SAS in a windowing environment.

Syntax

FS;

LIST Statement

Lists all of the parameters of the specified device entry in the Output window.

Default: _ALL_

See: “Running the GDEVICE Procedure in Program Mode” on page 1048

Syntax

LIST <device-entry> _ALL_ _NEXT_ _PREV_ <DUMP>;
Optional Arguments

device-entry
  specifies the one-level name of the device entry whose contents you want to list.

Restriction  The entry must exist in the current catalog.

_ALL_
  lists only the name, description, and creation date of all device entries in the current
  catalog. If no entries exist in the catalog, the GDEVICE procedure issues a message.

_NEXT_
  lists the contents of the next device entry. The GDEVICE procedure lists the first
  entry in the catalog if no entries have been previously listed.

_PREV_
  lists the contents of the previous device entry. If you have not previously listed the
  contents of a device entry, the GDEVICE procedure issues this message: No
  objects preceding current object.

DUMP
  lists detailed information about all device entries in the current catalog. Depending
  on the number of device entries in the catalog, the DUMP option can create a large
  amount of output.

MODIFY Statement

Changes the values in a device entry.

Restriction:  Not valid in Browse mode.

Requirement:  You must have Write access to the current catalog to modify a device entry. You
              must specify the CATALOG= option in the PROC GDEVICE statement to have Write
              access to the current catalog.

See:  "Creating or Modifying Device Entries" on page 1060
      "Example: Creating a Custom Device Entry with Program Statements" on page 1062

Syntax

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 that you want to modify. These can be any of the parameters that
  you can specify with the ADD statement, whether listed as required or optional for
  the ADD statement. See “ADD Statement” on page 1050 for more information.
  Refer to Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page
  515 for a description of each parameter.
Details

To modify a device entry, create your own catalog and then copy the device entries that you need into it. You can then change your personal copies of the device entries without affecting the original drivers in SASHELP.DEVICES. (To copy device entries, use either the COPY statement, the COPY command that is available after you select Import Device Entry from the DIRECTORY’ window's File menu, or the CATALOG procedure, which is part of Base SAS.)

**CAUTION:**

Be careful when modifying device entries in program mode. In program mode, you cannot cancel any modifications that you have just made. To change a value that you have modified, you must use another MODIFY statement to replace the original value or reset it to its default. (In the GDEVICE windows, you can enter the CANCEL command in the command line to cancel changes that you have made to the fields.)

QUIT Statement

Saves all modifications made to device entries during the procedure. Exits the GDEVICE procedure.

**Syntax**

QUIT | END | STOP;

RENAME Statement

Changes the name of the device entry to the name specified in the statement.

**Restriction:** Not valid in Browse mode.

**Requirement:** You must have Write access to the current catalog to rename a device entry. You must specify the CATALOG= option in the PROC GDEVICE statement to have Write access to the current catalog.

**Syntax**

RENAME device-entry 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 Windows

About the GDEVICE Windows

In the SAS windowing environment, you can use GDEVICE windows instead of program mode to view, modify, copy, create, or delete device entries.

Note: The GDEVICE windows are not available in SAS Studio.

You can perform tasks in the GDEVICE windows by entering values in the fields, by using the menus, and by issuing commands from the command line.

These are the thirteen GDEVICE windows in order of appearance:

• DIRECTORY Window
• Detail Window
• Parameters Window
• Gcolors Window
• Chartype Window
• Colormap Window
• Metagraphics Window
• Gprolog Window
• Gepilog Window
• Gstart Window
• Gend Window
• Host File Options Window
• Host Commands Window

The fields in these windows represent device entry parameters. The GDEVICE windows group the device parameters by topic. When you open a device entry in Edit mode, you can modify the fields directly. For a description of each field, see the corresponding parameter in Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515.

Note: Each entry in the dictionary states whether the entry (that is, the option or parameter) can be used by the GOPTIONS statement, the GDEVICE procedure, or any of the GDEVICE windows. Any entry that says it can be used by the GDEVICE procedure or a GDEVICE window is a device parameter.

Note: The parameters are sometimes an abbreviation of the field names. For example, in the Detail window, the “Driver query” field corresponds to the DRVQRY parameter, and the “Queued messages” field corresponds to the QMSG parameter.

GDEVICE Window Commands

You can navigate and manipulate the GDEVICE windows by entering commands on the command line, or selecting items from the menus. For a complete description of all the
GDEVICE window commands, open the Help for the GDEVICE windows. You can open the Help by entering **Help** on the command line or by selecting **Help ⇒ Using This Window**.

**Note:** In a Windows environment, the GDEVICE commands are presented on pop-up menus. Click the secondary mouse button on a GDEVICE window to access a pop-up menu.

**DIRECTORY Window**

This window is displayed when you start the GDEVICE procedure in window mode. It lists all the device entries in the default catalog or the catalog that you specified in the PROC GDEVICE statement. You can use it to do the following:

- copy, rename, or delete device entries in the catalog
- select a device entry whose parameters you want to browse or edit

You can enter these commands in the DIRECTORY window selection field:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>open the Detail window and browse (B) or, if you are in Edit mode, edit (S) the selected device entry.</td>
</tr>
<tr>
<td>D</td>
<td>delete the selected device entry. You cannot restore a device entry once it has been deleted.</td>
</tr>
<tr>
<td>E</td>
<td>open the Detail window and edit the selected device entry.</td>
</tr>
<tr>
<td>R</td>
<td>rename the device entry or the description, or both.</td>
</tr>
</tbody>
</table>

You cannot edit the TYPE and UPDATED fields in the DIRECTORY Window.

**Figure 37.1  The DIRECTORY Window**

After the DIRECTORY window appears, you can navigate through the GDEVICE windows by selecting the **View ⇒ Next Screen**.

**Detail Window**

This window contains device parameters that control basic characteristics of the device (for example, the size of the graphics output area).
**Parameters Window**

This window includes additional device parameters that affect how graphs are drawn. For example, you can define the following parameters:

- whether certain graphics primitives are drawn by your hardware or by SAS/GRAPH
- whether to feed paper to printers or plotters automatically
- whether to have SAS/GRAPH prompt you with messages under certain conditions

*Note:* If the device does not support a hardware characteristic, the catalog entry cannot enable the support.

**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.
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.

**Chartype 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.
**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.

---

**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 POSTPREGPROLOG= graphics options. See Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515 for descriptions of these options.

---

**Gepilog Window**

This window enables you to specify one or more hexadecimal strings that are sent to the device after graphics commands are sent. Additional commands can be sent with the PREGEPILLOG= and POSTGEPILLOG= graphics options. See Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515 for descriptions of these options.

---

**Gstart Window**

This window enables you to specify one or more hexadecimal strings that are placed at the beginning of each record of graphics data.

---

**Gend Window**

This window enables you to specify one or more hexadecimal strings that are placed at the end of each record of graphics data.
Host File Options Window

This window controls the output destination and formatting of the data stream produced by the driver. (Most of these values can also be specified with the GOPTIONS statement or with the FILENAME statement. See “Specifying the Graphics Output File Type for Your Graph” on page 109.)

Figure 37.8  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 37.9  The Host Commands Window

Creating or Modifying Device Entries

Write Access, Personal Catalogs, and Device Entry Names

In order to add, modify, or delete device entries, you must have Write access to the catalog. On multi-user systems, your site administrator is usually the only person who has Write access to the SASHELP.DEVICES catalog and can make any changes. Therefore, when creating new entries or modifying existing ones, individual users
usually work in a personal catalog. Be sure the catalog in which you store new or modified device entries is named DEVICES.

To use a device entry stored in a personal catalog, you must assign the GDEVICEn libref to the library that contains the device catalog. See “Device Catalogs” on page 1046.

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 GDEVICEn libraries before it searches SASHELP.DEVICES and uses the first device that it finds whose name matches the one you have specified. (See “Search Order of Device Catalogs” on page 1047.)

For example, suppose there is a customized copy of PSCOLOR in your GDEVICE0.DEVICES catalog as well as the original in SASHELP.DEVICES. If you specify DEV=PSCOLOR and the libref GDEVICE0 is assigned, SAS/GRAPH searches GDEVICE0.DEVICES first and uses the copy of PSCOLOR. Unless you cancel the GDEVICE0 libref, SAS/GRAPH will never find the original device entry in SASHELP.DEVICES. (To include SASHELP.DEVICES in the search path, you would need to cancel the GDEVICE0 libref.)

Creating a New Device Entry

Typically, you create a new device entry by copying an existing device and modifying its parameters to meet your needs. You can copy and modify a device entry in two ways:

• Use the DIR command to open the DIRECTORY window, and then use the COPY command to make a copy of an existing device entry. Edit the new entry and modify its parameters. The existing device entry can be from any catalog. (See help for information about using GDEVICE windows and commands. You can open the Help by entering Help on the command line or by selecting Help Using This Window.)

• In program mode, use the COPY statement to make a copy of the device entry. Use the MODIFY statement to change the parameters. See “Example: Creating a Custom Device Entry with Program Statements” on page 1062.

Note: If you copy and modify a device entry that is a Universal Printer shortcut device, then you also need to create a Universal Printer that has the same name. See “Device Categories and Modifying Default Output Attributes” on page 87 and “Configuring Universal Printing Using the Windowing Environment” in SAS Language Reference: Concepts for more information.

If you want to start with a blank device entry and fill in values for the parameters, use the EDIT command from the DIRECTORY window or use the ADD statement with program mode PROC GDEVICE.

With either method, you provide values for the parameters listed in “Required Arguments” on page 1050. If you copy and modify an existing entry, all the required parameters have values. If you create a new entry with GDEVICE windows, you are prompted to fill in the appropriate fields.

Note: When you change a field in a device entry that was provided by SAS (either the original device entry in SASHELP.DEVICES or a copy), SAS/GRAPH asks whether you really want to change the entry.
Modifying an Existing Device Entry

Typically, you modify an existing device entry when you want to change the device parameters permanently in order to customize a device entry. The process is similar to creating a new catalog entry. Copy the device entry that you want to modify into your personal catalog. Change the parameters in the new device entry. See “Example: Creating a Custom Device Entry with Program Statements” on page 1062 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 280 and “Precedence of Appearance Option Specifications” on page 281. See also “Style Attributes versus Device Entry Parameters” on page 274.

Example: Creating a Custom Device Entry with Program Statements

Features:
- COPY statement options
  - FROM=
  - NEWNAME=
- MODIFY statement options
  - COLORS=
  - DESCRIPTION=

Other features:
- LIBNAME statement

Sample library member:
- GDVCSTOM

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

This example uses the GDEVICE procedure statements COPY and MODIFY statements to create a customized version of the JPEG device. It copies the original entry into a personal devices catalog and changes the device parameters. After running this example, all output generated with the JPEG device will use the customized color list created by this example.
Output 37.1  Pie Chart Created with Default JPEG Device Entry

Output 37.2  Pie Chart Created with Customized JPEG Device Entry

Program

goptions reset=all border;
options nogstyle;
goptions device=jpeg;
proc gchart data=sashelp.class;
   pie age/discrete noheading value=none;
run;
quit;
libname gdevice0 ".";
proc gdevice nofs catalog=gdevice0.devices;
copy jpeg from=sashelp.devices newname=myjpeg;
modify myjpeg
description="JPEG with new color list"
colors=(black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F cx929cff);
quit;
goptions target=myjpeg;
proc gchart data=sashelp.class;
pie age/discrete noheading value=none;
rn;
quit;
options gstyle;

Program Description

Set the graphics environment.

goptions reset=all border;

Turn styles off so that the device color list is used.

options nogstyle;

Generate the pie chart using the JPEG device.

goptions device=jpeg;
proc gchart data=sashelp.class;
pie age/discrete noheading value=none;
rn;
quit;

Assign the libref GDEVICE0. The LIBNAME statement assigns the gdevice0 libref to the aggregate file storage location that will contain the DEVICES catalog and the modified JPEG device entry.

libname gdevice0 ".";

Start the GDEVICE procedure. The NOFS option causes GDEVICE to use program mode. The CATALOG= option assigns GDEVICE0.DEVICES as the current device catalog. If the DEVICES catalog does not already exist in the library, it is automatically created.

proc gdevice nofs catalog=gdevice0.devices;

Copy the original device entry from SASHELP.DEVICES to the current catalog. The NEWNAME= option specifies a name for the copy of JPEG that is placed in GDEVICE0.DEVICES. The name of a catalog entry cannot exceed eight characters.

copy jpeg from=sashelp.devices newname=myjpeg;

Modify the new entry. The DESCRIPTION= option specifies a new device description that appears in the catalog listing. The COLORS= option defines a new color list.

modify myjpeg
description="JPEG with new color list"
colors={black cx95c051 cxA359B2 cxD65259 cx69D6D2 cxFFB74F cx929cff};

Exit the procedure.
quit;

Test the new device entry. The TARGET= option specifies the new device. The GDEVICE0 libref is already defined, so SAS/GRAPH searches GDEVICE0 for the specified device entry. The GHART procedure generates a pie chart with the new color list.

goptions target=myjpeg;
proc gchart data=sashelp.class;
  pie age/discrete noheading value=none;
run;
quit;

Turn styles back on.

options gstyle;
Chapter 38
GFONT Procedure

Overview: GFONT Procedure

About the GFONT Procedure

The GFONT procedure displays fonts and creates SAS/GRAPH fonts for use in SAS/GRAPH programs. These fonts can contain standard Roman alphabet characters, foreign language characters, symbols, logos, or figures.
**Displaying Fonts**

You can use the GFONT procedure output when you want to do the following tasks:

- review the characters that are available in SAS/GRAPH fonts
- examine the default device-resident font for your device
- see the character codes associated with font characters
- view the hexadecimal values associated with font characters
- modify the color and height of font characters
- draw reference lines around font characters

See “Example 1: Displaying Fonts with Character Codes” on page 1093.

**Creating Fonts**

The GFONT procedure enables you to create and store any series of figures or alphabet fonts that you can digitize or draw using X and Y coordinates. Font characters or figures can be displayed with any SAS/GRAPH statement or option that allows for a font specification and a text string. See “Storing or Displaying User-Created Fonts: GFONT0 Libref” on page 1069 and “Font Data Sets and the GFONT Procedure” on page 1081 for more information.

**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 38.1  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 38.2  Characters from a Stroked Font*

A B C $ ? @

• 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 38.3  Filled and Outline Characters from Polygon Fonts*

A B C $ ? @

A B C $ ? @

In the GFONT procedure, the term line segment means a continuous line that can change direction. All font characters are drawn with line segments. The letter C is drawn with one line segment, whereas the letter A can be drawn with two.

Polygon characters can be drawn with one or more line segments. In a polygon font the following is true:
• A character can 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 or Displaying User-Created Fonts: GFONT0 Libref**

The GFONT procedure stores user-created SAS/GRAPH fonts in the location that is associated with the libref GFONT0. Before you create or display a user-created SAS/GRAPH font, submit a LIBNAME statement to associate the libref GFONT0 with a location where the font is stored, as follows:

```
LIBNAME gfont0 "SAS-data-library";
```

The GFONT0 library is the first place that SAS/GRAPH software searches for fonts. Always assign GFONT0 to the library that contains your personal SAS/GRAPH fonts. If
you have personal SAS/GRAPH fonts in more than one SAS library, assign them librefs in the sequence GFONT0, GFONT1, GFONT2, and so on. The search for entries terminates if there is a break in the numbering sequence. If the libref GFONT0 is not defined, by default SAS/GRAPH software begins searching for fonts in SASHELP.FONTS.

If a user-created SAS/GRAPH font has the same name as a font supplied by SAS, and if the libref GFONT0 has been defined, then the user-created SAS/GRAPH font is used, because GFONT0 is first in the search order.

To cancel or redefine the libref GFONTn, submit the following statement:

LIBNAME GFONTn;

Specifying Units for Creating Fonts

When the syntax of an option includes units, specify one of the units:

- CELLS
  - character cells
- CM
  - centimeters
- IN
  - inches
- PCT
  - percentage of the graphics output area
- PT
  - points

If you omit units, a unit specification is searched for in the following order:

1. the value of GUNIT= in a GOPTIONS statement
2. the value of units in the ODS STYLE
3. the default unit CELLS

Syntax: GFONT Procedure

- **Restriction:** This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

- **Requirement:** One font name is required. To display a font, include the NOBUILD option. To create a font, include the DATA= option.

- **Global statements:** FOOTNOTE, NOTE, TITLE

- **Note:** Before you run the GFONT procedure, use the DEVICE= option in a GOPTIONS statement or in an OPTIONS statement to specify an output device.
PROC GFONT Statement

The PROC GFONT statement can either create SAS/GRAPH fonts or display existing SAS/GRAPH fonts. The GFONT procedure names the font to be created or displayed. If the GFONT procedure creates a font, then an input data set name is required. You can modify the design and appearance of the fonts that you create or display, and specify a destination catalog for graphics output.

**Requirement:** When you create or display a user-created font, you must define the libref GFONT0. See “Storing or Displaying User-Created Fonts: GFONT0 Libref” on page 1069.

**Syntax**

The syntax for displaying a font is as follows:

```sql
PROC GFONT NAME=SAS/GRAPH-font | device-resident-font | system-font
NOBUILD <options-for-displaying-fonts>
```

The syntax for creating a font is as follows:

```sql
PROC GFONT NAME=SAS/GRAPH-font
DATA=font-data-set <options-for-creating-fonts>
```

**Summary of Optional Arguments**

**Creating fonts**

- **BASELINE=** specifies the vertical coordinate in the font data set that is the baseline of the characters.
- **CAPLINE=** specifies the vertical coordinate in the font data set that is the capline of the characters.
- **CHARSPACETYPE=** specifies the type of intercharacter spacing.
  - **DATA | FIXED | NONE | UNIFORM**
- **CODELEN=** specifies the length in bytes of the CHAR variable.
  - **1 | 2**
- **FILLED** specifies that the characters in a polygon font are filled.
- **KERNDATA=** specifies the data set that contains kerning information.
- **MWIDTH=** specifies the width of a character in a uniform font, where `character-width` is the number of font units.
- **NODISPLAY** specifies that the GFONT procedure is not to display the font that it is creating.
NOKEYMAP
specifies that the current key map is ignored when you create and then use
the font that is created.

RESOL=1...4
controls the resolution of the fonts by specifying the number of bytes (1
through 4) for storing coordinates in the font.

ROMHEX
specifies that hexadecimal values are displayed automatically below the font
characters when the GFONT procedure displays the font.

SHOWROMAN
specifies that character codes are displayed automatically below the font
characters when the GFONT procedure displays the font.

SPACEDATA=space-data-set
specifies the SAS data set that contains font spacing information.

UNIFORM
specifies that characters are spaced uniformly rather than proportionately.

Displaying fonts

CTEXT=text-color
specifies a color for the body of the characters.

GOUT=<libref:output-catalog
specifies the SAS catalog in which to save the graphics output generated by
the display of the font.

HEIGHT=character-height<units>
specifies the height of the font characters in number of units, n.

NOKEYMAP
specifies that the current key map is ignored when displaying the font and its
character codes or hexadecimal values.

NOROMAN
turns off the automatic display of character codes that are created when you
use the SHOWROMAN option during font creation.

NOROMHEX
turns off the automatic display of hexadecimal values for single-byte
characters.

REFCOL=reference-line-color
specifies a color for reference lines.

REFLINES
draws reference lines around each displayed character.

ROMCOL=code-color
specifies the color of the character codes or hexadecimal values that are
displayed with the SHOWROMAN and ROMHEX options.

ROMFONT=font
specifies the font for character codes and hexadecimal values that are
displayed by the SHOWROMAN and ROMHEX options.

ROMHEX
displays hexadecimal values below the font characters.

ROMHT=height<units>
specifies the height of the character codes and the hexadecimal values that
are displayed with the SHOWROMAN and ROMHEX options in number of
units, n.

SHOWALL
displays the font with a space for every possible character position whether a font character exists for that position.

SHOWROMAN
displays character codes below the font characters even if they are not displayed automatically with the font.

**Required Arguments**

**DATA= font-data-set**
specifies the SAS data set that the GFONT procedure uses to build the font. The data set must be sorted by the variables CHAR and SEGMENT.

**Default**
The GFONT procedure uses the most recently created data set.

**See**
“The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*

“About Data Set Options” in *SAS Data Set Options: Reference*

**Example**
“Example 2: Creating Figures for a Symbol Font” on page 1094

**NAME= SAS/GRAPH-font | device–resident font | system–font**
specifies font to be displayed. The name can be the name of a font. Name can be any of the following types of fonts:

- SAS/GRAPH font that is stored in the SASHELP.FONTS catalog or a SAS/GRAPH 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 SAS/GRAPH output files.

- 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.

- device-resident font that is burned into the chips in a device's hardware. These fonts are specific to the device being used and are not portable between devices. Some device resident fonts such as Helvetica can also be present as system fonts.

**Alias**
N=

**Note**
Device resident font names must be enclosed in quotation marks.

**NAME= SAS/GRAPH-font**
assigns a name to the font that you create. *SAS/GRAPH-font* is the name of a catalog entry, and must be a valid SAS name of no more than eight characters. You cannot specify NONE, or the name of a SAS/GRAPH font that is shipped with SAS/GRAPH software.

**Alias**
N=

**Example**
“Example 2: Creating Figures for a Symbol Font” on page 1094

**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
Options for Displaying Fonts

Options for displaying a font can be used when you create a font if you also display it (that is, you do not use the NODISPLAY option in the PROC GFONT statement). However, none of the display options affect the design and appearance of the stored font except the NOKEYMAP, SHOWROMAN, and ROMHEX options. See also “Specifying Units for Creating Fonts” on page 1070.

CTEXT=text-color
specifies a color for the body of the characters. If you do not use the CTEXT= option, a color specification is searched for in the following order:

- the CTEXT= option in the procedure statement
- the CTEXT= option in a GOPTIONS statement
- the color specified in the ODS style
- the first color in the color list

Alias CT=

Note The CTEXT= value is not stored as part of the font.

Example “Example 2: Creating Figures for a Symbol Font” on page 1094

GOUT=<libref:output-catalog>
specifies the SAS catalog in which to save the graphics output generated by the display of the font. You can use the GREPLAY procedure to view the output that is stored in the catalog.

If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library, and creates the catalog if it does not exist.

See “Specifying the Catalog Name and Entry Name for Your GRSEGs ” on page 120

HEIGHT=character-height<units>
specifies the height of the font characters in number of units, n. Height is measured from the minimum font measurement to the capline.

Alias H=

Default 2

Example “Example 1: Displaying Fonts with Character Codes” on page 1093

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 key map.
Note Only the characters that are mapped into your current key map are available.

NOROMAN
  turns off the automatic display of character codes that are created when you use the
  SHOWROMAN option during font creation.

  Alias  NR

NOROMHEX
  turns off the automatic display of hexadecimal values for single-byte characters that
  are created when you use the ROMHEX option during font creation.

  Alias  NOHEX

REFCOL=reference-line-color
  specifies a color for reference lines. If you do not use the REFCOL= option, a color
  specification is searched for in the following order:
  •  the CTEXT= option in a GOPTIONS statement
  •  the color specified in the ODS style
  •  the first color in the color list

REFLINES
  draws reference lines around each displayed character. Vertical reference lines show
  the width of the character. Horizontal reference lines show the font maximum and
  the font minimum, as well as the baseline and the capline.

  See  “Font Terminology and Characteristics” on page 1068

ROMCOL=code-color
  specifies the color of the character codes or hexadecimal values that are displayed
  with the SHOWROMAN and ROMHEX options. If you do not use the ROMCOL= option, a color
  specification is searched for in the following order:
  •  the color specified by the CTEXT= option in a GOPTIONS statement
  •  the color specified in the current style or, if the NOGSTYLE option is specified,
    then the default color is black for the Java and ActiveX devices and the first color
    in the color list for all the other devices

  Alias  RC=

  Note  The ROMCOL= value is not stored as part of the font.

  Example  “Example 1: Displaying Fonts with Character Codes” on page 1093

ROMFONT=font
  specifies the font for character codes and hexadecimal values that are displayed by
  the SHOWROMAN and ROMHEX options. If you do not use the ROMFONT= option, a font specification is searched for in the following order:
  •  the value of the ODS STYLE variable
  •  the FTEXT= option in a GOPTIONS statement
  •  fonts supplied by SAS
  •  the device-resident font
Alias RF=

Example “Example 1: Displaying Fonts with Character Codes” on page 1093

**ROMHEX**

Displays hexadecimal values below the font characters. If you use both the ROMHEX and SHOWROMAN options, both the character codes and the hexadecimal values are displayed. You can also use the ROMHEX option when you create a font.

Alias HEX

See ROMHEX “ROMHEX” on page 1080 (for creating fonts)

**ROMHT=height<units>**

Specifies the height of the character codes and the hexadecimal values that are displayed with the SHOWROMAN and ROMHEX options in number of units, n. If you do not use the ROMHT= option, a height specification is searched for in the following order:

- the HEIGHT specified in the ODS STYLE
- the HTEXT= option in a GOPTIONS statement

Alias RH=

Default 1

Example “Example 1: Displaying Fonts with Character Codes” on page 1093

**SHOWALL**

Displays the font with a space for every possible character position whether a font character exists for that position. The characters that are displayed are those available under your current key map, unless you use the NOKEYMAP option. The SHOWALL option usually is used in conjunction with the ROMHEX option, to display all possible hexadecimal values. If, under your current key map, a font character is available for a position, it is displayed above the hexadecimal value. If no character is available for a position, the space above the hexadecimal value is blank. You can use the SHOWALL option to show where undefined character positions fall in the font.

**SHOWROMAN**

Displays character codes below the font characters even if they are not displayed automatically with the font. If you use both the SHOWROMAN option, and the ROMHEX option, both the character codes, and the hexadecimal values are displayed. You can also use the SHOWROMAN option when you create a font.

Alias SR

Example “Example 1: Displaying Fonts with Character Codes” on page 1093

**Options for Creating Fonts**

Options to display a font can be used when you create a font if you also display it (that is, you do not use the NODISPLAY option in the PROC GFONT statement). However, none of the display options affect the design and appearance of the stored font except the
NOKEYMAP, SHOWROMAN, and ROMHEX options. See also “Specifying Units for Creating Fonts” on page 1070.

BASELINE=y specifies the vertical coordinate in the font data set that is the baseline of the characters. The baseline is the line upon which the letters rest. If you do not use the BASELINE= option, the GFONT procedure uses the lowest vertical coordinate of the first character in the font data set.

Alias B=

CAPLINE=y specifies the vertical coordinate in the font data set that is the capline of the characters. The capline is the highest point of normal that case the capline, and the font maximum are the same. See Figure 38.1 on page 1068 for an illustration of capline, and font maximum.

If you use the CAPLINE= option, when the height of a character is calculated, any part of the character that is above the capline is ignored in the calculation.

You can use this option to prevent an accented capital like A from being shortened to accommodate the accent. If you do not use the CAPLINE= option, the capline and the font maximum are the same. The A is shortened to make room for the accent below the capline. However, if the CAPLINE= option is used, the top of the letter A is at the capline, and the accent is drawn above the capline, and below the font maximum.

Alias C=

CHARSPACETYPE=DATA | FIXED | NONE | UNIFORM specifies the type of intercharacter spacing. The following values are valid:

DATA specifies that the first observation for each character sets the width of that character. When CHARSPACETYPE=DATA, the PTYPE variable is required, and the observation that specifies the width of the character must have a PTYPE value of W. See “Overview: The Font Data Set” on page 1081 for details about the PTYPE variable.

Intercharacter spacing is included in the character's width. If the first observation for the letter A specifies a character width of 10 units, and the A occupies 8 units, the remaining 2 units serve as intercharacter spacing.

Note The character can extend beyond the width that you specified in the first observation if desired.

FIXED adds a fixed amount of space between characters based on the font size. The width of the individual character is determined by the data that generates the character.

NONE specifies that no space is added between characters. The width of the individual character is determined by the data that generates the character. This type of spacing is useful for script fonts in which the characters should appear connected.

UNIFORM specifies that the amount of space that is used for each character is uniform, not proportional. Each character occupies the same amount of space. In uniform spacing the letters m and i occupy the same amount of space, in proportional
spacing m occupies more space than i. In uniform spacing, the character is always centered in the space, and a fixed space is added between characters.

When UNIFORM is specified, the amount of space that is used for each character is one of the following:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Default</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSP=</td>
<td>CHARSPACETYPE=FIXED</td>
<td>Specifying CHARSPACETYPE=UNIFORM is the same as using the UNIFORM option.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Alias</th>
<th>Default</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1</td>
<td>If you specify a double-byte character set, the KERNDATA= option and SPACEDATA= option are ignored.</td>
</tr>
</tbody>
</table>

Example "Example 2: Creating Figures for a Symbol Font” on page 1094

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.

<table>
<thead>
<tr>
<th>Alias</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>KERN=</td>
<td>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.</td>
</tr>
</tbody>
</table>

See “Overview: The Kern Data Set” on page 1089

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 38.4 on page 1079 shows the effect of decreasing the space that is allowed for equally spaced characters.
NODISPLAY

specifies that the GFONT procedure is not to display the font that it is creating.

Alias  ND

NOKEYMAP

specifies that the current key map is ignored when you create and then use the font that is created. The character codes that you enter are not mapped in any way before being displayed. As a result, the created font is never affected by any setting of the KEYMAP= graphics option.

By default, the NOKEYMAP option is not used. In that case, when you build a font, the current key map is applied to the values in the CHAR variable.

However, your current key map might not be symmetrical; two or more input character codes might be mapped to the same output character. For example, if A is mapped to B, then both A and B map to B, but nothing maps to A. In this case, more than one code in your input data set can map to the same character in the resulting font. For example, if A and B are values of CHAR, both map to B. If this happens, a message that indicates the problem characters is displayed in the SAS log. To solve this problem, do one of the following tasks:

• change the character code of one of the characters
• eliminate one of the characters
• use the NOKEYMAP option

The NOKEYMAP option works correctly only if the end user’s host or controller encoding is the same as the encoding used to create the input data set.

See  NOKEYMAP on page 1074 (for displaying fonts)

CAUTION

Fonts created with the NOKEYMAP option are never affected by any setting of the KEYMAP= graphics option.

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.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Number of Distinct Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>32,766</td>
</tr>
<tr>
<td>3</td>
<td>2,147,483,646</td>
</tr>
<tr>
<td>4</td>
<td>2,147,483,646</td>
</tr>
</tbody>
</table>

**ROMHEX**

specifies that hexadecimal values are displayed automatically below the font characters when the GFONT procedure displays the font. If you use the ROMHEX option for a font that you create, you can later use the NOROMHEX option to suppress display of the hexadecimal values.

**SHOWROMAN**

specifies that character codes are displayed automatically below the font characters when the GFONT procedure displays the font. If you use the SHOWROMAN option for a font that you create, you can later use the NOROMAN option to suppress display of the character codes.

**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.

**UNIFORM**

specifies that characters are spaced uniformly rather than proportionately. Using the UNIFORM option is the same as specifying CHARSPACETYPE=UNIFORM.
Font Data Sets and the GFONT Procedure

To create a font, you must create a data set that contains font information. Typically, you use a DATA step to create a SAS data set from which the GFONT procedure generates the font. The data set is referred to as the font data set, and you can specify it with the DATA= argument.

To produce the font, invoke the GFONT procedure and specify the data set that contains the font information. In addition, you can include options to modify the design and appearance of the font. For example, the following statement uses the data set FONTDATA to generate the font MYLOG:

```sas
proc gfont data=fontdata name=mylogo;
```

For a demonstration of the font creation process, see “Example 2: Creating Figures for a Symbol Font” on page 1094.

The GFONT procedure uses three types of data sets: the font data set, the kern data set, and the space data set. Each type of data set must contain certain variables and meet certain requirements. The following sections explain what each data set contains, how it is built, and what the requirements of the variables are.

The Font Data Set

Overview: The Font Data Set

The font data set consists of a series of observations that describe the font. The variables in the font data set must be assigned certain names and types. Table 38.1 on page 1082 describes the variables that are included in the font data set. These variables are described in “Font Data Set Variables” on page 1082 in more detail.

The font data set must include the horizontal and vertical coordinate values. It also includes line segment numbers that the GFONT procedure uses to generate each character. In addition, each observation must include a character code that is associated with the font character and is used to specify the font character in a text string. The font data set also determines whether the font is stroked or polygon. A font data set that generates a polygon font produces an outline font by default. You can use the FILLED option with the same data set to generate a filled font.

Specify the font data set with the DATA= argument, for example:

```sas
data mylib.fontdata;
  proc gfont data=fontdata name=mylogo;
run;
```
Table 38.1 Table of Font Data Set Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
<th>Valid Values</th>
<th>With Stroked Fonts</th>
<th>With Polygon Fonts</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>The character code associated with the font character</td>
<td>Character</td>
<td>1 or 2</td>
<td>Keyboard characters or hexadecimal values</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>LP</td>
<td>The type of line segment being drawn, either a line or a polygon</td>
<td>Character</td>
<td>1</td>
<td>L or P</td>
<td>Optional</td>
<td>Required</td>
</tr>
<tr>
<td>PTYPE</td>
<td>The type of data in the observation</td>
<td>Character</td>
<td>1</td>
<td>V or C or W</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>SEGMENT</td>
<td>The number of the line segment or polygon being drawn</td>
<td>Numeric</td>
<td>Number</td>
<td>Number</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>X</td>
<td>The horizontal coordinate</td>
<td>Numeric</td>
<td>Number</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Y</td>
<td>The vertical coordinate</td>
<td>Numeric</td>
<td>Number</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
</tbody>
</table>

Font Data Set Variables

**CHAR**

provides a code for the character or figure that you are creating. CHAR is a character variable with a length of 1 or 2. CHAR is required for all fonts.

**CAUTION:**

Using reserved or undefined hexadecimal codes as CHAR values might require the use of the NOKEYMAP option.

The CHAR variable takes any character as its value, including keyboard characters and hexadecimal values from '00'x to 'FF'x. (If you use hexadecimal values as CHAR values, your font might not work correctly under a key map that is different from the one under which the font was created. Positions that are not defined in one key map might be defined in another.)

When you specify the code for the character in a text string, the associated font character is drawn. For example, if you create a Roman alphabet font, typically the characters that you specify for CHAR are keyboard characters that match the character in the font. All of the observations that build the letter A have a CHAR value of A. When you specify 'A' in a text string, this creates an A in the output.

However, if you build a symbol font, the symbols might not have corresponding keyboard characters. In that case, you select a character or hexadecimal value to
represent each symbol in the font and assign it to CHAR. For example, in the Special font, the letter G is assigned as the code for the fleur-de-lis symbol. When you specify the code in a text string, the associated symbol is displayed.

Note: If the CODELEN= option is set to 2, the values for CHAR represent two characters, such as AA, or a four-digit hexadecimal value, such as '00A5'x.

**LP**

tells the GFONT procedure whether the coordinates of each segment form a line or a polygon. LP is a character variable with a length of 1. The LP variable is required for polygon fonts but is optional for stroked fonts. You can assign the LP variable either of the following values:

L lines

P polygons

Every group of line segments with an LP value of P is designated as a polygon. If the observations do not draw a completely closed figure, the program closes the figure automatically. For example, the following observations do not contain an LP variable. Observations that do not contain an LP variable create a shape like the one in Figure 38.5 on page 1083.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CHAR</th>
<th>SEG</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 38.5** LP Value of L
An LP variable with a value of P for all observations added to the data set produces a complete box like the one in Figure 38.6 on page 1084.

**Table 38.3  Observations to Create the Following Figure**

<table>
<thead>
<tr>
<th>OBS</th>
<th>CHAR</th>
<th>SEG</th>
<th>X</th>
<th>Y</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>P</td>
</tr>
</tbody>
</table>

**Figure 38.6  LP Value of P**

The LP variable enables you to mix lines and polygons. These observations create the figure consisting of a polygon and a line segment as shown in Figure 38.7 on page 1085.

**Table 38.4  Observations to Create the Following Figure**

<table>
<thead>
<tr>
<th>OBS</th>
<th>CHAR</th>
<th>SEG</th>
<th>X</th>
<th>Y</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>L</td>
</tr>
</tbody>
</table>
**PTYPE**

tells the GFONT procedure what type of data is in the observation. PTYPE is a character variable of length 1 that is optional. For each observation, the PTYPE variable assigns a characteristic to the point that is determined by the X and Y values. You can assign the PTYPE variable to any of these values:

- **V**
  - normal point in the line segment
  - **Note**: If a PTYPE variable is not specified, then all points are assumed to be V-type points.

- **C**
  - center of a circular arc joining two V points
  - **Restriction**: Arcs are limited to 106 degrees or less.
  - **Example**: Figure 38.8 on page 1086

- **W**
  - width value for CHARSPACETYPE=DATA. An observation with a PTYPE value of W must always be the first observation for a character. The observation gives the minimum and maximum X values for the character. The Y variable observation contains the maximum X value. Usually, these values include a little extra space for intercharacter spacing.
  - **Restriction**: Use a PTYPE of W only if you have specified CHARSPACETYPE=DATA. Otherwise, the points are ignored.
  - **See**: CHARSPACETYPE= on page 1077
The following observations illustrate how the PTYPE variable is used to draw an arc similar to Figure 38.8 on page 1086.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CHAR</th>
<th>SEG</th>
<th>X</th>
<th>Y</th>
<th>LP</th>
<th>PTYPE</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>40</td>
<td>60</td>
<td>P</td>
<td>W</td>
<td>Define width of character as 20 font units, which is the number of units from left margin, 40, to right margin, 60</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>1</td>
<td>45</td>
<td>40</td>
<td>P</td>
<td>V</td>
<td>Start line segment at position 45,40</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>1</td>
<td>45</td>
<td>50</td>
<td>P</td>
<td>V</td>
<td>Draw a line to position 45,50, which is start point of arc</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>1</td>
<td>45</td>
<td>40</td>
<td>P</td>
<td>C</td>
<td>Draw an arc whose center is at 45,40</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>1</td>
<td>55</td>
<td>40</td>
<td>P</td>
<td>V</td>
<td>Finish drawing the arc at 55,40</td>
</tr>
</tbody>
</table>

After the figure was generated, a grid was overlaid on it to show the location of the points.

Figure 38.8 Using the PTYPE Variable to Create an Arc

Note the following:

- Three observations are required to draw an arc: observation 3 and observation 5 denote the start point and the end point of the arc. Observation 4 locates the center of the arc.
- The figure is closed because the line segments have an LP value of P (polygon).
- The font that contains the figure of the arc was created with a similar PROC GFONT statement:
proc gfont data=arc name=arcfig charspacetype=data filled;

The CHARSPACETYPE= DATA option specifies that the first observation sets the width of the character. The FILLED option fills the area of the arc.

SEGMENT

numbers the line segments that compose a character or symbol. SEGMENT is a required numeric variable. All observations for a given line segment have the same segment number. To start a new line segment, change the segment number.

When the GFONT procedure draws a stroked character with more than one line segment (for example, the letter E), or a polygon character with a hole (for example, the letter A), it needs to know when one line stops and where the next line begins. To indicate when one line stops and where the next line begins you can do either of the following:

• Change the segment number when a new line begins. If the value of LP is L (line), a change in segment number tells the GFONT procedure not to connect the last point in line segment 1 and the first point in line segment 2.
  • The last point in line segment 1 ends the line.
  • The first point in line segment 2 starts a new line.

If the value of LP is P (polygon), a change in segment numbers causes the following:

• The last point in line segment 1 joins to the first point in line segment 1, which closes the polygon.
• A new polygon starts. If the value of CHAR has not changed, the new polygon is part of the same character.

Use this method for characters that consist of two polygons such as a question mark. This method is preferred, unless you are creating a polygon character with a hole in it.

• Keep the same segment number for all lines. Insert an observation with missing values for X and Y. Insert the new observation between the observation that marks the end of the first line, and the observation that begins the next line. For example, if you are drawing the letter O, insert an observation with a missing value between the line that draws the outer circle and the beginning of the line that draws the inner circle.

The first method is preferred, unless you are creating a polygon character with a hole in it. The second method is preferred when creating a polygon with a character with a hole in it. In this case, you should separate the lines with a missing value and keep the same segment numbers.

If you use separate line segments when you create a polygon with a hole, the results can be unpredictable. For example, the observations from a data set called BOXES were used to draw the hollow square in Figure 38.9 on page 1088.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CHAR</th>
<th>SEG</th>
<th>X</th>
<th>Y</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>P</td>
</tr>
</tbody>
</table>
Note the observation 5 has missing values for X and Y. This separates the observations that draw the inner box from those that draw the outer box. The segment number is the same for all the observations.

*Figure 38.9 on page 1088* was created with a similar GFONT statement:

```
proc gfont data=boxes name=boxes filled;
```

The FILLED option is included, and only the space between the two squares is filled.

The data points that form the figure are laid out on a grid shown next to the square.

*Figure 38.9 Drawing Nested Polygons*

\[
\begin{array}{c}
\begin{array}{c}
\text{X and } Y \\
\text{specify the horizontal and vertical coordinates of the points for each character. Their values describe the position of the points on the character. These variables have the following characteristics:}
\end{array}
\end{array}
\]

- They must be numeric.
- They must be named X and Y for the horizontal and vertical coordinates, respectively.
- The values specified by them can be in any range.
- They both must describe the character in the same scale or font units.
- Vertical (Y) coordinates for all characters should be defined on the same baseline.

**Interaction** When you specify PTYPE=W, both X and Y contain horizontal coordinate values.
Creating a Font Data Set

Create a font data set by digitizing the shape of the characters or figures either manually or with special digitizing equipment. To create a font data set by digitizing the characters manually:

1. Determine the coordinate points for each line segment by drawing the characters on a grid.
2. Lay out the observations for each character. Each observation describes a move from one point to another along a line segment. For each line segment, enter the coordinate points in the order in which they are drawn. For a stroked font, when you start a new line segment, change the segment number. For a polygon font, when you start a new polygon, change the line segment number.
   If the polygon has a hole in it, as in the letter O, keep the line segment number and separate the lines with a missing value. Use the same value for CHAR for all of the observations that describe one character.
3. Create a SAS data set that contains the variables CHAR, SEGMENT, X, and Y, and read in the data for each observation. Include the variables LP and PTYPE if necessary.
4. Sort the data set by CHAR and SEGMENT.
5. Assign the font data set with the DATA= argument.

This process is illustrated in “Example 2: Creating Figures for a Symbol Font” on page 1094.

The Kern Data Set

Overview: The Kern Data Set

The kern data set consists of observations that specify how much space to add or remove between any two characters when they appear in combination. This process, called kerning, increases or decreases space between the characters. Kerning usually is applied to certain pairs of characters that have too much space between them. Reducing the space between characters might result in part of one character extending over the body of the next. Examples of some combinations that should be kerned are AT, AV, AW, TA, VA, and WA.

You can apply kerning to the intercharacter spacing that you specify with the CHARSPACETYPE= option (except for uniform fonts). Assign the kern data set with the KERNDATA= option.

Kern Data Set Variables

The kern data set must contain these variables:

CHAR1
   specifies the first character in the pair to be kerned. CHAR1 is a character variable with a length of 1.
CHAR2
specifies the second character in the pair to be kerned. CHAR2 is a character variable with a length of 1.

XADJ
specifies the amount of space to add or remove between the two characters. XADJ is a numeric variable that uses the same font units as the font data set. The value of XADJ specifies the horizontal adjustment to be applied to CHAR2 whenever CHAR1 is followed immediately by CHAR2. Negative numbers decrease the spacing, and positive numbers increase the spacing.

Creating a Kern Data Set

Each observation in a kern data set names the pair of characters to be kerned and the amount of space to be added or deleted between them. To create a kern data set, follow these steps:

1. Select the pairs of characters to be kerned, and specify the space adjustment (in font units) for each pair as a positive number (more space) or negative number (less space).
2. Create a SAS data set that contains the variables CHAR1, CHAR2, and XADJ. Define one observation for each pair of characters and the corresponding space adjustment, for example:

```sas
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:

```sas
proc gfont data=fontdata
  name=font2
  charspacetype=data
  kerndata=kern1
  nodisplay;
run;
```

“Creating a Kern Data Set” on page 1090 illustrates how to use the KERNDATA= option to create a font in which the space between specified pairs of letters is reduced. The characters A, D, and T are shown as the word DATA. The first line uses the unkerned font, FONT1, and the second line uses the kerned font, FONT2. Note that the characters in FONT2 are spaced more closely than the characters in FONT1.

The following TITLE statements specify the kerned and unkerned fonts and are used with the GSLIDE procedure to produce Figure 38.10 on page 1091.

```sas
title2 lspace=6 f=font1 h=10 j=l "DATA";
title3 lspace=4 f=font2 h=10 j=l "DATA";
```
The Space Data Set

Overview: The Space Data Set

As the height (point size) of a font increases, less space is required between letters in relation to their height. If the height decreases, more space might be needed. The space data set tells the GFONT procedure how much to increase or decrease the intercharacter spacing for a given point size. Like kerning, spacing is added to or subtracted from the intercharacter spacing that is specified by the CHARSPACETYPE= option. Spacing is applied uniformly to all characters.

Values that are specified in the space data set are added to the normal intercharacter spacing and any kerning data. Normal intercharacter spacing is determined by the CHARSPACETYPE= option.

Space Data Set Variables

The space data set must contain these variables:

SIZE
  specifies the point size of the font. SIZE is a numeric variable.

ADJ
  specifies the spacing adjustment for the point size in hundredths (1/100) of a point. (A point is equal to 1/72 of an inch.) ADJ is a numeric variable. Positive values for ADJ increase the space between characters. Negative values for ADJ reduce the space between characters.

Creating a Space Data Set

Each observation in a space data set specifies a point size (SIZE) and the amount of space (ADJ) to be added or subtracted between characters when a font of that point size is requested. When you specify a point size that is not in the space data set, the adjustment for the next smaller size is used. To create a space data set, follow these steps:

1. Determine the amount of adjustment that is required for typical point sizes.
2. Create a data set that contains the variables SIZE and ADJ. Create one observation for each point size and corresponding space adjustment as follows:

```plaintext
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:

```plaintext
proc gfont data=fontdata
  name=font3
  charspacetype=data
  spacedata=space1
  nodisplay;
run;
```

**Figure 38.11** on page 1092 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 38.11** on page 1092:

```plaintext
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 38.11**  Comparison of Text with and without Spacing Adjustments
## Examples: GFONT Procedure

### Example 1: Displaying Fonts with Character Codes

**Features:**
- GFONT statement options
  - HEIGHT=
  - NOBUILD
  - ROMCOL=
  - ROMFONT=
  - ROMHT=
  - SHOWROMAN

**Sample library member:** GFODISFO

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

This illustrates the SHOWROMAN option, which displays the character codes that are associated with the font characters that are being displayed. This display shows which keyboard character you enter to produce the Greek character that you want displayed. The example also illustrates how to modify the appearance of both the font characters, and the character codes.

<table>
<thead>
<tr>
<th>The GREEK Font with Character Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 &quot; # $ % &amp; ' ( ) * + , - . / 0 1 2 3 4 5</td>
</tr>
<tr>
<td>6 7 8 9 : ; &lt; = &gt; ? @ A 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</td>
</tr>
<tr>
<td>a 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</td>
</tr>
<tr>
<td>{</td>
</tr>
</tbody>
</table>

**Program**

```sas
options reset=all border hsize=5.5in vsize=4.14in device=png;
title "The GREEK Font with Character Codes";
```
Proportional font sizes and heights can be specified with the HEIGHT= option. Height values can be specified in any unit of measure supported by the PROC GGRAPH procedure, with a default of font units. Height values can be set for all fonts, for a single font with the FONT= option, or for a single character with the FONTCHAR= option.

Example 2: Creating Figures for a Symbol Font

Features:
- GFONT statement options
  - CTEXT=
  - DATA=
  - FILLED
  - NAME=
  - RESOL=

Other features:
- LIBNAME statement

Sample library member:
- GFOCRFIG

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

Create three simple figures for a symbol font. Each figure is laid out on a grid that is 64 font units square. The third figure is a circle with a slash through it. Output 38.1 on page 1095 shows the figure and some of its coordinate points laid out on a grid.
Output 38.1 Diagram of Circle with Slash Figure

Program

LIBNAME gfont0 ".";

goptions reset=all border gunit=pct
device=png;

data figures;
  input char $ ptype $ x y segment lp $;
datalines;
  A W 0 64 0 P /* triangle pointing right */
  A V 4 4 1 P
  A V 60 32 1 P
  A V 4 60 1 P
  A V 4 4 1 P
  B W 0 64 0 P /* heart */
  B V 32 2 1 P
  B V 44 17 1 P
  B V 58 40 1 P
Program Description

Assign the librefs and set the graphics environment. The LIBNAME statement associates the libref GFONT0 with the SAS library in which the font catalog is stored.
Create the font data set FIGURES for a triangle, a heart, and a circle with slash. The first figure, a right-pointing triangle that is assigned the character code A, is a polygon drawn with three straight lines.

```
data figures;
  input char $ ptype $ x y segment lp $;
datalines;
  A W 0 64 0 P /* triangle pointing right */
  A V 4 4 1 P
  A V 60 32 1 P
  A V 4 60 1 P
  A V 4 4 1 P
```

The second figure, a heart that is assigned the character code B, uses the PTYPE variable combination V-C-V to draw the arcs that make up the top of the heart. Each side requires two arcs. Because the arcs are continuous, the observation that marks the end of one arc is also the beginning of the next arc. The heart drawing begins at the bottom point and continues counterclockwise.

```
B W 0 64 0 P /* heart */
B V 32 2 1 P
B V 44 17 1 P
B V 58 40 1 P
B C 46 47 1 P
B V 56 58 1 P
B C 46 47 1 P
B V 32 52 1 P
B C 18 47 1 P
B V 8 58 1 P
B C 18 47 1 P
B V 6 40 1 P
B V 20 17 1 P
B V 32 2 1 P
```

The third figure, a circle with a slash through it, assigned the character code C, consists of three polygons: a circle and two empty arcs. An observation with missing values separates the observations defining each of the three polygons. The outer circle is defined by the first group of observations. The empty arcs are drawn with three continuous arcs using the PTYPE variable pattern V-C-V-C-V-C-V. The straight line that closes the arc is drawn automatically by the GFONT procedure in order to complete the polygon. Because all the polygons are part of one character, the continuous space that they define is filled.

```
C W 0 64 0 P /* circle with slash */
C V 32 64 1 P
C C 32 32 1 P
C V 64 32 1 P
C C 32 32 1 P
C V 32 0 1 P
C C 32 32 1 P
C V 0 32 1 P
C C 32 32 1 P
C V 32 64 1 P
```
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;
Overview: GKPI Procedure

About KPI Charts and the GKPI Procedure

The GKPI procedure creates graphical key performance indicator (KPI) charts. KPIs are metrics that help a business monitor its performance and measure its progress toward specific goals. The procedure produces five KPI chart types:

<table>
<thead>
<tr>
<th>Chart Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slider KPI Charts</td>
<td>1100</td>
</tr>
<tr>
<td>Bullet Graph KPI Charts</td>
<td>1100</td>
</tr>
<tr>
<td>Dial KPI Charts</td>
<td>1101</td>
</tr>
<tr>
<td>Speedometer KPI Charts</td>
<td>1101</td>
</tr>
<tr>
<td>Traffic Light KPI Charts</td>
<td>1102</td>
</tr>
</tbody>
</table>

Specifying Qualitative Range Boundaries and Actual KPI Values

Controlling the Display of Boundary and Tick Mark Values

Controlling Qualitative Range Colors

Default Colors

Defining Active and Inactive Color Lists

Example: Specifying an Inactive Color List

Example: Specifying an Active Color List

Specifying Active Colors Only for Specific Ranges (Using Null Colors)

Specifying Color Names

Syntax: GKPI Procedure

PROC GKPI Statement

DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT,
SPEEDOMETER, VTRAFFICLIGHT, VBULLET, and VSLIDER Statement

Examples: GKPI Procedure

Example 1: Using the Default Colors as the Active Colors

Example 2: Creating a Gray Scale Bullet Graph

Example 3: Creating a Dial KPI Chart

Example 4: Defining a Speedometer

Example 5: Defining a Speedometer with Reversed Colors

Example 6: Creating a Traffic Light
- slider (vertical or horizontal)
- bullet graph (vertical or horizontal)
- dial
- speedometer
- traffic light (vertical or horizontal)

The GKPI procedure produces a KPI chart based on a series of qualitative range boundaries and an actual KPI value that you specify. If you specify a target value, the KPI chart also displays the target value. The procedure uses a set of default colors for the KPI chart, but you can specify your own colors.

**Note:** The only device supported for the GKPI procedure is JAVA IMG. If you do not specify DEVICE=JAVA IMG, then SAS/GRAPH sets the DEVICE option to JAVA IMG.

**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=’gkpimage.png’ option in the GOPTIONS statement with the GSLIDE or GANNO procedures to generate the GRSEG.

### Slider KPI Charts

Slider KPI charts display a bar divided into qualitative ranges according to the boundary values that you specify. The actual value of the KPI is indicated with a triangle pointer the left (for a vertical slider) or on the top (for a horizontal slider). This actual value indicator is the same color as the range that contains the actual KPI value. The target value, if it is specified, is displayed as a smaller triangle on the right side (or bottom) of the slider.

### Bullet Graph KPI Charts

Bullet graphs display a bar divided into qualitative ranges according to the boundary values that you specify. The actual value of the KPI is indicated with a black line, or bullet, down the center of the graph. The target value, if it is specified, is displayed as a vertical line (in a horizontal bullet graph) or a horizontal line (in a vertical bullet graph) across the graph.
**Dial KPI Charts**

Dial KPI charts display a dial divided into qualitative ranges according to the boundary values. The actual value of the KPI is indicated with a large, white triangle pointer. The target value, if it is specified, is displayed as a small, black triangle. The center of the dial is the same color as the range that contains the actual KPI value.

**Speedometer KPI Charts**

Speedometer KPI charts display a speedometer with the tick marks evenly spaced around the dial and colored ranges that correspond to the range boundaries that you specify. Speedometers can be displayed as a full speedometer, as a half speedometer, or as a quarter speedometer. The actual value of the KPI is indicated by a long pointer. The target value, if it is specified, is displayed as a small, black triangle.
In each display type, tick marks are evenly spaced but do not correspond to colored range boundaries. The numbered band in the full speedometer is always divided into ten sections (using 11 tick marks). The numbered band in the half speedometer is divided into five sections (six tick marks), and the quarter speedometer is divided into three sections (four tick marks).

**Traffic Light KPI Charts**

Traffic light KPI charts display a traffic light that contains one light for each qualitative range. The range that contains the actual value is displayed in color. The remaining ranges are gray. In other words, only one “light” is “turned on” at a time. Traffic lights do not display target values.

**Specifying Qualitative Range Boundaries and Actual KPI Values**

To generate a KPI chart, you must specify a list of range boundaries using the BOUNDS= option and an actual KPI value using the ACTUAL= option. The values can be positive numbers, negative numbers, or missing (ACTUAL=.), but the BOUNDS= list must be in either ascending or descending order and must contain at least two numbers (in order to define a single range). For example, the following code defines a horizontal slider with range boundaries in ascending order from –8 to 10 and an actual KPI value of 6:

```plaintext
goptions device=javaimg;
proc gkpi;
   hslider actual=6 bounds=(-8 -5 0 3 5 10);
run;
quit;
```

The boundaries can also be specified in descending order, for example:

```plaintext
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 1106 for information about controlling range colors.

The actual KPI value can fall outside of the highest or lowest boundaries, but the GKPI procedure treats such values as if they occur at the edge of the highest or lowest boundaries. For example, suppose the actual KPI value is –10, but the lowest boundary value is –8:

```
hslider actual=-10 bounds=(-8 -5 0 3 5 10)
```

PROC GKPI displays the actual KPI value indicator at –8.

If you specify a missing value for the actual KPI value (ACTUAL=.), then the GKPI procedure does not generate a KPI chart.

### Controlling the Display of Boundary and Tick Mark Values

In some cases, there might not be enough space to display all of the boundary values or, for speedometers, tick mark values without some of the values colliding together. In these cases, the GKPI procedure typically drops some or all of the boundary or tick mark values, depending on the amount of space available, the font size being used, and the values that need to be displayed.

If the GKPI procedure drops values, you can try the following solutions:

- increasing the size of the KPI chart using the XPIXELS= / YPIXELS= or HSIZE= / VSIZE= options in the GOPTIONS statement
- reducing the size of the boundary value font using the BFONT= option
- applying a SAS format to the boundary values using the FORMAT= option

See “Example 3: Creating a Dial KPI Chart” on page 1119 for an example that uses the XPIXELS=, YPIXELS=, BFONT=, and FORMAT= options.
Controlling Qualitative Range Colors

Default Colors

If you define only one range or more than five ranges, the GKPI procedure uses the same value of gray (hexadecimal RGB value B2B2B2) for all ranges.

*Figure 39.1  GKPI Procedure Gray Scale Default*

If you define two to five ranges, the GKPI procedure uses some or all of the colors shown in *Figure 39.2 on page 1104* as the default colors.

*Figure 39.2  GKPI Procedure Default Colors*

For example, if you define only three ranges, the GKPI procedures uses the red, yellow, and green colors shown in *Figure 39.2 on page 1104*.

*Table 39.1 on page 1104* lists the hexadecimal values for each of these default colors.

*Table 39.1  Hexadecimal Values for GKPI Procedure Default Colors*

<table>
<thead>
<tr>
<th>Color</th>
<th>Hexadecimal RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>cxD06959</td>
</tr>
<tr>
<td>Orange</td>
<td>cxE1A05D</td>
</tr>
<tr>
<td>Yellow</td>
<td>cxF1DC63</td>
</tr>
<tr>
<td>Yellow-Green</td>
<td>cxBDCD5F</td>
</tr>
</tbody>
</table>
The traffic light also uses the default colors, but it applies them slightly differently. If you define only one range, then the procedure displays the light for that range in gray. If you define two to five ranges, the traffic light uses the default color listed in Table 39.1 on page 1104 only for the light corresponding to the range that contains the actual KPI value. All other lights are gray. If you define more than five ranges, all lights are displayed in gray, but the light corresponding to the range that contains the actual KPI value is displayed in dark gray. In other words, only one light in a traffic light is “turned on” at a time. All other lights are “turned off.”

Figure 39.3  Traffic Light Default Colors

Colors are applied in the same direction, regardless of whether the range boundaries are in ascending or descending order. Colors for horizontal sliders and bullet graphs are applied from left to right. Colors for traffic lights, vertical sliders, and vertical bullet graphs are applied from bottom to top. Colors for dials and speedometers are applied clockwise.
Defining Active and Inactive Color Lists

You can define two different color lists: a list of active colors and a list of inactive colors. The active color list is defined with the ACTIVECOLORS= option, and the inactive color list is defined with the COLORS= option. Neither option is required.

Each color in a list corresponds to a range in the KPI chart. That is, the first color is applied to the first range, the second color is applied only to the second range, and so on. A range is displayed in its active color if the actual KPI value falls in that range. All ranges that do not contain the actual KPI value are displayed in their inactive colors. If you do not specify an active color list, then the range that contains the actual KPI value is also displayed in its inactive color.

Note: The TRAFFICLIGHT statement supports both the COLORS= option and the ACTIVECOLORS= option. However, if both options are specified, the COLORS= option is ignored. All ranges that do not contain the actual KPI value appear gray.

You can specify colors for the ranges using any of the color-naming schemes supported by SAS/GRAPH. See “Specifying Color Names” on page 1108.

If you specify the COLORS= option, then you must specify a color for each range. That is, the number of entries in the COLORS= list must be one less than the number of entries in the BOUNDS= list. If you specify the ACTIVECOLORS= list, you do not have to specify a color for every range. See “Specifying Active Colors Only for Specific Ranges (Using Null Colors)” on page 1107 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 39.2 on page 1104.

goptions reset=all device=javaimg;
proc gkpi mode=raised;
   hslider actual=0.28
     bounds=(0 .22 .35 .50) /
         colors=(PaleTurquoise MediumTurquoise Teal);
   run;
quit;

The actual KPI value falls into the second range, and there are no active colors specified, so the second color in the COLORS= list, MediumTurquoise, is used for the second range and for the actual KPI value indicator.

Example: Specifying an Active Color List

The following example defines the inactive colors for all of the ranges to be the same value of gray, cxB2B2B2. For the active colors, it specifies the default values for the red, yellow, and green colors listed in Table 39.1 on page 1104.
The actual KPI value is 0.28, which falls into the second range, so the second color listed in the ACTIVECOLORS= color list, cxF1DC63, which is yellow, is used as the color for the second range instead of gray.

If the actual KPI value is changed to 0.43, then the third color in the ACTIVECOLORS= color list, cx84AF5B, which is green, is used for the third range instead of gray.

**Specifying Active Colors Only for Specific Ranges (Using Null Colors)**

If you specify a null color for a range in the ACTIVECOLORS= list, then either the default color or the color in the COLORS= list, if one is specified, is used for that range even if it contains the actual KPI value.

To specify a null color, you can specify `null` for the color or enter a comma-delimited list with no space between the commas. For example, if you have five ranges and you want red to be used only for the lowest and highest ranges, then you can specify the ACTIVECOLORS= list in either of the following forms:

```
activecolors=(red,null,null,null,red)
activecolors=(red,,,,red)
```

The default colors (or the color in the COLORS= list) are used for ranges two through four even if the actual KPI value falls into one of these ranges.

The ACTIVECOLORS= list does not have to specify a color for each range, but the one-to-one correspondence between the colors that are specified and the ranges is maintained. For example, supposed you define five ranges and you specify the following:

```
activecolors=(red green)
```

The GKPI procedure treats this specification as if you had entered the following:

```
activecolors=(red,green,null,null,null)
```

Red applies only to the first range, and green applies only to the second range. Default colors (or COLORS= colors) apply to all the other ranges.
Specifying Color Names

You can specify colors for the ranges using any of the color-naming schemes supported by SAS/GRAPH. For a complete description of these color-naming schemes, see Table 39.2 on page 1108. The following table shows examples of each of the color naming schemes:

<table>
<thead>
<tr>
<th>Color-Naming Scheme</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)</td>
</tr>
<tr>
<td>RGBA*</td>
<td>COLORS=(a98FB9833 aDDA0DD66 aFFDAB999 aDB7093CC aB0E0E6FF)</td>
</tr>
<tr>
<td>CMYK</td>
<td>COLORS=(&quot;FF00FF00&quot; &quot;00FFFF00&quot; &quot;FFFFFF00&quot;)</td>
</tr>
<tr>
<td>HLS</td>
<td>COLORS=(H14055FF H0F060FF H0B485FF H07880FF)</td>
</tr>
<tr>
<td>HSV</td>
<td>COLORS=(V0F055FF V010FFFF V03BFFFF V12C55E8)</td>
</tr>
<tr>
<td>Gray Scale</td>
<td>COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)</td>
</tr>
<tr>
<td>SAS Registry Colors</td>
<td>COLORS=(palegreen plum peachpuff palevioletred powderblue)</td>
</tr>
<tr>
<td>CNS Color Names</td>
<td>COLORS=(&quot;very light purplish blue&quot; &quot;light vivid green&quot; &quot;medium strong yellow&quot; &quot;dark grayish green&quot;)</td>
</tr>
</tbody>
</table>

* RGBA color mode is not supported by Java devices. RGBA color mode is supported by ActiveX devices when the output is used in Microsoft applications.

Syntax: GKPI Procedure

Restriction: The only device supported for the GKPI procedure is JAVAIMG. If you do not specify DEVICE=JAVAIMG, then the procedure sets this option automatically.

Requirement: At least one DIAL, HSLIDER, HBULLET, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, or VSLIDER statement is required.

Global statements: FOOTNOTE, GOPTIONS (BORDER, VSIZE, HSIZE, XPIXEL, IBACK, CBACK, CTEXT, HTEXT, FTEXT only), TITLE

Supports: Run-group processing

Tips: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, "Using Run-Group Processing," on page 67 for more information.
When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See “About Filename Indexing” on page 119. To reset the current index value and overwrite output files that already exist, use the following statement:

```ods graphics / reset=index;```

See: “Overview: GKPI Procedure” on page 1099

---

**PROC GKPI Statement**

Identifies the display mode.

### Syntax

`PROC GKPI <MODE= BASIC | MODERN | RAISED>;`

### Optional Argument

PROC GKPI statement options affect all graphs produced by the procedure.

**MODE=BASIC | MODERN | RAISED**

specifies the display mode of KPI chart images.

**BASIC**

creates a two-dimensional image.

**MODERN**

creates an image with a streamlined appearance.
RAISED

creates a three-dimensional version of the basic KPI.

Default    BASIC

DIAL, HBULLET, HSLIDER, HTRAFFICLIGHT, SPEEDOMETER, VTRAFFICLIGHT, VBULLET, and VSLIDER Statement

Creates a chart in one of seven display types.

Requirement: The ACTUAL= value and the BOUNDS= list are required.

Syntax

<DIAL ACTUAL=value BOUNDS=bounds-list </options>>;
<HBOUNDLET | BULLET ACTUAL=value BOUNDS=bounds-list </options>>;
<VBULLET ACTUAL=value BOUNDS=bounds-list </options>>;
<HSLIDER | SLIDER ACTUAL=value BOUNDS=bounds-list </options>>;
<VSLIDER ACTUAL=value BOUNDS=bounds-list </options>>;
<HTRAFFICLIGHT ACTUAL=value BOUNDS=bounds-list </options>>;
<VTRAFFICLIGHT | TRAFFICLIGHT ACTUAL=value BOUNDS=bounds-list </options>>;
<SPEEDOMETER ACTUAL=value BOUNDS=bounds-list </options>>;

Summary of Optional Arguments

Appearance options

ACTIVECOLORS=(color–1 color–2 ...color–n)
specifies the list of active colors for each qualitative range.

AFONT=(<F="fontname" <BOLD> </ITALIC> > <C=color> <H=text-height <units> > )
specifies the name, color, and text height for the font used for the actual KPI value label.
AVALE | NOAVALE
  specifies whether to display the actual KPI value label.

BFONT=(<F="fontname"</BOLD><ITALIC><C=color><H=text-height
<units>>)
  specifies the name, color, and text height of the font used for the boundary
  and tick mark values.

BVALUE | NOBVALUE
  specifies whether to display the boundary values.

COLORS=(color–1 color–2 ...color–n)
  specifies the list of inactive colors for each qualitative range.

FORMAT="SAS-format"
  specifies a SAS format for the boundary and actual values.

LABEL= "string"
  specifies a label for the graphic.

LFONT=(<F="fontname"</BOLD><ITALIC><C=color><H=text-height
<units>>)<J=LEFT | RIGHT | CENTER>
  specifies the name, color, and text height of the font to use for the label that is
  specified by the LABEL= option.

LOWBOUNDARY | NOLOWBOUNDARY
  specifies whether the KPI chart displays as if the KPI value falls in the lower
  range or the upper range when the actual KPI value falls directly on a range
  boundary.

TARGET= data-value
  specifies the numeric value of the target key performance indicator.

TYPE=FULL | HALF | QUARTER
  specifies the size of the display for speedometers.

Drill-down option
  LINK="url"
  adds a drill-down link to the KPI chart.

Output file description options
  DESCRIPTION= "description"
  specifies the description of the output.

NAME="name"
  specifies the name of the graphics output file.

Required Arguments

ACTUAL= data-value
  specifies the actual value of the key performance indicator. The actual data value can
  fall outside the bounds specified with the BOUNDS= option, but the GKPI
  procedure will display the actual value indicator at the outermost edge of the KPI
  chart. If you specify a missing value (ACTUAL=.), then the GKPI procedure does not
  generate a KPI chart.

See  “Specifying Qualitative Range Boundaries and Actual KPI Values” on page
1102

BOUNDS=(bound–1 bound-2 ...bound-n)
  specifies a list of defined boundary values. The values can be negative or positive,
  but you must specify the list in either ascending or descending order. Separate each
  boundary value with a blank space.
See “Specifying Qualitative Range Boundaries and Actual KPI Values” on page 1102

Optional Arguments
You can specify as many options as needed and list them in any order.

**ACTIVECOLORS=**(color-1 color-2 ...color-n)
specifies the list of active colors for each qualitative range. This list is the colors that you want to use for each range when that range contains the actual KPI value. You do not have to specify a color for each range in the KPI chart. The default colors shown in Figure 39.2 on page 1104 (or the colors specified by the COLORS= option) are used for each range for which active colors are not specified. The number of entries in the ACTIVECOLORS= list cannot exceed the number of ranges that are defined. That is, the maximum number of active colors is one less than the number of entries in the BOUNDS= list. Separate each color with either a blank space or a comma.

See “Controlling Qualitative Range Colors ” on page 1104

**Example**
“Example 1: Using the Default Colors as the Active Colors” on page 1117

**AFONT=**(F=fontname BOLD ITALIC C=color H=text-height units>
(specifies the name, color, and text height for the font used for the actual KPI value label.

Style reference Font attribute of the GraphLabelText element

See “Font Suboptions” on page 1116

**Example**
AFONT=(f="Comic Sans MS" c=red h=15pt)

**Example**
“Example 3: Creating a Dial KPI Chart” on page 1119

**AVALE|** NOAVALUE
specifies whether to display the actual KPI value label.

Alias AVALE| NOAVALUE

Default AVALE

**Example**
“Example 6: Creating a Traffic Light” on page 1123

**BFONT=**(F=fontname BOLD ITALIC C=color H=text-height units>
specifies the name, color, and text height of the font used for the boundary and tick mark values. If you increase the size of the font to the point where labels would collide, then the intermediate labels are not displayed. The GKPI procedure displays only the lowest and highest boundary labels.

Style reference Font attribute of the GraphValueText element

See “Font Suboptions” on page 1116

**Example**
BFONT=(font="Arial" color=H14055FF height=.25in)
BVALUE | NOBVALUE
specifies whether to display the boundary values.

Alias BVAL | NOBVAL
Default BVALUE

COLORS=(color–1 color–2 ...color–n)
specifies the list of inactive colors for each qualitative range. This list is the colors that you want to be used for each range when that range does not contain the actual KPI value. You must specify a color for each range in the KPI chart. That is, the number of entries in the COLORS= list must be one less than the number of entries in the BOUNDS= list. Separate each color with either a blank space or a comma.

For all KPI charts except traffic lights, if you define two to five ranges, the GKPI procedure applies a default set of colors ranging from red to green. If you define only one or more than five ranges, the default color for all ranges is gray.

For traffic lights, the default color for all ranges is gray. This option is ignored by the TRAFFICLIGHT statement if the ACTIVECOLORS= option is specified.

See “Controlling Qualitative Range Colors” on page 1104

Examples
“Example 1: Using the Default Colors as the Active Colors” on page 1117

“Example 2: Creating a Gray Scale Bullet Graph” on page 1118

“Example 5: Defining a Speedometer with Reversed Colors” on page 1122

“Example 6: Creating a Traffic Light” on page 1123

DESCRIPTION= "description"
specifies the description of the output. The maximum length for the description is 256 characters. The description does not appear on the graph. The default is “Key performance indicator”.

The descriptive text is shown in the “description” portion of each of the following:

• the Results window.
• the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement.
• the chart description for web output. See “Chart Descriptions for Web Presentations” on page 189 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="percent 8.0" to display values as percentages or format="datetime7." to display SAS datetime values in the format ddmmmyy. You can also specify a user-defined format.
LABEL= "string"

specifies a label for the graphic. The label is displayed at the top of graph, beneath the title, if a title is specified.

Note

By default, labels are displayed at the top center of the graphics output area, and the KPI chart is displayed in the center of the output area. To reduce the space between labels and the KPI chart, reduce the size of the graphics output area by specifying the XPIXELS= /YPIXELS= or HSIZE= /VSIZE= options in the GOPTIONS statement. See “The Graphics Output and Device Display Areas” on page 70 for more information.

Examples

“Example 4: Defining a Speedometer” on page 1121

“Example 5: Defining a Speedometer with Reversed Colors” on page 1122

“Example 6: Creating a Traffic Light” on page 1123

LFONT=(<F=fontName> <BOLD> <ITALIC> > <C=color> <H=text-height <units>> > <J=LEFT | RIGHT | CENTER> )

specifies the name, color, and text height of the font to use for the label that is specified by the LABEL= option. You can also specify whether the label is left-justified, centered, or right-justified within the graphics output area.

Style reference

Font attribute of the GraphLabelText element

See

“LABEL= “string”” on page 1114 and “Font Suboptions” on page 1116

Example

LFONT=(<F="Albany AMT/italic" C=cornflowerblue H=.25cm J=right)

Example

“Example 4: Defining a Speedometer” on page 1121

LINK=“url”

adds a drill-down link to the KPI chart. Enclose the URL in quotation marks.

Restriction

The length of the URL specified must be no greater than 256 characters. Specifying a URL in excess of this limit leads to truncation of the URL in the output.
LOWBOUNDARY | NOLOWBOUNDARY
specifies whether the KPI chart displays as if the KPI value falls in the lower range or the upper range when the actual KPI value falls directly on a range boundary. This option controls the color that is used for dial centers, traffic lights, and actual value indicators. It also controls which range is displayed in the active color, if an active color list is specified. The default is LOWBOUNDARY, which tells the GKPI procedure to use the color of the lower range. If you specify NOLOWBOUNDARY, then PROC GKPI uses the color of the higher range. Figure 39.4 on page 1115 illustrates the effect of this option on dial centers and on the actual KPI value indicator in a horizontal slider when both a range boundary and the actual KPI value is 30.

Figure 39.4  LOWBOUND and NOLOWBOUND Effect on Indicator Colors

Alias
LOWBOUND | NOLOWBOUND

Default
LOWBOUNDARY

Example
“Example 3: Creating a Dial KPI Chart” on page 1119

NAME="name"
specifies the name of the graphics output file.
The following applies to name:

- For the graphics output filename:
  - All characters are represented in lowercase.
  - The JAVA IMG device must be used in order to generate a graphics output file.
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.
  - The name can include special characters.
  - Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique filename for the output. See “About Filename Indexing” on page 119.
TARGET= data-value
specifies the numeric value of the target key performance indicator. If you specify a missing value (TARGET=.), then the GKPI procedure generates a KPI chart without a target value indicator.

Restriction  Not supported by the TRAFFICLIGHT statement

Examples
“Example 2: Creating a Gray Scale Bullet Graph” on page 1118
“Example 3: Creating a Dial KPI Chart” on page 1119
“Example 4: Defining a Speedometer” on page 1121

TYPE=FULL | HALF | QUARTER
specifies the size of the display for speedometers.

Default  FULL

Restriction  Valid for SPEEDOMETER statement only

See  “Speedometer KPI Charts” on page 1101

**Font Suboptions**
You can control the fonts that are used to display the boundary and tick mark values, the actual KPI values, and the labels with BFONT=, AFONT=, and LFONT= options, respectively. Each of these options takes a font specification on the FONT= suboption.

<FONT="fontname</BOLD><ITALIC>" > <COLOR=color> <HEIGHT=text-height<units>> <JUSTIFICATION=LEFT | CENTER | RIGHT>

*Note:* The JUSTIFICATION= suboption is supported by the LFONT= option only.

**FONT= "fontname </BOLD><ITALIC> "**
specifies the font name. You can specify only system fonts (such as TrueType and UNIX system fonts), not SAS/GRAPH fonts. For more information, see “SAS/GRAPH, System, and Device-Resident Fonts” on page 300.

Alias  F=

See  Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299

**COLOR=color**
specifies the text color. You can specify the color in any of the color-naming schemes recognized by SAS/GRAPH.

Alias  C=

See  “Specifying Color Names” on page 1108

**HEIGHT=text-height <units>**
specifies the font height in units. You can specify the text height in units of points (PT), centimeters (CM), inches (IN), or percentage of the graphics output area (PCT).

Alias  H=
JUSTIFICATION= LEFT | RIGHT | CENTER
specifies whether the text is left-justified, centered, or right-justified within the graphics output area. You can specify LEFT, RIGHT, or CENTER.

Alias J=

Default CENTER

Restriction This suboption is supported by the LFONT= option only.

Examples: GKPI Procedure

Example 1: Using the Default Colors as the Active Colors

Features: PROC GKPI statement option: MODE=RAISED
HSLIDER statement options COLORS=
ACTIVECOLORS=

Sample library member: GKPGRSLD

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

The default colors described in “Default Colors” on page 1104 can be used as the active colors instead of the inactive colors. This example uses the same value of gray for all ranges for the inactive color. It uses the red, orange, yellow-green, and green colors shown in Figure 39.2 on page 1104 as the active colors.

Program

goptions reset=all device=javaimg xpixels=180 ypixels=110;
proc gkpi mode=raised;
hslider actual=-6.7 bounds=(-10 -5 0 5 10) / 
  activecolors=(cxD06959 cxE1A05D cxBD5DF cx84AF5B);
run;
quit;
Program Description

**Set the graphics environment.** Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```
goptions reset=all device=javaimg xpixels=180 ypixels=110;
```

**Generate the KPI chart.** Specify the range boundaries, actual KPI value, and colors. Boundary values can be positive or negative or both, but must be specified in either ascending or descending order. All colors are specified as hexadecimal RGB values. The same value of gray, cxB2B2B2, is used as the inactive color for all ranges. The default colors listed in Table 39.1 on page 1104 are used as the active colors.

```plaintext
proc gkpi mode=raised;
hslider actual=-6.7 bounds=(-10 -5 0 5 10) /
  activecolors=(cxD06959 cxE1A05D cxBDCD5F cx84AF5B);
run;
```

**End the procedure.** The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```plaintext
quit;
```

Example 2: Creating a Gray Scale Bullet Graph

**Features:**
- PROC GKPI statement option: MODE=RAISED
- VBULLET statement options
  - COLORS=
  - TARGET=

**Sample library member:**
- GKPGGBUL

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

This example creates a vertical bullet graph. It uses a gray scale color scheme that provides a good contrast between ranges. This color scheme can be used in output that is included in publications that are not in color.
Program

goptions reset=all device=javaimg xpixels=130 ypixels=250;
proc gkpi mode=raised;
vbullet
  actual=.58 bounds=(0 .22 .38 .52 .68 1) / target=.75
  colors=(cx747474 cx8C8C8C cxB2B2B2 cxD2D2D2 cxE6E6E6);
run;
quit;

Program Description

Set the graphics environment. Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

goptions reset=all device=javaimg xpixels=130 ypixels=250;

Generate the KPI chart. Specify the range boundaries, actual KPI value, target value, and colors. The gray scale colors are specified using hexadecimal RGB values.

proc gkpi mode=raised;
vbullet
  actual=.58 bounds=(0 .22 .38 .52 .68 1) / target=.75
  colors=(cx747474 cx8C8C8C cxB2B2B2 cxD2D2D2 cxE6E6E6);
run;

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

quit;

Example 3: Creating a Dial KPI Chart

Features:

PROC GKPI statement option: MODE=RAISED
DIAL statement options
  AFONT=
  BFONT=
  FORMAT=
Program

```sas
goptions reset=all device=javaimg xpixels=240 ypixels=200;
proc gkpi mode=raised;
dial actual=.46 bounds=(0 .23 .46 .65 .79 1) /
   target=.9 nolowbound format="percent8.0"
   afont=(f="Albany AMT" height=.5cm)
   bfont=(f="Albany AMT" height=.4cm) ;
run;
quit;
```

Program Description

**Set the graphics environment.** Use the XPIXELS and YPIXELS options in the GOPTIONS statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

```sas
      goptions reset=all device=javaimg xpixels=240 ypixels=200;
```

**Generate the KPI chart.** Specify the range boundaries, actual KPI value, and target value. In this case, the target value falls on a range boundary. The NOLOWBOUNDARY option specifies that the KPI chart behaves as if the actual KPI value falls in the higher range. The AFONT= and BFONT= options specify the fonts for the actual value and the range boundary values, respectively. The FORMAT= option specifies the SAS format for the values in the chart.

```sas
proc gkpi mode=raised;
dial actual=.46 bounds=(0 .23 .46 .65 .79 1) /
   target=.9 nolowbound format="percent8.0"
   afont=(f="Albany AMT" height=.5cm)
   bfont=(f="Albany AMT" height=.4cm) ;
run;
```
End the procedure. The You must close the destination to generate output.

quit;

Example 4: Defining a Speedometer

Features:
- PROC GKPI statement option: MODE=RAISED
- SPEEDOMETER statement options:
  - COLORS=
  - FORMAT=
  - LABEL=
  - LFONT=
  - TARGET=

Sample library member: GKPSPD

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

---

Program
goptions reset=all device=javaimg xpixels=210 ypixels=200;
proc gkpi mode=raised;
  speedometer actual=.72 bounds=(0 .40 .60 1) / target=.85
  lfont=(f="Albany AMT" height=.5cm) label="Average Capacity"
  format="percent8.0";
run;
quit;

Program Description

Set the graphics environment. The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options scale the KPI charts to a size that would be appropriate for use in a dashboard.

goptions reset=all device=javaimg xpixels=210 ypixels=200;
Generate the KPI chart. Specify the range boundaries, actual KPI value, and target value. The LFONT= option specifies the font for the label. The FORMAT= option specifies the SAS format for the values in the chart.

```
proc gkpi mode=raised;
  speedometer actual=.72 bounds=(0 .40 .60 1) / target=.85
    lfont=(f="Albany AMT" height=.5cm) label="Average Capacity"
    format="percent8.0";
run;
```

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```
quit;
```

Example 5: Defining a Speedometer with Reversed Colors

**Features:**
- PROC GKPI statement option: MODE=BASIC
- SPEEDOMETER statement options
  - COLORS=
  - LABEL=

**Sample library member:** GKPSPCLR

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

```
goptions reset=all device=javaimg xpixels=210 ypixels=200;
proc gkpi mode=basic;
  speedometer actual=12 bounds=(0 25 50 100) /
    colors=(cx84AF5B cxF1DC63 cxD06959)
    label="Cancellations";
run;
quit;
```
Program Description

**Set the graphics environment.** The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options scale the KPI charts to a size that would be appropriate for use in a dashboard.

```sas
options reset=all device=javaimg xpixels=210 ypixels=200;
```

**Generate the KPI chart.** Specify the range boundaries, actual KPI value, target value, and colors. The green, yellow, and red colors listed in Figure 39.2 on page 1104 are specified in reverse order so that green begins at zero.

```sas
proc gkpi mode=basic;
speedometer actual=12 bounds=(0 25 50 100) /
    colors=(cx84AF5B cxF1DC63 cxD06959)
    label="Cancellations";
run;
```

**End the procedure.** The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

```sas
quit;
```

---

**Example 6: Creating a Traffic Light**

**Features:**
- PROC GKPI statement option: MODE=RAISED
- TRAFFICLIGHT statement options
  - COLORS=
  - LABEL=
  - NOAVALUE

**Sample library member:** GKPTRAFF

**Note:**
- The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example creates a traffic light that uses primary green, yellow, and red colors. Colors are applied to vertical KPI charts from the bottom up, so to get red at the top, you must specify red last in the list of colors.
Program

goptions reset=all device=javaimg xpixels=120 ypixels=210;

proc gkpi mode=raised;
  trafficlight actual=598 bounds=(1500 900 600 0) /
  colors=(green yellow red) noavalue
  label="New Subscriptions";
run;
quit;

Program Description

Set the graphics environment. The XPIXELS and YPIXELS graphics options reduce the size of the graphics output area and therefore reduce both the size of the KPI chart and the distance between the label and the KPI chart. These options statement to scale the KPI charts to a size that would be appropriate for use in a dashboard.

  goptions reset=all device=javaimg xpixels=120 ypixels=210;

Generate the KPI chart. Specify the range boundaries, actual KPI value, and colors. The NOAVALUE option turns off the display of the actual KPI value. The colors are specified as SAS Registry Color names.

  proc gkpi mode=raised;
  trafficlight actual=598 bounds=(1500 900 600 0) /
  colors=(green yellow red) noavalue
  label="New Subscriptions";
run;

End the procedure. The GKPI procedure supports RUN-group processing, so it is recommended that you enter the QUIT statement to end the procedure.

  quit;
Overview: GOPTIONS Procedure

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 375 for a list of the graphics options that you can set with the GOPTIONS statement. See Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515 for a complete description of each graphics option.

The list of graphics options is displayed in the SAS Log window and includes the names of the options, the current values, and a brief description of each one. You can use PROC GOPTIONS statement options to control what information is listed and where it appears in the Log window. Output 40.1 on page 1126 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.
SAS/GRAPH software options and parameters
(executing in DMS Process environment)

NOACCESSIBLE
NOADMGDF
ALTDESC
ALT text
ASPECT=
NOAUTOCOPY
NOAUTOFEED
AUTOSIZE=
catalog rows and
columns
BINDING=DEFAULTEDGE
NOBORDER
CBACK=
CBY=
CELL
CHARACTERS
CHARTYPE=
CIRCLEARC
NOCOLLATE
COLORS=( )
CPATTERN=
CSYMBOL=
CTEXT=
CTITLE=
DASH
DASHSCALE=
DELAY=
DEVADDR=
DEVICE=
DEVMAP=DEFAULT
DISPLAY
DISPOSAL=NONE
DRVINIT=
DRVTERM=
NODUPLEX
NOERASE
EXTENSION=
NOFASTTEXT
routines;
templated replay

Note: All of the graphics options that are displayed by the GOPTIONS procedure are
described in Chapter 25, “Graphics Options and Device Parameters Dictionary,” on
page 515.

Syntax: GOPTIONS Procedure

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)>;

Summary of Optional Arguments

Item request options
AXIS
requests a list of all current AXIS definitions.
FOOTNOTE
requests a list of all of the current FOOTNOTE and TITLE definitions.
LEGEND
requests a list of all of the current LEGEND definitions.
OPTION=graphics-option
requests information about the specified graphics option.
PATTERN
requests a list of all of the current PATTERN definitions.
SYMBOL
requests a list of all of the current SYMBOL definitions.
TITLE
requests a list of all of the current TITLE and FOOTNOTE definitions.

Listing format options
CENTIMETERS
displays the values of the HORIGIN=, HSIZE=, PAPERFEED=, PAPERLIMIT=, VORIGIN=, and VSIZE= graphics options in units of centimeters (CM).
NOLIST
suppresses the display of graphics options.
NOLOG
displays the output in the OUTPUT window instead of the Log window.
SHORT
suppresses the descriptions of the graphics options and displays the graphics options values in an alphabetical list in paragraph form.

Optional Arguments
You can specify as many options as you want and list them in any order.

AXIS
requests a list of all current AXIS definitions. AXIS also lists the current values for all graphics options, unless you use the NOLIST option. If you have not defined any AXIS statements, the GOPTIONS procedure issues a message.
CENTIMETERS
displays the values of the HORIGIN=, HSIZE=, PAPERFEED=, PAPERLIMIT=, VORIGIN=, and VSIZE= graphics options in units of centimeters (CM). These graphics options use units of IN or CM only, and their values are always stored as inches even if you specify CM. Therefore, the GOPTIONS procedure displays these values in inches, unless you specify the CENTIMETERS option.

Alias    CM

Note    The CENTIMETERS option does not affect the graphics options that can use unit specifications of CELLS, CM, IN, PCT, and PT.

FOOTNOTE
requests a list of all of the current FOOTNOTE and TITLE definitions. FOOTNOTE also lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any FOOTNOTE or TITLE statements, the GOPTIONS procedure issues a message.

Alias    F

Example    “Example 1: Displaying TITLE and FOOTNOTE Statements” on page 1129

LEGEND
requests a list of all of the current LEGEND definitions. LEGEND lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any LEGEND statements, the GOPTIONS procedure issues a message.

Alias    L

NOLIST
suppresses the display of graphics options. Use the NOLIST option in conjunction with the appropriate statement request option when you want to list only the current AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, or TITLE definitions.

Alias    N

Example    “Example 1: Displaying TITLE and FOOTNOTE Statements” on page 1129

NOLOG
displays the output in the OUTPUT window instead of the Log window.

OPTION=graphics-option
requests information about the specified graphics option. For these options, requesting the information about one option also displays the value of its corresponding option, as follows:

- HSIZE= and VSIZE=
- HPOS= and VPOS=
- XMAX= and YMAX=
- XPIXELS= and YPIXELS=

PATTERN
requests a list of all of the current PATTERN definitions. The PATTERN option lists the current values for all of the graphics options unless you use the NOLIST option.
If you have not defined any PATTERN statements, the GOPTIONS procedure issues a message.

**Alias P**

**SHORT**

suppresses the descriptions of the graphics options and displays the graphics options values in an alphabetical list in paragraph form.

**Example** “Example 2: Displaying Graphics Options without the Description” on page 1130

**SYMBOL**

requests a list of all of the current SYMBOL definitions. The SYMBOL option lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any SYMBOL statements, the GOPTIONS procedure issues a message.

**Alias S**

**TITLE**

requests a list of all of the current TITLE and FOOTNOTE definitions. The TITLE option lists the current values for all of the graphics options, unless you use the NOLIST option. If you have not defined any FOOTNOTE or TITLE statements, the GOPTIONS procedure issues messages.

**Alias T**

---

**Examples: GOPTIONS Procedure**

---

**Example 1: Displaying TITLE and FOOTNOTE Statements**

**Features:** PROC GOPTIONS statement options FOOTNOTE and NOLIST

**Sample library member:** GOPTIFT

This example uses the FOOTNOTE option to display the current definitions of both the FOOTNOTE and TITLE statements. It also uses the NOLIST option to suppress the list of graphics options. **Output 40.2 on page 1129** shows the listing that appears in the Log window.

**Output 40.2 Using the NOLIST Option (GOPTIFT)**

```
TITLE1 HEIGHT=6 COLOR=BLUE FONT=SWISSB "Production Quality" ;
TITLE2 HEIGHT=4 COLOR=BLUE FONT=SWISSB "January through June";
FOOTNOTE1 HEIGHT=3 COLOR=GREEN FONT=SWISS "Data from SASDATA.QUALITY" ;
FOOTNOTE2 HEIGHT=3 COLOR=GREEN FONT=SWISS "* denotes approximations";
```
Program

goptions reset=all;

option h=6 c=blue f=swissb "Production Quality";
option h=4 c=blue f=swissb "January through June";
option h=3 c=green f=swiss "Data from SASDATA.QUALITY";
option h=3 c=green f=swiss "* denotes approximations";

proc goptions nolist footnote;
run;

Program Description

Clear all global statements.

option reset=all;

Define titles and footnotes.

option h=6 c=blue f=swissb "Production Quality";
option h=4 c=blue f=swissb "January through June";
option h=3 c=green f=swiss "Data from SASDATA.QUALITY";
option h=3 c=green f=swiss "* denotes approximations";

Produce the listing. The NOLIST and FOOTNOTE options control the information that
appears in the Log window.

proc goptions nolist footnote;
run;

Example 2: Displaying Graphics Options without the Description

Features: PROC GOPTIONS statement option SHORT
Sample library member: GOPSHORT

This example uses the SHORT option to display only the values of graphics options
without the description of each graphics option. Output 40.3 on page 1131 shows the
listing that appears in the Log window.
Using the SHORT Option (GOPSHORT)

Program

goptions reset=all;

proc goptions short;

run;

Program Description

Set the graphics environment. The values of the graphics options specified in this statement appear in the Log listing.

  goptions reset=all;

Produce the listing. The SHORT option suppresses the display of the description of each graphics option.

  proc goptions short;

  run;
Chapter 41

GPLOT Procedure

Overview: GPLOT Procedure ........................................ 1134
- About the GPLOT Procedure ...................................... 1134
- About Plots of Two Variables ..................................... 1134
- About Plots with a Classification Variable ...................... 1135
- About Bubble Plots .................................................. 1136
- About Plots with Two Vertical Axes .............................. 1137
- About Interpolation Methods ...................................... 1138

Parts of a Plot .......................................................... 1138

About the Input Data Set ............................................. 1140
- Data Set Requirements .............................................. 1140
- Missing Values ....................................................... 1140
- Values Out of Range ................................................ 1140
- Sorted Data ........................................................... 1140
- Logarithmic Axes ..................................................... 1141

Syntax: GPLOT Procedure ............................................ 1141
- PROC GPLOT Statement ........................................... 1141
- BUBBLE Statement ................................................ 1143
- BUBBLE2 Statement ............................................... 1158
- PLOT Statement ...................................................... 1168
- PLOT2 Statement ..................................................... 1189

Examples: GPLOT Procedure ........................................ 1206
- Example 1: Generating a Simple Bubble Plot .................. 1206
- Example 2: Labeling and Sizing Plot Bubbles ................ 1208
- Example 3: Adding a Right Vertical Axis ....................... 1211
- Example 4: Plotting Two Variables ............................... 1214
- Example 5: Connecting Plot Data Points ......................... 1217
- Example 6: Generating an Overlay Plot ......................... 1219
- Example 7: Filling Areas in an Overlay Plot .................... 1223
- Example 8: Plotting Three Variables ............................. 1226
- Example 9: Plotting with Different Scales of Values ........ 1231
- Example 10: Creating Plots with Drill-down Functionality for the Web . . . . 1235
Overview: GPLOT Procedure

About the GPLOT Procedure

The GPLOT procedure plots the values of two or more variables on a set of coordinate axes (X and Y). The coordinates of each point on the plot correspond to two variable values in an observation of the input data set. The procedure can also generate a separate plot for each value of a third (classification) variable. It can also generate bubble plots in which circles of varying proportions representing the values of a third variable are drawn at the data points.

The procedure produces a variety of two-dimensional graphs including the following plots:

• simple scatter plots
• overlay plots in which multiple sets of data points are displayed on one set of axes
• plots against a second vertical axis
• bubble plots
• logarithmic plots (controlled by the AXIS statement)

In conjunction with the SYMBOL statement, the GPLOT procedure can produce join plots, high-low plots, needle plots, and plots with simple or spline-interpolated lines. The SYMBOL statement can also display regression lines on scatter plots.

The GPLOT procedure is useful for the following tasks:

• displaying a long series of data, and showing trends and patterns
• interpolating between data points
• extrapolating beyond existing data with the display of regression lines and confidence limits

About Plots of Two Variables

Plots of two variables display the values of two variables as data points on one horizontal axis (X) and one vertical axis (Y). Each pair of X and Y values forms a data point.

The following figure shows a simple scatter plot that plots the values of the variable HEIGHT on the vertical axis and the variable WEIGHT on the horizontal axis. By default, the PLOT statement scales the axes to include the maximum and minimum data values and displays a symbol at each data point. It labels each axis with the name of its variable or an associated label and displays the value of each major tick mark.
Figure 41.1 Scatter Plot of Two Variables (GPLVRBL1(a))

The program for this plot is in “Example 4: Plotting Two Variables” on page 1214. For more information about producing scatter plots, see “PLOT Statement” on page 1168.

You can also overlay two or more plots (multiple sets of data points) on a single set of axes. You can also apply a variety of interpolation techniques to these plots. See “About Interpolation Methods” on page 1138.

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.
By default, plots with a classification variable generate a legend. In the code that generates the plot for "Example 8: Plotting Three Variables" on page 1226, a SYMBOL statement connects the data points. It also specifies the plot symbol that is used for each value of the classification variable (CITY). For more information about how to produce plots with a classification variable, see “PLOT Statement” on page 1168.

**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 41.3 on page 1137 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 41.3  Bubble Plot (GPLBUBL1)

The program for this plot is in “Example 1: Generating a Simple Bubble Plot” on page 1206. For more information about producing bubble plots, see “BUBBLE Statement” on page 1143.

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 41.4  Plot with a Right Vertical Axis (GPLSCVL1)
The program for this plot is in “Example 9: Plotting with Different Scales of Values” on page 1231. For more information about how to produce plots with a right vertical axis, see “PLOT2 Statement” on page 1189 and “BUBBLE2 Statement” on page 1158.

About Interpolation Methods

In addition to these graphs, you can produce other types of plots such as box plots or high-low-close plots by specifying various interpolation methods with the SYMBOL statement. Use the SYMBOL statement to do the following tasks:

• connect the data points with straight lines
• specify regression analysis to fit a line to the points and can display lines for confidence limits
• connect the data points to the zero line on the vertical axis
• display the minimum and maximum values of Y at each X value and mark the mean value. Also, display standard deviations that connect the data points with lines or bars, generate box plots, or plot high-low-close stock market data
• specify that a pattern fills the polygon that is defined by data points
• smooth plot lines with spline interpolation
• use a step function to connect the data points

The “SYMBOL Statement” on page 412 describes all interpolation methods.

Parts of a Plot

Some terms used with GPLOT procedure are illustrated in Figure 41.5 on page 1139 and Figure 41.6 on page 1139.
**Figure 41.5** GPLOT Procedure Terms

- Vertical axis label (y variable)
- Horizontal axis label (x variable)
- Frame
- Axis area
- Reference line
- Plot line
- Plot symbol
- Minor tick marks
- Major tick marks
- Mark values

**Figure 41.6** Additional GPLOT Procedure Terms

- Area 1
- Area 2
- Area 3
- Offset
- Percent
- Month

Parts of a Plot 1139
About the Input Data Set

Data Set Requirements

The input data set that is used by the GPLOT procedure must contain at least one variable to plot on the horizontal axis and one variable to plot on the vertical axis. Typically, the horizontal axis shows an independent variable (time, for example), and the vertical axis shows a dependent variable (temperature, for example). With the exception of the Java and ActiveX device drivers, variables can be either character or numeric. For the ActiveX and Java device drivers, the horizontal axis can contain either character or numeric values, but the vertical axis can contain only numeric variables. Graphs are automatically scaled to the values of the character data or to include the values of numeric data. However, you can control scaling with procedure options or with associated AXIS statements.

Missing Values

If the value of either of the plot variables is missing, the GPLOT procedure does not include the observation in the plot. If you specify interpolation with a SYMBOL definition, the plot is not broken at the missing value. To break the plot line or area fill at the missing value, use the PLOT statement's SKIPMISS option. The SKIPMISS option is enabled only for JOIN interpolation.

Values Out of Range

Data values can be excluded from a graph by restricting the range of axis values with the VAXIS= or HAXIS= options or with the ORDER= option in an AXIS statement. When an observation contains a value outside of the specified axis range, the GPLOT procedure excludes the observation from the plot and issues a message to the log.

If you specify interpolation with a SYMBOL definition, by default values outside of the axis range are excluded from interpolation calculations. As a result, interpolated values for the plot might change. Values that are omitted from interpolation calculations have a particularly noticeable effect on the high-low interpolation methods: HILO, STD, and BOX. In addition, regression lines and confidence limits represent only part of the original data.

To specify that values out of range are included in the interpolation calculations, use the MODE= option in a SYMBOL statement. When MODE=INCLUDE, values that fall outside of the axis range are included in interpolation calculations but excluded from the plot. The default (MODE=EXCLUDE) omits observations that are outside of the axis range from interpolation calculations. See the MODE= option in the “SYMBOL Statement” on page 412 for details.

Sorted Data

Data points are plotted in the order in which the observations are read from the data set. Therefore, if you use any type of interpolation that generates a line, sort your data by the horizontal axis variable.
Logarithmic Axes

If your data contain logarithmic values or if the data values vary over a wide range or contain large values, you might want to specify a logarithmic axis for the horizontal or vertical axis. Logarithmic axes can be specified with the AXIS statement options LOGBASE= and LOGSTYLE=. See the “AXIS Statement” on page 345 for a complete discussion.

Syntax: GPLOT Procedure

Restriction: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

Requirements: An input data set is required. If not specified, an input data set must have been created before issuing the PROC GPLOT statement.

At least one PLOT or BUBBLE statement is required. A PLOT2 or BUBBLE2 statement can be used in conjunction with a PLOT or BUBBLE statement.

Global statements: AXIS, FOOTNOTE, LEGEND, PATTERN, TITLE

Supports: RUN-group processing

Note: The procedure can include the SAS/GRAPH statement BY on page 370, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

PROC GPLOT DATA=input-data-set
<ANNOTATE=Annotate-data-set>
<GOUT=<libref:output-catalog>
<IMAGEMAP= output-data-set>
<UNIFORM>;

    BUBBLE plot-request(s) <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.
Requirement: An input data set is required. If not specified, an input data set must have been created before issuing the PROC GPLOT statement.

Syntax

PROC GPLOT DATA=input-data-set
<ANNOTATE=Annotate-data-set>
<GOUT=<libref>output-catalog>
<IMAGEMAP=output-data-set>
<UNIFORM>;

Required Argument

DATA=input-data-set

specifies the SAS data set that contains the variables to plot. By default, the procedure uses the most recently created SAS data set.

See “The SAS Data Set: Your Key to the SAS System” in Step-by-Step Programming with Base SAS

“About Data Set Options” in SAS Data Set Options: Reference

“About the Input Data Set” on page 1140

Optional Arguments

ANNOTATE=Annotate-data-set

specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635

GOUT=<libref, >output-catalog

specifies the SAS catalog in which to save the graphics output that is produced by the GPLOT procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See “Specifying the Catalog Name and Entry Name for Your GRSEGs ” on page 120

IMAGEMAP=output-data-set

creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192 and “Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198
UNIFORM

specifies that the same axis scaling is used for all graphs that are produced by the procedure. The same axis scaling would apply across Run-groups as well. By default, the range of axis values for each axis is based on the minimum and maximum values in the data. Therefore, they can vary from graph to graph and among BY groups. Using the UNIFORM option forces the value range for each axis to be the same for all graphs. Take for example, a procedure that produces multiple graphs with both left and right vertical axes. The UNIFORM option scales all of the left axes the same and all of the right axes the same. The scaling is based on the minimum and maximum data values.

In addition, UNIFORM forces the assignment of SYMBOL statements for the category variable without regard to the BY-group variable. If a legend is generated, UNIFORM makes the legend the same across graphs.

Restriction

Partially supported by Java and ActiveX

BUBBLE Statement

Creates bubble plots in which a third variable is plotted against two variables represented by the horizontal and vertical axes; the value of the third variable controls the size of the bubble.

**Requirement:** At least one plot request is required.

**Global statements:** AXIS, FOOTNOTE, TITLE

**Syntax**

BUBBLE plot-request(s) </ option(s)>;

**Summary of Optional Arguments**

**Appearance options**

ANNOTATE=Annotate-data-set

specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

**Bubble appearance options**

BCOLOR=bubble-color

specifies the color for the bubbles.

BFILL=SOLID | GRADIENT

enables you to generate solid or gradient-filled bubbles.

BFONT=font

specifies the font to use for bubble labels.

BLABEL

labels the bubbles with the values of the third variable.

BSIZE=AREA | RADIUS

specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure.

BSIZE=multiplier
specifies an overall scaling factor for the bubbles so that you can increase or
decrease the size of all bubbles by this factor.

Catalog options

DESCRIPTION="description"
specifies a description of the output.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics
output file, if one is created.

Horizontal axis options

AUTOHREF
draws reference lines at all major tick marks on the horizontal axis.

CAUTOHREF=reference-line-color
specifies colors for reference lines drawn at major tick marks on the
horizontal axis, as specified by the AUTOHREF option.

CHREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the horizontal
axis.

HAXIS=value-list | AXIS<1 ... 99>
specifies major tick mark values for the horizontal axis or assigns an axis
definition.

HMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major
tick mark on the horizontal axis.

HREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the horizontal axis at
points that are specified by value-list.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed.

HZERO
specifies that tick marks on the horizontal axis begin in the first position or
end in the last position with a value of zero, depending on the horizontal
variable values.

LAUTOHREF=reference-line-type
specifies the line type for reference lines at major tick marks on the
horizontal axis, as specified by the AUTOHREF option.

LHREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the horizontal
axis.

WAUTOHREF=reference-line-width
specifies the line width for all reference lines at major tick marks on the
horizontal axis as determined by the AUTOHREF option.

WHREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the horizontal axis.

ODS options

HTML=variable
identifies the variable in the input data set whose values create links or data
tips or both.

URL=character-variable
specifies a character variable whose values are URLs.
Plot appearance options

CAXIS=axis-color
specifies the color for the axis line and all major and minor tick marks.

CFRAME=background-color
fills the axis area with the specified color.

CTEXT=text-color
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels.

DATAORDER='entry-description'
plots character of midpoint-type data in data order instead of the default alphabetical order.

FRAME | NOFRAME
specifies whether a line is drawn around the axis area.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

GRID
draws reference lines at all major tick marks on both axes.

IFRAME=fileref | 'external-file'
identifies the image file you that want to apply to the axis area of the plot.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of an image to fill the axis area (TILE) or to stretch a single instance of an image to fill the axis area (FIT).

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Vertical axis options

AUTOVREF
draws reference lines at all major tick marks on the vertical axis.

CAUTOVREF=reference-line-color
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=reference-line-type
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the vertical axis.

VAXIS=value-list | AXIS<1 ...99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=reference-line-width
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis.

**Required Argument**

plot-request(s)
each specifies the variables to plot and produces a separate graph. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request must have this form:

*y-variable*x-variable=bubble-size
plots the values of two variables and draws a circle (bubble) at each data point. The value of the third variable determines the size of the bubble.

*y-variable
variable plotted on the left vertical axis.

*x-variable
variable plotted on the horizontal axis.

*bubble-size
variable that specifies the size of the bubbles. Bubble-size must be numeric. If the value of bubble-size is positive, bubbles are drawn with a solid line. If it is negative, bubbles are drawn with a dashed line.

**Notes**
If you specify the 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.

**Optional Arguments**
Options in a BUBBLE statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635
**AUTOHREF**

draws reference lines at all major tick marks on the horizontal axis. LAUTOHREF=, CAUTOHREF=, and WAUTOHREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the HAXIS= option.

**AUTOVREF**

draws reference lines at all major tick marks on the vertical axis. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

**BCOLOR=bubble-color**

specifies the color for the bubbles. You might not specify the BCOLOR= option. In that case, the bubble color becomes the color of the default style (GSTYLE) or the color specified by the current ODS style (if used).

<table>
<thead>
<tr>
<th>Style reference</th>
<th>ContrastColor attribute of the GraphOutline, GraphData1, and TwoColorAltRamp elements</th>
</tr>
</thead>
</table>

**Examples**

- “Example 2: Labeling and Sizing Plot Bubbles” on page 1208
- “Example 3: Adding a Right Vertical Axis” on page 1211

**BFILL=SOLID | GRADIENT**

enables you to generate solid or gradient-filled bubbles. By default, the JAVA and ActiveX devices create solid bubbles.

- **BFILL=SOLID** fills the bubbles with the color specified by the BCOLOR= option. If the BFILL option is not specified, then the color is specified by the current style. If you are using specific ODS style, the color comes from the contrast color attribute within the GraphData1 style element.

- **BFILL=GRADIENT** starts with the current axis area color and gradually transitions to the color specified with the BCOLOR= option or the color of the current style. If you are using an ODS style, the colors are controlled by the startcolor and endcolor attributes of the TwoColorAltRamp style element.

**Restriction**

Not supported by Java and ActiveX

**Note**

The SAS/GRAPH ActiveX control displays negative values as empty circles.

**BFONT=font**

specifies the font to use for bubble labels. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about how to specify font. If you omit the BFONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the font specified by the current style
3. the default hardware font

<table>
<thead>
<tr>
<th>Style reference</th>
<th>Font attribute of the GraphValueText element</th>
</tr>
</thead>
</table>

**Restriction**

Not supported by Java and ActiveX
See the option “BLABEL” on page 1148 for information about the location and color of labels

Example “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

**BLABEL**
labels the bubbles with the values of the third variable. If the variable has a format, the formatted value is used. By default, bubbles are not labeled.

The procedure normally places labels directly outside of the circle at 315 degrees rotation. If a label in this position does not fit in the axis area, other 45-degree placements (that is, 45, 135, and 225 degrees) are attempted. If the label cannot be placed at any of the positions (45, 135, 225, or 315 degrees) without being clipped, the label is omitted. However, labels can collide with other bubbles or previously placed labels.

Labels are displayed in the color specified by the CTEXT= option. If you omit the CTEXT=option, the default is the color of the current style.

Example “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

**BScale=** **AREA | RADIUS**
specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure. By default, BSCALE=AREA.

The value that is assigned to the BSCALE= option affects how large the bubbles appear in relation to each other. For example, suppose the third variable value is twice as big for one bubble as it is for another. If BSCALE=AREA, the area of the larger bubble is twice the area of the smaller bubble. If BSCALE=RADIUS, the radius of the larger bubble is twice the radius of the smaller bubble and the larger bubble has more than twice the area of the smaller bubble.

Restriction Not supported by Java and ActiveX

**BSIZE=multiplier**
specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

In web output, the Java applets and the ActiveX Control override the default value. To prevent this override, specify a value for the BSIZE= option, rather than relying on the default value.

Restriction Partially supported by Java and ActiveX

Example “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

**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 CAUTOVREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.
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.

**Interaction**
- The CAUTOHREF= option overrides the CHREF= option for lines drawn with the AUTOHREF option

**CHREF=** *reference-line-color | (reference-line-color) | (reference-line-color-list)*
specifies the color of reference lines drawn perpendicular to the horizontal axis. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. Specifying in parentheses a single color applies that color only to the first reference line drawn with the HREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the HREF= option. The syntax of the color list requires parentheses and each color separated by spaces *(color1 color2 ... colorN)*. If you do not specify the CHREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

**Alias**  
CH=

**Interaction**
The CAUTOHREF= option overrides the CHREF= option for lines drawn with the AUTOHREF option

**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 a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices
In an AXIS statement, the COLOR= suboption of either a LABEL= option or a
VALUE= option overrides the CTEXT= option. It also determines the color of the
axis label, or the color of the tick mark values, respectively.

Alias            C=, CT=

Style reference  Color attributes of the GraphValueText and the GraphLabelText
elements

CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the vertical axis. This
option affects reference lines drawn with the AUTOVREF, VREF, and GRID
options. Specifying without parentheses a single color applies that color to all
reference lines. The CAUTOVREF= option overrides the CVREF= option for lines
drawn with the AUTOVREF option. Specifying in parentheses a single color applies
that color only to the first reference line drawn with the VREF= option. Specifying a
color list applies colors sequentially to successive reference lines drawn with the
VREF= option. The syntax of the color list requires parentheses and each color
separated by spaces (color1 color2 ... colorN). If you do not specify the CVREF=
option, the GPLOT procedure uses the color specified by the CAXIS= option. If
neither option is specified, the default color is retrieved from the current style or
from the first color in the color list if the NOGSTYLE option is specified.

Alias            CV=

DATAORDER='entry-description'
plots character of midpoint-type data in data order instead of the default alphabetical
order.

Restriction      Supported by Java and ActiveX only

DESCRIPTION="description"
specifies a description of the output. The maximum length for description is 256
characters. The description does not appear in the output. The descriptive text is
shown in each of the following:
- the chart description for web output (depending on the device driver). See “Chart
  Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in
  an ODS HTML statement, assuming that the output is generated while the
  contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the SAS Explorer.
- the Description field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution
options, which work as they do when used on TITLE, FOOTNOTE, and NOTE
statements. Refer to “Substituting BY Line Values in a Text String” on page 959.
The 256-character limit applies before the substitution takes place for these options.
Thus, if in the SAS program the description text exceeds 256 characters, it is
truncated to 256 characters, and then the substitution is performed.

Alias            DES=
Based on the specified value of bubble-size in the plot-request, either
BBELE PLOT OF chart-variable by chart-variable or BUBBLE PLOT
OF chart-variable by chart-variable sized by chart-variable

FRAME | NOFRAME
specifies whether a line 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 back wall color, use the CFRAME= option.
To fill the axis area with a back wall image, use the IFRAME= 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.
By default, the procedure scales the axis and provides an appropriate number of tick
marks. For a description of value-list, see “HAXIS=value-list | AXIS<1 ... 99>” on
page 1178 for the PLOT statement. To assign labels to horizontal reference lines,
specify an axis definition that contains the REFLABEL= option. The labels are
applied in sequence to all reference lines drawn with the AUTOHREF and HREF=
options.

If you assign an axis definition that does not currently exist, the option is ignored.

If data values fall outside of the range that is specified by the HAXIS= option, then
by default the outlying data values are not used in interpolation calculations.

For web output that is generated with a Java or ActiveX device driver, certain
options of the AXIS statement are not supported. For details, see the “AXIS
Statement” on page 345.

Restriction  Partially supported by Java and ActiveX

See  “About the Input Data Set” on page 1140 for more information about
values out of range

Example  “Example 2: Labeling and Sizing Plot Bubbles” on page 1208
HMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the horizontal axis. Minor tick marks are not labeled. The HMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias HM=

Example “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

HREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the horizontal axis at points that are specified by value-list. For a description of value-list see “HAXIS=value-list | AXIS<1 …99>” on page 1178 for the PLOT statement. To specify labels for these reference lines, use the HAXIS= option.

Values can be listed in any order, but should be within the range of values represented by the chart response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks

The LHREF=, CHREF=, and WHREF= options can be used to change the line types, colors, and widths of these reference lines, respectively.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed. For web output that is generated with a Java device driver, the horizontal axis data must be numeric.

Restriction Partially supported by Java and ActiveX

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

Restriction For Java and ActiveX, only the ACTXIMG device is supported for the BUBBLE2 statement.

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

HZERO
specifies that tick marks on the horizontal axis begin in the first position or end in the last position with a value of zero, depending on the horizontal variable values.

When all horizontal variable values are positive, the first tick mark on the horizontal axis has a value of zero. When all horizontal variable values are negative, the last tick mark on the horizontal axis has a value of zero.

The HZERO request is ignored if the horizontal variable contains a mix of positive and negative values. It is ignored when it is ordered with the HAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the horizontal variable contains any time-formatted values, such as date, time, or datetime. In most of these
cases when the HZERO option is ignored, the last tick mark on the horizontal axis has a value of zero.

**IFRAME=fileref | 'external-file'**
identifies the image file you want to apply to the axis area of the plot.

<table>
<thead>
<tr>
<th>Restriction</th>
<th>Not supported by Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactions</td>
<td>This option is overridden by the NOIMAGEPRINT goption. For information about the NOIMAGEPRINT goption see “IMAGEPRINT” on page 577.</td>
</tr>
<tr>
<td>See</td>
<td>“Displaying Images on Data Elements” on page 336</td>
</tr>
</tbody>
</table>

**IMAGESTYLE=TILE | FIT**
specifies whether to use multiple instances of an image to fill the axis area (TILE) or to stretch a single instance of an image to fill the axis area (FIT). The TILE value is the default.

| See | “IFRAME=fileref | 'external-file’” on page 1153 |

**LAUTOHREF=reference-line-type**
specifies the line type for reference lines at major tick marks on the horizontal axis, as specified by the AUTOHREF option. Line types are specified as whole numbers from 1 to 46, with 1 representing a solid line and the other values representing dashed lines. The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

| Style reference | LineStyle attribute of the GraphGridLines element |

**LAUTOVREF=reference-line-type**
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option. The reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line.

| Style reference | LineStyle attribute of the GraphGridLines element |

**LHREF=reference-line-type | (reference-line-type) | (reference-line-type-list)**
specifies line types for reference lines drawn perpendicular to the horizontal axis. The reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying in parentheses a single line type applies that line type only to the first reference line drawn with the HREF= option. Specifying a linetype list applies line types in sequence to successive reference lines drawn with the HREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (type1 type2 ... typeN). The default line type is retrieved from the current style. If the NOGSTYLE option is specified, the default value is 1, which draws a solid line.
LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the vertical axis. The
reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies
a solid line; values 2 through 46 specify dashed lines. This option affects reference
lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without
parenthesis a single line type applies that line type to all reference lines. The
LAUTOVREF= option overrides the LVREF= option for lines drawn with the
AUTOVREF option. Specifying in parentheses a single line type applies that line
type only to the first line drawn by the VREF= option. Specifying a line-type list
applies line types in sequence to successive reference lines drawn with the VREF=
option. The syntax of the line type list requires parentheses and line types separated
by spaces (type1 type2 ... typeN). The default line type is retrieved from the current
style. If the NOGSTYLE option is specified, the default value is 1, which draws a
solid line. To specify colors for these reference lines, use the CVREF= option. To
specify labels for these reference lines, use the VAXIS= option.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output
file, if one is created.

The following applies to name:
• The name can be up to 256 characters in length.
• Special characters in the name are converted to underscores.

• For the GRSEG entry name:
  • The name is truncated to eight characters.
  • The first character is always represented in uppercase, and all other
characters are represented in lowercase.
  • If the name begins with a number, an underscore is prepended to the name.
  • If the name duplicates an existing name, SAS/GRAPH appends a number or
increments the last number used to create a unique graph name (for example,
name1, name2, and so on). If necessary, the name is truncated so that the
name and appended number do not exceed eight characters.

• For the graphics output filename:
  • The filename is based on the NAME= value except when you use an ODS
LISTING destination, a DEVICE= option, and a file reference specifying an
output filename. In this case, the file reference specification overrides the
NAME= value. See “Controlling Graphics Output for ODS LISTING” on
page 118.
  • All characters are represented in lowercase.
  • If a number is added to the GRSEG name, the same number is added to the
output filename. See “About Filename Indexing” on page 119.
  • If the NAME= value is 8 characters or less, the filename is the GRSEG entry
name.
• If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

• The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default  GPLOT

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias  NOAXES

URL=character-variable

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions  This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the BUBBLE2 statement

Interaction  If you specify both the HTML= and URL= options, then the URL= option is ignored

Note  You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See  “Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

VAXIS=value-list | AXIS<1 ...99>

specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of the value-list see “HAXIS=value-list | AXIS<1 ...99>” on page 1178 for the PLOT statement. To assign labels to reference lines, specify an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the “AXIS Statement” on page 345.

Restriction  Partially supported by Java and ActiveX

Examples  “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

“Example 3: Adding a Right Vertical Axis” on page 1211
VMINOR=number-of-minor-ticks

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias  VM=

Example  "Example 2: Labeling and Sizing Plot Bubbles" on page 1208

VREF=value | (value) | (value-list)

draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of value-list see “HAXIS=value-list | AXIS<1 …99>” on page 1178 for the PLOT statement. The LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines, respectively. To specify labels for these reference lines, use the VAXIS= option.

VREVERSE

specifies that the order of the values on the vertical axis should be reversed.

VZERO

specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when it is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOHREF=reference-line-width

specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOHREF=reference-line-color” on page 1148.

Style reference  LineThickness attribute of the GraphGridLines element

WAUTOVREF=reference-line-width

specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOVREF=reference-line-color” on page 1148.

Style reference  LineThickness attribute of the GraphGridLines element

WHREF=reference-line-width | (reference-line-width) | (reference-line-width-list)

specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the HREF= option. Specifying a line width list applies line widths in
sequence to successive reference lines drawn with the HREF= option. The syntax of
the line-width list requires parentheses and line widths separated by spaces (width1
width2 ...widthN). The default line width is specified by the current style or by the
AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for
these reference lines, use the option “CHREF=reference-line-color | (reference-line-
color) | (reference-line-color-list)” on page 1149.

Style reference LineThickness attribute of the GraphGridLines element

WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis. Line
widths are specified as whole numbers. Specifying without parentheses a line width
applies that line width to all reference lines drawn with the AUTOVREF and VREF=
options. Note that the WAUTOVREF= option overrides WVREF=reference-line-
width for reference lines drawn with the AUTOREF option. Specifying in
parentheses a single line width applies that line width to the first reference line
drawn with the VREF= option. Specifying a line width list applies line widths in
sequence to successive reference lines drawn with the VREF= option. The syntax of
the line-width list requires parentheses and line widths separated by spaces (width1
width2 ...widthN). The default line width is specified by the current style or by the
AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for
these reference lines, use the option “CVREF=reference-line-color | (reference-line-
color) | (reference-line-color-list)” on page 1150.

Style reference LineThickness attribute of the GraphGridLines element

Details

Description

The BUBBLE statement specifies one or more plot requests that name the horizontal and
left vertical axis variables and the variable that controls the size of the bubbles. This
statement automatically does the following:

• centers each circle at a data point that is determined by the values of the vertical and
horizontal axes variables

• scales the axes to include the maximum and minimum data values

• labels each axis with the name of its variable or associated label

• displays each major tick mark value

• draws circles for values that are located within the axes

You can use statement options to control axis scaling, draw reference lines, or modify
the appearance of axes. You can also use statement options to control the display of the
bubbles, specify an axis area color or image, and specify annotation.

In addition, you can use global statements to modify axes (AXIS statement), and add
text to the graph (TITLE, NOTE, and FOOTNOTE statements). You can also use the
Annotate data set to enhance the plot.

Controlling the Display of Bubbles

The BUBBLE statement draws circles only for values that are located within the axes.
Observations with values that lie outside of the axis area are not plotted. If a bubble size
value causes a bubble to overlap the axis, the bubble is clipped against the axis line. The
bubbles for the highest axis value and lowest axis value might be clipped unless you
modify the axes in either of the following ways:
• by offsetting the first and last values
• by adding values to the range that is represented by the axis

Specify the range of values on an axis with the HAXIS= or VAXIS= option, or with AXIS definitions.

To add a right vertical axis, use a BUBBLE2 statement.

**BUBBLE2 Statement**

Creates a second vertical axis on the right side of a graph produced by an accompanying BUBBLE or PLOT statement. A second variable can be plotted against this axis.

**Requirement:**

You cannot use the BUBBLE2 statement alone. You can use it only with a BUBBLE or PLOT statement. At least one plot request is required.

**Global statements:**

AXIS, FOOTNOTE, TITLE

**Syntax**

```
BUBBLE2 plot-request(s) / option(s);
```

**Summary of Optional Arguments**

**Appearance options**

- **ANNOTATE=Annotate-data-set**
  
specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

**Bubble appearance options**

- **BCOLOR=bubble-color**
  
specifies the color for the bubbles.

- **BFILL=SOLID | GRADIENT**
  
enables you to generate solid or gradient-filled bubbles.

- **BFONT=font**
  
specifies the font to use for bubble labels.

- **BLABEL**
  
labels the bubbles with the values of the third variable.

- **BSCALE=AREA | RADIUS**
  
specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure.

- **BSIZE=multiplier**
  
specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

**ODS options**

- **HTML=variable**
  
identifies the variable in the input data set whose values create links or data tips or both.

- **URL=character-variable**
  
specifies a character variable whose values are URLs.
Plot appearance options

CAXIS=axis-color
   specifies the color for the axis line and all major and minor tick marks.

CFRAME=background-color
   fills the axis area with the specified color.

CTEXT=text-color
   specifies the color for all text on the axes, including tick mark values, axis
   labels, and bubble labels.

DATAORDER='entry-description'
   plots character of midpoint-type data in data order instead of the default
   alphabetical order.

FRAME | NOFRAME
   specifies whether a line is drawn around the axis area.

FRONTREF
   specifies that reference lines drawn by the AUTOREF or REF= options
   should be drawn in front of the bars.

GRID
   draws reference lines at all major tick marks on both axes.

NOAXIS
   suppresses the axes, including axis lines, axis labels, all major and minor tick
   marks, and tick mark values.

Vertical axis options

AUTOVREF
   draws reference lines at all major tick marks on the vertical axis.

CAUTOVREF=reference-line-color
   specifies the color of reference lines drawn at major tick marks on the vertical
   axis, as determined by the AUTOVREF option.

CVREF=reference-line-color | (reference-line-color | (reference-line-color-list)
   specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=reference-line-type
   specifies a line type for reference lines drawn at major tick marks on the
   vertical axis, as specified by the AUTOVREF option.

LVREF=reference-line-type | (reference-line-type | (reference-line-type-list
   specifies line types for reference lines drawn perpendicular to the vertical
   axis.

VAXIS=value-list | AXIS<1 …99>
   specifies the major tick mark values for the vertical axis or assigns an axis
   definition.

VMINOR=number-of-minor-ticks
   specifies the number of minor tick marks that are drawn between each major
   tick mark on the vertical axis.

VREF=value | (value | (value-list
   draws one or more reference lines perpendicular to the vertical axis at
   specified points.

VREVERSE
   specifies that the order of the values on the vertical axis should be reversed.

VZERO
   specifies that tick marks on the vertical axis begin in the first position or end
   in the last position with a value of zero, depending on the vertical variable
   values.
$WAUTOVREF=reference-line-width$

specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

$WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)$

specifies line widths for reference lines as determined by the vertical axis.

**Required Argument**

*plot-request(s)*

each specifies the variables to plot and produces a separate graph. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request must have this form:

$y$-variable$x$-variable$=$bubble-size

plots the values of two variables and draws a circle (bubble) at each data point. The value of the third variable determines the size of the bubble. All of these variables must be in the input data set:

$y$-variable

variable plotted on the right vertical axis; typically it is different from $y$-variable in the accompanying BUBBLE or PLOT statement.

$x$-variable

variable plotted on the horizontal axis; it is the same as $x$-variable in the accompanying BUBBLE or PLOT statement.

bubble-size

specifies the size of the bubbles. *Bubble-size* must be numeric. If the value of *bubble-size* is positive, bubbles are drawn with a solid line. If it is negative, bubbles are drawn with a dashed line.

**Optional Arguments**

**ANNOTATE=Annotate-data-set**

specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

Alias** ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635

**AUTOVREF**

draws reference lines at all major tick marks on the vertical axis. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

**BCOLOR=bubble-color**

specifies the color for the bubbles. You might not specify the BCOLOR= option. In that case, the bubble color becomes the color of the default style (GSTYLE) or the color specified by the current ODS style (if used).

Style reference

ContrastColor attribute of the GraphOutline, GraphData1, and TwoColorAltRamp elements

Examples

“Example 2: Labeling and Sizing Plot Bubbles” on page 1208
**BFILL=SOLID | GRADIENT**

enables you to generate solid or gradient-filled bubbles. By default, the JAVA and ActiveX devices create solid bubbles.

- BFILL=SOLID fills the bubbles with the color specified by the BCOLOR= option. If the BFILL option is not specified, then the color is specified by the current style. If you are using specific ODS style, the color comes from the contrast color attribute within the GraphData1 style element.

- BFILL=GRADIENT starts with the current axis area color and gradually transitions to the color specified with the BCOLOR= option or the color of the current style. If you are using an ODS style, the colors are controlled by the `startcolor` and `endcolor` attributes of the TwoColorAltRamp style element.

**Restriction**

Not supported by Java and ActiveX

**Note**

The SAS/GRAPH ActiveX control displays negative values as empty circles.

**BFONT=font**

specifies the font to use for bubble labels. See Chapter 21, “Specifying Fonts in SAS/GRAPH Programs,” on page 299 for details about how to specify `font`. If you omit the BFONT= option, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the font specified by the current style
3. the default hardware font

**Style reference**

Font attribute of the GraphValueText element

**Restriction**

Not supported by Java and ActiveX

**See**

the option “BLABEL” on page 1161 for information about the location and color of labels

**Example**

“Example 2: Labeling and Sizing Plot Bubbles” on page 1208

**BLABEL**

labels the bubbles with the values of the third variable. If the variable has a format, the formatted value is used. By default, bubbles are not labeled.

The procedure normally places labels directly outside of the circle at 315 degrees rotation. If a label in this position does not fit in the axis area, other 45-degree placements (that is, 45, 135, and 225 degrees) are attempted. If the label cannot be placed at any of the positions (45, 135, 225, or 315 degrees) without being clipped, the label is omitted. However, labels can collide with other bubbles or previously placed labels.

Labels are displayed in the color specified by the CTEXT= option. If you omit the CTEXT=option, the default is the color of the current style.

**Example**

“Example 2: Labeling and Sizing Plot Bubbles” on page 1208
**BSCALE=AREA | RADIUS**
specifies whether the bubble-scaling proportion is based on the area of the circles or the radius measure. By default, BSCALE=AREA.

The value that is assigned to the BSCALE= option affects how large the bubbles appear in relation to each other. For example, suppose the third variable value is twice as big for one bubble as it is for another. If BSCALE=AREA, the area of the larger bubble is twice the area of the smaller bubble. If BSCALE=RADIUS, the radius of the larger bubble is twice the radius of the smaller bubble and the larger bubble has more than twice the area of the smaller bubble.

**Restriction** Not supported by Java and ActiveX

**BSIZE=multiplier**
specifies an overall scaling factor for the bubbles so that you can increase or decrease the size of all bubbles by this factor.

In web output, the Java applets and the ActiveX Control override the default value. To prevent this override, specify a value for the BSIZE= option, rather than relying on the default value.

**Restriction** Partially supported by Java and ActiveX

**Example** “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

**CAUTOVREF=reference-line-color**
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

**Style reference** Color attribute of the GraphGridLines element

**CAXIS=axis-color**
specifies the color for the axis line and all major and minor tick marks. By default, the procedure uses the color of the current style.

The CAXIS= option is overridden by the COLOR= option in an AXIS definition. The COLOR= option in an AXIS definition is overridden by the COLOR= suboption of the MAJOR= and MINOR= options in an AXIS definition.

**Alias** CA=

**Style reference** Color attribute of the GraphAxisLines attribute

**CFRAME=background-color**
fills the axis area with the specified color. If the FRAME option is also in effect, the procedure determines the color of the frame according to the precedence list given for the FRAME option description. If the IFRAME= option is in effect, the specified image fills the axis area instead of the specified color.

**Style reference** Color attribute of the GraphWalls element

**CTEXT=text-color**
specifies the color for all text on the axes, including tick mark values, axis labels, and bubble labels. The GPLOT procedure searches for a color specification in this order:
1. colors specified for labels and values on assigned AXIS and LEGEND statements, which override the CTEXT= option specified in the PLOT statement
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

In an AXIS statement, the COLOR= suboption of either a LABEL= option or a VALUE= option overrides the CTEXT= option. It also determines the color of the axis label, or the color of the tick mark values, respectively.

Alias                      C=, CT=

Style reference            Color attributes of the GraphValueText and the GraphLabelText elements

**CVREF=**reference-line-color | (reference-line-color) | (reference-line-color-list)

specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and each color separated by spaces (color1 color2 ... colorN). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias                      CV=

**DATAORDER='**entry-description**'**

plots character of midpoint-type data in data order instead of the default alphabetical order.

Restriction                Supported by Java and ActiveX only

**FRAME | NOFRAME**

specifies whether a line 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 back wall color, use the CFRAME= option.

To fill the axis area with a back wall image, use the IFRAME= 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.

HTML=variable
identifies the variable in the input data set whose values create links or data tips or
both. The variable values are either links or data tips or both that are created in the
HTML file generated by the ODS statement. The links are URLs pointing to web
pages to display when the user clicks (drills down) on elements in the graph. Data
tips are detailed information or data values that are displayed as pop-up text when a
mouse pointer is positioned over elements in the graph.

Restriction For Java and ActiveX, only the ACTXIMG device is supported for the
BUBBLE2 statement.

See “Data Tips for Web Presentations” on page 191
“Adding Links and Enhancements with the URL=, HTML=, and
HTML_LEGEND= Options” on page 192

LAUTOVREF=reference-line-type
specifies a line type for reference lines drawn at major tick marks on the vertical
axis, as specified by the AUTOVREF option. The reference-line-type value can be a
whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46
specify dashed lines. The default line type is retrieved from the current style, or if the
NOGSTYLE option is specified, the default value is 1, which draws a solid line.

Style reference LineStyle attribute of the GraphGridLines element

LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the vertical axis. The
reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies
a solid line; values 2 through 46 specify dashed lines. This option affects reference
lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without
parenthesis a single line type applies that line type to all reference lines. The
LAUTOVREF= option overrides the LVREF= option for lines drawn with the
AUTOVREF option. Specifying in parentheses a single line type applies that line
type only to the first line drawn by the VREF= option. Specifying a line-type list
applies line types in sequence to successive reference lines drawn with the VREF=
option. The syntax of the line type list requires parentheses and line types separated
by spaces (type1 type2 ... typeN). The default line type is retrieved from the current
style. If the NOGSTYLE option is specified, the default value is 1, which draws a
solid line. To specify colors for these reference lines, use the CVREF= option. To
specify labels for these reference lines, use the VAXIS= option.

Alias LV=

Style reference LineStyle attribute of the GraphGridLines element
NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

URL=character-variable
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restrictions This option affects graphics output that is created through the ODS HTML destination only
Not supported by the JAVAIMG and ACTXIMG devices when using the BUBBLE2 statement

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

Note You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See “Overview of Enhancing Web Presentations” on page 188
“Example: GIF Output with Drill-Down Links” on page 163

VAXIS=value-list | AXIS<1 …99>
specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of the value-list see “HAXIS=value-list | AXIS<1 …99>” on page 1178 for the PLOT statement. To assign labels to reference lines, specify an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the “AXIS Statement” on page 345.

Restriction Partially supported by Java and ActiveX

Examples “Example 2: Labeling and Sizing Plot Bubbles” on page 1208
“Example 3: Adding a Right Vertical Axis” on page 1211

VMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias VM=

Example “Example 2: Labeling and Sizing Plot Bubbles” on page 1208

VREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of value-list see “HAXIS=value-list | AXIS<1 …99>” on page 1178 for the PLOT statement. The LVREF=, CVREF=, and WVREF= options
can be used to change the line types, colors, and widths of these reference lines, respectively. To specify labels for these reference lines, use the VAXIS= option.

VREVERSE
specifies that the order of the values on the vertical axis should be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when it is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOVREF=
reference-line-width
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOVREF=reference-line-color” on page 1162.

WVREF=
reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify colors for these reference lines, use the option “CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)” on page 1163.

Details

Options
Options for the BUBBLE2 statement are identical to the options for the BUBBLE statement with exception of the following, which are ignored if specified:

- AUTOHREF
- CAUTOHREF=
- CHREF=
- DESCRIPTION=
See “BUBBLE2 Statement” on page 1158 for complete descriptions of options that you can use with the BUBBLE2 statement.

**Description**

The BUBBLE2 statement specifies one or more plot requests that name the horizontal and right vertical axis variables and the variable that controls the size of the bubbles. This statement automatically does the following:

- scales the axes to include the maximum and minimum data values
- labels each axis with the name of its variable or an associated label
- displays each major tick mark value
- draws circles for values that are located within the axes

You can use statement options to control right vertical axis scaling, draw reference lines on the right vertical axis, and control the display of the bubbles. You can also use statement options to display an axis area color or image, and specify annotation.

In addition, you can use global statements to modify the axes (AXIS statement), and add text to the graph (TITLE, NOTE, and FOOTNOTE statements). You can also use the Annotate data set to enhance the plot.

**Coordinating BUBBLE and BUBBLE2 Plot Requests**

The BUBBLE2 statement draws circles only for values that are located within the axes. Bubbles are not drawn for values that lie outside of the axis range. If a bubble size value causes a bubble to overlap the axis, the bubble is clipped against the axis line.

In the BUBBLE2 statement, either the *y-variable* or *bubble-size* can differ from the variables in the BUBBLE statement. Here are some possible combinations of plot requests for BUBBLE and BUBBLE2 statement pairs and how they affect the plot:

- The vertical axis variables Y and Y2 are different, but the bubble size variable, S, is the same in both:
  ```
  bubble y*x=s;
  bubble2 y2*x=s;
  ```
  These plot requests generate a plot in which both sets of bubbles have the same value (size) but different locations on the graph.

- The vertical axis variables are the same, Y, but the bubble size variables, S and S2, are different:
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:

  \[
  \text{bubble } y \times x = s; \\
  \text{bubble2 } y \times x = s2;
  \]

  These plot requests produce the equivalent of an overlay plot in which two different sets of bubbles plotted against different vertical axes are displayed on the same graph.

The plot requests in the BUBBLE and BUBBLE2 statements must be evenly matched, for example:

\[
\text{bubble } y \times x = s; \quad b \times a = c; \\
\text{bubble2 } y2 \times x = s2; \quad b2 \times a = c2;
\]

These statements produce two graphs, and each has two vertical axes. The first pair of plot requests (\(Y \times X = S\) and \(Y2 \times X = S\)) produce one graph. In this graph the variable X is plotted on the horizontal axis. The variable Y is plotted on the left axis and the variable Y2 is plotted on the right axis. In this pair, the value of S is the same for both requests. The second pair of plot requests (\(B \times A = C\) and \(B2 \times A = C2\)) produce another graph. In this graph the variable A is plotted on the horizontal axis. The variable B is plotted on the left axis and the variable B2 is plotted on the right axis.

Any modifications to horizontal axes specifications must be identical for both statements. If they are different, the BUBBLE2 axis specification is ignored.

There might be a case where the scale of values for the left and right vertical axes is the same. You also want both axes to represent the same range of values. You would specify the range with a VAXIS= option in both the BUBBLE and BUBBLE2 statements.
CAXIS=axis-color
  specifies the color for the axis line and all major and minor tick marks.

CBASELINE=baseline-color
  specifies the color of the horizontal baseline for reference lines of a plot.

CFRAME=background-color
  fills the axis area with the specified color.

COUTLINE=outline-color
  specifies the color of the outline that is drawn around filled areas.

CTEXT=text-color
  specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME | NOFRAME
  specifies whether a line is drawn around the axis area.

FRONTREF
  specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

IFRAME=fileref | 'external-file'
  identifies the image file that you want to apply to the axis area of the plot.

IMAGESTYLE=TILE | FIT
  specifies whether to use multiple instances of the image to fill the axis area (TILE) or to stretch a single instance of the image to fill the (FIT). The TILE value is the default.

NOAXIS
  suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Catalog options

DESCRIPTION="description"
  specifies a description of the output.

NAME="name"
  specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Horizontal axis options

AUTOHREF
  draws reference lines at all major tick marks on the horizontal axis.

CAUTOHREF=reference-line-color
  specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option.

CHREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
  specifies the color of reference lines drawn perpendicular to the horizontal axis.

HAXIS=value-list | AXIS<1 …99>
  specifies major tick mark values for the horizontal axis or assigns an axis definition.

HMINOR=number-of-minor-ticks
  specifies the number of minor tick marks drawn between each major tick mark on the horizontal axis.

HREF=value | (value) | (value-list)
  draws one or more reference lines perpendicular to the horizontal axis at points that are specified by value-list.
HREVERSE
specifies that the order of the values on the horizontal axis be reversed.

HZERO
specifies that tick marks on the horizontal axis begin in the first position or
depend on the horizontal
variable values.

LAUTOHREF=reference-line-type
specifies a line type for reference lines drawn at major tick marks on the
horizontal axis, as specified by the AUTOHREF option.

LHREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the horizontal
axis.

WAUTOHREF=reference-line-width
specifies the line width for all reference lines at major tick marks on the
horizontal axis as determined by the AUTOHREF option.

WHREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the horizontal axis.

ODS options

HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data
tips or both.

HTML=variable
identifies the variable in the input data set whose values create links or data
tips or both.

URL=character-variable
specifies a character variable whose values are URLs.

Plot options

AREAS=n
fills all the areas below plot line n with a pattern.

GRID
draws reference lines at all major tick marks on both axes.

LEGEND | LEGEND=LEGEND<1 ...99>
generates a legend or specifies the legend to use for the plot.

NOLASTAREA
removes the last area of a plot when the value specified by the AREAS=
option exceeds the number of bounded areas in the plot.

NOLEGEND
suppresses the legend that is generated by a plot request of the type y-
variable*x-variable=third-variable.

OVERLAY
places all the plots that are generated by the PLOT statement on one set of
axes.

REGEQN
displays the regression equation that is specified in the INTERPOL= option
of the SYMBOL statement in the lower left corner of the plot.

SKIPMISS
breaks a plot line or an area fill at occurrences of missing values of the Y
variable.

Vertical axis options
AUTOVREF
draws reference lines at all of the major tick marks on the vertical axis.

CAUTOVREF=reference-line-color
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option.

CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the vertical axis.

LAUTOVREF=reference-line-type
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the AUTOVREF option.

LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options.

VAXIS=value-list | AXIS<1 …99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

VMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

VREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the vertical axis at specified points.

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

WAUTOVREF=reference-line-width
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option.

WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis.

**Required Argument**

plot-request(s)
each specifies the variables to plot and produces a separate graph, unless you specify OVERLAY. All variables must be in the input data set. Multiple plot requests are separated with blanks. You can plot character or numeric variables. A plot request can be any of these:

y-variable*x-variable<=n>
plots the values of two variables and can assign a SYMBOL definition to the plot.

y-variable
variable plotted on the left vertical axis.

x-variable
variable plotted on the horizontal axis.

n
number of the nth generated SYMBOL definition.
The \( n \)th generated SYMBOL definition is not necessarily the same as the \( n \)th SYMBOL statement. Plot requests of the form \( y\text{-variable}\times x\text{-variable}=n \) assign the SYMBOL definition that is designated by \( n \) to the plot that is produced by \( y\text{-variable}\times x\text{-variable} \). For more information, see “About Plot Requests That Assign a SYMBOL Definition” on page 1189.

\[(y\text{-variable(s)})\times(x\text{-variable(s)})\]
plots the values of two or more variables and produces a separate graph for each combination of \( Y \) and \( X \) variables. That is, each \( Y\times X \) pair is plotted on a separate set of axes unless you specify OVERLAY.

\( y\text{-variable(s)} \)
variables plotted on the left vertical axes.

\( x\text{-variable(s)} \)
variables plotted on the horizontal axes.

If you use only one \( y\text{-variable} \) or only one \( x\text{-variable} \), omit the parentheses for that variable, for example:

\[
\text{plot (temp rain)*month;}
\]
This plot request produces two plots, one of TEMP and MONTH and one of RAIN and MONTH.

\( y\text{-variable}\times x\text{-variable}=\text{third-variable} \)
plots the values of two variables against a third classification variable.

\( y\text{-variable} \)
variable plotted on the left vertical axis.

\( x\text{-variable} \)
variable plotted on the horizontal axis.

\( \text{third-variable} \)
classification variable against which \( y\text{-variable} \) and \( x\text{-variable} \) are plotted. \( \text{Third-variable} \) 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 \( \text{third-variable} \). This means that all plots are drawn on the same set of axes. Also, a legend is automatically generated to show the plot symbol and color for each value of the classification variable.

The following plot request produces a graph with a plot line for each department and a legend that shows the plot symbol for each department:

\[
\text{plot sales*weekday=dept;}
\]
For an example of a plot that specifies a \( \text{third-variable} \), see “Example 8: Plotting Three Variables” on page 1226.

\[\text{Note} \quad \text{If a BY statement is used to produce multiple plots, you can make the legend identical across graphs by specifying the UNIFORM option in the PROC GPLOT statement.} \]

You can use more than one type of plot request in a single PLOT statement (provided that you do not specify OVERLAY), for example:

\[
\text{plot temp*month rain*month=2;}
\]
**Optional Arguments**

Options in a PLOT statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

**ANNOTATE=** *Annotate-data-set*

specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.

**AREAS=** *n*

fills all the areas below plot line *n* with a pattern. The value of *n* specifies which areas to fill:

- **AREAS=** 1 fills the first area.
- **AREAS=** 2 fills both the first and second areas, and so on.

You might specify a value for the AREAS= option that is greater than the number of bounded areas in the plot. In this case, the area between the top plot line and the axis frame is filled.

Before an area can be filled, the data points that border the area must be joined by a line. Use a SYMBOL statement with one of these interpolation methods to join the data points:

- **INTERPOL=JOIN**
- **INTERPOL=STEP**
- **INTERPOL=R|series**
- **INTERPOL=SPLINE | SM | L**

See the “SYMBOL Statement” on page 412 for details about interpolation methods.

By default, the AREAS= option fills areas by rotating a solid fill through the list of colors defined in the current style. If the NOGSTYLE option is specified, the areas are filled by rotating a solid fill through the device's color list. If the graph needs more patterns, it rotates hatch patterns, beginning with the M2N0 pattern. See the "PATTERN Statement" on page 398 for more information about map and plot patterns. However, if color is limited to a single color with the CPATTERN= or COLORS= graphic options, the solid pattern is skipped and the first default pattern is M2N0. If the COLORS= graphic option specifies a single color, use as many SYMBOL statements as you have areas to fill. You must do this because the INTERPOL= setting does not automatically apply to multiple symbol definitions.

You can alter the default pattern behavior by specifying patterns and colors in PATTERN statements that specify map and plot patterns. A separate PATTERN definition is needed for each specified area.

If you specify the PATTERN statements, the AREAS= option uses the lowest numbered PATTERN statement first. If it runs out of patterns, it uses the default behavior for map and plot patterns. See the “PATTERN Statement” on page 398 for details.

Pattern definitions are assigned to the areas below the plot lines in the order the plots are drawn. The first area is that between the horizontal axis and the plot line that is drawn first. The second area is that above the first plot line and below the plot line that is drawn second, and so on. If the line that is drawn second lies below the line
that is drawn first, the second area is hidden when the first is filled. The plots with the lower line values must be drawn first to prevent one area fill from overlaying another. If the lines cross, only the part of an area that is above the previous line is visible.

Therefore, you must order your PLOT statements when creating multiple plots in combination with the OVERLAY option. Order them so that the plot request that produces the lowest line value is first (leftmost). The plot request that produces the next lowest line value is the second plot request, and so on.

If you produce multiple plots with a y-variable*x-variable=third-variable plot request, the lines are plotted in order of increasing third variable values. In that case, you must recode your data so that the lowest value of the third variable produces the lowest plot line. Likewise, the next lowest value produces the next lowest plot line, and so on.

If you use the VALUE= option in the SYMBOL statement, some symbols might be hidden. If you also specify reference lines with the AREAS= option, they are drawn behind the pattern fill.

You might use the AREAS= option to fill the areas between your plots while using the JAVA device to generate the plot. In this case, when you move your mouse pointer over a filled area in the plot output, a tooltip is displayed. It shows the values for the last observation in the data set for the group that the area represents. However, when you position your mouse pointer over a data point on a plot line, a tooltip shows the value for that point. This might confuse your viewers. To help avoid confusion in that case, you can turn off the tooltips by specifying the following on your ODS statement:

```
parameters=("Tips"="NONE")
```

**Restriction**

Partially supported by Java

**Notes**

The AREAS= option works only if all plot lines are generated by the same PLOT or PLOT2 statement.

If you have specified the NOGSTYLE option and the first color in your device's default color list is black, color rotation begins with the second color in the list. That is, there are no solid black patterns. See “How Default Patterns and Outlines Are Generated” on page 407 for more information.

**Example**

“Example 7: Filling Areas in an Overlay Plot” on page 1223

**AUTOHREF**

draws reference lines at all major tick marks on the horizontal axis. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LAUTOHREF=, CAUTOHREF=, and WAUTOHREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the HAXIS= option.

**AUTOVREF**

draws reference lines at all of the major tick marks on the vertical axis. If you also use the AREAS= option, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option in either the PROC GPLOT statement or the PLOT statement. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of
these reference lines. To specify labels for these reference lines, use the VAXIS= option.

**CAUTOHREF=reference-line-color**

specifies colors for reference lines drawn at major tick marks on the horizontal axis, as specified by the AUTOHREF option. The default color is either the value of the CAXIS= option or the first color in the color list.

**CAUTOVREF=reference-line-color**

specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOVREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

**CAXIS=axis-color**

specifies the color for the axis line and all major and minor tick marks. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

Alias CA=

**CBASELINE=baseline-color**

specifies the color of the horizontal baseline for reference lines of a plot. This option compensates for a reference line obscuring the axis. This can occur when using a combination of the AUTOVREF option, the CAUTOVREF= option, or the CVREF= option. They must be specifying any color other than black, and a vertical axis with no offset. For example, the light gray reference line drawn by the AUTOVREF option and a CVREF= graydd specification obscures the default black color of the axis line. This option enables you to set the color of the baseline to match the axis color.

Specify the baseline-color as a name or a quoted string. Two examples are presented. The first indicates a name. The second specifies an RGB color code:

cbaseline=black

cb='CX0B5FF'

Alias CB=

Default The value of the CAXIS= option is the default color if you do not specify an alternate color for the baseline. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Restriction This option is not supported by the ACTIVEX, ACTXIMG, JAVA, or JAVAIMG devices

Note The baseline is drawn after the plot elements are drawn, and either a default or specified color is applied. This means that the reference lines are drawn behind plot elements such as bars. Use the FRONTREF option if you want reference lines drawn in front of the bars.

See “Specifying Colors in SAS/GRAPH Programs” on page 314

“FRONTREF” on page 1178

**CFRAME=background-color**

fills the axis area with the specified color.
**CHREF=reference-line-color | (reference-line-color) | (reference-line-color-list)**

Specifies the color of reference lines drawn perpendicular to the horizontal axis. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOHREF= option overrides the CHREF= option for reference lines drawn with the AUTOHREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the HREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the HREF= option. The syntax of the color list requires parentheses and colors separated by spaces (color1 color2 ...colorN). If you do not specify the CHREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

**Alias** CH=

**Style reference** Color attribute of the GraphGridLines element

**COUTLINE=outline-color**

Specifies the color of the outline that is drawn around filled areas. The filled areas are generated when the SYMBOL statement specifies “INTERPOL=map/plot-pattern” on page 421 or the GOPTIONS statement specifies “INTERPOL” on page 579. The default outline color is specified in the current style. However, the NOGSTYLE option might be specified. In that case, the default color is the first color in the device's color list (the foreground color). Also, the default slice outline color is determined as follows:

- If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.
- If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

**Alias** CO=

**Style reference** Color attribute of the GraphOutlines element

**Restriction** Not supported by Java

**Interaction** The COUTLINE= option overrides the SYMBOL statement option CO=.
CTEXT=text-color
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements. They override the CTEXT= option specified in the PLOT statement. This also includes colors specified by the POINTLABEL= option in the SYMBOL statement.
2. the color specified by the CTEXT= option in the PLOT statement
3. the color specified by the CTEXT= option in the GOPTIONS statement
4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

Alias C=

Style reference Color attributes of the GraphValueText and the GraphLabelText elements

Note If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and colors separated by spaces (color1 color2 ... colorN). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

Style reference Color attribute of the GraphGridLines element

DESCRIPTION="description"
specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:
• the chart description for web output (depending on the device driver). See “Chart Descriptions for Web Presentations” on page 189 for more information.

• the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.

• the description and the properties for the output in the Results window.

• the description and properties for the catalog entry in the Explorer.

• the Description field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default Based on whether a SYMBOL definition \(<n>\) is specified in the plot request, either PLOT OF chart-variable by chart-variable or PLOT OF chart-variable by chart-variable identified by chart-variable

FRAME | NOFRAME

specifies whether a line is drawn around the axis area. The default is FRAME. If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

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 color, use the CFRAME= option.

To fill the axis area with an image, use the IFRAME= 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 horizontal reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOHREF and HREF= options.
The way you specify value-list depends on the type of variable:

- For numeric variables, value-list is either an explicit list of values, or a starting value and an ending value with an interval increment, or a combination of both forms:
  - \( n < ...n > \)
  - \( n \ TO \ n <BY \ increment> \)
  - \( n < ...n > \ TO \ n <BY \ increment > <n < ...n > > \)

If a numeric variable has an associated format, the specified values must be the unformatted values.

- For date-time values, value-list includes any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX, shown here as SAS-value:
  - '\SAS-value'i < ...'SAS-value'i>
  - '\SAS-value'i TO 'SAS-value' i<BY interval>

- For character variables, value-list is a list of unique character values enclosed in quotation marks and separated by blanks:
  - '\value-1' < ...'value-n'>

If a character variable has an associated format, the specified values must be the formatted values.

You are able to specify up to 256 characters for each variable value.

For a complete description of value-list, see “ORDER=(value-list)” on page 353 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 1140 for more information about values out of range.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the “AXIS Statement” on page 345.

Restriction: Partially supported by Java and ActiveX

Examples

- “Example 4: Plotting Two Variables” on page 1214
- “Example 5: Connecting Plot Data Points” on page 1217
- “Example 9: Plotting with Different Scales of Values” on page 1231

HMINOR=number-of-minor-ticks

specifies the number of minor tick marks drawn between each major tick mark on the horizontal axis. Minor tick marks are not labeled. The HMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias: HM=

Example

- “Example 2: Labeling and Sizing Plot Bubbles” on page 1208
HREF=\textit{value} | (\textit{value}) | (\textit{value-list})
draws one or more reference lines perpendicular to the horizontal axis at points that are specified by \textit{value-list}. For a description of \textit{value-list} see “HAXIS=\textit{value-list} | AXIS<1 … 99>” on page 1178 for the PLOT statement. To specify labels for these reference lines, use the HAXIS= option.

Values can be listed in any order, but should be within the range of values represented by the chart response axis. A warning is written to the SAS log if any of the points are off of the axis, and no reference line is drawn for such points. You can use the AUTOREF option to draw reference lines automatically at all of the major tick marks.

If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option.

The LHREF=, CHREF=, and WHREF= options can be used to change the line types, colors, and widths of these reference lines.

HREVERSE
specifies that the order of the values on the horizontal axis be reversed. For web output that is generated with a Java device driver, the horizontal axis data must be numeric. To specify line widths for these reference lines, use the WAUTOHREF= option.

Restriction Partially supported by Java and ActiveX

HTML=\textit{variable}
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

For Java and ActiveX, the PLOT statement is partially supported by the JAVA and ACTIVEX devices. It is fully supported by the JAVAIMG and ACTXIMG devices. The PLOT2 statement is supported only by the ACTXIMG device.

Restriction Partially supported by Java and ActiveX

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

HTML_LEGEND=\textit{variable}
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
HZERO
specifies that tick marks on the horizontal axis begin in the first position or end in the
last position with a value of zero, depending on the horizontal variable values. When
all horizontal variable values are positive, the first tick mark on the horizontal axis
has a value of zero. When all horizontal variable values are negative, the last tick
mark on the horizontal axis has a value of zero.

The HZERO request is ignored if the horizontal variable contains a mix of positive
and negative values. It is ignored when it has been ordered with the HAXIS= option
or the ORDER= option in an AXIS statement. It is also ignored if the horizontal
variable contains any time-formatted values, such as date, time, or datetime. In most
of these cases when the HZERO option is ignored, the last tick mark on the
horizontal axis has a value of zero.

IFRAME=fileref |'external-file'
identifies the image file that you want to apply to the axis area of the plot. fileref
must be a valid SAS fileref up to eight characters long and must have been
previously assigned with a FILENAME statement. external-file must specify the
complete filename of the image file that you want to use. The format of external-file
varies across operating environments.

Restriction Not supported by Java

Interactions This option is overridden by the NOIMAGEPRINT goption. For
information about the NOIMAGEPRINT goption, see
“IMAGEPRINT” on page 577.

This option is ignored if you specify the NOFRAME option or if you
specify the STYLE=0 option in the AXIS statement.

See the IMGAGESTYLE= option and “Displaying an Image in Graph
Frame” on page 334.

IMAGESTYLE=TILE | FIT
specifies whether to use multiple instances of the image to fill the axis area (TILE) or
to stretch a single instance of the image to fill the (FIT). The TILE value is the
default.

Restriction Not supported by Java

See “IFRAME=fileref |'external-file'” on page 1181

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.
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=LEGENDn to assign a defined LEGEND statement to the plot. The default legend is centered below the axis frame and identifies which colors and plot symbols represent the y-variables that you specify for the plots. To control the order of the legend entries for overlaid plots, use the ORDER= option in the LEGEND statement. Specify the list of variables in quotation marks in the preferred order. For example, the following causes the legend entry for y3 to be displayed first, y1 next, and y2 last:

```
legend1 order=('y3' 'y1' 'y2');
```

```
proc gplot data=mydata2;
plot (y1 y2 y3)*x / overlay  legend=legend1;
run;
```

For more information about the ORDER= option see “ORDER=(value-list) | DESCENDING” on page 383 in the LEGEND statement.

- 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=LEGENDn 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.

See “LEGEND Statement” on page 377

Example “Example 6: Generating an Overlay Plot” on page 1219

LHREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies line types for reference lines drawn perpendicular to the horizontal axis. The reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOHREF, HREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOHREF= option overrides the LHREF= option for lines drawn with the AUTOHREF option. Specifying in parentheses a single line type applies that line type only to the first reference line drawn with the HREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the HREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (type1 type2 ... typeN). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these reference lines, use the CHREF= option. To specify labels for these reference lines, use the HAXIS= option.

Alias LH=

Style reference LineStyle attribute of the GraphGridLines element

LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options. The reference-line-type value can be a whole number from 1 to 46. A value
of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn with the VREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (type1 type2 ... typeN). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

You might generate a needle plot with a Java or ActiveX device driver. In this case, the value of the LVREF= option is not applied to the default reference line that is drawn at zero. This occurs when the minimum value of the vertical axis is less than zero. This line is solid (not dashed).

Alias LV=

Style reference LineStyle attribute of the GraphGridLines element

Restriction Partially supported by Java and ActiveX

Example “Example 5: Connecting Plot Data Points” on page 1217

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- For the GRSEG entry name:
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAph appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- For the graphics output filename:
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
• If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.

• If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

• The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default

GPLOT

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

Alias NOAXES

NOLASTAREA

removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot. The last area is the area between the last plot line and the axis frame. You can use the NOLASTAREA option in PLOT and PLOT2 statements. However, when both PLOT and PLOT2 statements exist, the NOLASTAREA option is ignored in the PLOT statement. The NOLASTAREA option is also ignored in both PLOT and PLOT2 statements when the AREAS= option is not specified. It is also ignored when the number of bounded areas in the plot is less than or equal to the value specified by the AREAS= option.

Alias NOLASTFILL

Restriction Not supported by Java and ActiveX.

Note Certain conditions related to the PLOT2 statement might result in variations in the behavior of the NOLASTAREA option.

See “AREAS=n” on page 1173

NOLEGEND

suppresses the legend that is generated by a plot request of the type y-variable*x-variable=third-variable.

OVERLAY

places all the plots that are generated by the PLOT statement on one set of axes. The axes are scaled to include the minimum and maximum values of all of the variables. They also include the variable names or labels associated with the first pair of variables label the axes.

The OVERLAY option produces a legend if you include the LEGEND option or the LEGEND=n option in the PLOT statement.

OVERLAY is not enabled with plot requests of the form y-variable*x-variable=third-variable. However, you can achieve an overlay effect by using a PLOT and PLOT2 statement.
When generating output for the web with the JAVA, JAVAMETA, or JAVAIMG device drivers, the OVERLAY option cannot be used in the PLOT or 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 the INTERPOL= BOX, HILO, or STD, with or without the OVERLAY option.

**Restriction**  Partially supported by Java

**Examples**  “Example 6: Generating an Overlay Plot” on page 1219

“Example 7: Filling Areas in an Overlay Plot” on page 1223

**REGEQN**

displays the regression equation that is specified in the INTERPOL= option of the SYMBOL statement in the lower left corner of the plot. You cannot modify the format that is used for the equation.

The GPLOT regression equation is computed from the screen coordinates of the markers. Therefore, a graph might not be displayed if the chart area for the plot becomes so small that markers cannot be drawn. This is because there are no coordinates from which to build the regression equation. In such cases, the regression equation is no longer meaningful.

**Restriction**  Not supported by ActiveX

**Example**  “Example 4: Plotting Two Variables” on page 1214

**SKIPMISS**

breaks a plot line or an area fill at occurrences of missing values of the Y variable. By default, plot lines and area fills are not broken at missing values. The SKIPMISS option is available only with JOIN interpolation. If the SKIPMISS option is used, observations should be sorted by the independent (horizontal axis) variable. If the plot request is y-variable*x-variable=third-variable, observations should also be sorted by the values of the third variable.

**See**  “About the Input Data Set” on page 1140

**URL=character-variable**

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

**Restrictions**  This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the PLOT2 statement

**Interaction**  If you specify both the HTML= and URL= options, then the URL= option is ignored
You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

See

“Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

VAXIS=value-list | AXIS<1 ...99>
specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of value-list see “HAXIS=value-list | AXIS<1 ...99>” on page 1178 for the PLOT statement. To assign labels to reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the “AXIS Statement” on page 345.

Restriction

Partially supported by Java and ActiveX

Examples

“Example 4: Plotting Two Variables” on page 1214

“Example 5: Connecting Plot Data Points” on page 1217

VMINOR=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

Alias

VM=

Example

“Example 2: Labeling and Sizing Plot Bubbles” on page 1208

VREF=value | (value) | (value-list)
draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of value-list see “HAXIS=value-list | AXIS<1 ...99>” on page 1178 for the PLOT statement. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

Example

“Example 5: Connecting Plot Data Points” on page 1217

VREVERSE
specifies that the order of the values on the vertical axis be reversed.

VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values. When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when the vertical variable is ordered with the VAXIS= option.
option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

**WAUTOHREF=reference-line-width**

specifies the line width for all reference lines at major tick marks on the horizontal axis as determined by the AUTOHREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOHREF=reference-line-color” on page 1175.

Style reference  LineThickness attribute of the GraphGridLines element

**WAUTOVREF=reference-line-width**

specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOVREF=reference-line-color” on page 1175.

Style reference  LineThickness attribute of the GraphGridLines element

**WHREF=reference-line-width | (reference-line-width) | (reference-line-width-list)**

specifies line widths for reference lines as determined by the horizontal axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOHREF and HREF= options. Note that the WAUTOHREF= option overrides WHREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the HREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the HREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option “CHREF=reference-line-color | (reference-line-color) | (reference-line-color-list)” on page 1176.

Style reference  LineThickness attribute of the GraphGridLines element

**WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)**

specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option “CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)” on page 1177.

Style reference  LineThickness attribute of the GraphGridLines element
Details

Description
The PLOT statement specifies one or more plot requests that name the horizontal and left vertical axis variables, and can specify a third classification variable. This statement automatically does the following:

- scales the axes to include the maximum and minimum data values
- plots data points within the axes
- labels each axis with the name of its variable and displays each major tick mark value.

You can use statement options to manipulate the axes, modify the appearance of your graph, and describe catalog entries. You can use SYMBOL definitions to modify plot symbols for the data points, join data points, draw regression lines, plot confidence limits, or specify other types of interpolations. For more information about the SYMBOL statement, see “About SYMBOL Definitions” on page 1189.

In addition, you can use global statements to modify the axes; add titles, footnotes, and notes to the plot; or modify the legend if one is generated by the plot. You can also use an Annotate data set to enhance the plot.

Plot Requests with Multiple Variables
Plot requests with multiple variables produce a separate plot for every Y*X pair, unless you specify OVERLAY. For example, this statement produces four plots (the actual plots are produced on separate pages). See Figure 41.7 on page 1188.

plot (y b)*(x a);  

Figure 41.7 Graphs Generated by Multiple Plot Requests
About SYMBOL Definitions
SYMBOL statements control the appearance of plot symbols and lines, and define interpolation methods. They can specify the following:

- the shape, size, and color of the plot symbols that mark the data points
- plot line style, color, and width
- an interpolation method for plotting data
- how missing values are treated in interpolation calculations

SYMBOL definitions are assigned either by default by the GPLOT procedure or explicitly with a plot request.

If no SYMBOL definition is currently in effect, the GPLOT procedure produces a scatter plot of the data points using the default plot symbol. If you need more than one SYMBOL definition, the procedure rotates through the colors defined by the current style. If the NOGSTYLE option is specified, the procedure rotates through the device color list. If the current color list contains only one color, or if all the colors are used, additional plot symbols are used.

If SYMBOL definitions have been defined but not explicitly assigned by a plot request of the form y-variable*x-variable=n, the procedure assigns them in the order in which they are generated. For example, this statement creates three plots:

```
plot y*x b*a s*r;
```

The procedure assigns the first generated SYMBOL definition to Y*X, the second generated SYMBOL definition to B*A, and the third to S*R.

If more SYMBOL definitions are needed than have been defined, the procedure uses the default definitions for the plots that remain.

See the “SYMBOL Statement” on page 412.

About Plot Requests That Assign a SYMBOL Definition
Plot requests of the form y-variable*x-variable=n are useful when you use the OVERLAY option to produce multiple plots on one graph and you want to assign a particular SYMBOL definition to each plot.

With plot requests of this type it is important to remember that a single SYMBOL statement can generate multiple SYMBOL definitions, so that the SYMBOL definition that is designated by n might not be the same as the SYMBOL statement of the same number. That is, the third SYMBOL definition is not necessarily the same as the SYMBOL3 statement. For more information see the “SYMBOL Statement” on page 412.

PLOT2 Statement
Produces one or more plots with the vertical axis on the right side of the graph against which a second variable can be plotted.

**Requirement:**
You cannot use the PLOT2 statement alone. It can be used only with a PLOT or BUBBLE statement. At least one plot request is required.

**Global statements:**
AXIS, FOOTNOTE, LEGEND, PATTERN, SYMBOL, TITLE
Syntax

PLOT2 plot-request(s) <option(s)>;

Summary of Optional Arguments

Appearance options

ANNOTATE=Annotate-data-set
specifies a data set to annotate all graphs that are produced by the GPLOT procedure.

CAXIS=axis-color
specifies the color for the axis line and all major and minor tick marks.

CBASELINE=baseline-color
specifies the color of the horizontal baseline for reference lines of a plot.

CFRAME=background-color
fills the axis area with the specified color.

COUTLINE=outline-color
specifies the color of the outline that is drawn around filled areas.

CTEXT=text-color
specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions.

FRAME | NOFRAME
specifies whether a line is drawn around the axis area.

FRONTREF
specifies that reference lines drawn by the AUTOREF or REF= options should be drawn in front of the bars.

NOAXIS
suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

ODS options

HTML_LEGEND=variable
identifies the variable in the input data set whose values create links or data tips or both.

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both.

URL=character-variable
specifies a character variable whose values are URLs.

Plot options

AREAS=n
fills all the areas below plot line n with a pattern.

GRID
draws reference lines at all major tick marks on both axes.

LEGEND | LEGEND=LEGEND<1 ...99>
generates a legend or specifies the legend to use for the plot.

NOLASTAREA
removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot.

NOLEGEND
suppresses the legend that is generated by a plot request of the type \textit{y-variable}*\textit{x-variable}=\textit{third-variable}.

\textbf{OVERLAY}
places all the plots that are generated by the PLOT statement on one set of axes.

\textbf{REGEQN}
displays the regression equation that is specified in the \texttt{INTERPOL=} option of the SYMBOL statement in the lower left corner of the plot.

\textbf{SKIPMISS}
breaks a plot line or an area fill at occurrences of missing values of the \textit{Y} variable.

\textbf{Vertical axis options}

\textbf{AUTOVREF}
draws reference lines at all of the major tick marks on the vertical axis.

\texttt{CAUTOVREF}=reference-line-color
specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the \texttt{AUTOVREF} option.

\texttt{CVREF}=reference-line-color | (reference-line-color) | (reference-line-color-list)
specifies the color of reference lines drawn perpendicular to the vertical axis.

\texttt{LAUTOVREF}=reference-line-type
specifies a line type for reference lines drawn at major tick marks on the vertical axis, as specified by the \texttt{AUTOVREF} option.

\texttt{LVREF}=reference-line-type | (reference-line-type) | (reference-line-type-list)
specifies the type of reference lines drawn with the \texttt{AUTOVREF}, \texttt{VREF}, and \texttt{GRID} options.

\texttt{V AXIS}=value-list | AXIS<1 ..99>
specifies the major tick mark values for the vertical axis or assigns an axis definition.

\texttt{VMINOR}=number-of-minor-ticks
specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis.

\texttt{VREF}=value | (value) | (value-list)
draws one or more reference lines perpendicular to the vertical axis at specified points.

\texttt{VREVERSE}
specifies that the order of the values on the vertical axis be reversed.

\texttt{VZERO}
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values.

\texttt{WAUTOVREF}=reference-line-width
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the \texttt{AUTOVREF} option.

\texttt{WVREF}=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis.
**Required Argument**

*plot-request(s)*

each specifies the variables to plot and produces a separate graph, unless you specify the OVERLAY option. All variables must be in the input data set. Multiple plot requests are separated with blanks. A plot request can be any of these:

- \( \text{y-variable}*\text{x-variable}<=n> \)
  plots the values of two variables and can assign a SYMBOL definition to the plot.
  - \( \text{y-variable} \)
    variable plotted on the right vertical axis.
  - \( \text{x-variable} \)
    variable plotted on the horizontal axis.
  - \( n \)
    number of the \( n \)th generated SYMBOL definition.

- \( (\text{y-variable(s)})*(\text{x-variable(s)}) \)
  plots the values of two or more variable and produces a separate graph for each combination of Y and X variables.
  - \( \text{y-variable(s)} \)
    variables plotted on the right vertical axes.
  - \( \text{x-variable(s)} \)
    variables plotted on the horizontal axes.

- \( \text{y-variable}*\text{x-variable}=\text{third-variable} \)
  plots the values of two variables against a third classification variable
  - \( \text{y-variable} \)
    variable plotted on the right vertical axis.
  - \( \text{x-variable} \)
    variable plotted on the horizontal axis.
  - \( \text{third-variable} \)
    classification variable against which \( \text{y-variable} \) and \( \text{x-variable} \) 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 1168.

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 \( \text{y-variable} \) on the right vertical axis.

**Optional Arguments**

ANNOTATE=Annotate-data-set

specifies a data set to annotate all graphs that are produced by the GPLOT procedure. To annotate individual graphs created using a By statement or multiple action statements, use ANNOTATE= in the action statement.
AREAS=n
fills all the areas below plot line n with a pattern. The value of n specifies which
areas to fill:

• AREAS=1 fills the first area.
• AREAS=2 fills both the first and second areas, and so on.

You might specify a value for the AREAS= option that is greater than the number of
bounded areas in the plot. In this case, the area between the top plot line and the axis
frame is filled.

Before an area can be filled, the data points that border the area must be joined by a
line. Use a SYMBOL statement with one of these interpolation methods to join the
data points:

• INTERPOL=JOIN
• INTERPOL=STEP
• INTERPOL=R.series
• INTERPOL=SPLINE | SM | L

See the “SYMBOL Statement” on page 412 for details about interpolation methods.

By default, the AREAS= option fills areas by rotating a solid fill through the list of
colors defined in the current style. If the NOGSTYLE option is specified, the areas
are filled by rotating a solid fill through the device's color list. If the graph needs
more patterns, it rotates hatch patterns, beginning with the M2N0 pattern. See the
“PATTERN Statement” on page 398 for more information about map and plot
patterns. However, if color is limited to a single color with the CPATTERN= or
COLORS= graphic options, the solid pattern is skipped and the first default pattern is
M2N0. If the COLORS= graphic option specifies a single color, use as many
SYMBOL statements as you have areas to fill. You must do this because the
INTERPOL= setting does not automatically apply to multiple symbol definitions.

You can alter the default pattern behavior by specifying patterns and colors in
PATTERN statements that specify map and plot patterns. A separate PATTERN
definition is needed for each specified area.

If you specify the PATTERN statements, the AREAS= option uses the lowest
numbered PATTERN statement first. If it runs out of patterns, it uses the default
behavior for map and plot patterns. See the “PATTERN Statement” on page 398 for
details.

Pattern definitions are assigned to the areas below the plot lines in the order the plots
are drawn. The first area is that between the horizontal axis and the plot line that is
drawn first. The second area is that above the first plot line and below the plot line
that is drawn second, and so on. If the line that is drawn second lies below the line
that is drawn first, the second area is hidden when the first is filled. The plots with
the lower line values must be drawn first to prevent one area fill from overlaying
another. If the lines cross, only the part of an area that is above the previous line is
visible.

Therefore, you must order your PLOT statements when creating multiple plots in
combination with the OVERLAY option. Order them so that the plot request that
produces the lowest line value is first (leftmost). The plot request that produces the next lowest line value is the second plot request, and so on.

If you produce multiple plots with a y-variable*x-variable=third-variable plot request, the lines are plotted in order of increasing third variable values. In that case, you must recode your data so that the lowest value of the third variable produces the lowest plot line. Likewise, the next lowest value produces the next lowest plot line, and so on.

If you use the VALUE= option in the SYMBOL statement, some symbols might be hidden. If you also specify reference lines with the AREAS= option, they are drawn behind the pattern fill.

You might use the AREAS= option to fill the areas between your plots while using the JAVA device to generate the plot. In this case, when you move your mouse pointer over a filled area in the plot output, a tooltip is displayed. It shows the values for the last observation in the data set for the group that the area represents. However, when you position your mouse pointer over a data point on a plot line, a tooltip shows the value for that point. This might confuse your viewers. To help avoid confusion in that case, you can turn off the tooltips by specifying the following on your ODS statement:

```ods parameters= ("Tips"= "NONE")
```

**Restriction**

Partially supported by Java

**Notes**

The AREAS= option works only if all plot lines are generated by the same PLOT or PLOT2 statement.

If you have specified the NOGSTYLE option and the first color in your device's default color list is black, color rotation begins with the second color in the list. That is, there are no solid black patterns. See “How Default Patterns and Outlines Are Generated” on page 407 for more information.

**Example**

“Example 7: Filling Areas in an Overlay Plot” on page 1223

**AUTOVREF**

draws reference lines at all of the major tick marks on the vertical axis. If you also use the AREAS= option, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option in either the PROC GPLOT statement or the PLOT statement. LAUTOVREF=, CAUTOVREF=, and WAUTOVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

**CAUTOVREF=reference-line-color**

specifies the color of reference lines drawn at major tick marks on the vertical axis, as determined by the AUTOVREF option. If you do not specify the CAUTOVREF option, the default color is the value of the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

**CAXIS=axis-color**

specifies the color for the axis line and all major and minor tick marks. The default color is retrieved from the current style or from the device's color list if the NOGSTYLE option is specified.

**Alias**

CA=
CBASELINE=baseline-color

specifies the color of the horizontal baseline for reference lines of a plot. This option compensates for a reference line obscuring the axis. This can occur when using a combination of the AUTOVREF option, the CAUTOVREF= option, or the CVREF= option. They must be specifying any color other than black, and a vertical axis with no offset. For example, the light gray reference line drawn by the AUTOVREF option and a CVREF= graydd specification obscures the default black color of the axis line. This option enables you to set the color of the baseline to match the axis color.

Specify the baseline-color as a name or a quoted string. Two examples are presented. The first indicates a name. The second specifies an RGB color code:

```
cbaseline=black
cb='CX0BB5FF'
```

Alias CB=

Default The value of the CAXIS= option is the default color if you do not specify an alternate color for the baseline. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Restriction This option is not supported by the ACTIVEX, ACTXIMG, JAVA, or JAVA IMG devices.

Note The baseline is drawn after the plot elements are drawn, and either a default or specified color is applied. This means that the reference lines are drawn behind plot elements such as bars. Use the FRONTREF option if you want reference lines drawn in front of the bars.

See “Specifying Colors in SAS/GRAPH Programs” on page 314

“FRONTREF” on page 1197

CFRAME=background-color

fills the axis area with the specified color.

Style reference Color attribute of the GraphWalls element

Restriction CFRAME= is overridden by the NOFRAME and IFRAME= options. If the IFRAME= option is in effect, an image appears in the axis area instead of the color.

Interaction If the FRAME option is also in effect, the procedure behavior is modified. The procedure determines the color of the frame around the axis area according to the precedence list given within the FRAME option description.

COU TLINE=outline-color

specifies the color of the outline that is drawn around filled areas. The filled areas are generated when the SYMBOL statement specifies "INTERPOL=map/plot-pattern" on page 421 or the GOPTIONS statement specifies "INTERPOL" on page 579. The default outline color is specified in the current style. However, the NOGSTYLE option might be specified. In that case, the default color is the first color in the device's color list (the foreground color). Also, the default slice outline color is determined as follows:
• If you do not specify a PATTERN statement, the default outline color is the color defined in the current style.

• If you specify the NOGSTYLE option and no PATTERN statement, the default outline color is black for the Java or ActiveX devices. Otherwise, the default outline color is the foreground color. If you specify an EMPTY PATTERN statement, then the default outline color is the same as the fill color.

**Alias**

**CO=**

**Style reference**

Color attribute of the GraphOutlines element

**Restriction**

Not supported by Java

**Interaction**

The COUTLINE= option overrides the SYMBOL statement option CO=.

**CTEXT=text-color**

specifies a color for all text on the axes and legend, including axis labels, tick mark values, legend labels, and legend value descriptions. The GPLOT procedure searches for a color specification in this order:

1. colors specified for labels and values on assigned AXIS and LEGEND statements. They override the CTEXT= option specified in the PLOT statement. This also includes colors specified by the POINTLABEL= option in the SYMBOL statement.

2. the color specified by the CTEXT= option in the PLOT statement

3. the color specified by the CTEXT= option in the GOPTIONS statement

4. the color specified in the current style, or a NOGSTYLE option specification. With NOGSTYLE specified, the default color is black for the Java and ActiveX devices and the first color in the color list for all other devices

The LEGEND statement's VALUE= color is used for legend values, and its LABEL= color is used for legend labels.

The AXIS statement's VALUE= color is used for axis values, and its LABEL= color is used for axis labels. However, if the AXIS statement specifies only general axis colors with its COLOR= option, the CTEXT= color overrides the general COLOR= specification and is used for axis labels and values. The COLOR= color is still used for all other axis colors, such as tick marks.

**Alias**

**C=**

**Style reference**

Color attributes of the GraphValueText and the GraphLabelText elements

**Note**

If you use a BY statement in the procedure, the color of the BY variable labels is controlled by the CBY= option in the GOPTIONS statement.

**CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)**

specifies the color of reference lines drawn perpendicular to the vertical axis. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single color applies that color to all reference lines. The CAUTOVREF= option overrides the CVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single color applies that color only to the first reference line drawn with the VREF= option. Specifying a
color list applies colors sequentially to successive reference lines drawn with the VREF= option. The syntax of the color list requires parentheses and colors separated by spaces (color1 color2 ... colorN). If you do not specify the CVREF= option, the GPLOT procedure uses the color specified by the CAXIS= option. If neither option is specified, the default color is retrieved from the current style or from the first color in the color list if the NOGSTYLE option is specified.

Alias CV=

Style reference Color attribute of the GraphGridLines element

FRAME | NOFRAME
specifies whether a line is drawn around the axis area. The default is FRAME. If you also use a BUBBLE2 or PLOT2 statement and these plotting statements have conflicting frame specifications, FRAME is used as opposed to NOFRAME.

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 color, use the CFRAME= option.
To fill the axis area with an image, use the IFRAME= 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.

HTML=variable
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

For Java and ActiveX, the PLOT statement is partially supported by the JAVA and ACTIVEX devices. It is fully supported by the JAVA IMG and ACTX IMG devices. The PLOT2 statement is supported only by the ACTX IMG device.

Restriction Partially supported by Java and ActiveX

See “Data Tips for Web Presentations” on page 191

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
**HTML_LEGEND=variable**

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

**Restriction**

Not supported by Java and ActiveX

**See**

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

**Example**

“Example 10: Creating Plots with Drill-down Functionality for the Web” on page 1235

**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=LEGENDn to assign a defined LEGEND statement to the plot. The default legend is centered below the axis frame and identifies which colors and plot symbols represent the y-variables that you specify for the plots. To control the order of the legend entries for overlaid plots, use the ORDER= option in the LEGEND statement. Specify the list of variables in quotation marks in the preferred order. For example, the following causes the legend entry for y3 to be displayed first, y1 next, and y2 last:

```plaintext
legend1 order=('y3' 'y1' 'y2');
proc gplot data=mydata2;
plot (y1 y2 y3)*x / overlay legend=legend1;
run;
```

For more information about the ORDER= option see “ORDER=(value-list) | DESCENDING” on page 383 in the LEGEND statement.

- 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=LEGENDn 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.

**See**

“LEGEND Statement” on page 377

**Example**

“Example 6: Generating an Overlay Plot” on page 1219
LVREF=reference-line-type | (reference-line-type) | (reference-line-type-list)

specifies the type of reference lines drawn with the AUTOVREF, VREF, and GRID options. The reference-line-type value can be a whole number from 1 to 46. A value of 1 specifies a solid line; values 2 through 46 specify dashed lines. This option affects reference lines drawn with the AUTOVREF, VREF, and GRID options. Specifying without parentheses a single line type applies that line type to all reference lines. The LAUTOVREF= option overrides the LVREF= option for lines drawn with the AUTOVREF option. Specifying in parentheses a single line type applies that line type only to the first line drawn with the VREF= option. Specifying a line-type list applies line types in sequence to successive reference lines drawn with the VREF= option. The syntax of the line type list requires parentheses and line types separated by spaces (type1 type2 ... typeN). The default line type is retrieved from the current style, or if the NOGSTYLE option is specified, the default value is 1, which draws a solid line. To specify colors for these references lines, use the CVREF= option. To specify labels for these reference lines, use the VAXIS= option.

You might generate a needle plot with a Java or ActiveX device driver. In this case, the value of the LVREF= option is not applied to the default reference line that is drawn at zero. This occurs when the minimum value of the vertical axis is less than zero. This line is solid (not dashed).

<table>
<thead>
<tr>
<th>Alias</th>
<th>LV=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style reference</td>
<td>LineStyle attribute of the GraphGridLines element</td>
</tr>
<tr>
<td>Restriction</td>
<td>Partially supported by Java and ActiveX</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 5: Connecting Plot Data Points” on page 1217</td>
</tr>
</tbody>
</table>

NOAXIS

suppresses the axes, including axis lines, axis labels, all major and minor tick marks, and tick mark values.

<table>
<thead>
<tr>
<th>Alias</th>
<th>NOAXES</th>
</tr>
</thead>
</table>

NOLASTAREA

removes the last area of a plot when the value specified by the AREAS= option exceeds the number of bounded areas in the plot. The last area is the area between the last plot line and the axis frame. You can use the NOLASTAREA option in PLOT and PLOT2 statements. However, when both PLOT and PLOT2 statements exist, the NOLASTAREA option is ignored in the PLOT statement. The NOLASTAREA option is also ignored in both PLOT and PLOT2 statements when the AREAS= option is not specified. It is also ignored when the number of bounded areas in the plot is less than or equal to the value specified by the AREAS= option.

<table>
<thead>
<tr>
<th>Alias</th>
<th>NOLASTFILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>Not supported by Java and ActiveX.</td>
</tr>
<tr>
<td>Note</td>
<td>Certain conditions related to the PLOT2 statement might result in variations in the behavior of the NOLASTAREA option.</td>
</tr>
<tr>
<td>See</td>
<td>“AREAS=n” on page 1193</td>
</tr>
</tbody>
</table>

NOLEGEND

suppresses the legend that is generated by a plot request of the type y-variable*x-variable=third-variable.
OVERLAY
places all the plots that are generated by the PLOT statement on one set of axes. The
axes are scaled to include the minimum and maximum values of all of the variables.
They also include the variable names or labels associated with the first pair of
variables label the axes.

The OVERLAY option produces a legend if you include the LEGEND option or the
LEGEND=n option in the PLOT statement.

OVERLAY is not enabled with plot requests of the form y-variable*x-variable=third-variable. However, you can achieve an overlay effect by using a
PLOT and PLOT2 statement.

When generating output for the web with the JAVA, JAVAMETA, or JAVAIMG
device drivers, the OVERLAY option cannot be used in the PLOT or 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 the INTERPOL= BOX, HILO, or STD, with or
without the OVERLAY option.

Restriction Partially supported by Java

Examples “Example 6: Generating an Overlay Plot” on page 1219
“Example 7: Filling Areas in an Overlay Plot” on page 1223

REGEQN
displays the regression equation that is specified in the INTERPOL= option of the
SYMBOL statement in the lower left corner of the plot. You cannot modify the
format that is used for the equation.

The GPLOT regression equation is computed from the screen coordinates of the
markers. Therefore, a graph might not be displayed if the chart area for the plot
becomes so small that markers cannot be drawn. This is because there are no
coordinates from which to build the regression equation. In such cases, the
regression equation is no longer meaningful.

Restriction Not supported by ActiveX

Example “Example 4: Plotting Two Variables” on page 1214

SKIPMISS
breaks a plot line or an area fill at occurrences of missing values of the Y variable.
By default, plot lines and area fills are not broken at missing values. The SKIPMISS
option is available only with JOIN interpolation. If the SKIPMISS option is used,
observations should be sorted by the independent (horizontal axis) variable. If the
plot request is y-variable*x-variable=third-variable, observations should also be
sorted by the values of the third variable.

See “About the Input Data Set” on page 1140

URL=character-variable
specifies a character variable whose values are URLs. The variable values are URLs
for web pages to display when the user clicks (drills down) on elements in the graph.
### Restrictions
This option affects graphics output that is created through the ODS HTML destination only

Not supported by the JAVAIMG and ACTXIMG devices when using the PLOT2 statement

### Interaction
If you specify both the HTML= and URL= options, then the URL= option is ignored

### Note
You can specify this option starting with SAS 9.4M1. The ODS URL= option is supported by the Java device in the BUBBLE2 and PLOT2 statements.

### See
“Overview of Enhancing Web Presentations” on page 188

“Example: GIF Output with Drill-Down Links” on page 163

---

#### VAXIS=value-list | AXIS<1 ...99>

specifies the major tick mark values for the vertical axis or assigns an axis definition. For a description of value-list see “HAXIS=value-list | AXIS<1 ...99>” on page 1178 for the PLOT statement. To assign labels to reference lines, use an axis definition that contains the REFLABEL= option. The labels are applied in sequence to all reference lines defined with the AUTOVREF and VREF= options.

For web output that is generated with a Java or ActiveX device driver, certain options of the AXIS statement are not supported. For details, see the “AXIS Statement” on page 345.

### Restriction
Partially supported by Java and ActiveX

### Examples
“Example 4: Plotting Two Variables” on page 1214

“Example 5: Connecting Plot Data Points” on page 1217

---

#### VMINOR=number-of-minor-ticks

specifies the number of minor tick marks that are drawn between each major tick mark on the vertical axis. Minor tick marks are not labeled. The VMINOR= option overrides the NUMBER= suboption of the MINOR= option in an AXIS definition. You must specify a positive number.

### Alias
VM=

### Example
“Example 2: Labeling and Sizing Plot Bubbles” on page 1208

---

#### VREF=value | (value) | (value-list)

draws one or more reference lines perpendicular to the vertical axis at specified points. For a description of value-list see “HAXIS=value-list | AXIS<1 ...99>” on page 1178 for the PLOT statement. If the AREAS= option is also used, the filled areas cover the reference lines. To draw lines on top of the filled areas, use the FRONTREF option. LVREF=, CVREF=, and WVREF= options can be used to change the line types, colors, and widths of these reference lines. To specify labels for these reference lines, use the VAXIS= option.

### Example
“Example 5: Connecting Plot Data Points” on page 1217

---

#### VREVERSE

specifies that the order of the values on the vertical axis be reversed.
VZERO
specifies that tick marks on the vertical axis begin in the first position or end in the last position with a value of zero, depending on the vertical variable values. When all vertical variable values are positive, the first tick mark on the vertical axis has a value of zero. When all vertical variable values are negative, the last tick mark on the vertical axis has a value of zero.

The VZERO request is ignored if the vertical variable contains a mix of positive and negative values. It is ignored when the vertical variable is ordered with the VAXIS= option or the ORDER= option in an AXIS statement. It is also ignored if the vertical variable contains any time-formatted values, such as date, time, or datetime. In most of these cases when the VZERO option is ignored, the last tick mark on the vertical axis has a value of zero.

WAUTOVREF=reference-line-width
specifies the line width for all reference lines at major tick marks on the vertical axis as determined by the AUTOVREF option. Line widths are specified as whole numbers with the default value being 1. To specify a color for these reference lines, use the option “CAUTOVREF=reference-line-color” on page 1194.

WVREF=reference-line-width | (reference-line-width) | (reference-line-width-list)
specifies line widths for reference lines as determined by the vertical axis. Line widths are specified as whole numbers. Specifying without parentheses a line width applies that line width to all reference lines drawn with the AUTOVREF and VREF= options. Note that the WAUTOVREF= option overrides WVREF=reference-line-width for reference lines drawn with the AUTOREF option. Specifying in parentheses a single line width applies that line width to the first reference line drawn with the VREF= option. Specifying a line width list applies line widths in sequence to successive reference lines drawn with the VREF= option. The syntax of the line-width list requires parentheses and line widths separated by spaces (width1 width2 ...widthN). The default line width is specified by the current style or by the AXIS statement's WIDTH= option. (By default, WIDTH=1.) To specify a color for these reference lines, use the option “CVREF=reference-line-color | (reference-line-color) | (reference-line-color-list)” on page 1196.

Details

Options
Options for the PLOT2 statement are identical to the options for the PLOT statement except for these options, which are ignored if you specify them:

- AUTOHREF
- CAUTOHREF=
- CHREF=
- DESCRIPTION=
- HAXIS=
- HMINOR=
- HREF=
- HREVERSE=
Description
The PLOT2 statement specifies one or more plot requests that name the horizontal and right vertical axis variables. This statement automatically does the following:

- plots data points within the axes
- scales the axes to include the maximum and minimum data values
- labels each axis with the name of its variable and displays each major tick mark value

You can use statement options to manipulate the axes and modify the appearance of your graph. You can use SYMBOL definitions to modify plot symbols for the data points, join data points, draw regression lines, plot confidence limits, or specify other types of interpolation. For more information about the SYMBOL statement see “About SYMBOL Definitions” on page 1189.

Note: You might be using the PLOT2 statement to generate output with the Java or ACTIVEX device drivers. If you are also using the global statement SYMBOL, the value of the SYMBOL statement option INTERPOL= cannot be BOX, STD, or HILO.

In addition, you can use global statements to modify the axes; to add titles, footnotes, and notes to the plot; or to modify the legend if one is generated by the plot. You can also use an Annotate data set to enhance the plot.

Matching Plot Requests

Matching Requests between PLOT and PLOT2 Statements
The plot requests in both the PLOT and PLOT2 statements must be evenly matched as in this example:

```
plot y*x b*a;
    plot2 y2*x b2*a;
```

These statements produce two graphs, and each one has two vertical axes. The first pair of plot requests (Y*X and Y2*X) produce one graph. In this graph X is plotted on the horizontal axis, Y is plotted on the left axis, and Y2 is plotted on the right axis. The second pair of plot requests (B*A and B2*A) produce another graph. In that graph A is plotted on the horizontal axis, B is plotted on the left axis, and B2 is plotted on the right axis.
Using Multiple Plot Requests
Plot requests of the form \((y\text{-variable}(s))\ast(x\text{-variable}(s))\). Both the PLOT and PLOT2 statements generate multiple graphs (the actual plots are produced on separate pages). See Figure 41.8 on page 1204.

\[
\begin{align*}
\text{plot} & \ (y \ b) \ast (x \ a); \\
\text{plot2} & \ (y2 \ b2) \ast (x \ a); \\
\end{align*}
\]

Figure 41.8  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 \(y\text{-variable}\ast x\text{-variable}=\text{third-variable}\), 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 41.9 on page 1205. 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.

\[
\begin{align*}
\text{plot} & \ y \ast x=z; \\
\text{plot2} & \ y2 \ast x=z; \\
\end{align*}
\]
Using a Second Vertical Axis

Displaying the Same Values in a Different Scale
If your data contain the same variable values in two different scales, such as height in inches and height in centimeters, you can display one scale of values on the left axis and the other scale of values on the right axis. Both vertical axes might be calibrated so that they represent the same range of values. In this case, for each observation of X, the data points for Y and Y2 are the same.

For example, Y is height in inches and Y2 is height in centimeters. If the Y axis values range from 0 to 84 inches and the Y2 axis values range from 0 to 213.36 centimeters, the plot is displayed like Figure 41.10 on page 1205.

For these types of plots, the PLOT2 statement should use a SYMBOL statement that specifies INTERPOL=NONE and VALUE=NONE.

Displaying Different Values
Your data might contain variables with different data values (such as height and weight). In this case you can display one type of data on the left axis and another type of data on the right axis. Because the Y variable and the Y2 variable contain different data, two sets of data points are displayed on the graph. For example, if Y is height and Y2 is weight, the plot is like Figure 41.11 on page 1206.
Your data might contain two sets of values for the same type of data. In this case you can use the PLOT2 statement to generate a right axis that is calibrated the same as the left axis. This makes the data points on the right of the graph easier to read. For example, if Y is high temperatures and Y2 is low temperatures, you can create a graph like Figure 41.12 on page 1206.

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 1189.

Examples: GPLOT Procedure

Example 1: Generating a Simple Bubble Plot

Features:  BUBBLE statement option
This example shows a bubble plot in which each bubble represents a category of engineer. The plot shows engineers on the horizontal axis and average salaries on the vertical axis. Each bubble's vertical location is determined by the average salary for the category. Each bubble's size is determined by the number of engineers in the category: the more engineers, the larger the bubble.

**Program**

```sas
goptions reset=all border;

data jobs;
  length eng $5;
  input eng dollars num;
  datalines;
  Civil 27308 73273
  Aero  29844 70192
  Elec  22920 89382
  Mech  32816 19601
  Chem  28116 25541
  Petro 18444 34833
; 
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
```
axis1 offset=(5,5);
proc gplot data=jobs;
   format dollars dollar9.;
   bubble dollars*eng=num / haxis=axis1;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

data jobs;
   length eng $5;
   input eng dollars num;
datalines;
Civil  27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;

Define titles.

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";

Define axis characteristics. The OFFSET= option specifies an offset for the tick marks so that bubbles near an axis are not clipped.

axis1 offset=(5,5);

Generate bubble plot. The HAXIS= option assigns the AXIS1 statement to the horizontal axis. The salary averages are assigned a dollar format.

proc gplot data=jobs;
   format dollars dollar9.;
   bubble dollars*eng=num / haxis=axis1;
run;
quit;

Example 2: Labeling and Sizing Plot Bubbles

Features: BUBBLE statement options
       BCOLOR
       BLABEL
       BSIZE
       HAXIS=
       VAXIS=
Example 2: Labeling and Sizing Plot Bubbles

This example modifies the code in Example 1. It shows how BUBBLE statement options control the appearance of bubbles and their labels. It also shows how AXIS statements can modify the plot axes.

Program

goptions reset=all border;

data jobs;
  length eng $5;
  input eng dollars num;
  datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;
title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
axis1 label=none
   offset=(5,5);
axis2 order=(0 to 40000 by 10000)
   label=none;
proc gplot data=jobs;
   format dollars dollar9. num comma7.0;
   bubble dollars*eng=num / haxis=axis1
   vaxis=axis2
   vminor=1
   bcolor=darkred
   blabel
   bsize=3;
run;
quit;

Program Description

Set the graphics environment.
   goptions reset=all border;

Create the data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero 29844 70192
Elec 22920 89382
Mech 32816 19601
Chem 28116 25541
Petro 18444 34833
;

Define titles.
   title1 "Member Profile";
   title2 "Salaries and Number of Member Engineers";

Define axis characteristics. AXIS1 suppresses the horizontal axis label and uses the OFFSET= option to move the first and last major tick mark values away from the vertical axes. This ensures that the bubbles are not clipped. AXIS2 uses the ORDER= option to set major tick mark intervals. This could be done with the VAXIS= option in the BUBBLE statement, but then you could not suppress the axis label and alter other axis characteristics.

   axis1 label=none
      offset=(5,5);
   axis2 order=(0 to 40000 by 10000)
      label=none;
Generate bubble plot. The VMINOR= option specifies one minor tick mark for the vertical axis. The BLABEL option labels each bubble with the value of variable NUM. Then BCOLOR= option specifies the color for the bubbles. The BLABEL option labels the bubbles with the value of the third variable, which in this case is the number of engineers in the job category. The BSIZE option specifies the size of the bubbles.

```
proc gplot data=jobs;
  format dollars dollar9. num comma7.0;
  bubble dollars*eng=num / haxis=axis1
                 vaxis=axis2
                 vminor=1
                 bcolor=darkred
                 blabel
                 bsize=3;
  run;
  quit;
```

Example 3: Adding a Right Vertical Axis

**Features:**
- BUBBLE statement options
  - VAXIS=
  - HAXIS=
  - HMINOR=
  - VMINOR=
  - BLABEL

**Other features:**
- AXIS statement
- FORMAT statement
- GOPTIONS statement option
- BORDER

**Sample library member:**
- GPLAXIS1

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

This example modifies “Example 2: Labeling and Sizing Plot Bubbles” on page 1208 to show how a BUBBLE2 statement generates a right vertical axis. This axis displays the values of the vertical coordinates in a different scale from the scale that is used for the left vertical axis. Salary values are scaled by dollars on the left vertical axis and by yen on the right vertical axis.

BUBBLE and BUBBLE2 statement options control the appearance of the graph. In particular, the VAXIS options calibrate the axes so that the data points are identical and only one set of bubbles appears.

**Note:** If the data points are not identical, two sets of bubbles are displayed.
Program

goptions reset=all border;

data jobs;
   length eng $5;
   input eng dollars num;
   datalines;
Civil 27308 73273
Aero  29844 70192
Elec  22920 89382
Mech  32816 19601
Chem  28116 25541
Petro 18444 34833
;

data jobs2;
   set jobs;
   yen=dollars*125;
run;

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";
axis1 label=none
   offset=(5,5);
proc gplot data=jobs2;
   format dollars dollar7. yen comma9.0;
   bubble dollars*eng=num / haxis=axis1
      vaxis=10000 to 40000 by 10000
    hminor=0
    vminor=1
    blabel;
bubble2 yen*eng=num / vaxis=1250000 to 5000000 by 1250000
vminor=1;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the JOBS data set. The data set JOBS contains average salary data for several categories of engineer. It also indicates the number of engineers in each category.

data jobs;
  length eng $5;
  input eng dollars num;
  datalines;
  Civil  27308 73273
  Aero  29844 70192
  Elec  22920 89382
  Mech  32816 19601
  Chem  28116 25541
  Petro 18444 34833
  ;

Create the data set JOBS2 and calculate variable YEN. The DATA step uses a SET statement to read the JOBS data set.

data jobs2;
  set jobs;
  yen=dollars*125;
  run;

Define titles.

title1 "Member Profile";
title2 "Salaries and Number of Member Engineers";

Define horizontal-axis characteristics.

  axis1 label=none
  offset=(5,5);

Generate bubble plot with second vertical axis. In the BUBBLE statement, the HAXIS= option specifies the AXIS1 definition and the VAXIS= option scales the left axis. In the BUBBLE2 statement, the VAXIS= option scales the right axis. Both axes represent the same range of monetary values. The BUBBLE and BUBBLE2 statements ensure that the bubbles generated by each statement are identical by coordinating specifications on any options in these statements.

proc gplot data=jobs2;
  format dollars dollar7. num yen comma9.0;
  bubble dollars*eng=num / haxis=axis1
    vaxis=10000 to 40000 by 10000
    hminor=0
    vminor=1
  ;
Example 4: Plotting Two Variables

Features: PLOT statement options
- HAXIS=
- HMINOR=
- REGEQN
- VAXIS=

Other features: GOPTIONS statement option
- BORDER

SYMBOL statement

Sample library member: GPLVRBL1

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

In this example, the PLOT statement uses a plot request of the type y-variable*x-variable to plot the variable HEIGHT against the variable WEIGHT. The plot shows that weight generally increases with size.

The second example then requests the same plot with some modifications. As shown by the second output display, the second plot request specifies a regression analysis with confidence limits, and scales the range of values along the vertical and horizontal axes. It also displays the regression equation specified for the SYMBOL statement. Because the procedure supports RUN-group processing, you do not have to repeat the PROC GPLOT statement to generate the second plot.
Program

goptions reset=all border;

  title "Study of Height vs Weight";
  footnote1 j=l "Source: T. Lewis & L. R. Taylor";
  footnote2 j=l "Introduction to Experimental Ecology";

  proc gplot data=sashelp.class;
    plot height*weight;

  proc gplot data=sashelp.class;
    plot height*weight;

Regression Equation:
  Height = 33.1875 + 0.419659*Weight - 0.00135*Weight^2 + 1.6726*Weight^3
run;
footnote1; /* this clears footnote1 and footnote2 */
symbol1 interpol=rcc1m95
  value=circle
  cv=darkred
  ci=black
  co=blue
  width=2;
  plot height*weight / haxis=45 to 155 by 10
                 vaxis=48 to 78 by 6
                 hminor=1
                 regeqn;
run;
quit;

Program Description

Set the graphics environment.
  goptions reset=all border;

Define title and footnotes.
  title "Study of Height vs Weight";
  footnote1 j=l "Source: T. Lewis & L. R. Taylor";
  footnote2 j=l "Introduction to Experimental Ecology";

Generate a default scatter plot.
  proc gplot data=sashelp.class;
    plot height*weight;
  run;

Clear the footnotes to make room for the regression equation.
  footnote1; /* this clears footnote1 and footnote2 */

Define symbol characteristics. The INTERPOL= option specifies a cubic regression analysis with confidence limits for mean predicted values. The VALUE=and CV= options specify a plot symbol and color. The CI=, CO=, and WIDTH= options specify colors and a thickness for the interpolation and confidence-limits lines.
  symbol1 interpol=rcc1m95
    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
Example 5: Connecting Plot Data Points

Features:  PLOT statement options
          HMINOR=
          LVREF=
          VAXIS=
          VMINOR=
          VREF=

Other features:  GOPTIONS statement option
                BORDER
                SYMBOL statement

Sample library member:  GPLDTPT1

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

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).

Source: 1997 World Almanac
Program

goptions reset=all border;

data stocks;
   input year high low @@;

datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  941.08  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 1555.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;

   title1 "Dow Jones Yearly Highs";
   footnote1 j=l "Source: 1997 World Almanac";
   symbol1 interpol=join
      value=dot;
   proc gplot data=stocks;
      plot high*year / haxis=1955 to 1995 by 5
         vaxis=0 to 6000 by 1000
         hminor=3
         vminor=1
         vref=1000 3000 5000
         lvref=2;
   run;
   quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones
Industrial Average and the dates of the high and low values each year.

data stocks;
   input year high low @@;
Define title and footnote.

title1 "Dow Jones Yearly Highs";
footnotel j=l "Source: 1997 World Almanac"

Define symbol characteristics. Specifying INTERPOL=JOIN joins the data points with straight lines and the VALUE= option specifies the type of symbol used.

symbol1 interpol=join
value=dot;

Generate the plot and modify the axis values. The VAXIS= option sets major tick marks for the vertical axis. The HMINOR= and VMINOR= options specify the number of tick marks between major tick marks.

proc gplot data=stocks;
plot high*year / haxis=1955 to 1995 by 5
vaxis=0 to 6000 by 1000
hminor=3
vminor=1
vref=1000 3000 5000
lvref=2;
run;
quit;

Example 6: Generating an Overlay Plot

Features: PLOT statement options
COLOR=
In this example, one PLOT statement plots both the HIGH and LOW variables against the variable YEAR using two plot requests. The OVERLAY option in the PLOT statement determines that both plot lines appear on the same graph. The other PLOT options scale the vertical axis, add a reference line to the plot, and specify the number of minor tick marks on the axes. The SYMBOL, AXIS, and LEGEND statements modify the plot symbols, axes, and legend.

**Note:** If the OVERLAY option is not specified, each plot request generates a separate graph.
Program

goptions reset=all border;

data stocks;
    input year high low @@;

datalines;
1956  521.05  462.35  1957  520.77  419.79
1958  583.65  436.89  1959  679.36  574.46
1960  685.47  568.05  1961  734.91  610.25
1962  726.01  535.76  1963  767.21  646.79
1964  891.71  768.08  1965  969.26  840.59
1966  995.15  744.32  1967  941.08  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  1555.57 1502.29 1987  2722.42 1738.74
1988  2183.50 1879.14 1989  2791.41 2144.64
1990  2999.75 2365.10 1991  3168.83 2470.30
1992  3413.21 3136.58 1993  3794.33 3241.95
1994  3978.36 3593.35 1995  5216.47 3832.08
;

title1 "Dow Jones Yearly Highs and Lows";

footnote1 j=1 * Source: 1997 World Almanac"

;

symbol1 interpol=join
    value=dot
    colors=_style_;  
symbol2 interpol=join
    value=C
    font=marker
    colors=_style_;  

axis1 order=(1955 to 1995 by 5) offset=(2,2)
    label=none
    major=(height=2)
    minor=(height=1)
    ;

axis2 order=(0 to 6000 by 1000) offset=(0,0)
    label=none
    major=(height=2)
    minor=(height=1)
    ;

legend1 label=none
    position=(top center inside)
    mode=share;

proc gplot data=stocks;

Program Description

Set the graphics environment.

goptions reset=all border;

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average and the dates of the high and low values each year.

data stocks;
  input year high low @@;
datalines;
1956 521.05 462.35 1957 520.77 419.79
1958 583.65 436.89 1959 679.36 574.46
1960 685.47 568.05 1961 734.91 610.25
1962 726.01 535.76 1963 767.21 646.79
1964 891.71 768.08 1965 969.26 840.59
1966 995.15 744.32 1967 943.08 786.41
1968 985.21 825.13 1969 968.85 769.93
1970 842.00 631.16 1971 950.82 797.97
1972 1036.27 889.15 1973 1051.70 788.31
1974 891.66 577.60 1975 881.81 632.04
1976 1014.79 858.71 1977 999.75 800.85
1978 907.74 742.12 1979 897.61 796.67
1980 1000.17 759.13 1981 1024.05 824.01
1982 1070.55 776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1955.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08
;

Define title and footnote.

title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=1 " 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
Define axis characteristics.

axis1 order=(1955 to 1995 by 5) offset=(2,2)
  label=none
  major=(height=2)
  minor=(height=1)
;

axis2 order=(0 to 6000 by 1000) offset=(0,0)
  label=none
  major=(height=2)
  minor=(height=1)
;

Define legend characteristics. The LABEL= option suppresses the legend label. The
POSITION= option centers the legend inside the top of the axis frame. The MODE=
option shares the legend area with other graphics elements.

legend1 label=none
  position=(top center inside)
  mode=share;

Generate two plots and display them on the same set of axes. The OVERLAY
option specifies that both plot lines appear on the same graph. The LEGEND= option
assigns the LEGEND1 definition to the graph. The VAXIS= option sets major tick marks
for the vertical axis. The HMINOR= and VMINOR= options specify the number of tick
marks between major tick marks.

proc gplot data=stocks;
  plot high*year low*year / overlay legend=legend1
    vref=1000 to 5000 by 1000
    lvref=2
    haxis=axis1 hminor=4
    vaxis=axis2 vminor=1;
  run;
quit;

Example 7: Filling Areas in an Overlay Plot

Features: PLOT statement options
  AREAS=
  HAXIS=
  HMINOR=
  VAXIS=
  VMINOR=
  CAXIS=
  OVERLAY

Other features: GOPTIONS statement option
  BORDER
  SYMBOL statement

Sample library member: GPLFILL1
Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

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.

![Dow Jones Yearly Highs and Lows](image)

Source: 1997 World Almanac

Program

```sas
options reset=all border;

data stocks;
    input year high low @@;
    datalines;
1956  521.05  462.35 1957  520.77  419.79
1958  583.65  436.89 1959  679.36  574.46
1960  685.47  568.05 1961  734.91  610.25
1962  726.01  535.76 1963  767.21  646.79
1964  891.71  768.08 1965  969.26  840.59
1966  995.15  744.32 1967  943.08  786.41
1968  985.21  825.13 1969  968.85  769.93
1970  842.00  631.16 1971  950.82  797.97
1972 1036.27  889.15 1973 1051.70  788.31
1974  891.66  577.60 1975  881.81  632.04
1976 1014.79  858.71 1977  999.75  800.85
1978  907.74  742.12 1979  897.61  796.67
1980 1000.17  759.13 1981 1024.05  824.01
1982 1070.55  776.92 1983 1287.20 1027.04
1984 1286.64 1086.57 1985 1553.10 1184.96
1986 1555.57 1502.29 1987 2722.42 1738.74
1988 2183.50 1879.14 1989 2791.41 2144.64
```

```
1990 2999.75 2365.10 1991 3168.83 2470.30
1992 3413.21 3136.58 1993 3794.33 3241.95
1994 3978.36 3593.35 1995 5216.47 3832.08

; run;

title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=l " Source: 1997 World Almanac";
symbol1 interpol=join;
axis1 order=(1955 to 1995 by 5) offset=(2,2)
       label=none
       major=(height=2)
       minor=(height=1);
axis2 order=(0 to 6000 by 1000) offset=(0,0)
       label=none
       major=(height=2)
       minor=(height=1);
proc gplot data=stocks;
   plot low*year high*year / overlay
      haxis=axis1
      hminor=4
      vaxis=axis2
      vminor=1
      caxis=black
      areas=2;
run;
quit;

Program Description

Set the graphics environment. BORDER draws a border around the graph.

goptions reset=all border;

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average and the dates of the high and low values each year.

data stocks;
   input year high low @@;
datelines;
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 950.81 797.97
1972 1036.27 889.15 1973 1051.70 888.31
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

Example 7: Filling Areas in an Overlay Plot
Define the title and the footnote.

```
title1 "Dow Jones Yearly Highs and Lows";
footnote1 j=1 " Source: 1997 World Almanac";
```

Define the symbol characteristics. The INTERPOL= option specifies a line to connect data points. The line creates the fill boundary.

```
symbol1 interpol=join;
```

Define the axis characteristics.

```
axis1 order=(1955 to 1995 by 5) offset=(2,2)
   label=none
   major=(height=2)
   minor=(height=1);
axis2 order=(0 to 6000 by 1000) offset=(0,0)
   label=none
   major=(height=2)
   minor=(height=1);
```

Generate a plot with filled areas. The plot requests are ordered to draw the lowest plot first. Area 1 occupies the space between the lowest (first) plot line and the horizontal axis, and area 2 is below the highest (second) plot line. This arrangement prevents the pattern for area 1 from overlaying the pattern for area 2. AREAS=2 fills all the areas below the second plot line.

```
proc gplot data=stocks;
   plot low*year high*year / overlay
      haxis=axis1
      hminor=4
      vaxis=axis2
      vminor=1
      caxis=black
      areas=2;
   run;
quit;
```

---

**Example 8: Plotting Three Variables**

**Features:**
- PLOT classification variable
- PLOT statement options
  - HAXIS=
  - HMINOR=
  - LEGEND=
  - VAXIS=
  - VMINOR=

**Other features:**
- GOPTIONS statement option
This example shows that when your data contain a classification variable that groups the data, you can use a plot request of the form `y-variable*x-variable=third-variable` to generate a separate plot for every value of the classification variable, which in this case is CITY. With this type of request, all plots are drawn on the same graph and a legend is automatically produced. This legend identifies the values of `third-variable`. The default legend uses the variable name CITY for the legend label and the variable values for the legend value descriptions.

This example then modifies the plot request. As shown in the second output display, the plot is enhanced by using different symbol definitions and colors for each plot line. It is also enhanced by changing axes labels and scaling the vertical axes differently.
Program

goptions reset=all border;

proc format;
  value mmm_fmt
    1='Jan'
    2='Feb'
    3='Mar'
    4='Apr'
    5='May'
    6='Jun'
    7='Jul'
    8='Aug'
    9='Sep'
   10='Oct'
   11='Nov'
   12='Dec'
  ;
run;

data citytemp;
  input  month faren city $ @@;
  datalines;
  1      40.5    Raleigh     1      12.2    Minn
  1      52.1    Phoenix     2      42.2    Raleigh
  2      16.5    Minn        2      55.1    Phoenix
  3      49.2    Raleigh     3      28.3    Minn
  3      59.7    Phoenix     4      67.7    Phoenix
  4      45.1    Minn        4      57.1    Minn
  5      67.4    Raleigh     5      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
;
source: 1980 American Express
Appointment Book
83.8    Phoenix    60.2    Raleigh
50.0    Minn       72.2    Phoenix
50.0    Raleigh    32.4    Minn
59.8    Phoenix    41.2    Raleigh
18.6    Minn       52.5    Phoenix

; title1 "Average Monthly Temperature";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l " Appointment Book";
symbol1 interpol=join value=dot;
proc gplot data= citytemp;
   plot faren*month=city / hminor=0;
run;
symbol1 interpol=spline width=2 value=triangle c=steelblue;
symbol2 interpol=spline width=2 value=circle c=indigo;
symbol3 interpol=spline width=2 value=square c=orchid;
axis1 label=none
   order = 1 to 12 by 1
   offset=(2);
axis2 label=("Degrees" justify=right "Fahrenheit")
   order=(0 to 100 by 10);
legend1 label=none value=(tick=1 "Minneapolis");
format month mmm_fmt.;
plot faren*month=city / 
   haxis=axis1 hminor=0
   vaxis=axis2 vminor=1
   legend=legend1;
run;
quit;

Program Description

Set the graphics environment.
   goptions reset=all border;

Create a format for the month values. Format mmm_fmt formats the numeric month values into three-character month names.
   proc format;
      value mmm_fmt
         1='Jan'
         2='Feb'
         3='Mar'
         4='Apr'
         5='May'
         6='Jun'
         7='Jul'
         8='Aug'
         9='Sep'
        10='Oct'
Create the data set. CITYTEMP contains the average monthly temperatures of three cities: Raleigh, Minneapolis, and Phoenix.

```sas
data citytemp;
  input month faren city $ @@;
datalines;
1 40.5 Raleigh 1 12.2 Minn
1 52.1 Phoenix 2 42.2 Raleigh
2 16.5 Minn 2 55.1 Phoenix
3 49.2 Raleigh 3 28.3 Minn
3 59.7 Phoenix 4 59.5 Raleigh
4 45.1 Minn 4 67.7 Phoenix
5 67.4 Raleigh 5 57.1 Minn
5 76.3 Phoenix 6 74.4 Raleigh
6 66.9 Minn 6 84.6 Phoenix
7 77.5 Raleigh 7 71.9 Minn
7 91.2 Phoenix 8 76.5 Raleigh
8 70.2 Minn 8 89.1 Phoenix
9 70.6 Raleigh 9 60.0 Minn
9 83.8 Phoenix 10 60.2 Raleigh
10 50.0 Minn 10 72.2 Phoenix
11 50.0 Raleigh 11 32.4 Minn
11 59.8 Phoenix 12 41.2 Raleigh
12 18.6 Minn 12 52.5 Phoenix
;
```

Define title and footnotes.

```sas
title1 "Average Monthly Temperature";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l " Appointment Book";
```

Define symbol for the first plot. This statement specifies that a straight line connect data point. Because no color is specified, the default color behavior is used and each line is a different color.

```sas
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.

```sas
proc gplot data= citytemp;
  plot faren*month=city / hminor=0;
run;
```

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.
Example 9: Plotting with Different Scales of Values

**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

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

This example shows how a PLOT2 statement generates a right axis that displays the values of the vertical coordinates in a scale different from that used for the left axis.

This plot shows the average monthly temperature for Minneapolis. Temperature variables represent degrees centigrade (displayed on the left axis) and degrees Fahrenheit (displayed on the right axis). These temperature variables are plotted against the variable MONTH. Although the procedure produces two sets of data points, it calibrates the axes so that the data points are identical and it displays only one plot.
This example uses SYMBOL statements to define symbol definitions. By default, the SYMBOL1 statement is assigned to the plot that is generated by the PLOT statement, and SYMBOL2 is assigned to the plot generated by the PLOT2 statement.

Program

goptions reset=all border;
PROC FORMAT;
  VALUE MMM_FMT
  1='JAN'
  2='FEB'
  3='MAR'
  4='APR'
  5='MAY'
  6='JUN'
  7='JUL'
  8='AUG'
  9='SEP'
  10='OCT'
  11='NOV'
  12='DEC';
RUN;
DATA MINNTEMP;
  INPUT @10 MONTH
  @23 F2;  /* fahrenheit temperature for Minneapolis */
  C2=(F2-32)/1.8;  /* calculate centigrade temperature */
                  /* for Minneapolis */
  OUTPUT;
  DATALINES;
01JAN83  1 1  40.5  12.2  52.1
01FEB83  2 1  42.2  16.5  55.1
01MAR83  3 2  49.2  28.3  59.7
01APR83  4 2  59.5  45.1  67.7
01MAY83  5 2  67.4  57.1  76.3
01JUN83  6    3   74.4  66.9  84.6
01JUL83  7    3   77.5  71.9  91.2
01AUG83  8    3   76.5  70.2  89.1
01SEP83  9    4   70.6  60.0  83.8
01OCT83 10   4   60.2  50.0  72.2
01NOV83 11   4   50.0  32.4  59.8
01DEC83 12   1   41.2  18.6  52.5
;

title1 "Average Monthly Temp for Minneapolis";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l " Appointment Book";
symbol1 interpol=needle ci=blue cv=red value=star;
symbol2 interpol=none value=none;
axis1 label=none order=(1 to 12 by 1) offset=(2);
axis2 label=( "Degrees" justify=right " Centigrade")
    order=(-20 to 30 by 10);
axis3 label=( "Degrees" justify=left "Fahrenheit")
    order=(-4 to 86 by 18);
proc gplot data= minntemp;
    format month mmm_fmt.;
    plot c2*month / haxis=axis1 hminor=0
        vaxis=axis2 vminor=1;
    plot2 f2*month / vaxis=axis3 vminor=1;
run;
quit;

Program Description

Set the graphics environment.

    goptions reset=all border;

Create a format for the month values. Format mmm_frt formats the numeric month values into three-character month names.

    proc format;
        value mmm_fmt
            1='JAN'
            2='FEB'
            3='MAR'
            4='APR'
            5='MAY'
            6='JUN'
            7='JUL'
            8='AUG'
            9='SEP'
           10='OCT'
           11='NOV';
Create the data set and calculate centigrade temperatures. MINNTEMP contains average monthly temperatures for Minneapolis.

``` Sas
data minntemp;
input @10 month @23 f2; /* fahrenheit temperature for Minneapolis */
c2=(f2-32)/1.8; /* calculate centigrade temperature */
/* for Minneapolis */
output;
datalines;
01JAN83 1 1 40.5 12.2 52.1
01FEB83 2 1 42.2 16.5 55.1
01MAR83 3 2 49.2 28.3 59.7
01APR83 4 2 59.5 45.1 67.7
01MAY83 5 2 67.4 57.1 76.3
01JUN83 6 3 74.4 66.9 84.6
01JUL83 7 3 77.5 71.9 91.2
01AUG83 8 3 76.5 70.2 89.1
01SEP83 9 4 70.6 60.0 83.8
01OCT83 10 4 60.2 50.0 72.2
01NOV83 11 4 50.0 32.4 59.8
01DEC83 12 1 41.2 18.6 52.5
;
```

Define title and footnotes.

``` Sas
title1 "Average Monthly Temp for Minneapolis";
footnote1 j=l " Source: 1984 American Express";
footnote2 j=l " Appointment Book";
```

Define symbol characteristics. INTERPOL=NEEDLE generates a horizontal reference line at zero on the left axis and draws vertical lines from the data points to the reference line. CI= specifies the color of the interpolation line and CV= specifies the color of the plot symbol.

``` Sas
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.

``` Sas
symbol2 interpol=none value=none;
```

Define axis characteristics. In the AXIS2 and AXIS3 statements, the ORDER= option controls the scaling of the axes. Both axes represent exactly the same range of temperature. The distance between the major tick marks on both axes represent an equivalent quantity of degrees (10 for centigrade and 18 for Fahrenheit).

``` Sas
axis1 label=none order=(1 to 12 by 1) offset=(2);
axis2 label=( "Degrees" justify=right " Centigrade")
    order=( -20 to 30 by 10);
axis3 label=( "Degrees" justify=left "Fahrenheit")
```
Generate a plot with a second vertical axis. The previously defined mmm_fmt format is applied to the MONTH variable to format the month numbers into three-character month names. The HAXIS= option specifies the AXIS1 definition. The VAXIS= option specifies AXIS2 and AXIS3 definitions in the PLOT and PLOT2 statements. Axis labels and major tick mark values use the default color. The VMINOR= option specifies the number of minor tick marks for each axis.

```
proc gplot data= minntemp;
    format month mmm_fmt.;
    plot c2*month / haxis=axis1 hminor=0 vaxis=axis2 vminor=1;
    plot2 f2*month / vaxis=axis3 vminor=1;
run;
quit;
```

Example 10: Creating Plots with Drill-down Functionality for the Web

Features:
- PLOT statement options
  - HTML=
  - HTML_LEGEND=
- ODS HTML statement
  - BODY=
  - NOGTITLE
  - PATH=

Other features:
- GOPTIONS statement option
  - BORDER
- BY statement
- GOPTIONS statement

Sample library member: GPLDRIL1

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

This example shows how to create a plot with simple drill-down functionality for the web. If you display the plot in a web browser, you can select any plot point or legend symbol to display a report on monthly temperatures for the selected city.

The example explains how to use an ODS statement such as ODS HTML to generate a graph with drill-down links. It shows how to do the following actions:

- explicitly name the HTML files and direct the different types of output to different files
- use BY-group processing with ODS, and determine the anchor names for the different pieces of output
- use the PATH= option to specify the destination for the HTML and GIF files created by the ODS statement
• add an HTML HREF string to a data set to define a link target
• assign link targets with the HTML= and HTML_LEGEND= procedure options
• suppress the titles in the GIF files and display them in the HTML file

For more information about drill-down graphs, see “Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192.

This program modifies the code from sample GPLVRBL2, which shows how to generate separate plots for the formatted values of a classification variable. In this example, the code implements drill-down capability for the plot. This enables you to select any plot point or legend symbol to drill down to a report on the yearly temperatures for the corresponding city. The first figure shows the drill-down plot as it is viewed in a browser.

The second figure shows the report that appears when you select any plot point or legend symbol that corresponds to the data for Raleigh.
Program

filename odsout ".";
goptions reset=all border device=gif;
ods _all_ close;
ods html path=odsout
   body="city_plots.html"
   nogtitle;
data citytemp;
   input Month Fahrenheit City $ @@;
datalines;
1   40.5   Raleigh   1   12.2   Minn
1   52.1   Phoenix   2   42.2   Raleigh
2   16.5   Minn      2   55.1   Phoenix
3   49.2   Raleigh   3   28.3   Minn
3   59.7   Phoenix   4   59.5   Raleigh
4   45.1   Minn      4   67.7   Phoenix
5   67.4   Raleigh   5   57.1   Minn
5   76.3   Phoenix   6   74.4   Raleigh
6   66.9   Minn      6   84.6   Phoenix
7   77.5   Raleigh   7   71.9   Minn
7   91.2   Phoenix   8   76.5   Raleigh
8      70.2    Minn        8      89.1    Phoenix
9      70.6    Raleigh     9      60.0    Minn
9      83.8    Phoenix    10      60.2    Raleigh
10     50.0    Minn       10      72.2    Phoenix
11     50.0    Raleigh    11      32.4    Minn
11     59.8    Phoenix    12      41.2    Raleigh
12     18.6    Minn       12      52.5    Phoenix

; data newtemp;
   set citytemp;
   length citydrill $ 40;
   if city="Minn" then
      citydrill="HREF='city_reports.html#IDX1'";
   else if city="Phoenix" then
      citydrill="HREF='city_reports.html#IDX2'";
   else if city="Raleigh" then
      citydrill="HREF='city_reports.html#IDX3'";
   title1 "Average Monthly Temperature";
   footnote1 j=l " Click a data point or legend symbol";
   symbol1 interpol=join
      value=dot;
   proc gplot data=newtemp;
      plot fahrenheit*month=city / hminor=0
         html=citydrill
         html_legend=citydrill;
   run;
   quit;
   ods html path=odsout
   body="city_reports.html";
   proc sort data=newtemp;
      by city month;
   run;
   goptions reset=footnote;
   option nobyline;
   title1 "Monthly Temperatures in #byval(city)";
   proc report data=newtemp nowindows;
      by city;
      column city month fahrenheit;
      define city   / noprint group;
      define month  / group;
      define Fahrenheit / group;
   run;
   ods html close;
   ods html; /* Not required in SAS Studio */

**Program Description**

**Define graphics output location.**

filename odsout ".";
Set the graphics environment.

```bash
goptions reset=all border device=gif;
```

Close the currently open ODS destinations.

```bash
ods _all_ close;
```

Open an HTML output file in ODS.

```bash
ods html path=odsout
  body="city_plots.html"
  nogtitle;
```

Create the data set CITYTEMP. CITYTEMP contains the average monthly temperatures for three cities.

```bash
data citytemp;
  input  Month Fahrenheit City @@;
  datalines;
1   40.5    Raleigh  1   12.2    Minn
1   52.1    Phoenix  2   42.2    Raleigh
2   16.5    Minn    2   55.1    Phoenix
3   49.2    Raleigh  3   28.3    Minn
3   59.7    Phoenix  4   59.5    Raleigh
4   45.1    Minn    4   67.7    Phoenix
5   67.4    Raleigh  5   57.1    Minn
5   76.3    Phoenix  6   74.4    Raleigh
6   66.9    Minn    6   84.6    Phoenix
7   77.5    Raleigh  7   71.9    Minn
7   91.2    Phoenix  8   76.5    Raleigh
8   70.2    Minn    8   89.1    Phoenix
9   70.6    Raleigh  9   60.0    Minn
9   83.8    Phoenix 10  60.2    Raleigh
10  50.0    Minn   10  72.2    Phoenix
11  50.0    Raleigh 11  32.4    Minn
11  59.8    Phoenix 12  41.2    Raleigh
12  18.6    Minn   12  52.5    Phoenix
;
```

Add the HTML variable to CITYTEMP and create the NEWTEMP data set. The HTML variable CITYDRILL contains the target locations to associate with the different values of the variable CITY. Each location for CITYDRILL references the file city_reports.html, which this program creates. Each location ends with the default anchor name (IDX1, IDX2, and IDX3) that ODS assigns to the target output when it creates that output in file city_reports.html.

```bash
data newtemp;
  set citytemp;
  length citydrill $40;
  if city="Minn" then
    citydrill="HREF='city_reports.html#IDX1'";
  else if city="Phoenix" then
    citydrill="HREF='city_reports.html#IDX2'";
  else if city="Raleigh" then
    citydrill="HREF='city_reports.html#IDX3'";
```
Define a title and footnote and a symbol definition for the plots.

```plaintext
title1 "Average Monthly Temperature";
footnote1 j=l "Click a data point or legend symbol";
symbol1 interpol=join
   value=dot;
```

**Generate the plot.** Both HTML= and HTML_LEGEND= specify CITYDRILL as the variable that contains the targets for the drill-down links. The HTML= option determines that each plot point is a hot zone that links to target output. The HTML_LEGEND= option determines that the legend symbols are hot zones that link to target output. This GPLOT procedure generates the first piece of output in this program. Thus, the plot receives the first default anchor name, which is IDX.

```plaintext
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.

```plaintext
ods html path=odsout
   body="city_reports.html";
```

**Sort data set NEWTEMP in order by city.**

```plaintext
proc sort data=newtemp;
   by city month;
run;
```

**Clear the footnotes, and suppress the default BY line.**

```plaintext
goptions reset=footnote;
   option nobyline;
```

**Print a report of monthly temperatures for each city.** The BY statement determines that a separate report is generated for each city. Thus, the REPORT procedure generates three pieces of output. To assign anchor locations to this new output, ODS increments the last anchor name that was used (IDX). Therefore, ODS assigns the anchor names IDX1, IDX2, and IDX3 to the output. These are the anchor locations that were specified above as the anchor locations for variable CITYDRILL.

```plaintext
title1 "Monthly Temperatures in #byval(city)";
proc report data=newtemp nowindows;
   by city;
   column city month fahrenheit;
   define city / noprint group;
   define month / group;
   define Fahrenheit / group;
run;
```
**Close ODS HTML.** This closes the HTML output file.

```sas
ods html close;
```

**Open ODS HTML.** Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```
Overview: GRADAR Procedure

About the GRADAR Procedure

The GRADAR procedure creates radar charts, wind rose charts, and calendar charts. Note that the GRADAR procedure is not supported by the Java device drivers.

About Radar Charts

Radar charts show the relative frequency of data measures in quality control or market research problems. The chart statistics are displayed along spokes that radiate from the center of the chart. The charts are often stacked on top of one another with reference circles, which gives them the look of a radar screen. By default, the chart vertices—the points where the statistical values intersect the spokes—are based on the frequencies associated with the levels of a single numeric variable. Non-integer values of the chart variable are truncated to integers. The measures can be displayed in decreasing order, the
order in which they appear in the input data, increasing order of internal values, or lexicographic order of variable names. Radar charts are sometimes also called star charts or spider charts. The terms vary depending on the appearance of the chart and the variable types that the chart is depicting.

Note: A calendar chart, circular in format, is generated with the GRADAR procedure. 

Tip: Radar charts are most effective when the input data set is limited to a few hundred points.

Figure 42.1 Basic Radar Chart (GRRBASIC)

If the radar chart includes all of the axes and the reference lines, it is commonly referred to as a spider chart. The program for this chart is in “Example 1: Producing a Basic Radar Chart” on page 1266. For more information about producing a radar chart, see the “CHART Statement” on page 1247.

About Wind Rose Charts

A wind rose chart displays wind direction and speed over a period of time at a specific location. The segments in a wind rose chart represent the percentage of time that the wind blew from each direction.

The wind data must be sorted by direction. You can enter data for four, eight, or sixteen compass points.

The angular direction of the axes, as in this example, can be significant.
The program for this chart is in “Example 7: Creating a Wind Rose Chart” on page 1275. For more information about producing a radar chart, see the “CHART Statement” on page 1247.

**About Calendar Charts**

Calendar charts display twelve equal-sized segments, one for each month of the year. The color shading of each segment represents the magnitude of the frequency variable. This calendar chart is overlaid with climate data showing temperatures for each month of the years 2006 and 2007.
The program for this chart is in “Example 8: Creating a Calendar Chart” on page 1276. For more information about producing a radar chart, see the “CHART Statement” on page 1247.

Syntax: GRADAR Procedure

Restrictions: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

Not supported by Java

The SAS ‘DOLLAR8.2’ FORMAT is not supported by the ActiveX device.

Requirement: At least one CHART statement is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, TITLE

Supports: RUN-group processing

Note: The procedure can include the SAS/GRAPH statements BY on page 370 and NOTE on page 447, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

**PROC GRADAR**

```sas
PROC GRADAR <DATA=input-data-set>
<GOUT=<libref:output-catalog>
<ANNOTATE=Annotate-data-set>

   CHART chart-variable </option(s)>

QUIT;
```

**PROC GRADAR Statement**

Identifies the data set that contains the plot variables. Specifies an output catalog (optional).

Requirement: An input data set is required.

**Syntax**

```sas
PROC GRADAR <DATA=input-data-set>
<GOUT=<libref:output-catalog>
<ANNOTATE=Annotate-data-set>
```
**Optional Arguments**
PROC GRADAR statement options affect all graphs produced by the procedure.

**ANNOTATE=Annotate-data-set**
specifies a data set to add annotate elements to all graphs that are produced by the GRADAR procedure. To add annotate elements to individual graphs, use ANNOTATE= in the CHART statement.

**DATA=input-data-set**
specifies the SAS data set that contains the variable or variables to chart. By default, the procedure uses the most recently created SAS data set.

**GOUT=<libref>.output-catalog**
specifies the SAS catalog in which to save the graphics output produced by the GRADAR procedure.

---

**CHART Statement**
Creates the radar charts in which the length of the vertices along the spines represent the values of the chart statistic for the data categories.

**Requirement:** At least one chart variable is required.

**Syntax**
CHART chart-variable <option(s)>;

**Summary of Optional Arguments**

**Appearance options**

**ANNOTATE=Annotate-data-set**
specifies a data set to add annotate elements to charts produced by the CHART statement.

**CAXIS=color**
specifies a color for the chart frame and the spokes or grid lines of the chart.

**CFRAME=color | (variable)**
fills the area inside the frame or the individual tiles with the specified color.

**CFRAMESIDE=color | (variable)**
specifies the color for filling the area for the row labels displayed along the left side of a chart.

**CFRAMETOP=color | (variable)**
specifies the color for filling the area for the column labels that are displayed across the top of a chart.

**CSPOKES=spoke-color | (variable)**
specifies a color to use for the spokes in a chart.

**CSTARCIRCLES=color | (colors-list)**
specifies a color or list of colors for the circles that are requested with the
STARCIRCLES= option.

**CSTARFILL=**

**color** | (**color-list**)  
specifies a color or colors for filling the interior of stars when STARFILL= is
set to SOLID.

**CSTARS=**

**color** | (**color-list**)  
specifies a color or list of colors for the outlines of stars.

**CTEXT=**

**color**  
specifies a color for all text on the chart.

**CTILES=**

**(variable) | color**  
specifies one or more colors for the tiles.

**FONT=**

**font**  
specifies the font for all text strings in the radar chart.

**FRAME** | **NOFRAME**  
specifies whether a frame is drawn around the chart.

**HEIGHT=**

**height**  
specifies the height in cells for labels and legends.

**IFRAME=**

**fileref** | **"external-image-file"**  
specifies an image file to display inside the chart's frame.

**IMAGESTYLE=**

**TILE** | **FIT**  
specifies how to display the image file that is specified on the IFRAME= option.

**INBORDER**  
generates an inside border around the chart.

**INHEIGHT=**

**value**  
specifies the height for spoke labels.

**INTERTILE=**

**value**  
specifies the distance (in cells) between tiles in a chart.

**LSPOKES=**

**linetype**  
specifies a line type for the spokes in a radar chart.

**LSTARCIRCLES=**

**linetypes** | (**linetypes**)  
specifies one or more line types for the circles requested with the
STARCIRCLES= option.

**LSTARS=**

**(linetypes)**  
specifies the line types for the outlines of stars that are produced for a radar
chart.

**MAXNVERT=**

**n**  
specifies the maximum number of vertices, from 1 to 360, in the radar chart.

**NOLEGEND**  
suppresses the legend that is otherwise automatically displayed.

**SPIDERWEB**  
displays lines connecting the points where tick marks would be instead of
displaying the tick marks.

**SPKLABEL=**

**CATEGORY** | **ALL** | **NONE**  
labels the chart spokes with the category of the variable that is being charted.

**SPOKESCALE=**

**CATEGORY** | **VERTEX**  
specifies whether every spoke is drawn to the same scale, or whether each
spoke is drawn to a different scale.

**STARCIRCLES=**

**(values)**  
specifies reference circles that are superimposed on the stars that are
produced for a radar chart.
STARFILL= lists of (SOLID | EMPTY) one for each star
determines whether the stars in the radar chart are empty or filled with a solid
color.

STARINRADIUS=percent
specifies the inner radius of stars as a percent from 0 to 100.

STARLEGEND=CLOCK | CLOCK0 | NUMBER | DEGREES | NONE
specifies the style of the legend used to identify the vertices of stars that are
produced for a radar chart.

STARLEGENDLAB="legend-label"
specifies the label displayed to the left of the legend for stars requested with
the STARLEGEND= option.

STAROUTRADIUS=value
specifies the outer radius of stars as a percent up to 100.

START=value
specifies the vertex angle for the first variable that is specified in the CHART
statement.

TILELEGEND=(variable)
specifies a variable used to add a legend for CTILES= colors.

TILELEGENDLABEL="label"
specifies a label displayed to the left of the legend that is created when you
specify a TILELEGEND= variable.

WFRAME=n
specifies the width in pixels for the frame lines.

WINDROSECIRCLES=n
specifies the number of reference circles.

WSPOKES=line-width
specifies the width in pixels of the spokes in a radar chart.

WSTARCIRCLES=(line-widths)
specifies the width in pixels of the outline of circles requested by the
STARCIRCLES= and WINDROSECIRCLES= options.

WSTARS=line-width | (line-widths)
specifies the width in pixels of the outline of stars that are produced for a
radar chart.

Axis options

STARAXIS= (AXIS<1...99><, . . . ,AXIS<1...99>>)
assigns one or more axis definitions to the axis spokes in the radar chart.

WAXIS=
See WFRAME=.

Catalog entry description options

DESCRIPTION="description"
specifies a description of the output.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics
output file, if one is created.

Chart options

ACROSSVAR=variable
generates a radar chart for each value (including missing values) of the
specified variable and displays the charts from left-to-right across the output
area.
CALENDAR  
produces a chart displaying 12 equal-sized segments, one for each month of the year January through December.

DOWNVAR=variable  
generates a radar chart for each value of the specified variable, and displays the charts from top-to-bottom.

FREQ=numeric-variable  
specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic.

MISSING  
accepts a missing value as a valid midpoint for the chart variable.

MODE=SHARE | PROTECT | RESERVE  
specifies the display mode for a radar chart.

NCOLS=n  
specifies the number of columns in a chart.

NLEVELS=n  
specifies the number of colors used in the calendar chart to represent the magnitude of the frequency variable.

NOZEROREF  
turns off the zero reference circle when negative values are plotted.

NROWS=n  
specifies the number of rows in a chart.

OTHER="category"  
specifies a new category that merges all categories not selected because of the MAXNVERT= option.

OVERLAYVAR=overlay-variable  
creates a comparative radar chart using the levels of the overlay variable.

SPEED=speed-variable  
specifies the wind speed in wind rose charts.

STARTYPE=CORONA | POLYGON | RADIAL | SPOKE | WEDGE  
specifies the style of the stars that are produced for a radar chart.

SUMVAR=summary-variable  
specifies a numeric variable to be used to construct weighted radar charts.

WINDROSE  
specifies a wind rose chart.

**ODS options**

**HTML_LEGEND=variable**  
identifies the variable in the input data set whose values create links or data tips or both.

**URL=character-variable**  
specifies a character variable whose values are URLs.

**Required Argument**

*chart-variable(s)*  
specifies one or more variables that define the categories of data to be charted. The values of the chart variable determine the spokes in the corresponding radar chart. These values are the observations in the input data set for the chart variable. You must have at least three observations in the data set as it takes three points to define a plane. Technically, you can create a GRADAR chart with only one or two observations, but a true radar chart is not displayed.
**Optional Arguments**

Options in a CHART statement affect all graphs produced by that statement. You can specify as many options as needed and list them in any order.

- **ACROSSVAR=variable**
  generates a radar chart for each value (including missing values) of the specified variable and displays the charts from left-to-right across the output area. If used with the DOWN= option, the charts are drawn in left-to-right and top-to-bottom order. To limit the number of columns or rows that are displayed, specify the NCOLS= and NROWS= options.

  **Alias** ACROSS=

  **Restriction** This option is ignored if you specify the WINDROSE, CALENDAR, or OVERLAYVAR= options.

  **See** DOWNVAR= on page 1255, NCOLS= on page 1259, and NROWS= on page 1260

  **Examples** “Example 3: Tiling Radar Charts” on page 1268
              “Example 4: Using Multiple Classification Variables in Radar Charts” on page 1269

- **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 646.

  **See** Chapter 27, “Using Annotate Data Sets,” on page 635

- **CALENDAR**
  produces a chart displaying 12 equal-sized segments, one for each month of the year January through December. The color shading of each segment represents the magnitude of the frequency variable. Use the OVERLAY variable to subdivide each segment (for example, by year).

  **Restriction** When you specify the CALENDAR option, you must also specify the OVERLAYVAR= option.

  **Tip** Missing values are treated as zeros. For any month in which the frequency variable is missing, the color used for that chart segment is the color for zero.

  **See** FREQ= on page 1256, NLEVELS= on page 1260, and OVERLAYVAR= on page 1260

  **Example** “Example 8: Creating a Calendar Chart” on page 1276

- **CAXIS=color**
  specifies a color for the chart frame and the spokes or grid lines of the chart. The specified color must be a valid SAS/GRAPH color name.

  **Alias** CAXES=, CA=
For ActiveX, the default color is black, and is unchangeable. For other devices, when the NOGSTYLE system option is not specified, the default color is retrieved from the current style. If NOGSTYLE is specified, the default color is the first color in the color list.

**Style reference**
Color attribute of the GraphAxisLines element

**Restriction**
Not supported by ActiveX

**Interactions**
For the spokes of a chart, the CSPOKES= option overrides the CAXIS= option.

The NOGSTYLE option suppresses use of the current style’s color.

**Note**
Avoid using a style with a dark background when specifying an ActiveX device with its default color of black for its chart frame and spokes.

**See**
CFRAME= on page 1252 CSPOKES= on page 1252

---

**CFRAME=**
`color | (variable)`

fills the area inside the frame or the individual tiles with the specified color. You can specify a valid SAS/GRAPH color name, or a character variable whose value is the background color.

**Alias**
CFR=

**Interaction**
The CTILES= option overrides the CFRAMESIDE= option.

**See**
CAXIS= on page 1251

---

**CFRAMESIDE=**
`color | (variable)`

specifies the color for filling the area for the row labels displayed along the left side of a chart. The specified color must be a valid SAS/GRAPH color name or a character variable whose value is a valid SAS/GRAPH color name. If a label is associated with the classification variable, the specified color is also used to fill the area for this label. By default, these areas are not filled.

**Restriction**
The CFRAMESIDE= option is ignored unless you also specify the DOWNVAR= option.

---

**CFRAMETOP=**
`color | (variable)`

specifies the color for filling the area for the column labels that are displayed across the top of a chart. The specified color must be a valid SAS/GRAPH color name, or a character variable whose value is a valid SAS/GRAPH color name. If a label is associated with the classification variable, the specified color is also used to fill the area for this label. By default, these areas are not filled.

**Restriction**
The CFRAMESIDE= option is ignored unless you also specify the ACROSSVAR= option.

---

**CSPOKES=**
`spoke-color | (variable)`

specifies a color to use for the spokes in a chart. The specified color must be a valid SAS/GRAPH color name, or a character variable whose value is the color. The default color is specified by the current style or is the first color in the color list if the NOGSTYLE option is specified.
Alias CSPOKE=

Restriction The CSPOKES= option is ignored if you specify the CALENDAR or WINDROSE options.

CSTARCIRCLES=color | (colors-list)
specifies a color or list of colors for the circles that are requested with the STARCIRCLES= option. All specified colors must be valid SAS/GRAPH color names. By default, the color specified with the CSTARS= option is used. If the CSTARS= option is omitted, the default color is specified by the current style or is the first color in the color list if the NOGSTYLE option is specified.

Alias CSTARCIRCLE=

Example “Example 2: Overlaying Radar Charts” on page 1267

CSTARFILL=color | (color-list)
specifies a color or colors for filling the interior of stars when STARFILL= is set to SOLID. All specified colors must be valid SAS/GRAPH color names.

If STARFILL is set to SOLID, the GRADAR procedure fills the stars with the first set of colors that it finds from the following list:

- the color or colors specified on the CSTARFILL= option
- the color or colors specified on the CSTARS= option
- the color or colors specified by the current style or, if the NOGSTYLE option is specified, the colors in the device color list.

The number of colors that you specify depends on the number of stars in the chart.

- If the OVERLAY= option is not used, all stars are filled with the same color. Specify a single fill color. If the ACROSSVAR= option or the DOWNVAR= option are used, the specified color is applied to each star in the tiled display.
- If the OVERLAY= option is used, the chart contains multiple overlaid stars. In that case, specify a list of colors in parentheses. Make sure that there are at least as many colors in the list as there are stars in the chart. If you do not specify enough colors for each star to have a different color, the GRADAR procedure assigns colors from the current style (or the device color list) to the remaining stars. (If the NOGSTYLE option is specified, the color for the star positioned at subgroup \( n \) on the chart is the value of the color corresponding to the color at position \( n \) in the device color list.)

If the CSTARFILL= option is specified and the CSTARS= option is not specified for the outline, then the outline is the same as the CSTARFILL option.

If the STARFILL= option is not set or is set to EMPTY, then the CSTARFILL= option sets only the outline color. You can also use the CSTARS= option to set the outline color.

See CSTARS= on page 1253

CSTARS=color | (color-list)
specifies a color or list of colors for the outlines of stars. All specified colors must be valid SAS/GRAPH color names. The GRADAR procedure uses the first set of colors that it finds from the following list:

- the color or colors specified on the CSTARS= option
- the color or colors specified on the CSTARFILL= option
the color or colors specified by the current style or, if the NOGOSTYLE 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 NOGOSTYLE 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.)

### CSTAR=

Alias CSTAR=

See CSTARFILL= on page 1253

Example “Example 5: Modifying the Appearance of Radar Charts” on page 1271

**CTEXT=**color

Specifies a color for all text on the chart. The specified color must be a valid SAS/GRAPH color name.

#### Defaults

For ActiveX, the default color is black. For other devices, the GRADAR procedure uses the first color that it finds in the following list.

1. the CTEXT= option in a GOPTIONS statement
2. the color specified by the current style
3. if the NOGOSTYLE option is specified, the second color in the color list

#### Style reference

Color attribute of the GraphValueText element

#### Note

Use the CTEXT= option to specify a more visible color if you are using a style with a dark background, and an ActiveX device with its default text color of black.

**CTILES=(variable) | color**

Specifies either a character variable whose values are the fill colors for the tiles or a single color that is the fill color for all tiles. By default, the tiles are not filled.

If you specify a variable, the values of the specified variable must be identical for all observations with the same level of the classification variables. The same color can be used to fill more than one tile. Use the special value, EMPTY, to indicate that a tile is not to be filled.

The CTILES= option cannot be used in conjunction with the NOFRAME option or the CFRAME= option. You can use the TILELEGEND= option in conjunction with the CTILES= option to add an explanatory legend for the CTILES= option colors at the bottom of the chart.
### 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 189 for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the `Description` field of the PROC GREPLAY window.

The `description` can include the `#BYLINE`, `#BYVAL`, and `#BYVAR` substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959. 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.

### 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.

### FONT=font

specifies the font for all text strings in the radar chart. If you omit the FONT= option, the font that is specified by the FTEXT= graphics option is used. If neither option is specified, the default font is specified by the current style or, if the NOGSTYLE option is specified, by the current device.

**Style reference**

Font attribute of the GraphValueText element
FRAME | NOFRAME

specifies whether a frame is drawn around the chart. FRAME draws a frame inside the border specified by INBORDER (if INBORDER is specified). NOFRAME suppresses the frame.

By default, the frame color is specified by the current style or, if the NOGOSTYLE option is specified, is the first color in the color list. If you want to specify a different color, use the CFRAME= option to fill the frame with a specified color or the CAXIS= option to specify the color of the frame.

**Default**

FRAME

**Restriction**

The NOFRAME option cannot be specified with the CFRAME= option or the CTILES= option. This option is not supported by ActiveX.

**See**

CAXIS= on page 1251 and CFRAME= on page 1252

**Example**

“Example 7: Creating a Wind Rose Chart” on page 1275

---

**FREQ=numeric-variable**

specifies a variable whose values weight the contribution of each observation in the computation of the chart statistic. Each observation is counted the number of times that are specified by the value of numeric-variable for that observation. If the value of numeric-variable is missing, 0, or negative, the observation is not used in the statistic calculation. Non-integer values of numeric-variable are truncated to integers.

The statistics are not affected by applying a format to numeric-variable.

**See**

“Calculating Weighted Statistics” on page 1266

**Examples**

“Example 1: Producing a Basic Radar Chart” on page 1266

“Example 2: Overlaying Radar Charts” on page 1267

“Example 3: Tiling Radar Charts” on page 1268

---

**HEIGHT=height**

specifies the height in cells for labels and legends.

**Alias**

HLABEL=

**Interaction**

The HEIGHT= option overrides the HTEXT= option in a GOPTIONS statement. This does not change the size of titles or footnotes.

---

**HTML_LEGEND=variable**

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

**See**

“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192
IFRAME=fileref | "external-image-file"
specifies an image file to display inside the chart's frame. Fileref must be a valid SAS fileref up to eight characters long and must have been previously assigned with a FILENAME statement. External-image-file must specify the complete filename of the image file that you want to use. The format of external-image-file varies across operating environments. For more information, see “Displaying an Image in Graph Frame” on page 334.

Restriction Not supported by ActiveX

Interactions This option is overridden by the NOIMAGEPRINT graphics option.

This option is ignored if you specify the NOFRAME option or if you specify the STYLE=0 option in the AXIS statement.

IMAGESTYLE=TILE | FIT
specifies how to display the image file that is specified on the IFRAME= option. TILE copies the image as many times as needed to fit inside the frame. FIT stretches the image so that a single copy fits within the frame.

Note When used with the IFRAME option, the IMAGESTYLE option must be within the PROC statement. When used with the IBACK option, the IMAGESTYLE option must be specified in the GOPTIONS statement.

INBORDER
generates an inside border around the chart. This border is inside the border created by the BORDER option in the GOPTIONS statement, if it is specified.

Restriction Not supported by ActiveX

INHEIGHT=value
specifies the height for spoke labels. The default unit is PCT, which is percentage of graphics output area. The INHEIGHT= option overrides the HTEXT= option in a GOPTIONS statement. This option does not change the size of titles or footnotes.

Restriction Not supported by ActiveX

See HEIGHT=

Example “Example 6: Modifying Chart Axes” on page 1272

INTERTILE=value
specifies the distance (in cells) between tiles in a chart, and is used only with the ACROSSVAR= option and the DOWNVAR= option. By default, the tiles are contiguous (value=0).

Alias INTERCHART=

Default 0

Example “Example 4: Using Multiple Classification Variables in Radar Charts” on page 1269

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 line types for, SAS/GRAF uses the line types that you specify and defaults to 1 (solid) for the remaining circles.

Alias LSTARCIRCLE=

Default 1 (solid line)

LSTARS=(linetypes)

specifies the line types for the outlines of stars that are produced for a radar chart. By default, the outlines rotate through the list of line types. The default line type for the star positioned at subgroup n is the value of the line type corresponding to the position n in the list of line types.

The number of line types that you specify depends on the number of stars in the chart.

• If the OVERLAY= option is not used, all stars use the same line type. Specify a single fill line type. If the ACROSSVAR= option or the DOWNVAR= option are used, the specified line type is applied to each star in the tiled display.

• If the OVERLAY= option is used, the chart contains multiple overlaid stars. In that case, specify a list of line types in parentheses. Make sure that there are at least as many line types in the list as there are stars in the chart.

To specify line colors, use the CSTARS= option.

Alias LSTAR=

Example “Example 5: Modifying the Appearance of Radar Charts” on page 1271

MAXNVERT=n

specifies the maximum number of vertices, from 1 to 360, in the radar chart.

Alias MAXVERT=

MISSING

accepts a missing value as a valid midpoint for the chart variable. By default, observations with missing values are ignored.

MODE=SHARE | PROTECT | RESERVE

specifies the display mode for a radar chart.

SHARE

shares the drawing space between the text and the graph.

PROTECT

shares the drawing space but maintains a solid rectangle (using the background color) behind the text. This is useful when the text is illegible because of the image specified with the IFRAME= option or the color specified with the CFRAME= option.

RESERVE

reduces the size of the text and graph in order to accommodate both.

Default RESERVE
NAME="name"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to name:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

  *Note:* Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

**Default**  
RADAR

**NCOLS=n**

specifies the number of columns in a chart. You can use the NCOLS= option in conjunction with the NROWS= option. NCOLS=2 and NROWS=2 if two classification variables are specified. If used with the ACROSSVAR= or DOWNVAR= options, the default number of columns or rows is calculated from the number of classifications for the variables that are listed on the ACROSSVAR= or DOWNVAR= options. In that case, you can use the NCOLS= option and NROWS= option to limit the number of columns and rows that are specified.

**Alias**  
NCOL=
NLEVELS=n
specifies the number of colors used in the calendar chart to represent the magnitude of the frequency variable. The colors are shown in the legend as a color ramp ranging from white to the full intensity of one color.

Default 6

NOLEGEND
suppresses the legend that is otherwise automatically displayed.

NOZEROREF
turns off the zero reference circle when negative values are plotted. When a negative value is plotted, a dashed circle indicates the zero position. You cannot change the appearance of this zero reference circle, but you can turn it off with the NOZEROREF option. The zero reference circle does not appear if there are no negative values plotted.

Restriction Not supported by ActiveX

NROWS=n
specifies the number of rows in a chart. You can use the NROWS= option in conjunction with the NCOLS= option. See NCOLS= for details.

Alias NROW=

Default 1

Restriction Not supported by ActiveX

OVERLAYVARIABLE=overlay-variable
creates a comparative radar chart using the levels of the overlay variable. All charts are displayed in the same set of spokes. A maximum of 24 overlays can be displayed in a single radar chart.

Alias OVERLAY=
Restriction

This option cannot be used with the ACROSSVAR= option or the DOWNVAR= options.

Examples

“Example 2: Overlaying Radar Charts” on page 1267
“Example 8: Creating a Calendar Chart” on page 1276
“Example 6: Modifying Chart Axes” on page 1272

SPEED=speed-variable

specifies the wind speed in wind rose charts.

SPIDERWEB

displays lines connecting the points where tick marks would be instead of displaying the tick marks, using the same number of points for all axes as for the first axis. The default number of web lines is three.

If there is an AXIS statement in effect, then the web gets its values (such as number, thickness, and color) from the MAJOR= values for the axis drawn at the first position. (The default first position is 12 o'clock.)

Alias

SPIDER

Restriction

This option is ignored if you also specify the CALENDAR or WINDROSE option.

SPKLABEL=CATEGORY | ALL | NONE

labels the chart spokes with the category of the variable that is being charted.

You can specify this option starting with SAS 9.4M2. When you specify the SPKLABEL=ALL option, the GRADAR procedure generates a spoke (axis) label for each graph displayed in the graphics output area. Without the ALL specification, only the first graph displayed has the spoke (axis) labels, and labels for subsequent graphs are suppressed.

NONE suppresses the labels.

Default

CATEGORY However, if the STARLEGEND= option is specified, the default is NONE.

Restriction

Not supported by ActiveX device

SPOKESCALE=CATEGORY | VERTEX

specifies whether every spoke is drawn to the same scale, or whether each spoke is drawn to a different scale. When you specify the SPOKESCALE=CATEGORY option (or you do not specify the SPOKESCALE option), the GRADAR procedure determines the minimum and maximum value of all the spokes. Then, the vertices of all the spokes have that same maximum value and minimum value. When you specify the SPOKESCALE=VERTEX option, each vertex has its own maximum value and minimum value, and each vertex is labeled at the tick marks.

Restriction

If you specify SPOKESCALE=VERTEX, you must also specify the OVERLAYVAR=, ACROSSVAR=, or DOWNVAR= option.

STARAXIS= (AXIS<1...99>, . . . , AXIS<1...99>)

assigns one or more axis definitions to the axis spokes in the radar chart. GRADAR displays axis spokes clockwise, starting at the 12 o'clock position. The axis definitions that are specified using the STARAXIS= option are assigned consecutively to the spokes, starting from the first spoke. AXIS statements are
assigned in clockwise order. For example, the STARAXIS=(AXIS3, AXIS1, AXIS2) option assigns the AXIS3 statement's definition to the first axis spoke (at the 12 o'clock position), the AXIS1 statement's definition to the second axis spoke, and the AXIS2 statement's definition to the third axis spoke.

The axis definitions are assigned consecutively, and you cannot skip a spoke. For example, to assign a definition to the seventh spoke, you must also assign definitions to the first six spokes. However, you do not have to assign definitions to all of the spokes. Any remaining axis spokes on the GRADAR chart are displayed with the default settings. For example, if the STARAXIS= option specifies three definitions and the chart has more than three axis spokes, the fourth and remaining spokes are displayed with the default settings. If there are more definitions specified than there are axis spokes in the chart, the excess definitions are ignored.

Alias      STARAXES=
Example     “Example 6: Modifying Chart Axes” on page 1272

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 $\text{inradius} + h \times (\text{outradius} - \text{inradius})$.

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 $\text{inradius} + 0.5 \times (\text{outradius} - \text{inradius})$, or a circle halfway between the inner circle and the outer circle. Likewise, the value 0.25 specifies a circle one-fourth of the way from the inner circle to the outer circle.

To specify the line types for the circles, use the LSTARCIRCLES= option. To specify colors for the circles, use the CSTARCIRCLES= option.

Alias      STARCIRCLE=
Example     “Example 2: Overlaying Radar Charts” on page 1267

STARFILL= lists of (SOLID | EMPTY) one for each star
determines whether the stars in the radar chart are empty or filled with a solid color. Valid values are EMPTY (the default) and SOLID. If there are multiple stars in the chart, specify, in parentheses, a separate value for each star.

If the STARFILL=(SOLID) option and the CSTARFILL= option are not specified, then each star is filled with the colors specified on the CSTARS= option. If the CSTARS= option is not specified, then the fill colors are determined by the current style or, if the NOGSTYLE option is specified, by the device color list.

If STARFILL= is not set or is set to EMPTY, then CSTARFILL= is ignored.

STARINRADIUS=percent
specifies the inner radius of stars as a percent from 0 to 100. The inner radius of a star is the distance from the center of the star to the circle that represents the lower limit of the standardized vertex variables. The lower limit can correspond to the minimum value, a multiple of standard deviations below the mean, or a lower specification limit. The value must be less than the value that is specified with the STAROUTRADIUS= option. The default value is one-third of the outer radius.
STARLEGEND=CLOCK | CLOCK0 | NUMBER | DEGREES | NONE
specifies the style of the legend used to identify the vertices of stars that are
produced for a radar chart.

CLOCK
identifies the vertex variables by their positions on the clock (starting with
12:00).

CLOCK0
identifies the vertex variables by their positions on the clock (starting with 0:00
corresponding to 12:00).

NUMBER
identifies the vertex variables by numbers. The number 1 corresponds to 12
o'clock. Legend entries are assigned in clockwise order.

DEGREES
identifies the vertex variables by angles in degrees. Zero degrees corresponds to
12 o'clock.

NONE
suppresses the legend. This is the default.

Example   “Example 3: Tiling Radar Charts” on page 1268

STARLEGENDLAB="legend-label"
specifies the label displayed to the left of the legend for stars requested with the
STARLEGEND= option. The label can be up to 16 characters and must be enclosed
in quotation marks. The default label is Vertices:.

Example   “Example 3: Tiling Radar Charts” on page 1268

STAROUTRADIUS=value
specifies the outer radius of stars as a percent up to 100. The value must be greater
than the value of the STARINRADIUS= option. The inner radius of a star is the
distance from the center of the star to the circle that represents the lower limit of the
standardized vertex variables. The lower limit can correspond to the minimum value,
a multiple of standard deviations below the mean, or a lower specification limit.

Restriction Not supported by ActiveX

See   STARINRADIUS= on page 1262

STARSTART=value
specifies the vertex angle for the first variable that is specified in the CHART
statement. Vertex angles for the remaining variables are equally spaced clockwise
and assigned in the order listed. You can specify the value in the following ways:

• Clock position: If you specify the value as a time literal (between '0:00'T and
  '12:00'T), the corresponding clock position is used for the first vertex variable.
  For example, '12:00'T indicates the 12 o'clock position, '03:00'T the 3 o'clock
  position (90 degrees), and '09:00'T the 9 o'clock position (270 degrees).
• **Degrees**: To specify a value in degrees, you must specify a negative number. (This is to distinguish degrees from clock values, which are stored internally as positive numbers.) If you specify a negative number, the absolute value is used for the first vertex angle in degrees. Here, 0 degrees corresponds to 12:00, \(-90\) degrees to 3:00, and \(-270\) degrees to 9:00. Always specify the value in degrees as a negative number.

The default value is zero, so the first vertex variable is positioned at 12:00.

**STARTYPE=CORONA | POLYGON | RADIAL | SPOKE | WEDGE** specifies the style of the stars that are produced for a radar chart. The following keywords are available:

- **CORONA**: polygon with star-vertices emanating from the inner circle
- **POLYGON**: closed polygon
- **RADIAL**: rays emanating from the center
- **SPOKE**: rays emanating from the inner circle
- **WEDGE**: closed polygon with rays from the center to the full spoke length.

**Default**: WEDGE

**SUMV AR=summary-variable** specifies a numeric variable to be used to construct weighted radar charts. The values of **SUMVAR** can be positive, negative, zero, or missing. If **SUMVAR=** is not specified, the weights applied to the chart variable are assumed to be one. The chart is not affected by applying a format to **numeric-variable**.

**See**: “Calculating Weighted Statistics” on page 1266

**Example**: “Example 6: Modifying Chart Axes” on page 1272

**TILELEGEND=(variable)** specifies a variable used to add a legend for CTILES= colors. The variable can have a formatted length less than or equal to 32. If a format is associated with the variable, then the formatted value is displayed. The TILELEGEND= option must be used in conjunction with the CTILES= option for filling the tiles in a chart. If CTILES= is specified and TILELEGEND= is not specified, a color legend is not displayed.

The values of the CTILES= and TILELEGEND= variables should be consistent for all observations with the same level of the classification variables. The value of the TILELEGEND= variable is used to identify the corresponding color value of the CTILES= variable in the legend.

**Restriction**: Not supported by ActiveX

**TILELEGLABEL="label"** specifies a label displayed to the left of the legend that is created when you specify a TILELEGEND= variable. The label can be up to 16 characters and must be enclosed in quotation marks. The default label is *Tiles:*

**Restriction**: Not supported by ActiveX
**URL=character-variable**
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

**Restrictions**
This option affects graphics output that is created through the ODS HTML destination only.

Not supported by the GIF, PNG, and ACTXIMG devices

**See**
“Overview of Enhancing Web Presentations” on page 188
“Example: GIF Output with Drill-Down Links” on page 163

**WAXIS=**
See WFRAME=.

**WFRAME=n**
specifies the width in pixels for the frame lines.

**Alias**
WAXIS=

**Default**
1

**Restriction**
Not supported by ActiveX

**WINDROSE**
The input data set can contain data for four, eight, or 16 compass points. The data must be sorted by direction.

**See**
“About Wind Rose Charts” on page 1244

**Example**
“Example 7: Creating a Wind Rose Chart” on page 1275

**WINDROSECIRCLES=n**
specifies the number of reference circles.

**WSPOKES=line-width**
specifies the width in pixels of the spokes in a radar chart.

**Alias**
WSPOKE=

**Default**
1

**WSTARCIRCLES=(line-widths)**
specifies the width in pixels of the outline of circles requested by the STARCIRCLES= and WINDROSECIRCLES= options.

**Alias**
WSTARCIRCLE=

**Default**
1

**Restriction**
This option is ignored unless the STARCIRCLES= or WINDROSECIRCLES= option is specified.

**WSTARS=line-width | (line-widths)**
specifies the width in pixels of the outline of stars that are produced for a radar chart.

**Alias**
WSTAR=
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:

<table>
<thead>
<tr>
<th>Value of COUNT</th>
<th>Value of HEIGHT</th>
<th>Number of Times the Observation is Used</th>
<th>Value Used for HEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55</td>
<td>1</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>5</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>60</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Examples: GRADAR Procedure

Example 1: Producing a Basic Radar Chart

Features: CHART statement options
          FREQ=

Sample library member: GRRBASIC

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
During the manufacture of a metal-oxide semiconductor (MOS) capacitor, different cleaning processes were used by two manufacturing systems that were operating in parallel. Process A used a standard cleaning solution. Process B used a different cleaning mixture that contained less particulate matter. For five consecutive days the causes of failure with each process were observed, recorded, and saved in the SAS data set called FAILURE.

In a radar chart, the vertices are determined by the levels of a single variable, which is specified in the CHART statement. In this example, the variable CAUSE is specified as the chart variable. The spokes in the chart start at the twelve o'clock position and go in a clockwise order. The output shows that Contamination and Oxide Defects are the most frequently occurring problems.

The FREQ= option specifies variable COUNT to score vertex lengths. Thus, the values of COUNT weigh the contribution of each observation in the computation of the chart statistic.

goptions reset=all border;
title "Capacitor Failures";
proc gradar data=sashelp.failure;
   chart cause / freq=count;
run;
quit;

Example 2: Overlaying Radar Charts

Features: CHART statement options
           FREQ=
           OVERLAYVAR=

Sample library member: GRROVER
Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The most typical way that radar charts are displayed is to overlay the charts on top of each other. To produce an overlay chart, use the OVERLAY= option in the CHART statement. On the OVERLAY= option, specify the classification variable whose values determine the charts to be overlaid. This example shows two blocks of code. For overlay charts with multiple stars, the lines for the stars are rotated through different line styles and colors so that the different stars can be easily seen.

In this example, the OVERLAY= option specifies variable DAY as the overlay variable.

```
goptions reset=all border;
title "Capacitor Failures by Cleaning Process";
proc gradar data=sashelp.failure;
chart cause / freq=count
    overlayvar=process;
run;
quit;
```

Example 3: Tiling Radar Charts

**Features:** CHART statement options
- ACROSSVAR=
- FREQ=
- STARLEGEND=
- STARLEGENDLAB=

**Sample library member:** GRRTILE
As an alternative to overlaying multiple radar charts (see “Example 2: Overlaying Radar Charts” on page 1267), you can tile charts horizontally, vertically, or in both directions (see “Example 4: Using Multiple Classification Variables in Radar Charts” on page 1269) 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.

```sas
options reset=all border;
title "Capacitor Failures by Cleaning Process;"

proc gradar data=sashelp.failure;
  chart cause / acrossvar=process
    freq=count
    starlegend=clock
    starlegendlab="Failure Causes;"
run;
quit;
```

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.

**Example 4: Using Multiple Classification Variables in Radar Charts**

**Features:** CHART statement options
You can study the effects of two classifications simultaneously with a two-way comparative radar chart. This arrangement provides the opportunity to discover both one-way marginal effects and interaction effects. To produce the chart, use both the ACROSSVAR= and DOWNVAR= options.

The ACROSSVAR= option specifies variable DAY as the variable whose values determine the rows in the chart matrix. The DOWNVAR= option specifies variable PROCESS as the variable whose values determine the columns in the chart matrix. The STARTYPE= option determines that the stars are displayed with rays emanating from the inner circle. The NROWS= and NCOLS= options specify the number of rows and columns in the chart. The STARLEGEND=CLOCK option generates a legend that identifies spoke positions. Value CLOCK determines that the positions are identified using a clock metaphor.

```sas
options reset=all border;
title "Capacitor Failures by Cleaning Process and Day";
proc gradar data=sashelp.failure;
    chart cause / acrossvar=day
downvar=process
    freq=count
    startype=spoke;
```

The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
Example 5: Modifying the Appearance of Radar Charts

Features:  
CHART statement options  
CSTARS=  
FREQ=  
LSTARS=  
OVERLAYVAR=  
STARCIRCLES=  
WSTARS=

Sample library member:  
GRRAPEAR

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

For overlay charts with multiple stars, the lines for the stars are rotated through different line styles and colors so that the different stars can be easily seen. Rather than relying on the default rotation patterns, you can control the line colors, widths, and styles with the CSTARS=, LSTARS=, and WSTARS= options.

The CSTARS= option specifies a different color for each of the star outlines in the chart. The WSTARS= option specifies the width of the line for each star outline. The LSTARS= option specifies a solid line as the line style for each star outline. The STARCIRCLES= option determines that two reference circles are superimposed on the star charts. The value 1.0 determines that a circle with a radius equal to the spoke length is displayed. The value 0.5 determines that a circle is displayed halfway 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, \texttt{STARCIRCLES=(0.0 1.0)}
would draw circles at 4 and 12.

```sql
goptions reset=all border;
title "Capacitor Failures by Cleaning Process";

proc gradar data=sashelp.failure;
  chart cause / freq=count
  overlayvar=process
  cstars=(red, blue)
  wstars=2 2
  lstars=1 1
  starcircles=(0.5 1.0)
  cstarcircles=ltgray;
run;
quit;
```

### Example 6: Modifying Chart Axes

**Features:**
- \texttt{AXIS} statement options
  - \texttt{ORDER}=
  - \texttt{VALUE}=
- \texttt{CHART} statement options
  - \texttt{HEIGHT}=
  - \texttt{NOFRAME}
  - \texttt{STARAXIS}=
  - \texttt{STARINRADIUS}=
  - \texttt{SUMVAR}=
- \texttt{FORMAT} procedure

**Sample library member:**
- \texttt{GRRAXIS}

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

You can use \texttt{AXIS} statements to modify the tick marks that appear on the chart spokes and the order in which the tick mark labels appear. This example uses \texttt{AXIS} statements, in addition to formats, to modify the tick marks, tick mark labels, and axis labels on the radar chart spokes.
Program

libname library "library-specification";
proc format library=library;
  value mnth
    1="Jan/Feb" 2="Mar/Apr" 3="May/Jun"
    4="Jul/Aug" 5="Sep/Oct" 6="Nov/Dec";
run;

data goals;
  input month Division $ value @@;
  format month mnth.;
  format value percentn7.0;
datalines;
  1 Parts .43 1 Tools .82
  2 Parts .86 2 Tools .32
  3 Parts .70 3 Tools .65
  4 Parts .35 4 Tools .52
  5 Parts .84 5 Tools .62
  6 Parts .55 6 Tools .43
;run;

goptions reset=all border hsize=5.15in vsize=4.2in;
axis1 order=(0 to .4 by .4, .6 to 1 by .2) value=(height=3pct c=blue tick=1 "");
axis2 order=(0 to .4 by .4, .6 to 1 by .2) value=none;
proc gradar data=goals;
  chart month / sumvar=value
    staraxis=(axis1 axis2 axis2 axis2 axis2 axis2)
    noframe height=3.25
    starinradius=0
    overlayvar=division;
run;
Program Description

Create a user-defined format, MNTH, for the axis labels. Replace library-specification with the path to your library.

libname library "library-specification";
proc format library=library;
value mnth
  1="Jan/Feb"  2="Mar/Apr"  3="May/Jun"
  4="Jul/Aug"  5="Sep/Oct"  6="Nov/Dec";
run;

Create the input data set WORK.GOALS. The user-defined format is applied to the variable MONTH, and the SAS format PERCENTN7.0 is applied to the variable VALUE.

data goals;
  input month Division $ value @@;
  format month mnth.;
  format value percentn7.0;
datalines;
  1 Parts .43  1 Tools .82
  2 Parts .86  2 Tools .32
  3 Parts .70  3 Tools .65
  4 Parts .35  4 Tools .52
  5 Parts .84  5 Tools .62
  6 Parts .55  6 Tools .43
;
run;

Set the graphics environment.

goptions reset=all border hsize=5.15in vsize=4.2in;

Define tick marks for the chart spokes. The AXIS1 statement controls the tick marks for the first (12 o’clock) axis. The ORDER= option specifies the tick mark interval and the actual values that correspond to each tick mark. The format PERCENTN7.0 that was applied to the variable VALUE in the DATA step controls how the values are displayed for the tick mark labels. The VALUE= option specifies the size and color of the tick mark labels. The AXIS2 statement controls all axes except the first. For those axes, the VALUE= option specifies that no tick mark labels are displayed.

axis1 order=(0 to .4 by .4, .6 to 1 by .2) value=(height=3pct c=blue tick=1 "");
axis2 order=(0 to .4 by .4, .6 to 1 by .2) value=none;

Create the radar chart. The STARAXIS= option tells the procedure to apply the AXIS1 statement to the first axis and the AXIS2 statement to all other axes in the chart.

proc gradar data=goals;
  chart month / sumvar=value
    staraxis=(axis1 axis2 axis2 axis2 axis2 axis2)
    noframe height=3.25
    starinradius=0
    overlayvar=division;
run;
Example 7: Creating a Wind Rose Chart

Features:
- CHART statement options
  - NOFRAME
  - SPEED=
  - SUMVAR=
  - WINDROSE

Sample library member:
GRRWNDRS

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

The wind rose chart displays wind direction and speed over a period of time at a specific location. The segments represent the percentage of time that the wind blew from each direction.

goptions reset=all border;

data wind;
  input Direction $ Speed $ Percent @@;
datalines;
N   1-9 1.7 N 10-19 1.0 NE 1-9 1.4
NE 10-19 .8 E 1-9 2.4 E 10-19 1.4
SE 1-9 1.2 SE 10-19 .4 S 1-9 2.7
S 10-19 1.6 SW 1-9 3.7 SW 10-19 3.2
W 1-9 3.1 W 10-19 3.4 NW 1-9 2.1
NW 10-19 1.7
run;

proc gradar data=wind;
Example 8: Creating a Calendar Chart

Features:
- CHART statement options
  - FREQ=
  - CALENDAR
  - CSTARS=
  - OVERLAYVAR=

Sample library member:
- GRRCALEN

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

The CALENDAR option produces a chart with 12 equal-sized segments, one for each month of the year. The color shading of each segment represents the magnitude of the frequency variable. Use the OVERLAY variable to subdivide each segment (for example, by year).

goptions reset=all border;

data climate;
  input Year Month $ Temperature @@;
datalines;
2006 Jan 16 2006 Feb 19 2006 Mar 22 2006 Apr 33
2006 May 41 2006 Jun 60 2006 Jul 55 2006 Aug 41

Example 8: Creating a Calendar Chart

proc gradar data=climate;
  chart month / freq=temperature
  calendar
  overlayvar=year;
  run;
quit;
Chapter 43
GREPLAY Procedure

Overview: GREPLAY Procedure ........................................... 1280
Catalog Entries ................................................................. 1281
Displaying the List of Templates Provided by SAS/GRAPH ............ 1282
Duplicate Entry Names ......................................................... 1283
Ways to Use the GREPLAY Procedure .................................... 1283
Sizing and Naming Your Graphs for Replay (Best Practice) ........... 1283
Considerations for Replaying a Graph .................................... 1284
  Replaying a Graph with a Border ..................................... 1284
  Replaying a Graph with Titles and Footnotes .......................... 1284
Syntax: GREPLAY Procedure ................................................ 1285
  PROC GREPLAY Statement ............................................. 1286
  ? Statement .................................................................. 1290
  BYLINE Statement ....................................................... 1291
  CC Statement ............................................................... 1291
  CCOPY Statement ......................................................... 1291
  CDEF Statement ........................................................... 1292
  CDELETE Statement ..................................................... 1293
  CMAP Statement ......................................................... 1293
  COPY Statement .......................................................... 1294
  DELETE Statement ....................................................... 1294
  DEVICE Statement ....................................................... 1295
  FS Statement .............................................................. 1295
  GOUT Statement .......................................................... 1295
  GROUP Statement .......................................................... 1296
  IGOUT Statement ........................................................ 1296
  LIST Statement ............................................................. 1297
  MODIFY Statement ....................................................... 1297
  MOVE Statement .......................................................... 1299
  NOBYLINE Statement .................................................... 1299
  PREVIEW Statement .................................................... 1300
  QUIT Statement ........................................................... 1300
  REPLAY Statement ....................................................... 1300
  TC Statement .............................................................. 1301
  TCOPY Statement .......................................................... 1301
  TDEF Statement ........................................................... 1302
  TDELETE Statement ...................................................... 1307
  TEMPLATE Statement ................................................... 1307
  TREPLAY Statement ....................................................... 1307
Overview: GREPLAY Procedure

The GREPLAY procedure displays and manages graphics output that is stored in SAS catalogs. The GREPLAY procedure also creates templates and color maps that you can use when you replay your graphics output. The GREPLAY procedure operates in line mode, batch mode, and in the SAS windowing environments.

With the GREPLAY procedure, you can perform any of the following actions:

- Lay out multiple graphs on one page; this output can be used to create dashboards.
- Create an inset graph.
- Select one or more catalog entries from the same catalog for replay, and direct this output to your display or other devices such as plotters and printers.
- Use, create, or modify templates. Use templates to describe positioning on a single display, for graphics output stored in one or more graph catalog entries.
- Create new graphics output by replaying one or more catalog entries into panels within a template.
- Use, create, or modify color maps. Use color maps to map current colors to different colors.
- List templates in SASHELP.TEMPLT.
- Manage GRSEG, TEMPLATE, and CMAP entries in SAS catalogs by doing the following:
- Rearranging or creating logical groupings of catalog entries that contain graphics output.
- Renaming, deleting, or copying catalog entries that contain graphics output, templates, and color maps.

Figure 43.1 on page 1281 shows four catalog entries that were replayed into a template and displayed as a single graph.

**Figure 43.1 Graphics Output in a Template**

![Chart showing age distribution and average height](chart)

---

**Catalog Entries**

The GREPLAY procedure can perform actions on three types of catalog entries:

**GRSEG entries**
store output from SAS/GRAPH procedures. The GREPLAY procedure uses two types of graphics catalogs: the input-catalog and the output-catalog. The input-catalog is the catalog that contains the graphics output that you want to replay. The output-catalog is the catalog in which graphics output that is produced by the template facility is stored. Both of these catalogs are GSEG catalogs. The same GSEG catalog can be used as the input-catalog and the output-catalog.

**TEMPLATE entries**
store templates created with the GREPLAY procedure. The catalog in which template entries are stored is referred to as the template catalog. SAS provides sample templates in SASHELP.TEMPLT. TEMPLATE entries can also be stored in GSEG catalogs.

You can use templates directly from SASHELP.TEMPLT to replay your graphics output, or you can copy these templates to a different catalog and edit the copied entries. Graphics output replayed using a template creates a new GRSEG entry.
CMAP entries
store color maps created with the GREPLAY procedure. The catalog in which color map entries are stored is referred to as the color map catalog. CMAP entries can be stored in GSEG catalogs. They can also be stored in other catalogs. You can copy, edit, or use these color maps to replay your graphics output. Graphics output replayed using a color map does not create a GRSEG entry.

You can store all of the previous entry types in a single SAS catalog, or you can store them in separate catalogs and use a different catalog for each type of entry. A single SAS catalog can contain graphics output, color maps, and templates.

Because the GREPLAY procedure operates on catalog entries, you must assign at least one catalog before you perform any tasks. The GREPLAY procedure has several ways to assign catalogs shown in Table 43.1 on page 1282.

<table>
<thead>
<tr>
<th>Catalog</th>
<th>How to Assign</th>
</tr>
</thead>
</table>
| input     | IGOUT= option in the PROC GREPLAY statement  
                    IGOUT statement  
                    IGOUT field in the PROC GREPLAY window |
| output    | GOUT= option in the PROC GREPLAY statement  
                    GOUT statement  
                    GOUT field in the PROC GREPLAY window |
| template  | TC= option in the PROC GREPLAY statement  
                    TC statement  
                    TC field in the PROC GREPLAY window |
| color map | CC= option in the PROC GREPLAY statement  
                    CC statement  
                    CC field in the PROC GREPLAY window |

Note: Image entries can exist in catalogs, but are not recognized by the GREPLAY procedure.

Displaying the List of Templates Provided by SAS/GRAPH

To write the list of templates stored in SASHELP.TEMPLT to the SAS log, submit the following code:

```
proc greplay nofs
   tc=sashelp.templt;
   list tc;
run;
```
Duplicate Entry Names

The GREPLAY procedure uses the following naming conventions to prevent duplication of names, or overwriting entries:

- For entry names that are less than eight characters long, the procedure adds a numeric suffix to the entry's name. The total number of characters is limited to eight.
- For entry names that are eight characters long or longer, the procedure drops the number of characters that are needed to add a numeric suffix while maintaining the eight-character limit. For example, if you copy an entry TITLEONE to a catalog that already contains an entry with that name, the procedure assigns the name TITLEON1 to the copied entry.

Note: The GREPLAY procedure uses the same naming conventions for entries created by the template facility.

Note: See also “About Filename Indexing” on page 119.

Ways to Use the GREPLAY Procedure

You can view, replay, or manage catalog entries in two ways:

- by submitting code-based GREPLAY procedure statements. The GREPLAY procedure automatically uses code-based statements if you are running in batch mode or in line mode in a non-windowing environment. See “Using Code-based Statements to Run the GREPLAY Procedure” on page 1314.
- 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 1309.

If you are in the SAS windowing environment, you can toggle between the windows and code-based statements while you run the GREPLAY procedure.

For more information, see “FS Statement” on page 1295 and “NOFS ” on page 1289.

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.
Considerations for Replaying a Graph

Replaying a Graph with a Border

In SAS 9.1.3, when you request that a border be drawn around your graph, the border is drawn around the graph inside the graph GRSEG. In SAS 9.2 and later versions, when you request that a border be drawn around your graph, if the ODS output destination that you are using supports drawing a border around the graph, the border is drawn outside of the graph GRSEG by ODS. Otherwise, the border is drawn around the graph inside the graph GRSEG as is done in SAS 9.1.3.

Note: When the BORDER option is used with the GSLIDE procedure, a border is drawn around the slide inside the slide GRSEG regardless of whether the ODS destination supports drawing a border around the graph.

The ODS HTML destination supports drawing a border around a graph when a border is requested, while the ODS LISTING, ODS PDF, ODS RTF, and ODS PRINTER destinations do not. When you replay a graph with a border, if you replay the graph to the same ODS destination that was used to generate the original graph, the graph should replay without any problems. However, in the following replay scenarios, you might encounter unexpected results with respect to the border in the replayed graph:

- When you replay a graph with a border that was generated with the ODS LISTING, PDF, RTF, or PRINTER destination to the ODS HTML destination, the replayed graph might have a double border drawn around it. The double border occurs when the original graph GRSEG already contains a border and the ODS HTML destination draws a second border around it.

- When you replay a graph with a border that was generated with the ODS HTML destination to the ODS LISTING, PDF, RTF, or PRINTER destination, the replayed graph does not have a border. The border is missing because the border in the original graph is not contained in the graph GRSEG.

In the case of a double border, to remove the second border, you can specify the NOBORDER graphics option when you replay the graph. In the case of the missing border, you can create a GREPLAY template that draws a border around the output panel and then replay the GRSEG using that template. In your template code, specify the COLOR=border-color option in the GREPLAY procedure TDEF statement. The COLOR= option draws a border around the graph in the new GRSEG. See “COLOR=border-color” on page 1304 and “Example 4: Replaying a Graph to Multiple Destinations” on page 1326.

Replaying a Graph with Titles and Footnotes

If your graph contains titles, footnotes, or both, and you use the NOGTITLE and NOGFOOTNOTE options on your ODS statement, the titles and footnotes are rendered outside of the graph. They are not included in the GRSEG in that case. See “Controlling...” on page 1284 and “Example 3: Replaying Graphs into a Template” on page 1323.
As a result, when you replay the graphs using the GREPLAY or TREPLAY statement, the titles and footnotes do not appear in the replay output. If you include titles and footnotes with your graphs and you intend to replay the graphs using GREPLAY, make sure that you do not use the NOGTITLE and NOGFOOTNOTE options in your ODS statement.

Syntax: GREPLAY Procedure

Restriction: Not supported by Java or ActiveX
Requirement: Use the NOFS option in the PROC GREPLAY statement when running in a non-windowing environment, batch mode, or in line mode in a windowing environment. At least one statement is required.

Supports: RUN-group processing

Note: Write access to a catalog is needed to modify, add, or delete catalog entries. Only GRSEG entry types can be replayed with the GREPLAY procedure.

Tip: When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

PROC GREPLAY
<BYLINE>
<CC=color-map-catalog>
<CMAP=color-map-entry-type>
<FS>
<GOUT=<libref>:output-catalog>
<IGOUT=<libref>:input-catalog>
<IMAGEMAP=output-data-set>
<NOBYLINE>
<NOFS>
<PRESENTATION>
<TC=template-catalog>
<TEMPLATE=template-entry>;

? 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;
PROC GREPLAY Statement

Determines whether the procedure starts in a windowing or non-windowing environment. Defines whether the session is used for catalog management or output presentation.

Syntax

PROC GREPLAY
  <BYLINE>
  <CC=color-map-catalog>
  <CMAP=color-map-entry-type>
  <FS>
  <GOUT=libref:output-catalog>
  <IGOUT=libref:input-catalog>
  <IMAGEMAP=output-data-set>
  <NOBYLINE>
  <NOFS>
  <PRESENTATION>
  <TC=template-catalog>
  <TEMPLATE=template-entry>;  

Summary of Optional Arguments

BY-line options

  BYLINE
    specifies that the BY statement information for the SAS catalog entries should be displayed.

  NOBYLINE
    specifies that the BY statement information for the SAS catalog entries should be suppressed.
Catalog options

- **CC=** `color-map-catalog` specifies the color map catalog where the color map entries are stored.
- **CMAP=** `color-map-entry-type` specifies the type of catalog entry to use with the GREPLAY procedure.
- **GOUT=** `<libref:` output-catalog` specifies the graphics output catalog.
- **IGOUT=** `<libref:` input-catalog` specifies the input catalog that stores the graphics output that you want to use with the GREPLAY procedure.
- **TC=** `template-catalog` specifies the template catalog to use with the GREPLAY procedure, and identifies the template catalog where the template entry is stored to replay your graphics.
- **TEMPLATE=** `template-entry` identifies the template entry to use with the GREPLAY procedure.

Image map options

- **IMAGEMAP=** `output-data-set` used with the REPLAY statement to create a temporary SAS data set that is used to generate an image map in an SVG file when replaying output to the LISTING destination.

Window options

- **FS** specifies that the GREPLAY procedure should use full-screen windows.
- **NOFS** specifies that the GREPLAY procedure should use line mode.
- **PRESENTATION** specifies that the GREPLAY procedure should open the PRESENTATION window, and use the catalog specified by the IGOUT= option as the input catalog.

Optional Arguments

- **BYLINE** specifies that the BY statement information for the SAS catalog entries should be displayed.

  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.

  Example: “Example 5: Creating a Color Map” on page 1330
**CMAP=**<output-data-set>

specifies the type of catalog entry to use with the GREPLAY procedure. A color map entry option must have a catalog entry type of CMAP.

**Note**
To replay graphics output using a color map entry, you must specify a color map catalog with the CC= option and a color map entry with the CMAP= option.

**Example**
“Example 5: Creating a Color Map” on page 1330

**FS**
specifies that the GREPLAY procedure should use full-screen windows.

Default If your device supports windows, the GREPLAY procedure uses windows. If your device does not support windows, the procedure begins execution in line mode, and the FS option has no effect.

**GOUT=**<libref:output-catalog>
specifies the graphics output catalog. New GRSEG entries or GRSEG entries from other catalogs can be copied to an output catalog. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library, and creates the GSEG catalog if it does not exist.

**Note**
The output catalog can be the same catalog specified in the IGOUT= option.

**See**
“Catalog Entries” on page 1281

**IGOUT=**<libref:input-catalog>
specifies the input catalog that stores the graphics output that you want to use with the GREPLAY procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary WORK library.

**Notes**
The input catalog can be the same catalog specified in the GOUT= option.

If the specified input catalog cannot be found, the GREPLAY procedure stops processing, and the following error message is written to the SAS log:

```
ERROR: Memname input-catalog is unknown.
```

**Example**
“Example 2: Replaying GSLIDE Procedure Output in a Template” on page 1321

**IMAGEMAP=**<output-data-set>

This option is not necessary when you are replaying output to the HTML destination. The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

**Restriction**
The IMAGEMAP= option is valid with the REPLAY statement only.

**See**
“Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options” on page 192

“Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes” on page 198
**NOBYLINE**

specifies that the BY statement information for the SAS catalog entries should be suppressed.

Default: BY statement information is displayed

**NOFS**

specifies that the GREPLAY procedure should use line mode.

Default: If your device does not support windows: NOFS

Example: “Example 1: Creating a Template” on page 1318

**PRESENTATION**

specifies that the GREPLAY procedure should open the PRESENTATION window, and use the catalog specified by the IGOUT= option as the input catalog. The PRESENTATION option is often used in applications to prevent the application users from deleting or reordering catalog entries. You can only replay graphics output from the PRESENTATION window.

Note: The PRESENTATION option overrides the NOFS option on full-screen devices.

**TC=template-catalog**

specifies the template catalog to use with the GREPLAY procedure, and identifies the template catalog where the template entry is stored to replay your graphics.

Note: To replay graphics output using a template catalog, you must also assign the template entry with the TEMPLATE= option.

Example: “Example 1: Creating a Template” on page 1318

**TEMPLATE=template-entry**

identifies the template entry to use with the GREPLAY procedure. The template entry must have a catalog entry type of TEMPLATE.

Note: To replay graphics output using a template entry, you must also assign a current template catalog with the TC= option. If the template entry is not in the template catalog, an error message is written to the SAS log.

Example: “Example 2: Replaying GSLIDE Procedure Output in a Template” on page 1321

**Details**

**Invoking the GREPLAY Procedure**

The mode of operation for the PROC GREPLAY statement depends on both the environment in which the statement is submitted and whether the NOFS option is included.
Table 43.2 Ways of Invoking the GREPLAY Procedure

<table>
<thead>
<tr>
<th>Environment</th>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>windowing</td>
<td>PROC GREPLAY;</td>
<td>GREPLAY procedure windows</td>
</tr>
<tr>
<td>windowing</td>
<td>PROC GREPLAY NOFS;</td>
<td>line mode</td>
</tr>
<tr>
<td>nonwindowing</td>
<td>PROC GREPLAY;</td>
<td>line mode</td>
</tr>
</tbody>
</table>

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

? requiredargument(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.
BYLINE Statement
Displays BY statement information directly beneath the primary description of the catalog entries when you list the input catalog contents.

Note: BY statement information is displayed by default.
See: “NOBYLINE Statement” on page 1299

Syntax
BYLINE;

CC Statement
Specifies a color map catalog, and enables you to change the color map catalog without exiting the procedure.

Syntax
CC libref.color-map-catalog;

Required Argument
color-map catalog
identifies the SAS catalog where color map entries are stored.

CCOPY Statement
Copies a color map from one color map catalog to another color map catalog. Creates a duplicate color map within the color map catalog.

Requirement: Assign a color map catalog before using the CCOPY statement.

Syntax
CCOPY <libref><color-map-catalog>.color-map-entry<.CMAP>;

Required Argument
<libref><color-map-catalog>.color-map-entry<.CMAP>
identifies the color map entry to be copied.
color map catalog
is the color map catalog that contains the color map to be copied.
color map entry
    is the name of the color map entry.

CMAP
    is the color map entry type.

Note  If a color map entry with the same name exists in the color map catalog,
duplicate entry names are resolved as described in “Duplicate Entry Names”
on page 1283.

See  “CC Statement” on page 1291

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.

Requirement:  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

Required Argument
color-map-entry
    identifies a color map entry.

If the color map entry is not in the color map catalog, then the procedure creates a
color map entry. If the color map entry exists in the color map catalog, then the
GREPLAY procedure modifies or adds to that color map entry.
See CC statement

Example “Example 5: Creating a Color Map” on page 1330

Optional Arguments

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.

Note: The GREPLAY procedure does not prompt you to confirm your request to delete color maps entries.

Syntax

CDELETE color map entry(s) | _ALL_ ;

Required Arguments

color map entry(s)
identifies one or more color map entries to delete from the color map catalog. You can submit one entry, or a list of entries in one delete statement.

_ALL_
deletes all of the color map entries from the color map catalog.

Alias CDEL

CMAP Statement

Assigns a color map entry to be used when replaying graphics output.
Requirement: Assign a color map catalog before using the CMAP statement.
See: CC
“Example 5: Creating a Color Map” on page 1330.

Syntax
CMAP color-map-entry;

Required Argument
color-map-entry
identifies the color map entry, contained in the current color map catalog, to use when replaying your graphics output. If the color map entry is not in the current color map catalog, the GREPLAY procedure issues an error message.

COPY Statement
Copies one or more GRSEG catalog entries from the input catalog to the output catalog.

Requirement: Assign an input catalog and an output catalog before using the COPY statement.
Note: The COPY statement cannot create a duplicate catalog entry that contains graphics output in the same catalog.
See: “GOUT Statement” on page 1295 and “IGOUT Statement” on page 1296

Syntax
COPY entry-id(s) | _ALL_;

Required Arguments
entry-id(s)
is the number (in the order in which they were created) or name of a catalog entry or group of entries to be copied from the input catalog to the output catalog. Entries must contain graphics output. Multiple catalog entries can contain both numbers and names.
_ALL_
copies all graphics output entries in the input catalog to the output catalog.

DELETE Statement
Deletes SAS catalog entries containing graphics output from the current input catalog.

Note: The GREPLAY procedure does not prompt you to confirm your request to delete an entry containing graphics output.

Syntax
DELETE entry-id(s) | _ALL_;
**Required Arguments**

`entry-id(s)`

is the number (in the order in which they were created) or name of a catalog entry or a group of entries to be deleted from the input catalog. Entries must contain graphics output. Multiple catalog entries can contain both numbers and names.

_=ALL_  
deletes all graphics output entries in the input catalog.

**DEVICE Statement**

Specifies the device driver.

**Requirement:** You must specify a device driver that your graphics device can support, and is available to your SAS session.

**Syntax**

`DEVICE device-name;`

**Required Argument**

`device-name`

specifies the device driver to use when you replay graphics output. The device driver that you specify becomes the current device. It is used for subsequent replays and the output of other graphics procedures. This device driver remains in effect until you change the device driver.

**FS Statement**

Toggles from line mode to the GREPLAY procedure windows.

**Requirement:** Device must support windows

**See:** “NOFS” on page 1289

**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.
GOUT <libref.:>output-catalog;

Required Argument

<libref.:>output-catalog
identifies the SAS catalog to use as an output catalog.

Default WORK.GSEG

GROUP Statement

Creates groups of entries in the current input catalog.

Syntax

GROUP entry-id(s);

Required Argument

tentry-id(s)
is the number (in the order in which they were created) or name of a catalog entry. All entries specified in the GROUP statement are included in one group, and identified with a group header. You can submit one catalog entry or a list of catalog entries with one GROUP statement. A list of catalog entries can contain both catalog entry numbers and catalog entry names.

Restriction

The number of entries in a group cannot exceed 400.

Details

Only one group can be created per group statement. The default name for a group header is GROUP. The default description for the group header is REPLAY GROUP. Duplicate entry names are resolved as described in “Duplicate Entry Names” on page 1283.

To change the name or description of a group, use the MODIFY statement. To manage and display groups of entries use the DELETE, COPY, and REPLAY statements.

IGOUT Statement

Assigns the SAS input catalog used by the GREPLAY procedure.

Note: You can change the input catalog without exiting the procedure by using the IGOUT statement.

Syntax

IGOUT <libref.:>input-catalog;
**Required Argument**

<libref:input-catalog>
identifies the SAS catalog with entries that contain graphics output that you want to replay.

*Note*  If the specified input catalog cannot be found, the GREPLAY procedure stops processing, and the following error message is written to the SAS log:

```
ERROR: Memname input-catalog is unknown.
```

**LIST Statement**

Lists entries in the input, template, and color map catalogs, as well as the contents of templates and color maps in the SAS log.

*Note:*  Entries are listed in creation date order.

*See:*  “Example 5: Creating a Color Map” on page 1330

**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 Argument

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. When multiple pairs are specified, the entry IDs can be a mix of
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="new-entry-name"
specifies a new name for the GRSEG catalog entry.
The following applies to name:

• The name can be up to 256 characters in length.
• Special characters in the name are converted to underscores.

• For the GRSEG entry name:
  • The name is truncated to eight characters.
  • The first character is always represented in uppercase, and all other
    characters are represented in lowercase.
  • Duplicate names are not allowed. If the name specified already exists in the
    GRSEG catalog, the name modification is rejected.
  • Special characters are not allowed. You must use letters, numbers, and
    underscores.

Default The original entry name

DES="description"
specifies a description of the output.
The maximum length for description is 256 characters. The description does
not appear in the output. The descriptive text is shown in each of the
following:

• the chart description for web output (depending on the device driver that
  you are using). See “Chart Descriptions for Web Presentations” on page
  189 for more information.
• the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
• the description and the properties for the output in the Results window.
• the description and properties for the catalog entry in the Explorer.
• the Description field of the PROC GREPLAY window.

Default        The original graph description

---

**MOVE Statement**

Rearranges entries in the input catalog by moving entries before or after other entries.

**Syntax**

MOVE entry-id-1 AFTER | BEFORE entry-id-2;

**Required Arguments**

*entry-id-1*

specifies the number (in the order in which they were created) or name of a catalog entry or a group header that is to be moved.

*entry-id-2*

is the number (in the order in which they were created) or name of a catalog entry or a group header.

**AFTER | BEFORE**

specifies whether entry-id-1 should be moved 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:**   "BYLINE Statement" on page 1291

**Syntax**

NOBYLINE;
PREVIEW Statement
Displays the panel outlines for one or more templates using the current device. Use the TC statement to specify the template catalog before using the PREVIEW statement.

**Note:** When a template is previewed, graphics output is produced, and stored in a catalog named WORK.GTEM. The temporary catalog is deleted when you end your session.

**Tip:** When previewing templates, press End or Enter, to move to the next template in the list.

**Syntax**
PREVIEW template-entry(s) | _ALL_;

**Required Arguments**
- **template-entry(s)** identifies one or more template entries contained in the template catalog. You can preview one entry or a list of entries with one PREVIEW statement.
- **_ALL_** previews all templates in the current template catalog.

QUIT Statement
Exits the GREPLAY procedure.

**Aliases:** END
STOP

**Syntax**
QUIT;

REPLAY Statement
Identifies one or more entries for replay from the input catalog.

**Alias:** PLAY

**Notes:** If any entry specified in a REPLAY statement is not found in the input catalog, the GREPLAY procedure issues a message to the SAS log. The GREPLAY procedure continues to replay valid entries.

When you replay your graphs, use the same device and orientation that you used when you generated the original graphs. If you use a different device or orientation, your replayed graphs might be distorted.

For a graph with a border, or a graph that includes titles and footnotes, if the border, titles, and footnotes are rendered outside of the GRSEG, they are not included in the replayed graph. See “Considerations for Replaying a Graph” on page 1284.
Syntax

REPLAY entry-id(s) | _FIRST_ | _LAST_ | _ALL_;

Required Arguments

entry-id(s)

is the number (in the order in which they were created) or name of a catalog entry or a group of entries in the input catalog. Entries must contain graphics output. Multiple entries can contain both numbers and names. This statement specifies both the entry named GRAPH, and the third entry in the catalog: replay graph 3;

_ALL_

replays all entries in the input catalog.

_FIRST_

replays the first entry in the input catalog.

_LAST_

replays the last entry in the input catalog.

TC Statement

Specifies the template catalog for the GREPLAY procedure.

Note: SAS supplies several templates in the SASHELP.TEMPLT catalog.

Tip: Use the TC statement to change the template catalog without exiting the procedure.

Syntax

TC template-catalog;

Required Argument

template-catalog

identifies the SAS catalog where templates are to be stored or identifies the name of a SAS catalog that contains templates.

TCOPY Statement

Copies templates from a catalog to the template catalog, or creates a duplicate of a template within the template catalog.

Requirement: Assign a template catalog before using the TCOPY statement.

See: “TC Statement” on page 1301.

Syntax

TCOPY <template-catalog->template-entry<.TEMPLATE>>;
**Required Argument**

`<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 1283.

**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:

```sas
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:

```sas
tcopy newtemp.template;
```

---

**TDEF Statement**

Defines or modifies templates in the template catalog.

**Requirement:** Assign a template catalog before using the TDEF statement.

**See:** TC statement

“Example 1: Creating a Template” on page 1318

**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

Summary of Optional Arguments

Appearance options

CLIP
specifies that any panel behind this panel should be clipped.

COLOR=border-color
specifies the panel border color.

Catalog entry description options

DES="description"
specifies the template entry description.

Panel coordinate options

LLX=x
specifies the X coordinate of the lower left corner of the panel.

LLY=y
specifies the Y coordinate of the lower left corner of the panel.

LRX=x
specifies the X coordinate of the lower right corner of the panel.

LRY=y
specifies the Y coordinate of the lower right corner of the panel.

ULX=x
specifies the X coordinate upper left corner of the panel.

ULY=y
specifies the Y coordinate upper left corner of the panel.

URX=x
specifies the X coordinate upper right corner of the panel.

URY=y
specifies the Y coordinate upper right corner of the panel.

XLATEX=distance
specifies the distance to move the X coordinates of the panel.
XLATEY=distance
specifies the distance to move the Y coordinates of the panel.

**Panel management options**

COPY=panel-number
specifies the panel number definition to be copied to this panel.

DEF
specifies a default panel with specific coordinates.

DELETE
deletes a panel.

PANEL-NUMBER=
identifies the panel number being defined or modified.

**Panel rotation options**

ROTATE=degrees
specifies the rotation angle for the panel.

**Panel scaling options**

SCALEX=factor
specifies the scale factor for the X coordinates in the panel.

SCALEY=factor
specifies the scale factor for the Y coordinates in the panel.

**Required Argument**

`template-entry`
identifies an existing or new template. If the template is not in the template catalog, it is created by the GREPLAY procedure. If the `template-entry` is in the template catalog, it is modified by the procedure.

Only one template entry is required, but if you specify only the template name without any option, modifications are not made, and a template is not created.

**Optional Arguments**

CLIP
specifies that any panel behind this panel should be clipped. Only the graphics output to be placed in the CLIP panel can appear in the space that the panel occupies. If a previous panel occupies all or part of that space, CLIP is ignored.

COLOR=border-color
specifies the panel border color. In SAS 9.4M4 and in earlier releases, you can specify a valid color name or color code that does not exceed 8 characters. Starting with SAS 9.4M5, you can specify a valid color in any of the color-naming schemes described in “Color-Naming Schemes” on page 317 that does not exceed 64 characters or a valid color code that does not exceed 8 characters.

**Restriction**
In SAS 9.4M4 and in earlier releases, the color name cannot exceed 8 characters.

**Notes**
If you do not specify a border color, then the GREPLAY procedure does not draw a border around the output panel when you replay graphics output to the output panel. In that case, unless the original graphic output contains a border around the graphic, the output panel
A template that contains a panel without a border color is assigned a color when it is previewed with the PREVIEW statement.

**COPY=**`panel-number`

specifies the panel number definition to be copied to this panel.

**DEF**

specifies a default panel with specific coordinates.

### Table 43.3 Default Panel Characteristics

<table>
<thead>
<tr>
<th>Panel Corner</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower left</td>
<td>(0,0)</td>
</tr>
<tr>
<td>upper left</td>
<td>(0,100)</td>
</tr>
<tr>
<td>upper right</td>
<td>(100,100)</td>
</tr>
<tr>
<td>lower right</td>
<td>(100,0)</td>
</tr>
</tbody>
</table>

**DELETE**

deletes a panel.

Alias **DEL**

**DES=**"description"

specifies the template entry description. The maximum length for the template entry description is 256 characters.

Default *** new template ***

**LLX=x**

specifies the X coordinate of the lower left corner of the panel. Units for `x` are a percentage of the graphics output area.

**LLY=y**

specifies the Y coordinate of the lower left corner of the panel. Units for `y` are a percentage of the graphics output area.

**LRX=x**

specifies the X coordinate of the lower right corner of the panel. Units for `x` are a percentage of the graphics output area.

**LRY=y**

specifies the Y coordinate of the lower right corner of the panel. Units for `y` are a percentage of the graphics output area.

**PANEL-NUMBER=**

identifies the panel number being defined or modified.

**ROTATE=**`degrees`

specifies the rotation angle for the panel. Panel corner coordinates are automatically adjusted.
SCALEX=\textit{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=\textit{factor}
specifies the scale factor for the Y coordinates in the panel. Use this scale factor to increase or decrease the panel size in the Y direction, or to reverse the Y coordinates for the panel.

ULX=\textit{x}
specifies the X coordinate upper left corner of the panel. Units for \textit{x} are a percentage of the graphics output area.

ULY=\textit{y}
specifies the Y coordinate upper left corner of the panel. Units for \textit{y} are a percentage of the graphics output area.

URX=\textit{x}
specifies the X coordinate upper right corner of the panel. Units for \textit{x} are a percentage of the graphics output area.

URY=\textit{y}
specifies the Y coordinate upper right corner of the panel. Units for \textit{y} are a percentage of the graphics output area.

XLATEX=\textit{distance}
specifies the distance to move the X coordinates of the panel. Units for \textit{distance} are a percentage of the graphics output area.

XLATEY=\textit{distance}
specifies the distance to move the Y coordinates of the panel. Units for \textit{distance} are a percentage of the graphics output area.

\textbf{Details}

To zoom in on the graphics output, use coordinate values less than 0 and greater than 100. These values can be used with the LLX= option, LLY= option, LRX= option, LRY= option, ULX= option, ULY= option, URX= option, and the URY= option. You can see the replayed graphics output portion in the graphics output area in the range from 0\% to 100\%.

The values that you specify for the SCALEX= option, and the SCALEY= option are used to change the size and panel orientation. The scale factors are used for the corresponding X and Y panel coordinates. If you submit:

```
scalex=.5
scaley=2
```

the X coordinates are scaled to half the original size, and the Y coordinates are scaled to twice the original size.

If you supply a scale factor of 0, all of the coordinates are set to the same value. If you use a scale factor of 1, nothing happens. If you use a scale factor greater than 1, the values of the coordinates are increased and hence the size of the panel increases. If you use a scale factor less than 1 but greater than 0, the values of the coordinates are reversed, and the panel (and any graphics output replayed in the panel) is reversed.
TDELETE Statement

Deletes templates from the template catalog.

Alias: TDEL

Note: The GREPLAY procedure does not prompt you to confirm your request to delete templates.

Syntax

TDELETE template-entry(s) | _ALL_;

Required Arguments

template-entry(s)

identifies a template entry to be deleted from the template entry catalog. You can submit one entry or a list of entries in one TDELETE statement.

_ALL_

deletes all template entries in the template entry catalog.

TEMPLATE Statement

Assigns a template to use when replaying graphics output.

Requirement: Assign a template catalog before using the TEMPLATE statement.

Note: If you specify a template that is not in the template catalog, before you assign a template entry catalog, the GREPLAY procedure issues an error message.

Syntax

TEMPLATE template-entry;

Required Argument

template-entry

identifies an existing template to use when replaying graphics output. Use the TREPLAY statement to replay graphics output in the template.

Example  “Example 1: Creating a Template” on page 1318

TREPLAY Statement

Replays graphics entries into template panels. TREPLAY copies one or more entries from the graphics input catalog into a template-entry in the graphics output catalog, using positioning information provided by the template.

Alias: TPLAY
Requirement: Before issuing the TREPLAY statement, specify a graphics input catalog with the GREPLAY IGOUT statement, assign a template entry catalog with the GREPLAY TC statement, and choose a template with the GREPLAY TEMPLATE statement.

Notes: When you replay your graphs, use the same device that you used when you generated the original graphs. If you use a different device, your replayed graphs might be distorted.

For a graph with a border, or a graph that includes titles and footnotes, if the border, titles, and footnotes are rendered outside of the GRSEG, they are not included in the replayed graph. See “Considerations for Replaying a Graph” on page 1284.

See: “IGOUT Statement” on page 1296
“TC Statement” on page 1301
“TEMPLATE Statement” on page 1307
“Example 2: Replaying GSLIDE Procedure Output in a Template” on page 1321

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 Argument

template-panel-number:entry-id

specifies the template panel number and the name of the template entry.

  template-panel-number

  specifies the number of the panel in the template into which you want to replay the entry. This number determines the position of the graph in the new entry in the graphics output catalog.

  entry-id

  specifies the name or number of the entry in the graphics input catalog that is to be added to the new entry in the graphics output catalog.

Optional Arguments

DES=“description”

specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

  • the chart description for web output (depending on the device driver that you are using). See “Chart Descriptions for Web Presentations” on page 189 for more information.
  • the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
  • the description and the properties for the output in the Results window.
  • the description and properties for the catalog entry in the Explorer.
  • the Description field of the PROC GREPLAY window.

Default Graphics Replay
NAME="entry-name"

specifies the entry name for the template.

The following applies to name:

• The name can be up to 256 characters in length.
• Special characters in the name are converted to underscores.
• **For the GRSEG entry name:**
  • The name is truncated to eight characters.
  • The first character is always represented in uppercase, and all other characters are represented in lowercase.
  • If the name duplicates an existing name, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique graph name (for example, name1, name2, and so on).

Default: **TEMPLATE**

See: “Duplicate Entry Names” on page 1283

**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:

```sas
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**

**Opening the GREPLAY Procedure Windows**

In the SAS windowing environment, you can use the GREPLAY windows instead of code-based statements to replay and manage catalog entries.

*Note:* The GREPLAY windows are not available in SAS Studio.

You perform tasks that use the GREPLAY procedure windows by entering values in the fields that are displayed in the windows and by issuing commands from the command line. To open the GREPLAY windows, submit the PROC GREPLAY statement without the NOFS option:

```sas
proc greplay;
```

SAS/GRAPH then opens the PROC GREPLAY window. The GREPLAY procedure has five windows:

• PROC GREPLAY window
• PRESENTATION window
• DIRECTORY window
Figure 43.2 on page 1310 shows how these windows relate to each other. Each window can be scrolled backward or forward as needed to display additional fields and information.

**GREPLAY Window Commands**

When using GREPLAY windows, tasks are performed by entering values in the fields. Each window can be scrolled forward, or backward to display additional fields and information. You can navigate, and manipulate GREPLAY windows by entering commands on the command line. For a complete description of each window and its fields, open the Help for the GREPLAY windows. You can open the Help by pressing the F1 key or by selecting Help → Using This Window. For information about navigating among these windows, see “Commands for Using the GREPLAY Procedure Windows” on page 1313.

**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.
This window replays graphics output without modifying or deleting entries, templates, or color maps. Once you have created and organized your catalog, you can use the PRESENTATION window in an application for replaying graphics output.

DIRECTORY Window

This window lists the catalog entry names, gives a brief description of each entry, and indicates the date on which each entry was created or last modified. Although all catalog entry types are displayed in the DIRECTORY window, you can manage only entries of the type CMAP and TEMPLATE from this window.

This window enables you to design templates that are used to present graphics. Templates are designed by specifying the coordinates of its panels and assigning the order in which panels are filled. Once you have entered the coordinates of a panel, you can modify them by using the Scale, Xlate (translate), and Rotate utility fields. These utility fields recalculate the coordinate values automatically.
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 43.6  Color Mapping Window
## Commands for Using the GREPLAY Procedure Windows

<table>
<thead>
<tr>
<th>Location</th>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GREPLAY statement</td>
<td>Open PROC GREPLAY window.</td>
<td>Submit the PROC GREPLAY statement without using the PRESENTATION or NOFS options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open PRESENTATION window.</td>
<td>Submit the PROC GREPLAY statement and include the PRESENTATION and IGOUT= options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC GREPLAY window</td>
<td>Open PRESENTATION window.</td>
<td>Specify a catalog and issue the PRES command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open DIRECTORY window.</td>
<td>Specify a template catalog and issue the TC command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify a color map catalog and issue the CC command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open TEMPLATE DESIGN window</td>
<td>Specify a template catalog and issue the following command: edit template-name.template</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open COLOR MAPPING window.</td>
<td>Specify a color map catalog and issue the following command: edit color-map-name.cmap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECTORY window</td>
<td>Open TEMPLATE DESIGN window</td>
<td>Place an S beside the name of an existing template.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue the following command: edit template-name.template</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Open COLOR MAPPING window.</td>
<td>Place an S beside the name of an existing color map.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Issue the following command: edit color-map-name.cmap</td>
</tr>
</tbody>
</table>
Using Code-based Statements to Run the GREPLAY Procedure

**Invoking the GREPLAY Procedure**

If you prefer to run code-based statements in a windowing environment, invoke the GREPLAY procedure with the NOFS option as follows:

```plaintext
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 43.5** GREPLAY Procedure Statements for Managing Color Maps, Templates, and Catalogs

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign a color map catalog</td>
<td>CC statement</td>
</tr>
<tr>
<td>copy a color map from another catalog, or within the same catalog</td>
<td>CCOPY statement</td>
</tr>
<tr>
<td>define or modify a color map in the current catalog</td>
<td>CDEF statement</td>
</tr>
<tr>
<td>assign a color map to use when you replay graphics output</td>
<td>CMAP statement</td>
</tr>
<tr>
<td>delete unneeded GRSEG entries</td>
<td>DELETE statement</td>
</tr>
<tr>
<td>assign a template catalog</td>
<td>TC statement</td>
</tr>
<tr>
<td>copy a template from another catalog, or within the same catalog</td>
<td>TCOPY statement</td>
</tr>
<tr>
<td>delete a template</td>
<td>TDELETE statement</td>
</tr>
<tr>
<td>define a template</td>
<td>TDEF statement</td>
</tr>
<tr>
<td>display the panel outlines for a template</td>
<td>PREVIEW statement</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy GRSEG entries from an input catalog to an output catalog*</td>
<td>COPY statement</td>
</tr>
<tr>
<td>group GRSEG entries</td>
<td>GROUP statement</td>
</tr>
<tr>
<td>move GRSEG entries</td>
<td>MOVE statement</td>
</tr>
<tr>
<td>delete GRSEG entries</td>
<td>DELETE statement</td>
</tr>
<tr>
<td>modify GRSEG entry names or descriptions in an input catalog</td>
<td>MODIFY statement</td>
</tr>
<tr>
<td>replay GRSEG entries from an input catalog</td>
<td>REPLAY statement</td>
</tr>
<tr>
<td>replay GRSEG entries into template panels</td>
<td>TREPLAY statement</td>
</tr>
</tbody>
</table>

* You must assign an output catalog before copying graphics output.

Using the GREPLAY Procedure to Replay Catalog Entries

To select catalog entries for replay, first assign an input catalog that contains the graphics output that is to be replayed. Then assign the entry with the REPLAY statement. To select a catalog entry or entries for replay:

1. Start the GREPLAY procedure.
2. Define the input catalog that contains the graphics to be replayed with the IGOUT= option.
3. Specify the entry or entries that you want to replay (GCHART in the example that follows) with the REPLAY statement.
4. End the GREPLAY procedure with the QUIT statement.
For example, the following statements replay the GRSEG entry named GCHART from the catalog WORK.GSEG, which is assigned with the IGOUT= option:

```sql
proc greplay igout=work.gseg nofs;
   replay gchart;
quit;
```

To replay all the graphics output stored in the WORK.GSEG catalog submit this code:

```sql
proc greplay nofs;
   igout work.gseg;
   replay _all_ ;
quit;
```

*Note: Graphics output is created only when you use the GREPLAY procedure with a template.*

### Using the GREPLAY Procedure to Create Custom Templates

You can use the GREPLAY procedure to create custom templates. Custom templates are typically used to perform the following actions:

- control the layout of multiple graphs on one page, which is useful for dashboards
- replay graphics output from several catalog entries, or from the same catalog, on one display or page
- change the shape of your graphics output
- change the size of your graphics output

To define and view a custom template:

1. Start the GREPLAY procedure with the NOFS option.
2. Assign a template catalog with the TC= option.
3. Define a template with the TDEF statement.
4. Preview the template with the PREVIEW statement.
5. End the GREPLAY procedure with the QUIT statement.

Before you create a template, you must assign a template catalog. If you are using the GREPLAY procedure in line mode, use the TDEF statement to define a template and the PREVIEW statement to preview a template. For example, the following statements define and preview a template named TEMPLT:

```sql
proc greplay nofs tc=sasuser.cat;
   tdef templt 1/def;
   preview templt;
quit;
```
Using the GREPLAY Procedure to Replay Graphics Output in a Template

You can use the GREPLAY procedure to create new graphics output by replaying existing graphics output in templates. Templates are often used to replay several graphics entries from the same catalog on one display or page. The GREPLAY procedure creates new graphics output when replaying graphics output with a template.

You can create your own templates, or you can use the templates provided with SAS/GRAPH that are stored in the SASHELP.TEMPLT catalog.

The following guidelines describe how to generate two graphs, and replay the graphics output on one page using a SASHELP.TEMPLT entry:

1. Generate two graphs with PROC GCHART using the default names (GCHART GCHART1).
2. Start the GREPLAY procedure with the NOFS option specified.
3. Define the input catalog with the IGOUT= option (WORK.GSEG).
4. Assign the template catalog with the TC= option (SASHELP.TEMPLT contains the template entries).
5. Assign a template to replay your graphs with the TEMPLATE statement (V2).
6. Assign the graphs that you want to replay with the TREPLAY statement (GCHART GCHART1).
7. End the GREPLAY procedure with the QUIT statement.

For example, the following statements replay the entries GRAPH1 and GRAPH2 into the V2 template, which is stored in the catalog SASHELP.TEMPLT. The TC statement specifies the catalog that contains the template, and the TEMPLATE statement specifies the template. The TREPLAY statement assigns each entry to a panel. (The V2 template has two panels, so there is an assignment for panel 1 and panel 2.)

```sas
proc gchart data=sashelp.class;
   hbar age/discrete;
   run;
   hbar height;
   run;
quit;
proc greplay igout=work.gseg nofs;
   tc sashelp.templt;
   template v2;
   treplay 1:gchart 2:gchart1;
quit;
```

Note: If the GOUT= option is not specified when creating the charts, then the output is stored in the temporary WORK.GSEG catalog.

When you replay graphics output in a template, the new GRSEG output that is created by the GREPLAY procedure is automatically provided a default name, Template, and it is stored in the output catalog WORK.GSEG. The default GRSEG description is “Graphics Replay”.

Using the GREPLAY Procedure to Create Color Maps

Color maps are useful for assigning unavailable colors on your current device to your graph. A color map is a list of up to 256 pairs of colors. By mapping the original colors to a different list of colors, you can change the colors in your graphics output.

To create a color map named CLRMAP, perform the following actions:

1. Start the GREPLAY procedure with the NOFS option.
2. Assign a color map catalog with the CC= option.
3. Define the output catalog with the GOUT= option.
4. Define a color map with the CDEF statement.
5. Remap your colors.
6. End the GREPLAY procedure with the QUIT statement.

Before you create a color map, you must assign a color map catalog. The following example defines a color map named CLRMAP:

```plaintext
proc greplay cc=clrmap gout=work nofs;
   cdef clrmap 1 / cyan : blue;
quit;
```

When you assign a color map and replay graphics output, the following occurs:

- The stored GRSEG entry or entries, retain the original foreground colors.
- The colors used to replay the graphics are not saved with the original graphics output.
- Graphics output is not created when you replay graphics output using a color map.

Examples: GREPLAY Procedure

Example 1: Creating a Template

<table>
<thead>
<tr>
<th>Features:</th>
<th>REPLAY statement options:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOFS</td>
</tr>
<tr>
<td></td>
<td>TC=</td>
</tr>
<tr>
<td></td>
<td>TDEF statement</td>
</tr>
<tr>
<td></td>
<td>TEMPLATE statement</td>
</tr>
</tbody>
</table>

| Sample library member: | GRECRTM1 |
Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

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 43.1 on page 1319 shows the template definition written to the SAS log file. The template defined here is also used in “Example 2: Replaying GSLIDE Procedure Output in a Template” on page 1321.

Output 43.1  Defining a Template

```
64        /* list template contents */
65      list template;
66   quit;
```

Program
goptions reset=all border;
proc greplay tc=work.tempcat nofs;
tdef newtemp des="Five panel template"
   1/llx=0 lly=10
      ulx=0 uly=50
      urx=50 ury=50
      lrx=50 lry=10
      color=navy

   2/llx=0 lly=50
      ulx=0 uly=90
      urx=50 ury=90
      lrx=50 lry=50
      color=lime

   3/llx=50 lly=50
      ulx=50 uly=90
      urx=100 ury=90
      lrx=100 lry=50
```
color=yellow

4/l1x=50 lly=10
ulx=50 uly=50
urx=100 ury=50
lrx=100 lry=10
color=cyan

5/l1x=0 lly=0
ulx=0 uly=100
urx=100 ury=100
lrx=100 lry=0
color=lipk;
template newtemp;
list template;
quit;

Program Description
Here is a detailed example of the example program.

Set the graphics environment.

goptions reset=all border;

Start the GREPLAY procedure. NOFS starts the procedure in line-mode. The TC=
option assigns TEMPCAT as the template catalog.

proc greplay tc=work.tempcat nofs;

Define a template with five panels. The TDEF statement defines a template named
NEWTEMP, and places it in the previously defined template catalog. Each definition
identifies the panel number, and specifies the four corner's coordinates. The COLOR=
option draws a border for each panel in the specified color.

tdef newtemp des="Five panel template"
  1/l1x=0 lly=10
   ulx=0 uly=50
   urx=50 ury=50
   lrx=50 lry=10
   color=navy

  2/l1x=0 lly=50
   ulx=0 uly=90
   urx=50 ury=90
   lrx=50 lry=50
   color=lime

  3/l1x=50 lly=50
   ulx=50 uly=90
   urx=100 ury=90
   lrx=100 lry=50
   color=yellow

  4/l1x=50 lly=10
   ulx=50 uly=50
Assign the template. The TEMPLATE statement assigns the created template NEWTEMP as the template.

```
template newtemp;
```

Write the template contents to the SAS log.

```
list template;
quit;
```

---

**Example 2: Replaying GSLIDE Procedure Output in a Template**

**Features:**
GREPLAY statement options:
- GOUT=,
- IGOUT=
- TEMPLATE=
TREPLAY statement

**Other features:**
PROC GSLIDE

**Sample library member:**
GRERGOT1

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

This example requires template NEWTEMP, which is created in “Example 1: Creating a Template” on page 1318. In order to run this program, you must first run the program in “Example 1: Creating a Template” on page 1318 to create the NEWTEMP template. Otherwise, an error will occur in this program.

This program uses template NEWTEMP, which is created by the program in “Example 1: Creating a Template” on page 1318. The TREPLAY statement in this example replays four catalog entries that contain graphics output into template NEWTEMP. The NEWTEMP template contains four equally sized panels, and one large, full-size panel. Note that assignments are made to all but one panel. Because the fourth panel is not listed in the TREPLAY statement, it does not appear in the graphics output. The HSIZE= option, and the VSIZE= option are adjusted, to reflect the overall output dimension. Alternatively, you could use XPIXELS= and YPIXELS= to adjust the graphics output size.
**Program**

```sas
options reset=all border hsize=5.14in vsize=4.2in;
proc gslide gout=grafcat;
   title c=navy "Graph Number Three";
   run;
   title c=lime "Graph Number One";
   run;
   title c=orange "Graph Number Two";
   run;
proc gslide gout=grafcat;
   title c=purple "Common Title";
   footnote c=blue "Common Footnote";
   run;
proc greplay igout=grafcat gout=excat tc=tempcat nofs;
   template=newtemp;
   treplay 1:gslide
      2:gslide1
      3:gslide2
      5:gslide3;
   quit;
```

**Program Description**

Here is a detailed example of the example program.

*Set the graphics environment.* The HSIZE= option, and the VSIZE= option are set for the overall output dimensions.

```sas
options reset=all border hsize=5.14in vsize=4.2in;
```
Generate three graphs in the WORK.GRAFCAT catalog. The GSLIDE procedure creates three text slides, and stores them in GRAFCAT as specified by the GOUT= option. These are stored as GSLIDE, GSLIDE1, and GSLIDE2.

```plaintext
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.

```plaintext
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.

```plaintext
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.

```plaintext
treplay 1:gslide
  2:gslide1
  3:gslide2
  5:gslide3;
quit;
```

---

**Example 3: Replaying Graphs into a Template**

**Features:**

- GREPLAY statement options:
  - IGOUT=
  - GOUT=
  - NOFS
  - TC=
  - TEMPLATE=
- DEVICE statement
- TREPLAY statement

This example replays four graphs into template L2R2. The following figure shows the output of this example program.
Program

data work.classtemp (drop=name );
   length Gender $ 6;
   set sashelp.class;
   if sex="F" then Gender="Female";
   else Gender="Male";
run;
proc sort data=work.classtemp out=work.class;
   by weight height;
run;

goptions reset=all hsize=2.75in vsize=2.06in device=png nodisplay;
axis1 label=none style=0 major=none value=none;
axis2 label=('Age');
axis3 label=('Height') order=50 to 75 by 5;
axis4 label=('Weight') order=50 to 150 by 25 minor=(n=1);
legend1 label=none value=('Male' 'Female') Position=(right middle outside)
   across=1;
legend2 label=none value=('Male' 'Female');
symbol i=join;
proc gchart data=work.class gout=work.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=work.class gout=work.excat;
Here is a detailed example of the example program.

Prepare the data for the graphs. Drop variables NAME and SEX, and add a variable called GENDER. Change variable values, and sort the data for the line graph.

```
data work.classtemp (drop=name);
    length Gender $ 6;
    set sashelp.class;
    if sex="F" then Gender="Female";
    else Gender="Male";
run;
proc sort data=work.classtemp out=work.class;
    by weight height;
run;
```

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 device=png nodisplay;
```

Create axes definitions for the graphs.

```
axis1 label=none style=0 major=none value=none;
axis2 label="Age";
axis3 label="Height" order=50 to 75 by 5;
axis4 label="Weight" order=50 to 150 by 25 minor=(n=1);
```

Create legend definitions for the graphs.

```
legend1 label=none value="(Male* Female)" Position=(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 WORK.EXCAT catalog. Generate a vertical bar chart, a horizontal bar chart, a pie chart, and a subgrouped plot.

```
proc gchart data=work.class gout=work.excat;
    vbar age/discrete hminor=0 subgroup=gender
        inside=freq raxis=axis1 maxis=axis2
    noframe legend=legend1;
```
Define the size of the template. Each graph is replayed into a separate panel in the template. The template size accommodates the four smaller graphs.

goptions reset=all hsize=5.5in vsize=4.12in;

Replay the graphs with a template to create one graph. The graphs stored in WORK.EXCAT are replayed to create one graph. The graph is also stored in the WORK.EXCAT catalog.

proc greplay gout=work.excat igout=work.excat nofs
c  tc=sashelp.templt template=l2r2;
device png;
treplay 1:gchart 2:gchart1 3:gchart2 4:gplot;
quit;

Example 4: Replaying a Graph to Multiple Destinations

Features: GREPLAY statement options:
IGOUT=
NOFS
TC=
DELETE statement
TDEF statement
TDELETE statement
TREPLAY statement

Other features:
FILENAME statement
ODS HTML statement
ODS PDF statement
ODS RTF statement
GOPTIONS statement:
BORDER
DEVICE=
GSFNAME=
NOBORDER
NODISPLAY
XPIXELS=
YPIXELS=

OPTIONS statement ORIENTATION= option
PROC GCHART
The GREPLAY procedure replays the graph shown in Output 43.2 on page 1327 to a TIF file, and the ODS HTML, PDF, and RTF destinations. A template is used to generate the initial graph. The template creates a margin and a blue border around the graph. The XPIXELS= and YPIXELS= graphics options are used to set the graph size. The GREPLAY procedure TREPLAY statement is used to replay the graph to each of the destinations. For the ODS PDF and RTF destinations, the system option ORIENTATION= is set to LANDSCAPE to orient the graph in landscape in both documents.

Output 43.2  Vertical Bar Chart Exported to Multiple Destinations

```
Program

%let goutpath=.;
%let goutname=height;

proc greplay nofs tc=work.tempcat igout=work.gseg;
   delete &goutname;
   tdelete grtemp;
run;
quit;

proc greplay tc=work.tempcat nofs;
   tdef grtemp des="One panel template with border"
      l/llx=5  lly=5
```

Here is a detailed example of the example program.

**Define variables to use for specifying the graph output path and filename.** The output path is set to the current working directory in this example. Using macro variables makes it easier to change the path and filename, if necessary.

%let goutpath=.;
%let goutname=height;

**Delete the old GRSEG and template, if they exist.** The DELETE statement deletes entry HEIGHT from the WORK catalog, which is specified by the IGOUT= option. The TDELETE statement deletes template GRTEMP in the WORK.TEMPCAT catalog, which is specified by the TC= option.
Create a one-panel template for the graph. This template specifies a 5% margin around the graph. The COLOR= option specifies a blue border around the graph.

```
proc greplay tc=work.tempcat nofs;
   tdef grtemp des="One panel template with border"
   1/llx=5   lly=5
   ulx=5   uly=95
   urx=95  ury=95
   lrx=95  lry=5
   color=cx7C95CA;
run;
quit;
```

Set the graphics options. The GOPTIONS statement NODISPLAY option prevents the chart from being displayed when it is generated. The NOBORDER option disables the graph border, since template GRTEMP draws its own border. Options XPIXELS= and YPIXELS= set the graph dimensions to 520-by-520 pixels.

```
goptions reset=all nodisplay noborder xpixels=520 ypixels=520;
```

Close the currently open ODS destinations.

```
ods _all_ close;
```

Open the LISTING destination. Use the LISTING style.

```
ods listing style=listing;
```

Generate the chart. The NAME= option in the VBAR statement sets the name of the GRSEG entry to HEIGHT.

```
title "Average Height by Age and Sex";
proc gchart data=sashelp.class;
   vbar age / sumvar=height discrete type=mean group=sex name="&goutname";
run;
quit;
```

Replay GRSEG entry HEIGHT to TIFF file height.tif. The GOPTIONS statement DEVICE= option sets the output device to TIFFP, and the GSFNAME= option specifies the output filename as height.tif. The TREPLAY statement replays GRSEG entry HEIGHT to entry THEIGHT using the panel 1 template in template GRTEMP.

```
goptions display device=tiffp gsfname=tifout;
proc greplay nofs tc=work.tempcat igout=work.gseg;
   template grtemp;
   treplay 1:&goutname name="t&goutname";
run;
quit;
```
Change device to PNG and disable borders. Because a border is included in the original GRSEG, the GOPTIONS statement NOBORDER option is used to prevent ODS from drawing another border around the image.

```sas
options reset=all device=png noborder xpixels=520 ypixels=520;
```

Replay GRSEG entry HEIGHT to the HTML, PDF, and RTF destinations. The OPTIONS statement ORIENTATION=LANDSCAPE option sets the graph orientation to landscape in the PDF and RTF documents. This option does not affect the ODS HTML destination. The LISTING style is used with each destination.

```sas
options orientation=landscape;
ods listing close;
ods html path="&goutpath" file="&goutname..html" style=listing;
ods pdf file="&goutpath\&goutname..pdf" style=listing;
ods rtf file="&goutpath\&goutname..rtf" style=listing image_dpi=72;
proc greplay nofs tc=work.tempcat igout=work.gseg;
    template grtemp;
        treplay 1:&goutname name="t&goutname";
    run;
quit;
```

Reset the ORIENTATION= system option to PORTRAIT.

```sas
options orientation=portrait;
```

Close all of the open destinations, and then open the HTML destination.

```sas
ods _all_ close;
ods html; /* Not required in SAS Studio */
```

---

### Example 5: Creating a Color Map

**Features:**
- GREPLAY statement options:
  - CC=
  - GOUT=
- CDEF statement
- CMAP statement
- LIST statement

**Sample library member:**
- GRECRRCM1

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

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 43.3 on page 1331 shows a partial SAS log listing.
Output 43.3  Defining a Color Map (GRECRCM1)

```
75       /* list  color map contents */
76     list cmap;
MYCOLOR       Special Color Map
   FROM       TO
   1    PINK      RED
   2    CYAN      BLUE
   3    LIG       GREEN
77  quit;
```

Program

```
goptions reset=all border;
proc greplay cc=clrmap gout=excat nofs;
cdef mycolor des="Special Color Map"
   1 / pink   : red
   2 / cyan   : blue
   3 / lig    : green;
cmap mycolor;
   list cmap;
quit;
```

Program Description

Here is a detailed example of the example program.

Set the graphics environment.

```
goptions reset=all border;
```

Start the GREPLAY procedure. The CC= option assigns CLRMAP as the color map catalog. The GOUT= option assigns EXCAT as the graphics output catalog. In order for the color map to be created, both options must be assigned.

```
proc greplay cc=clrmap gout=excat nofs;
```

Define a color map. The CDEF statement defines a color map named MYCOLOR that contains three color pairs.

```
cdef mycolor des="Special Color Map"
   1 / pink   : red
   2 / cyan   : blue
   3 / lig    : green;
```

Specify a current color map, and write contents to the SAS log. The CMAP statement assigns MYCOLOR as the current color map. The contents of CMAP are listed in the SAS log.

```
cmap mycolor;
```
list cmap;
quit;
Overview: GSLIDE Procedure

About the GSLIDE Procedure

The GSLIDE procedure is useful for creating text slides for presentations. You can overlay text slides on other graphics output with the GREPLAY procedure. The GSLIDE procedure produces graphics output that consists of text and straight lines that are generated by TITLE, FOOTNOTE, and NOTE statements. In addition, the procedure provides an easy way to add titles, notes, and footnotes to output that is produced entirely with an Annotate data set.

The GSLIDE procedure is not supported by the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG graphics devices. If you use one of these devices with the GSLIDE procedure, the SAS/GRAPH software uses the PNG device driver to generate the output instead and writes the following note to the SAS log:

```
NOTE: The graph is being created as a PNG file because the type of graph you requested is not supported by the <device-name> client.
```

where `<device-name>` is JAVA, JAVAIMG, ACTIVEX, or ACTXIMG.

About Text Slides

Text slides contain text and graphics that are generated by SAS/GRAPH statements.
Figure 44.1 on page 1334 shows a slide containing text that was produced with TITLE, FOOTNOTE, and NOTE statements.

**Figure 44.1** Text Slide Produced by the GSLIDE Procedure (GSLTEXTS)

The program for this slide is in “Example 1: Producing Text Slides” on page 1340.

**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 27, “Using Annotate Data Sets,” on page 635 for more information about creating and displaying Annotate data sets.

Figure 44.2 on page 1335 shows output from an Annotate data set that is displayed with titles and footnotes that were generated by TITLE and FOOTNOTE statements.
The program for this slide is in “Example 2: Displaying Annotate Graphics” on page 1342.

---

**Syntax: GSLIDE Procedure**

**Restriction:** The GSLIDE procedure is not supported by the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG devices. If you use one of these devices with the GSLIDE procedure, the SAS/GRAPH software uses the PNG device driver to generate the output instead.

**Requirement:** At least one of these is required: a TITLE, FOOTNOTE, or NOTE statement; an appearance option; the BORDER graphics option.

**Global statements:** FOOTNOTE, TITLE

**Supports:** RUN-group processing

**Note:** The procedure can include the SAS/GRAPH NOTE statement. See “TITLE, FOOTNOTE, and NOTE Statements”.

**Tip:** When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

```sas
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)>;

Summary of Optional Arguments

Appearance options

ANNOTATE=Annotate-data-set
specifies a data set that includes Annotate variables that identify graphics commands and parameters.

BORDER
draws a border around the graphics output area, which includes the title area, the footnote area, and the procedure output area.

CFRAME=frame-color
draws a frame around the procedure output area in the specified color.

FRAME
draws a frame around the procedure output area.

IFRAME=fileref | 'external-file'
specifies the image file that you want to apply to the procedure output area.

IMAGESTYLE=TILE | FIT
specifies whether to tile the image specified with the IFRAME= option to fill the backplane or to stretch the image to fit the backplane.

LFRAME=line-type
specifies the line type for a frame and draws a frame around the procedure output area.

WFRAME=n
specifies the width of the frame where \( n \) is a number.

Catalog options

DESCRIPTION='description'
specifies a description of the output.

GOUT=<libref>.output-catalog
specifies the SAS catalog in which to save the graphics output produced by the GSLIDE procedure.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Image map options

IMAGEMAP= output-data-set
creates a temporary SAS data set that is used to generate an image map.
Optional Arguments
You can specify as many options as you want and list them in any order.

**ANNOTATE=Annotate-data-set**
specifies a data set that includes Annotate variables that identify graphics commands and parameters.

<table>
<thead>
<tr>
<th>Alias</th>
<th>ANNO=Annotate-data-set</th>
</tr>
</thead>
<tbody>
<tr>
<td>See</td>
<td>Chapter 27, “Using Annotate Data Sets,” on page 635</td>
</tr>
<tr>
<td>Example</td>
<td>“Example 2: Displaying Annotate Graphics” on page 1342</td>
</tr>
</tbody>
</table>

**BORDER**
draws a border around the graphics output area, which includes the title area, the footnote area, and the procedure output area. A color specification for the border is searched for in the following order:
1. the CTITLE= option in a GOPTIONS statement.
2. the CTEXT= option in a GOPTIONS statement.
3. the color of the current style. If the NOGSTYLE option is specified, then the color is the first color in the device's color list.

| See   | “Adding Frames, Borders, and Images” on page 1339 |

**CFRAME=frame-color**
draws a frame around the procedure output area in the specified color. If you use both the CFRAME= and FRAME options, the FRAME option is ignored. If you use the IFRAME= option, the specified image fills the background of the slide.

| Note | The CFRAME= option does not color the background of the slide. |
| See   | “Adding Frames, Borders, and Images” on page 1339 |
| Example | “Example 1: Producing Text Slides” on page 1340 |

**DESCRIPTION='description'**
specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:
- the chart description for web output (depending on the device driver that you are using). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the Description field of the PROC GREPLAY window.

<table>
<thead>
<tr>
<th>Alias</th>
<th>DES=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Graphics text slide</td>
</tr>
</tbody>
</table>
FRAME
draws a frame around the procedure output area. By default, the frame color is the
color of the current style. If the NOGSTYLE option is specified, then the color is the
first color in the device's color list. If you want to specify a different color for the
frame, use the CFRAME= option instead.

See “Adding Frames, Borders, and Images” on page 1339

GOUT=<libref.>
output-catalog
specifies the SAS catalog in which to save the graphics output produced by the
GSLIDE procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the
temporary library called WORK and creates the catalog if it does not exist.

See “Specifying the Catalog Name and Entry Name for Your GRSEGs ” on page 120

IFRAME=fileref | 'external-file'
specifies the image file that you want to apply to the procedure output area.

Interaction This option is overridden by the NOIMAGEPRINT graphics option.

See the IMAGESTYLE= option

IMAGEMAP= output-data-set
creates a temporary SAS data set that is used to generate an image map in an SVG
file when you are sending output to the LISTING destination. (This option is not
necessary when you are sending output to the HTML destination.) The drill-down
URLs in the image map must be provided by variables in the input data set. These
variables are identified to the procedure with the HTML= and HTML_LEGEND=
options.

See “Adding Links and Enhancements with the URL=, HTML=, and
HTML_LEGEND= Options” on page 192 and “Enhancing Drill-Down
Behavior in SVG Presentations Using HTML Attributes” on page 198

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 24.21 on
page 443 . By default, the line type is specified by the current style. LFRAME=1,
which produces a solid line, is the default.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output
file, if one is created.

The following applies to name:
• The name can be up to 256 characters in length.
• Special characters in the name are converted to underscores.
• For the GRSEG entry name:
  • The name is truncated to eight characters.
• The first character is always represented in uppercase, and all other characters are represented in lowercase.

• If the name begins with a number, an underscore is prepended to the name.

• If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, name1, name2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

• For the graphics output filename:
  • The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  • All characters are represented in lowercase.
  • If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  • If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  • If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

• The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GSLIDE

WFRAME=n
specifies the width of the frame where n is a number. The thickness of the frame increases directly with n, but the thickness of the line can vary from device to device. By default, the line width is specified by the current style. WFRAME=1, which is the thinnest line, is the default. The WFRAME= option also draws the frame.

See “Adding Frames, Borders, and Images” on page 1339

Example “Example 1: Producing Text Slides” on page 1340

Details

Adding Frames, Borders, and Images
Like the BORDER option in a GOPTIONS statement, the BORDER option in the PROC GSLIDE statement draws a box around the graphics output area. However, the border generated by the GSLIDE procedure remains in effect only for the duration of the procedure.

Both BORDER options use the color specified by the CTITLE= or CTEXT= graphics option if either of these options is used. Otherwise, the border color is the color specified
by the current style. If the NOGSTYLE option is specified, then the color is the first color in the device’s color list.

The BORDER option draws a box around the graphics output area. The FRAME option draws a box or frame around the procedure output area. In this case, titles and footnotes are outside of the frame. (See “Overview” on page 69 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 32, “GANNO Procedure,” on page 779 for details.

**Using RUN Groups**

Although the GSLIDE procedure has no action statements, it can use RUN-group processing. This displays all currently defined titles, footnotes, notes, and specified annotation, each time you submit a RUN statement. TITLE and FOOTNOTE statements that are defined while the GSLIDE procedure is active remain in effect after the procedure ends. NOTE definitions remain in effect until the GSLIDE procedure ends, at which time they are canceled. To cancel NOTE definitions while the procedure is active, specify RESET=NOTE in a GOPTIONS statement or submit a null NOTE statement. See Chapter 7, “Using Run-Group Processing,” on page 67 for details.

---

**Examples: GSLIDE Procedure**

**Example 1: Producing Text Slides**

**Features:**

- PROC GSLIDE options BORDER, CFRAME=, and WFRAME=

**Other features:**

- NOTE Statement

**Sample library member:**

- GSLTEXTS

**Note:**

The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
This example uses FOOTNOTE, NOTE, and TITLE statements to produce a text slide. PROC GSLIDE statement options add both a border and a frame.

**Program**

```sql
options reset=all cback="light blue" border;

title color="dark red" "New Directions";
footnote1 j=1 " ABC Engineering, Inc";
footnote2 j=1 " January 2011" ;

proc gslide border
cframe="dark red"
wframe=4;
note height=14;
note height=5
justify=center
color="white"
"Goals and strategies"
justify=center
"for the coming year";
run;
quit;
```

**Program Description**

Set the graphics environment.

```sql
options reset=all cback="light blue" border;
```

Define titles and footnotes.

```sql
title color="dark red" "New Directions";
footnote1 j=1 " ABC Engineering, Inc";
footnote2 j=1 " January 2011" ;
```
Generate the slide and define additional text. The BORDER option draws a box around the entire graphics output area. The CFRAME= option draws a dark red box around the procedure output area. The WFRAME= option specifies the thickness of the frame. The COLOR= option specifies the color of the note text. The first NOTE statement, which has no text, simply leaves a large blank line above the text specified by the second NOTE statement. The second JUSTIFY= option causes a line break.

```
proc gslide border
cframe="dark red"
wframe=4;
note height=14;
note height=5
   justify=center
   color="white"
   "Goals and strategies"
   justify=center
   "for the coming year"
run;
quit;
```

**Example 2: Displaying Annotate Graphics**

**Features:** PROC GSLIDE option ANNOTATE=

**Other features:** Annotate data set

**Sample library member:** GSLANNOT

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

In this example, the GSLIDE procedure displays Annotate graphics along with current TITLE and FOOTNOTE definitions. See Chapter 27, “Using Annotate Data Sets,” on page 635 for information about creating Annotate data sets.
Program

```sas
options reset=all border;

data art;
    length function color style $ 8;
    input function $ x y color $ style $;
    xsys="5"; ysys="5";
    datalines;
    poly   30  20  blue    solid
    polycont  50  20 .
    polycont  40  50 .
    poly   50  20 green x1
    polycont  70  50 .
    polycont  60  50 .
    poly   40  50 red  l1
    polycont  60  50 .
    polycont  50  80 .
;
    title "Number 17";
    footnote1 h=2 "'Art is anything you can get away with.'";
    footnote2 j=r "D. H. Benson ";
    footnote4 angle=90;
    footnote5 angle=-90;
    proc gslide annotate=art
            border
            wframe=6
            cframe=red;
        run;
    quit;
```

Program Description

Set the graphics environment.

```sas
options reset=all border;
```

Create the Annotate data set, ART. ART contains the commands that draw the design of triangles.

```sas
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.

```plaintext
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.

```plaintext
 proc gslide annotate=art
   border
   wframe=6
   cframe=red;
run;
quit;
```
Overview: GTILE Procedure

The GTILE procedure creates charts that consist of rectangles or squares that are divided into tile-shaped segments. These charts are sometimes referred to as rectangular treemaps. The GTILE charts represent the relative sizes of tiles to one another and to the whole. The GTILE procedure provides three statements that you can use to define the layout in order to visualize your data. The statements require one numeric variable. This variable defines the size of the chart tiles.

The TILEBY= statement is followed by any number of numeric or character variables that are delineated by a comma or a blank space. See “Chart Variables” on page 1346. 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.

Chart Variables

The GTILE procedure produces charts based on the values of the chart's size variable, and the values of a TILEBY level variable. The chart's size variable must be numeric. All the values are treated as discrete. The sum of the chart's size variable value determines the size of each tile. The chart's size variable is also used to color each tile, unless a color variable is specified with the COLORVAL= option.

At least one TILEBY= variable is required. The values of this variable or variable list determine the tile categories, as well as the chart levels (or subsets).

The levels are visually represented in the chart by line colors and line style. The top level is indicated by a thick line and the darkest color. The next level is represented by a thinner line and the darkest color. The third level and any subsequent levels are represented by the thickest line and the darkest color. Each level is also represented in the navigation status bar in the top left corner of the chart.

Note: For Java, an indicator to the left of the legend identifies the name of the chart's size variable.

Default Tile Color

In a GTILE tile chart, each tile represents that tile’s size value relative to the other tiles and to the total size for the entire chart. Consider the following simple example.

Example Code 1  Simple Three-Level Tile Chart

```plaintext
filename odsout ";
data tiledata;
  input cat1 $1-4 cat2 $5-10 cat3 $11-13 size;
datalines;
  TOP BIG   B1 4
  TOP BIG   B2 3
  TOP SMALL S1 2
  TOP SMALL S2 1
;run;
ods _all_ close;
goptions reset=all device=activex xpixels=320 ypixels=260 border;
ods graphics / reset=index;
ods html path=odsout file="tilechart.html";
proc gtile data=tiledata;
  tile size tileby=(cat1, cat2, cat3);
run;
quit;
ods html close;
```
Here is the output.

**Figure 45.1  Output for Simple Three-Level Tile Chart**

By default, the detail level is set to 3 to match the number of variables that are specified by the GTILE statement TILEBY= option. This displays all of the tile segments. The label level is set to 1, which displays the label for the top-level tile TOP. In this case, the size of each tile segment is derived directly from the data. The size value also determines the tile color. For continuous colors, the size value is used to select a color from the color ramp. The smallest tile is assigned the starting color in the color ramp, and the largest tile is assigned the ending color. The remaining tiles are assigned colors between the minimum and maximum size values sequentially in ascending order.

When discrete colors are used, by default, each tile is assigned a color from a GraphData1–GraphDataN style element based on size in ascending order. The smallest tile is assigned the color from GraphData1, the next largest tile is assigned the color from GraphData2, and so on.

---

**Tile Summarization**

When continuous colors are used, tile summarization can occur when the detail level is set to a value that is less than the levels specified in the TILEBY= option. When a tile is summarized, its segments are merged into the parent tile. The size of the summarized tile is the sum of the segment sizes. The color value of a summarized tile is a size-weighted average of the segment color values. Since the sum of the weighting values is always 1 in this case, the size-weighted color value is computed as follows:

\[
SummarizedColorValue = \sum_{i=1}^{n} \left( \frac{SegmentSize_i}{ParentTileSize} \times SegmentColorValue_i \right)
\]

The resulting color provides a visual indication that the segments have been merged into one.

*Note:* Tile summarization does not occur when discrete colors are used.
If the detail level is set to 2 in Example Code 45.1 on page 1346, for example, each CAT2 tile color represents a size-weighted average of its CAT3 segment color values. The following figure shows the result.

**Figure 45.2 Summarization of Segments BIG and SMALL**

In the output, a data tip shows the results of the summarization of segment BIG. The last two items in the data tip show the tile size and color values respectively. The data tip shows that variable SIZE is assigned to the size role. The size value is the sum of the segment B1 and B2 size values 4 and 3. Because the COLORVAR= option is not specified, the data tip shows that the SIZE variable is also assigned to the color role. The color value is the size-weighted average of the segment B1 and B2 color values 4 and 3, which computes to 3.57 using the summarized color value equation shown previously.

*Note:* For information about how the COLORVAR= option affects the color values, see “Using a Color Variable” on page 1349.

A similar summarization occurs for segment SMALL, which results in a size of 3 and a size-weighted color value of 1.67. If the detail level is set to 1 in Example Code 45.1 on page 1346, only the top-level tile TOP is shown. It is a summarization of all of its constituent segments. The following figure shows the result.

**Figure 45.3 Summarization of Tile TOP**
Using a Color Variable

By default, the tile colors are determined by the SIZE variable values. You can use the COLORVAR= option in the GTILE statement to specify a different variable that determines the tile colors. The color variable values can be character or numeric. When the COLORVAR= option is specified, the variable values are first sorted in ascending order. For discrete values, the smallest value is assigned the GraphData1 color. The remaining values are assigned a GraphData2–GraphDataN color sequentially in ascending order. For continuous colors, the smallest value is assigned the first color in the color ramp, and the highest value is assigned the last color. The remaining colors are assigned a color from the color ramp sequentially between the minimum and maximum colors in ascending order.

Here is Example Code 45.1 on page 1346 modified to use variable COLOR as the chart color variable.

**Example Code 2  Three-Level Tile Chart with a Color Variable**

```sas
filename odsout ".";

data tiledata;
   input cat1 $1-4 cat2 $5-10 cat3 $11-13 size color;
datalines;
TOP BIG   B1 4 10  
TOP BIG   B2 3 20  
TOP SMALL S1 2 30  
TOP SMALL S2 1 40  
;
run;

ods _all_ close;
goptions reset=all device=activex xpixels=320 ypixels=260 border;
ods graphics / reset=index;
ods html path=odsout file="tilechartcolorvar.html";

proc gtile data=tiledata;
   tile size tileby=(cat1, cat2, cat3) /
      colorvar=color;
run;
quit;
ods html close;
/* Not required in SAS Studio */
```
Here is the output.

As shown, the tile color values now follow the COLOR variable and are reversed from the colors used in the default case shown in Figure 45.1 on page 1347. As in the default case, for continuous colors, if tile summarization occurs, a size-weighted color value is computed as described in “Tile Summarization” on page 1347. If the detail level in Example Code 45.2 on page 1349 is changed to 2, for example, segment BIG is summarized as shown in the following figure.

In the output, the data tip for segment BIG shows that the COLOR variable is assigned the color role. Summarization for the remaining cases is similar.

Missing Values, Negative Values, and Zero Values

When the GTILE procedure finds missing values, negative values, or zero values, it does the following:
• The chart’s size variable requires nonmissing, positive values to create a tile for that observation. If the value of the size variable is missing, a negative value, or a zero value, the observation is not included in the chart.

• The TILEBY= variable displays tiles with missing values, negative values, and zero values. Each tile is included in the chart, and its value is displayed on the tile and in the chart’s data tip.

• The COLORVAR= variable displays missing values in a color that can be distinguished from the colors in the color ramp.

Note: For Java, an indicator to the right of the legend identifies the color assigned to missing values.

Assigning Custom Colors

Custom Color Ramp Colors

When COLORTYPE=CONTINUOUS, the COLORRAMP= option enables you to customize the chart’s color ramp. You can specify colors for the color ramp using any of the color-naming schemes supported by SAS/GRAPH.

Table 45.1 Examples of Specifying Colors

<table>
<thead>
<tr>
<th>Color-Naming Scheme</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>COLORS=(cx98FB98 cxDDA0DD cxFFDAB9 cxDB7093 cxB0E0E6)</td>
</tr>
<tr>
<td>RGBA*</td>
<td>COLORS=(a98FB9833 aDDA0DD66 aFFDAB999 aDB7093CC aB0E0E6FF)</td>
</tr>
<tr>
<td>CMYK</td>
<td>COLORS=(*FF00FF00&quot; &quot;00FFFF00&quot; &quot;FFFFFF00&quot;)</td>
</tr>
<tr>
<td>HLS</td>
<td>COLORS=(H14055FF H0F060FF H0B485FF H07880FF)</td>
</tr>
<tr>
<td>HSV</td>
<td>COLORS=(V0F055FF v010FFFF v03BFFFF v12C55E8)</td>
</tr>
<tr>
<td>Gray Scale</td>
<td>COLORS=(GRAY4F GRAY6D GRAY8A GRAYC3)</td>
</tr>
<tr>
<td>SAS Registry Colors</td>
<td>COLORS=(palegreen plum peachpuff palevioletred powderblue)</td>
</tr>
<tr>
<td>CNS Color Names</td>
<td>COLORS=(&quot;very light purplish blue&quot; &quot;light vivid green&quot; &quot;medium strong yellow&quot; &quot;dark grayish green&quot;)</td>
</tr>
</tbody>
</table>

* RGBA color mode is not supported by Java devices. RGBA color mode is supported by ActiveX devices when the output is used in Microsoft applications.

For information about color naming schemes, see Chapter 22, “Using Colors in SAS/GRAPH Programs,” on page 313.

The COLORPOINTS= option enables you to specify the points at which the colors transition on the color ramp. The values in the COLORPOINT= option map directly to
the colors that are listed in the COLORRAMP= option. Each value specifies the
transition point as a percentage of the total color ramp range. You can use the
MINLEGENDVALUE= and MAXLEGENDVALUE= options to modify the legend
range. See “Specifying a Custom Range for the Color Ramp” on page 1352.

Custom Discrete Colors

When COLOTYPE=DISCRETE, the COLORS= graphics option enables you to
customize the chart’s colors. By default, a color from the GraphData1–GraphDataN
color list is assigned to each unique value of the color variable. You can use the
COLORS= graphics option to override the default colors with your own color list as
shown in Table 45.1 on page 1351. For information about the COLOR= graphics option,
see “COLORS” on page 532.

Specifying a Custom Range for the Color Ramp

By default, the range for a color ramp is set to the range of the color variable. The
MINLEGENDVALUE= and MAXLEGENDVALUE= options enable you to override
the default range with your own range. When one or both of the MINLEGENDVALUE=
and MAXLEGENDVALUE= options are used, the range is computed as shown in the
following table.

<table>
<thead>
<tr>
<th>Options Used</th>
<th>Color Ramp Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default)</td>
<td>min-color-variable-value to max-color-variable-value</td>
</tr>
<tr>
<td>MINLEGENDVALUE=n1 and MAXLEGENDVALUE=n2</td>
<td>n1 to n2</td>
</tr>
<tr>
<td>MINLEGENDVALUE=n1</td>
<td>n1 to max-color-variable-value</td>
</tr>
<tr>
<td>MAXLEGENDVALUE=n1</td>
<td>min-color-variable-value to n1</td>
</tr>
</tbody>
</table>

If you specify a custom range and the value of the color variable falls outside of the
specified range, the tile or tiles associated with the out-of-range color value are colored
gray.

Note: The out-of-range tile color is always gray and cannot be changed.

Specifying Custom Menu Items in the Pop-up Menu

You can add custom menu items to the pop-up menu for all drill-down levels of a tile chart or for specific drill-down levels only. For each custom item, you can specify the
URL to a web resource that is opened in a web browser when the user selects that item
in the menu. To add a custom item to the pop-up menu, you must include two parameters for each custom item in the PARAMETERS= option in the ODS HTML statement. The
first parameter specifies the text of the menu item, and the second specifies the URL that is opened when the item is selected. The following table lists the parameters that you can use.

<table>
<thead>
<tr>
<th>Menu Item Context</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All drill-down levels</td>
<td>MENUnLABEL</td>
<td>Specifies the text of custom menu item ( n ), where ( n ) is the position of the custom menu item starting with 1 (first custom menu item).</td>
</tr>
<tr>
<td></td>
<td>MENUnVALUE</td>
<td>Specifies the URL of a web resource that is to be opened when the user selects custom menu item ( n ).</td>
</tr>
<tr>
<td>A specific drill-down level only</td>
<td>MENUn.mLABEL</td>
<td>Specifies the text of custom menu item ( n ) at drill-down level ( m ) only.</td>
</tr>
<tr>
<td></td>
<td>MENUn.mVALUE</td>
<td>Specifies the URL of a web resource that is to be opened when the user selects custom menu item ( n ) at drill-down level ( m ).</td>
</tr>
</tbody>
</table>

When using these parameters, you must comply with the following:

- Specify the MENUnLABEL and MENUnVALUE parameters, or the MENUn.mLABEL and MENUn.mVALUE parameters, but not both. These parameter types cannot be mixed. You must use one or the other.
- For each MENUnLABEL or MENUn.mLABEL parameter, you must specify a corresponding MENUnVALUE or MENUn.mVALUE parameter.
- For MENUn.mLABEL and MENUn.mVALUE, \( m \) must not exceed the actual number of drill-down levels in the tile chart.

For the MENUnLABEL and MENUn.mLABEL parameters, you can insert in the menu item text the value of variables that are listed in the TILEBY= option in the GTILE statement for the chart. To do this, list the variables that you want to use in the WEBLINK_LIST parameter in the PARAMETERS= option in the ODS HTML statement. Next, insert a {&v} placeholder in the menu text string where you want a variable value to appear. For \( v \), specify the position of the desired variable in the WEBLINK_LIST list, which starts with 1. You can specify as many placeholders as you want. At run time, each \&{v} placeholder is replaced with the value of its variable.

For an example, see “Example 5: Specifying Custom Items in the Chart Pop-Up Menu” on page 1374.

**Syntax: GTILE Procedure**

**Restrictions:** This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

The GTILE procedure supports the JAVA, JAVAIMG, ACTIVEX, and ACTXIMG devices only. Prior to SAS 9.4M5, if you specify any other device, an error message is written to the SAS log and the tile chart is not generated. Starting with SAS 9.4M5, if you specify any other device, the tile chart is generated using the JAVAIMG device instead. A note is written to the SAS log indicating the device change in that case.
The plot legend does not include the size variable when the ACTIVEX or ACTXIMG device is used. To include the size variable in the legend, use the JAVA or JAVAIMG device.

Custom chart tips and drill-down links are not supported.

The chart legend cannot be suppressed.

For a discrete legend, binning is not supported.

For a discrete legend, user-specified legend values are not supported. The legend values are determined by the data.

**Requirements:** You must specify the JAVA, JAVAIMG, ACTIVEX, or ACTXIMG device in the GOPTION statement.

At least one FLOW, TILE, or TOGGLE statement is required.

**Global statements:** FOOTNOTE, GOPTIONS, TITLE

**Supports:** RUN-Group Processing

**Note:** The procedure can include the SAS/GRAPH statements BY on page 370 and NOTE on page 447, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

**Tips:** When using procedures that support RUN-group processing, include a QUIT statement after the last RUN statement. Using the QUIT statement is especially important when the procedure is supposed to completely terminate within the boundaries of an ODS destination (for example, ODS PDF; procedure-code; ODS PDF CLOSE;). See Chapter 7, “Using Run-Group Processing,” on page 67 for more information.

When the graphics output is sent to a graphics output file, this procedure uses ODS Graphics Indexing when necessary to avoid overwriting existing output files. See “About Filename Indexing” on page 119. To reset the current index value and overwrite output files that already exist, use the following statement:

```sas
ods graphics / reset=index;
```

**PROC GTILE Statement**

Identifies the data set containing the chart variables.

**Requirement:** An input data set is required.

**Syntax**

```sas
PROC GTILE <DATA=input-data-set>;
```

**FLOW**

```sas
FLOW size-variable TILEBY=(levels-variable | levels-variable-list) </option(s)>;
```

**TILE**

```sas
TILE size-variable TILEBY=(levels-variable | levels-variable-list) </option(s)>;
```

**TOGGLE**

```sas
TOGGLE size-variable TILEBY=(levels-variable | levels-variable-list) </option(s)>;
```
Optional Argument
PROC GTILE statement options affect all graphs produced by the procedure.

DATA=input-data-set
specifies the SAS data set that contains the variables to chart. By default, the procedure uses the most recently created SAS data set.

See “About the Input Map Data Set” in SAS/GRAPH and Base SAS: Mapping Reference
See “The SAS Data Set: Your Key to the SAS System” in Step-by-Step Programming with Base SAS
See “About Data Set Options” in SAS Data Set Options: Reference

FLOW, TILE, and TOGGLE Statement
Create a tile chart in one of the three display layouts.

Requirement: At least one numeric chart size variable and one TILEBY= variable is required.

Global statements:
FOOTNOTE, GOPTIONS, TITLE

Supports: Drill-down functionality

Syntax
FLOW size-variable TILEBY=(levels-variable | levels-variable-list) <option(s)>;
TILE size-variable TILEBY=(levels-variable | levels-variable-list) <option(s)>;
TOGGLE size-variable TILEBY=(levels-variable | levels-variable-list) <option(s)>;

Summary of Optional Arguments
Appearance options

CDEFAULT=missing-value-color
specifies the color to be assigned to tiles when the COLORVAR= value is missing.

COLORPOINTS=(point-1 point-2 ... point-n)
specifies where the colors need to transition on the color ramp with respect to the normalized values of the color variable.

COLORRAMP=(color-ramp color-list)
specifies the colors to be distributed continuously across the range of data values.

COLORTYPE=CONTINUOUS | DISCRETE
specifies whether a color ramp or discrete colors are used to color the tiles.

COLORVAR=color-variable
specifies a variable whose values determine the color of the tiles.

DETAILLEVEL=1-to-the-number-of-variables-specified-by-TILEBY=
specifies the number of levels to display.

LABELLEVEL=1-to-the-number-of-levels-variables-specified-by-TILEBY=
specifies the number that corresponds to the level of labels to display.
MAXLEGENDVALUE=\textit{max-value}
\begin{itemize}
\item specifies the maximum endpoint value of a continuous legend range.
\end{itemize}

MINLEGENDVALUE=\textit{min-value}
\begin{itemize}
\item specifies the minimum endpoint value of a continuous legend range.
\end{itemize}

NOLEGEND
\begin{itemize}
\item suppresses the legend that is automatically generated.
\end{itemize}

Catalog entry description options

DESCRIPTION=\textit{"description"}
\begin{itemize}
\item specifies a description of the output.
\end{itemize}

NAME=\textit{"name"}
\begin{itemize}
\item specifies the name of any graphics output file created.
\end{itemize}

Midpoint options

BASELINE=\textit{midpoint}
\begin{itemize}
\item specifies the midpoint value for the tiles.
\end{itemize}

Required Arguments

\textit{size-variable}
\begin{itemize}
\item specifies a numeric variable from the input data set. The values of this variable are used to determine the size of each tile.
\end{itemize}

Note
If the COLORVAR= option is not specified, then the size variable is used to assign a color to each tile. In that case, the size variable is assigned both the size and color roles in the chart data tips.

TILEBY=\textit{(levels-variable | levels-variable-list)}
\begin{itemize}
\item 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.
\end{itemize}

Optional Arguments

The options in a GTILE statement affect all chart levels. Specify as many options as you want and list them in any order.

BASELINE=\textit{midpoint}
\begin{itemize}
\item specifies the midpoint value for the tiles.
\end{itemize}

CDEFAULT=\textit{missing-value-color}
\begin{itemize}
\item 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.
\end{itemize}
Figure 45.4  CDEFAULT=yellow

Alias  

Restriction  Partially supported by ActiveX.

Interaction  This option is ignored when COLORTYPE=DISCRETE.

**COLORPOINTS=(point-1 point-2 ... point-n)**

specifies where the colors need to transition on the color ramp with respect to the normalized values of the color variable. The colors are specified by the COLORRAMP= option. A value must be provided for each color that is specified by the COLORRAMP= option. You can use a space or a comma to separate the values. Each transition point is specified as a percentage of the total range of the color ramp. For example, if COLORRAMP=(red green blue), COLORPOINTS=(0, 0.75 1), and the range of the color variable is 100, the color ramp begins with red at 0, transitions to green at 75, and then transitions to blue at 100. By default, the range is set to the range of the color variable. The range can be modified with the MAXLEGENDVALUE= and MINLEGENDVALUE= options.

Default  The color transition points are distributed equally along the color ramp.

Range  0 to 1

Requirements  The COLORRAMP= option must be specified before you can use the COLORPOINTS= option. The number of values that are specified in the COLORPOINTS= option must exactly match the number of colors that are specified in the COLORRAMP= option.

Interactions  This option is ignored when COLORTYPE=DISCRETE. The MAXLEGENDVALUE= and MINLEGENDVALUE= options modify the range of the color ramp.
COLORRAMP=(color-ramp color-list)
specifies the colors to be distributed continuously across the range of data values. The three-color gradient legend provides a key to the value of the colors plotted on the chart. The legend label is the variable used to color the tiles. The legend displays the variable's minimum value, the maximum value, and the midpoint value.

Note: When the COLORRAMP= option is used with the ACTIVEX device, the graph viewers cannot change the color ramp colors using the ActiveX Control Properties dialog box. In that case, the color gradient color pickers are disabled on the Tile tab.

Two colors are required to create a color ramp. However, the number of colors that can be provided is not limited. These values specify the minimum, and the maximum values of the color ramp. If only two colors are specified, the legend midpoint is not be labeled. The delimiter can be either blank spaces or commas. All of the color-naming schemes supported by SAS/GRAPH are valid.

Figure 45.5 COLORRAMP=(cxbcd3aa cxa9aeb)
**Figure 45.6**  \(\text{COLORRAMP}=(\text{cxbcd3aa cx5f8e97 cxae9aeb})\)

**Aliasing**

RAMP=

**Style reference**

Color attribute of the ThreeColorRamp element

**Interaction**

This option is ignored when \(\text{COLORTYPE} = \text{DISCRETE}\).

**Note**

When the \(\text{COLORRAMP} = \) option is used to specify custom colors and the ACTIVEX device is used to generate the chart, the chart users cannot change the chart’s gradient colors from the Tile tab of the ActiveX Control Properties dialog box. In that case, Color under \text{Color gradient} is disabled on the Tile tab.

**Examples**

“Example 2: Specifying the \(\text{COLORRAMP} = \) and \(\text{COLORVAR} = \) Options” on page 1367

“Example 3: Specifying the \(\text{COLORPOINTS} = \) Option and a Custom Color Ramp Range” on page 1370

**\(\text{COLORTYPE} = \text{CONTINUOUS} | \text{DISCRETE}\)**

specifies whether a color ramp or discrete colors are used to color the tiles. When \(\text{COLORTYPE} = \text{CONTINUOUS}\), a color ramp is used to color the tiles. A continuous legend is used in that case to denote the tile colors. When \(\text{COLORTYPE} = \text{DISCRETE}\), discrete colors are used to color the tiles. A discrete legend is used to denote the tile colors in that case.

**Defaults**

CONTINUOUS when \(\text{COLORVAR} = \) is not specified or when \(\text{COLORVAR} = \) specifies a numeric variable

DISCRETE when \(\text{COLORVAR} = \) specifies a character variable

**Style references**

GraphData1–GraphDataN (DISCRETE only)
the color attribute of the ThreeColorRamp element (CONTINUOUS only)

Restrictions

The legend is always positioned at the bottom of the chart. Because the LEGEND statement is not supported, the legend cannot be relocated.

When COLORTYPE=DISCRETE, you must use the COLORS= graphics option to change the chart colors from the default GraphData1 through GraphDataN colors. See “COLORS” on page 532.

Interaction

When COLORTYPE=DISCRETE, the COLORPOINTS=, COLORRAMP=, DETAILLEVEL=, MINLEGENDVALUE=, and MAXLEGENDVALUE= options are ignored.

Note

When the COLORTYPE=DISCRETE option is used and the ACTIVEX or JAVA device is used to generate the chart, the chart users cannot change the chart’s gradient colors using the ActiveX Control or Java Applet pop-up menu. For the ActiveX Control, Colors and Center are disabled on the Tile tab on the Properties dialog box in that case. For the Java Applet, GradientEditor is disabled on the Data Options dialog box in that case.

See

COLORRAMP= on page 1358

Example

“Example 4: Specifying Discrete Tile Colors” on page 1373

COLORVAR=color-variable

specifies a variable whose values determine the color of the tiles. The variable can be character or numeric. The values of this variable are sorted in ascending order. The smallest value is assigned to the first color in the color ramp, and the largest value is assigned the last color in the color ramp. Each of the remaining values is assigned a color from the gradient list of colors between the first, and the last colors in the color ramp.

Figure 45.7  COLORVAR=cylinders
The size variable is used to determine the tile colors

When COLOTYPE=DISCRETE and the data contains duplicate TILEBY= values, the COLORVAR= values for the duplicate TILEBY= values are dropped from the chart. However, when the ACTIVEX or ACTXIMG device is used, the dropped COLORVAR= values appear in the legend. In that case, the legend contains items that are not represented in the chart. To remove the dropped COLORVAR= values from the legend, use the JAVA or JAVA IMG device instead.

See
- “Default Tile Color” on page 1346
- “Using a Color Variable” on page 1349

**DESCRIPTION=**

```
specifies a description of the output. The maximum length for description is 256
characters. The description does not appear in the output. The descriptive text is
shown in each of the following:

- the chart description for web output (depending on the device driver that you are
  using). See “Chart Descriptions for Web Presentations” on page 189 for more
  information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS
  HTML statement, assuming that the output is generated while the contents page
  is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the Description field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution
options, which work as they do when used on TITLE, FOOTNOTE, and NOTE
statements. Refer to “Substituting BY Line Values in a Text String” on page 959.
The 256-character limit applies before the substitution takes place for these options.
The 256-character limit applies before the substitution takes place for these options.
Thus, if in the SAS program the description text exceeds 256 characters, it is
truncated to 256 characters, and then the substitution is performed.
```

<table>
<thead>
<tr>
<th>Alias</th>
<th>DES=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Tile chart of tileby</td>
</tr>
<tr>
<td>Restriction</td>
<td>Partially supported by ActiveX and Java.</td>
</tr>
</tbody>
</table>

**DETAILLEVEL=**

```
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.
```

FLOW, TILE, and TOGGLE Statement  1361
If the DETAILLEVEL= option value is less than the number of tile-by variables and continuous colors are used, tile summarization occurs. See “Tile Summarization” on page 1347.
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 45.10*  LABELLEVEL= Option Is Not Used

If LABELLEVEL=3, the third-level labels are displayed. Once you navigate past the LABELLEVEL specified, subsequent levels display their respective labels.

*Figure 45.11*  LABELLEVEL=3

Alias  LLEVEL=

Default  1
MAXLEGENDVALUE=\textit{max-value}

specifies the maximum endpoint value of a continuous legend range. By default, the maximum legend endpoint value is set to the highest data value. The MAXLEGENDVALUE= option overrides the default value. Tiles whose data value exceeds MAXLEGENDVALUE are colored gray.

**Interaction**

This option is ignored when COLORTYPE=DISCRETE.

**See**

“Specifying a Custom Range for the Color Ramp” on page 1352

**Example**

“Example 3: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range” on page 1370

MINLEGENDVALUE=\textit{min-value}

specifies the minimum endpoint value of a continuous legend range. By default, the minimum legend endpoint value is set to the lowest data value. The MINLEGENDVALUE= option overrides the default value. Tiles whose data values are less than MINLEGENDVALUE are colored gray.

**Interaction**

This option is ignored when COLORTYPE=DISCRETE.

**See**

“Specifying a Custom Range for the Color Ramp” on page 1352

**Example**

“Example 3: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range” on page 1370

NAME="\textit{name}"

specifies the name of any graphics output file created.

The following applies to \textit{name}:

- **For the graphics output filename:**
  - All characters are represented in lowercase.
  - The ACTXIMG or JAVA IMG device must be used in order to generate a graphics output file.
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.
  - The name can include special characters.
  - Each time the graph is generated in a SAS session, SAS/GRAPH adds a number to the name, or increments the last number used to create a unique filename for the output. See “About Filename Indexing” on page 119.

**Default**

graph.png

NOLEGEND

suppresses the legend that is automatically generated.

**Note:** This option is valid starting with SAS 9.4M2.

When this option is specified, the legend bar and, for graphs generated with the JAVA and JAVA IMG devices, the size legend, are not generated for the graph.

**Default**

A legend is generated for the size variable or, if specified, the COLORVAR= variable.
Details

The FLOW, TILE, and TOGGLE statements specify the type of layout, the variables that define the chart levels, and the tile sizes used to display the data. One of the following display layouts is required:

FLOW
creates a chart that honors data order and can be read from left to right.

TILE
creates a chart that orders data by value, descending from bottom left to top right.

TOGGLE
creates a chart that honors data order and can be read from left to right. When changing levels, the display toggles from one row to one column.

Examples: GTILE Procedure

Example 1: Simple GTILE with the COLORVAR= Option

Features: TILE statement options TILEBY= and COLORVAR=
Data set: SASHELP.SHOES
Sample library member: GTLSIMPL
Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

In this example, PROC GTILE generates the following chart for the SASHELP.SHOES data set. The size of each tile represents the number of stores. The COLORVAR=SALES option specifies that the color of each tile represents the sales revenue for that tile. The visualization of the data with the GTILE procedure makes it easy to see the data extremes for sales revenue relative to the number of stores.
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.

**Output 45.2**  Subset of SASHELP.SHOES Where Region=“Canada”

Program

```sas
filename odsout ".";
odc _all_ close;
goptions reset=all device=java noborder;
odc html path=odsout file='shoe_sales.html';
proc gtile data=sashelp.shoes;
tile stores tileby=(region subsidiary)
```
Program Description

Here is a detailed description of this program.

Create a file reference for the output. The current working directory is specified in this example.

```sas
filename odsout ";
```

Close the currently open ODS destinations.

```sas
ods _all_ close;
```

Set the graphics options.

```sas
goptions reset=all device=java noborder;
```

Open ODS HTML. The PATH= option specifies the file reference that was defined previously. The FILE= option specifies a name for the HTML output file.

```sas
ods html path=odsout file='shoe_sales.html';
```

Generate the tile chart using SALES as the color variable. The chart variable STORES specifies the size of the tiles. The TILE layout arranges the tiles. The TILEBY=(levels-list) variable list defines the tile segments and the chart levels. The COLORVAR=SALES option specifies the variable to use to color the tiles.

```sas
proc gtile data=sashelp.shoes;
   tile stores tileby=(region subsidiary)
   / colorvar=sales ;
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```sas
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```

---

**Example 2: Specifying the COLORRAMP= and COLORVAR= Options**

<table>
<thead>
<tr>
<th>Features:</th>
<th>FLOW statement options</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLORVAR=</td>
<td>COLORRAMP=</td>
</tr>
</tbody>
</table>

Data set: SASHELP.ORSALES Subset
Sample library member: GTLCOLOR

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

In this example, PROC GTILE generates the following chart, which displays 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.

Output 45.3 Tile Chart of SASHELP.ORSALES Where PRODUCT_LINE="SPORTS"
Clicking on the “Skates” product_group subsets the “Skates” observations by year as shown in the following chart. The updated display provides information about the “Skates” profits by year.

**Output 45.4**  
Tile Chart of SASHELP.ORSALES Where PRODUCT_LINE="SPORTS" and PRODUCT_GROUP="Skates"

### Program

```r
filename odsout ";

data sports_only;
  set sashelp.orsales;
  if product_line="Sports";
  format profit dollar12.;
  format quantity comma12.;
run;
ods _all_ close;
goptions reset=all device=java noborder;
ods html path=odsout file='sport_sales.html';
proc gtile data=sports_only;
  flow quantity tileby=(product_group year) /
    colorvar=profit
    /* reverse the colors so that blue is highest */
    colorramp=(CXDD6060 CXFFFFFF CX6497EB);
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

### Program Description

Here is a detailed description of this program.
Create a file reference for the output. The current working directory is specified in this example.

```sas
filename odsout ";.
```

Subset the data, format the quantity variable, and the profit variable.

```sas
data sports_only;
  set sashelp.orsales;
  if product_line="Sports";
  format profit dollar12.;
  format quantity comma12.;
run;
```

Close the currently open ODS destinations.

```sas
ods _all_ close;
```

Set the graphics options.

```sas
goptions reset=all device=java noborder;
```

Open ODS HTML. The PATH= option specifies the file reference for the output that was defined previously. The FILE= option specifies a name for the HTML output file.

```sas
ods html path=odsout file='sport_sales.html';
```

Generate the tile chart using PROFIT as the color variable, 1 as the detail level, and a three-color color ramp. 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.

```sas
proc gtile data=sports_only;
  flow quantity tileby=(product_group year) /
    colorvar=profit
    /* reverse the colors so that blue is highest */
    colorramp=(CXDD6060 CXFFFFFF CX6497EB);
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.

```sas
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```

Example 3: Specifying the COLORPOINTS= Option and a Custom Color Ramp Range

**Features:**
- TILE statement options
- COLORVAR=
In this example, PROC GTILE generates the chart shown in Output 45.5 for SASHELP.SHOES. The TILE statement defines the layout of the data. The tile sizes are determined by the number of stores in each subsidiary of each region. The COLORVAR=SALES specifies that the color of each tile represents the sales revenue for that subsidiary in that region. The COLORRAMP= option specifies that the color ramp colors consist of light red, light orange, medium yellow, and green. The COLORPOINTS= option specifies that the colors on the color ramp start at 0, and transition at 20%, 40%, and 100% of the legend value range. The MINLEGENDVALUE= and MAXLEGENDVALUE= options specify a custom legend range of 150000 to 3300000. Tiles whose SALES value falls outside of the legend range are colored gray as shown.

**Output 45.5**  Tile Chart with Custom Color Points and a Custom Color Ramp Range

**Program**

```plaintext
filename odsout ".";
ods _all_ close;
ods html path=odsout file="colorpoints.html";
goptions reset=all noborder device=javaimg border;
proc gtile data=sashelp.shoes;
tile stores tileby=(region subsidiary) /
    colorvar=sales
    colorramp=(lightred lightorange mediumyellow green)
    colorpoints=(0 0.2 0.4 1)
```

**Data set:** SASHELP.SHOES
Program Description

Here is a detailed description of the SAS program.

Create a file reference for the output. The current working directory is specified in this example.

filename odsout ";

Close the currently open ODS destinations.

ods _all_ close;

Open ODS HTML. The PATH= option specifies the output file reference that was defined previously. The FILE= option specifies a name for the HTML output file.

ods html path=odsout file="colorpoints.html";

Specify the graphics options. Remove the border around the chart and specify the JAVAIMG device to create a non-interactive image.

goptions reset=all noborder device=javaimg border;

Generate the tile chart using SALES as the color variable.

proc gtile data=sashelp.shoes;
    tile stores tileby=(region subsidiary) /
        colorvar=sales

Define the color ramp and specify the color transition points. Specify the colors light red, light orange, medium yellow, and green for the color ramp colors. Set the transition points at 0%, 20%, 40%, and 100%.

    colorramp=(lightred lightorange mediumyellow green)
    colorpoints=(0 0.2 0.4 1)

Specify a custom color ramp range of 150000 to 3300000. Any tiles that fall outside of this range are colored gray.

    minlegendvalue=150000
    maxlegendvalue=3300000

Specify a label level of 2.

    labellevel=2;
    run;
    quit;

Close ODS HTML. Closing ODS HTML closes the HTML output file.

    ods html close;
Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```

### Example 4: Specifying Discrete Tile Colors

**Features:**
- FLOW statement options `COLORVAR=` and `COLORTYPE=`

**Data set:**
- SASHELP.ORSALES

In this example, PROC GTILE generates the chart shown in **Output 45.6** for a subset of SASHELP.ORSALES. The FLOW statement defines the layout of the data. The tile sizes are determined by the total profits that were generated by each product group for each year. The `COLORVAR=YEAR` specifies that the color of each tile represents the sales revenue for that product group for that year. The `COLORTYPE=` option specifies that discrete colors are to be used. The GOPTIONS statement `COLORS=` option is used to specify that the discrete colors `CXXFCCCC` (light red), `CXXCCFF` (light blue), `CXXCCFFC` (light green), and `CXXFFFFC` (light yellow) are to be used in the chart instead of the default GraphData1 through GraphData4 colors.

**Output 45.6**  
Tile Chart That Uses Discrete Tile Colors

---

**Program**

```sas
filename odsout ".";
ods _all_ close;
ods html path=odsout file="discretecolors.html";
goptions reset=all border device=javaimg
colors=(cxffcccc cxccccff cxccffcc cxffffcc);
proc gtile data=sashelp.orsales (where=(product_line="Sports"));
flow profit tileby=(product_group year) /
```

---
Program Description
Here is a detailed description of the SAS program.

Create a file reference for the output. The current working directory is specified in this example.
```sas
filename odsout ".";
```

Close the currently open ODS destinations.
```sas
ods _all_ close;
```

Open ODS HTML. The PATH= option specifies the output file reference that was defined previously. The FILE= option specifies a name for the HTML output file.
```sas
ods html path=odsout file="discretecolors.html";
```

Specify the graphics options. Specify JAVAIMG as the device and specify custom chart colors.
```sas
goptions reset=all border device=javaimg
  colors=(cxffcccc cxccccff cxccffcc cxffffcc);
```

Generate the tile chart using YEAR as the color variable and DISCRETE as the color type.
```sas
proc gtile data=sashelp.orsales (where=(product_line="Sports");
  flow profit tileby=(product_group year) /
    colorvar=year
    colortype=discrete;
run;
quit;
```

Close ODS HTML. Closing ODS HTML closes the HTML output file.
```sas
ods html close;
```

Open ODS HTML. Open an ODS destination for subsequent programs. This is not required in SAS Studio.
```sas
ods html; /* Not required in SAS Studio */
```

Example 5: Specifying Custom Items in the Chart Pop-Up Menu

Features: TILE statement option TILEBY=

Other features: Applet parameters
```sas
  MENUn.mLABEL
  MENUn.mVALUE
```
Variable substitution using the \{&v\} variable placeholder

**Data set:** SASHELP.PRDSALE

In this example, applet parameters `MENU.nLABEL` and `MENU.nVALUE` are used to add custom menu items to the pop-up menus in a tile chart of product sales data. Applet parameter `WEBLINK_LIST` and the \{&v\} variable placeholder are used to customize the menu labels at run time based on the user’s selection. The tile chart shows product sales data at two levels. The top level, Country, shows the total product sales data for Canada, Germany, and the USA. The second level, Product, shows the sales data for a selected country.

To supplement the information in this chart, custom menu item **View Total Product Sales** is added to the pop-up menu at the Country level. This item appears when the user right-clicks a tile on the Country level as shown in the following figure.

This menu item is linked to a horizontal 3-D bar chart that shows the total sales data for all three countries. If the user selects this menu item, the sales chart shown in the following figure appears in a web browser window.
Custom menu item View Total Product Sales for {&1} is added at the Product level. The {&1} placeholder is used to display at run time the country for which the data is being displayed. This item is visible when the user drills down to the Product level for a country, and then right-clicks a tile. The following figure shows the menu item for Canada.

![Menu Item](image)

Notice that {&1} is replaced with CANADA in the menu item label. If the user selects this menu item, the sales chart shown in the following figure appears in a web browser window.

![Sales Chart](image)

**Total Product Sales for Canada**

<table>
<thead>
<tr>
<th>Product</th>
<th>Actual Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED</td>
<td>$47,729</td>
</tr>
<tr>
<td>CHAIR</td>
<td>$50,239</td>
</tr>
<tr>
<td>DESK</td>
<td>$52,187</td>
</tr>
<tr>
<td>SOFA</td>
<td>$50,135</td>
</tr>
<tr>
<td>TABLE</td>
<td>$46,700</td>
</tr>
</tbody>
</table>

**Program**

```plaintext
/* Specify the ODS HTML output path */
%let odsoutp=.;
/* Generate a total sales chart */
ods _all_ close;
ods graphics / reset=index;
goptions reset=all device=png border hsize=4.5in vsize=3in;
ods html style=pearl file="totalsales.html"
```
Example 5: Specifying Custom Items in the Chart Pop-Up Menu

   path=&odsoutp;
title1 lspace=1 "Total Sales";
proc gchart data=sashelp.prdsale;
   hbar3d country / sumvar=actual sum width=4.5 space=4;
   format actual dollar16.0;
run;
quit;
ods html close;

   /* Create a macro that generates a country sales chart */
%macro genchart(country=);
   /* Open ODS HTML */
ods html style=pearl file="%upcase(&country)-sales.html"
   path=&odsoutp;
   /* Generate the sales chart for the specified country */
   title lspace=1 "Total Product Sales for &country";
   proc gchart data=sashelp.prdsale;
      hbar3d product / sumvar=actual subgroup=region sum
         width=3.2 space=3;
      where country=%upcase("&country");
      format actual dollar16.0;
   run;
   quit;
   /* Close ODS HTML */
ods html close;
%mend genchart;

   /* Use macro %GENCHART to generate the country sales charts */
%genchart(country=Canada);
%genchart(country=Germany);
%genchart(country=U.S.A.);

   /* Generate the tile chart. Tile by country and product */
goptions reset=all border device=java hsize=4.5in vsize=3in;
ods html style=pearl file="tilechart.html"
   path=&odsoutp
   PARAMETERS=(
      "WEBLINK_LIST" = "country"
      "MENU1.1LABEL" = "View Total Product Sales"
      "MENU1.1VALUE" = "&odsoutp/totalsales.html"
      "MENU1.2LABEL" = "View Total Product Sales for {&1}"
      "MENU1.2VALUE" = "&odsoutp/{&1}-sales.html"
   );
proc gtile data=sashelp.prdsale;
   tile actual tileby=(country product) / name="tilechart";
run;
quit;
ods html close;
   /* Reset */
goptions reset=all;
title;
ods html; /* Not required in SAS Studio */
Program Description
Here is a detailed description of the code for this example.

Specify the path for the output.

```sas
/* Specify the ODS HTML output path */
%let odsoutp=.;
```

Generate the total sales chart for the top level. This code generates a horizontal 3-D bar chart of the total actual sales data for Canada, Germany, and the USA.

```sas
/* Generate a total sales chart */
ods _all_ close;
ods graphics / reset=index;
geoptions reset=all device=png border hsize=4.5in vsize=3in;
ods html style=pearl file="totalsales.html"
   path="&odsoutp";
title1 lspace=1 "Total Sales";
proc gchart data=sashelp.prdsale;
   hbar3d country / sumvar=actual sum width=4.5 space=4;
   format actual dollar16.0;
run;
quit;
ods html close;
```

Create a macro that can be used to generate a product sales chart for each country. The macro generates a horizontal 3-D bar chart of the actual product sales data for a specific country. The country name is incorporated into the output filename for easy identification. The %UPCASE macro function is used to ensure consistent case in the output filename for reliable linking.

```sas
/* Create a macro that generates a country sales chart */
%macro genchart(country=);
/* Open ODS HTML */
ods html style=pearl file="&upcase(&country)-sales.html"
   path="&odsoutp";
/* Generate the sales chart for the specified country */
title lspace=1 "Total Product Sales for &country";
proc gchart data=sashelp.prdsale;
   hbar3d product / sumvar=actual subgroup=region sum
      width=3.2 space=3;
   where country=&upcase(&country);
   format actual dollar16.0;
run;
quit;
/* Close ODS HTML */
ods html close;
%mend genchart;
```

Generate the country sales charts.

```sas
/* Use macro %GENCHART to generate the country sales charts */
%genchart(country=Canada);
%genchart(country=Germany);
%genchart(country=U.S.A.);
```
Set the graphics options and open the ODS HTML destination with the appropriate parameter values. The MENU1.1LABEL and MENU1.1VALUE parameters define the custom menu item in the pop-up menu at the Country level. The MENU1.2LABEL and MENU1.2VALUE parameters specify the custom menu item in the pop-up menu at the Product level. The MENU1.mVALUE parameters specify the URL to the appropriate sales information. In the MENU1.2VALUE and MENU1.2VALUE parameter values, \{&1\} is a placeholder for the value of the first variable in the WEBLINK_LIST, Country. At run time, \{&1\} is replaced with the value of Country for the selected country tile. In order to use variable substitution for the Country variable, Country must also be included in the GTILE statement TILEBY= option, which is done in the next statement.

```sas
/* Generate the tile chart. Tile by country and product */
goptions reset=all border device=java hsize=4.5in vsize=3in;
ods html style=pearl file="tilechart.html"
   path=&odsoutp
   PARAMETERS={("WEBLINK_LIST" = "country"
                  "MENU1.1LABEL" = "View Total Product Sales"
                  "MENU1.1VALUE" = "/&odsoutp/totalsales.html"
                  "MENU1.2LABEL" = "View Total Product Sales for \{&1\}" 
                  "MENU1.2VALUE" = "/&odsoutp/\{&1\}-sales.html"
    );
```

**Generate the tile chart.** This code creates a tile chart of the actual sales data tiled by country and product. Because Country appears in the TILEBY= option, it can be included in the applet parameter WEBLINK_LIST list as shown in the previous statement.

```sas
proc gtile data=sashelp.prdsale;
   tile actual tileby=(country product) / name="tilechart";
run;
quit;
```

**Close ODS HTML.** Closing ODS HTML closes the HTML output file.

```sas
ods html close;
```

**Reset the graphics options and graph titles.**

```sas
/* Reset */
goptions reset=all;
title;
```

**Open ODS HTML.** Open an ODS destination for subsequent programs. This is not required in SAS Studio.

```sas
ods html; /* Not required in SAS Studio */
```
Overview: G3D Procedure

The G3D procedure enables you to produce three-dimensional surface plots and scatter plots.

Surface Plots

Surface plots represent the shape of the surface that is described by the values of three variables, X, Y, and Z. The values of the X and Y variables are plotted to form a
horizontal plane. The values of the Z variable, create a vertical axis that is perpendicular to the X-Y plane. Combined, these three axes, form a three-dimensional surface.

The surface plot in the following figure displays various depths of a lake. The dimensions of the lake are plotted on the X-Y axes. The Z variable is plotted as the third dimension. The coordinates of each point correspond to the values of the three numeric variable values in an observation from the selected input data set.

**Figure 46.1  G3D Surface Plot**

With the PLOT statement, you can do the following actions:

- show the three-dimensional shape of your data (useful for examining data trends).
- change the data ranges that are displayed.
- rotate and tilt the plot to enhance viewing angles.
- customize the axes.

The program for this figure is featured in “Example 1: Generating a Surface Plot” on page 1404. For more information about producing surface plots, see the “PLOT Statement” on page 1387.

**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.
With the SCATTER statement, you can do the following actions:

- change the symbols used to represent your data points
- categorize your data with colors, shapes, sizes
- change the data ranges that are displayed
- rotate and tilt the plot to enhance the viewing angles
- customize the axes

For more information about producing scatter plots, see the “SCATTER Statement” on page 1396.

---

**G3D Procedure Terms**

The following illustration provides the terminology used to describe the elements of the three-dimensional plots generated with G3D.
The Input Data Set

About the G3D Data Set

The G3D procedure requires three numeric variables to produce a plot. The input data set forms a rectangular grid from the values of X and Y. One value of Z is required for each X-Y grid location. If multiple observations have the same Z value for any X-Y combination, only the last observation is plotted.

*Note:* The Java and ActiveX drivers support multiple points with identical X-Y combinations.

Data for Surface Plots

The G3D procedure requires nonmissing Z values for at least 50% of the grid cells. When the procedure cannot produce a satisfactory surface plot because of missing Z values, a warning message is issued and a graph might not be produced. To correct this problem, you can grid the data set with the G3GRID procedure. The G3GRID procedure interpolates the necessary values to produce a data set with nonmissing Z values for every X-Y combination. The G3GRID procedure can also smooth data for use with the G3D procedure. The output data set produced by the G3GRID procedure can be used as the input data set for the G3D procedure. See Chapter 47, “G3GRID Procedure,” on page 1417 for more information.
Data for Scatter Plots

In order to properly scale the axes, the G3D procedure requires at least two observations. These observations must contain unique values for each of the three variables that are specified in the plot request. If these requirements are not met, an error message is issued, and a graph is not produced.

Changing Data Ranges

For both surface plots and scatter plots, the range of the Z axis is defined by the minimum and maximum data values for Z. To increase or decrease the range of the Z axis, you can use the ZMIN= option and the ZMAX= option in the PLOT or SCATTER statements. To restrict the range of an X axis or a Y axis, you can use a WHERE clause in the PROC step to subset the data. A DATA Step with a WHERE clause, or an IF statement can also be used to subset the data.

Note: See “SCATTER Statement” on page 1396 for information about controlling axis labels and tick mark values with SCATTER statement options.

Rotating and Tilting the Plot

For both surface plots and scatter plots, you can rotate the X-Y plane around the Z axis, or tilt the X-Y plane toward you. When you rotate a plot, you can view data from any angle around the three-dimensional graph. Rotating a plot is useful for bringing into view data points that might be obscured by other data points. Tilting a plot enables you to accentuate the location of data points.

The following diagram illustrates how the TILT= option, and the ROTATE= option change the viewing angles of a plot.

Figure 46.4  Rotating and Tilting a Plot
Controlling the Axes

Because the relationship between a plot's surface and the actual data values can be difficult to interpret, the readability of the plot can be enhanced by changing the number of tick marks on the axes, or restricting the vertical axis range.

The G3D procedure supports AXIS definitions for Java and ActiveX only. However, you can use the functionality of PLOT and SCATTER statements to do the following:

- suppress the axes
- suppress axis labels
- suppress tick mark values
- specify the number of tick marks
- specify minimum and maximum values for the Z axis
- specify whether grid lines connect axis tick marks

The font and height of the graph's text can be changed with the GOPTIONS FTEXT= option and the GOPTIONS HTEXT= option, respectively. The GOPTIONS FBY= option can be used to specify the font for the BY-labels for BY-group graphs.

Syntax: G3D Procedure

Restrictions: This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

Requirement: At least one PLOT or SCATTER statement is required.

Global statements: AXIS, FOOTNOTE, GOPTIONS, TITLE

Supports: BY-group processing

Notes: When plotting a large data set with the G3D procedure, SAS might not be able to allocate enough memory to process the data. In that case, the G3D procedure step terminates, and an error message is written to the SAS log. To correct the problem, increase the amount of memory that is available to SAS or reduce the data through subsetting or some other means.

When the ACTIVEX device is used to generate the graph, the graph data tips show the interpolated values rather than the actual data values.

The procedure can include the SAS/GRAPH statements BY on page 370 and NOTE on page 447, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See Chapter 24, “SAS/GRAPH Statements,” on page 343 and SAS DATA Step Statements: Reference for more information.

PROC G3D <DATA=input-data-set>
PROC G3D Statement

Can identify the data set that contains the plot variables. Can also specify an annotate data set and an output catalog.

Syntax

PROC G3D <DATA= input-data-set>
<ANNOTATE= annotate-data-set>
<GOUT= <libref.> output-catalog> ;

Optional Arguments

PROC G3D statement options affect all graphs produced by the procedure.

ANNOTATE= annotate-data-set

specifies an annotate data set to annotate all of the graphs that are produced by the G3D procedure. To annotate individual graphs, use the ANNOTATE= option in the action statement.

Alias ANNO=

See Chapter 27, “Using Annotate Data Sets,” on page 635

DATA= input-data-set

specifies the SAS data set that contains the variables to plot. By default, the G3D procedure uses the most recently created SAS data set.

See “The SAS Data Set: Your Key to the SAS System” in Step-by-Step Programming with Base SAS

“About Data Set Options” in SAS Data Set Options: Reference

“The Input Data Set” on page 1384

GOUT= <libref.> output-catalog

specifies the SAS catalog in which to save the graphics output that is produced by the G3D procedure. If you omit the libref, the output is placed in the temporary catalog WORK.GSEG. The temporary output catalog is created if it does not already exist.

See Chapter 11, “SAS/GRAPH Output,” on page 105

PLOT Statement

Creates three-dimensional surface plots using values of three numeric variables from the input data set.

Restriction: The AXIS statement is partially supported by the Java and ActiveX devices only.
**Requirement:**
One plot request is required.

**Global statements:**
AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE

**Note:**
The procedure can include the FORMAT, LABEL, and WHERE statements.

## Syntax

```
PLOT y*x=z </option(s)>;
```

### Summary of Optional Arguments

**Appearance options**

- `ANNOTATE=annotate-data-set` specifies an annotate data set to annotate plots that are produced by the PLOT statement.
- `CBOTTOM=bottom-surface-color` specifies a color for the bottom of the plot surface.
- `CTOP=top-surface-color` specifies a color for the top of the plot surface.
- `ROTATE=angle-list` specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis.
- `SIDE` produces a surface graph with a side wall.
- `TILT=angle-list` specifies one or more angles to tilt the graph toward you.
- `XYTYPE=0 | 1 | 2 | 3` specifies the direction of lines that are used to represent the plot's surface.

**Axes options**

- `CAXIS=axis-color` specifies a color for all the axes lines and tick marks.
- `CTEXT=text-color` specifies a color for the axis labels and axis tick mark values.
- `GRID` draws reference lines at the major tick marks on all axes.
- `NOAXIS` specifies that the plot has no axes, axes labels, or tick mark values.
- `NOLABEL` specifies that the plot has no axis labels or tick mark values.
- `XAXIS= AXIS<1...<99>` assigns an axis definition.
- `XTICKNUM=number-of-major-tick–marks` specifies the number of major tick marks that are located on a plot's x axis.
- `YAXIS= AXIS <1...<99>` assigns an axis definition.
- `YTICKNUM=number-of-major-tick–marks` specifies the number of major tick marks that are located on a plot's Y axis.
- `ZAXIS= AXIS<1...<99>`
assigns an axis definition.

ZMAX=maximum-axis-value
specifies the maximum value that is displayed on a plot's Z axis.

ZMIN=minimum-axis-value
specifies the minimum value that is displayed on a plot's Z axis.

ZTICKNUM=number-of-major-tick-marks
specifies the number of major tick marks that are located on a plot's Z axis.

Catalog entry description options

DESCRIPTION="description"
specifies a description of the output.

NAME="name"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Required Argument

y\times x=z;
specifies the three numeric variables from the input data set:

Y
is the horizontal variable whose values are plotted on the Y axis

X
is the horizontal variable whose values are plotted on the X axis

Z
is the vertical variable whose values are plotted on the Z axis

Optional Arguments

Options in a PLOT statement affect all graphs that are produced by that statement. You can specify as many options as you want, and list them in any order.

ANNOTATE=annotate-data-set
specifies an annotate data set to annotate plots that are produced by the PLOT statement.

Alias      ANNO=

See       Chapter 27, “Using Annotate Data Sets,” on page 635

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
CTEXT=**text-color**  
specifies a color for the axis labels and axis tick mark values. The G3D procedure uses the first color that it finds from the following list:

- colors specified for labels and values on assigned AXIS statements, which override the CTEXT= option in the PLOT statement. (Colors specified in AXIS statements are supported by the Java and ActiveX devices only.)
- the color specified by the CTEXT= option in the PLOT statement
- the color specified by the CTEXT= option in a GOPTIONS statement
- the color specified in the current style, or the first color in the color list for all of the other devices

If the NOGSTYLE system option is specified, the CTEXT= option color is assigned as follows:

- for the Java and ActiveX devices the default color is black
- for all other devices, the first color in the device's color list

### Style Reference

| Color attribute of the GraphValueText and the GraphLabelText elements |

### Notes

If you use a BY statement in the procedure, the color of the BY variable label is controlled by the CBY= option in the GOPTIONS statement.

For Java and ActiveX, specific text options specified in the AXIS statement override the CTEXT= option.

CTOP=**top-surface-color**  
specifies a color for the top of the plot surface.

### Style Reference

| Color attribute of the GraphData1 element |

### Example

“Example 2: Generating a Rotated Surface Plot” on page 1405

DESCRIPTION=“**description**”  
specifies a description of the output. The maximum length for description is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on which device driver you are using). See “Chart Descriptions for Web Presentations” on page 189 for more information.
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The description can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to “Substituting BY Line Values in a Text String” on page 959.
The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

<table>
<thead>
<tr>
<th>Alias</th>
<th>DES=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>3D surface plot of (z) by (x) and (y)</td>
</tr>
<tr>
<td>Restriction</td>
<td>Partially supported for ActiveX and Java</td>
</tr>
<tr>
<td>See</td>
<td>“Substituting BY Line Values in a Text String” on page 959</td>
</tr>
</tbody>
</table>

**GRID**

draws reference lines at the major tick marks on all axes.

<table>
<thead>
<tr>
<th>Restriction</th>
<th>Not supported by Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>“Example 2: Generating a Rotated Surface Plot” on page 1405</td>
</tr>
</tbody>
</table>

**NAME=\"name\"**

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to `name`:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

**For the GRSEG entry name:**

- The name is truncated to eight characters.
- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, `name1`, `name2`, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.

**For the graphics output filename:**

- The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
- All characters are represented in lowercase.
- If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
- If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.
Note: Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default: G3D

NOAXIS
specifies that the plot has no axes, axes labels, or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set, or with the AXIS statement for Java and ActiveX.

Alias: NOAXES

NOLABEL
specifies that the plot has no axis labels or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set, or with the AXIS statement for Java and ActiveX.

ROTATE=angle-list
specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis. Specify the values in degrees. The values specified in the angle-list can be negative or positive. If you specify a sequence of angles, separate graphs are produced for each angle. The angles that are specified in the ROTATE= option are paired with any angles that are specified with the TILT= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The angle-list list is in one of the following forms:

- an explicit list of values: n <...n>
- a starting value and an ending value with an interval increment: n TO n <BY increment>
- a combination of both forms: n <...n> TO n <BY increment> <n <...n> >

Default: 70 degrees

Restriction: Not supported by ActiveX

Example: “Example 2: Generating a Rotated Surface Plot” on page 1405

SIDE
produces a surface graph with a side wall.

Restriction: Partially support by Java

Example: “Example 3: Generating a Tilted Surface Plot” on page 1407

TILT=angle-list
specifies one or more angles to tilt the graph toward you. The values must be specified in degrees. The valid values specified in the angle-list are 0 through 90. To generate a sequence of graphs, specify multiple angles, a graph is generated for each angle. The angles that are specified in the TILT= option are paired with any angles that are specified in the ROTATE= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The angle-list is in one of the following forms:
• an explicit list of values: \( n <...n> \)
• a starting value and an ending value with an interval increment: \( n \ TO \ n <BY \ increment> \)
• a combination of both forms: \( n <...n> \ TO \ n <BY \ increment> <n <...n> > \)

Default: 70 degrees

Example: “Example 3: Generating a Tiled Surface Plot” on page 1407

**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 XTYPE= option are as follows:

• XYTYPE=0 (Java and ActiveX only) No lines are displayed. The plot is displayed as a solid surface.
• XYTYPE=1 draws lines that are parallel to the X axis. The surface is displayed by using lines that represent Y-axis values.
• XYTYPE=2 draws lines that are parallel to the Y axis. The surface is displayed by using lines that represent X-axis values.
• XYTYPE=3 draws lines that are parallel to both the X and Y axes. Displays the surface by using lines that represent values for both X and Y.

Restriction: Not supported by Java

Example: “Changing the Surface Appearance” on page 1395

**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.
ZMAX=maximum-axis-value
specifies the maximum value that is displayed on a plot's Z axis. Defining the ZMAX= option value greater than the data that is in the input data set, extends the plot's Z axis. Defining the ZMAX= option value less than the maximum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping.

The value of the ZMAX= option must be greater than the value of the ZMIN= option.

Default The maximum value of the Z variable

Restriction Not supported by Java

Example “Example 2: Generating a Rotated Surface Plot” on page 1405

ZMIN=minimum-axis-value
specifies the minimum value that is displayed on a plot's Z axis. Defining the ZMIN= option value less than the minimum value in the input data set extends the plot's Z axis. Defining the ZMIN= value greater than the minimum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping.

The value of the ZMIN= option must be less than the value of the ZMAX= option.

Default The minimum value of the Z variable

Restriction Not supported by Java

Example “Example 2: Generating a Rotated Surface Plot” on page 1405

ZTICKNUM=number-of-major-tick-marks
specifies the number of major tick marks that are located on a plot's Z axis. At least two values are needed to generate the axis.

Default 4 (except ActiveX is 5)

Restriction Not supported by Java

Details

Description
The PLOT statement specifies one plot request that identifies the three numeric variables to plot. The statement also does the following actions:

• scales the axes to include the minimum data values and the maximum data values for each of the plotted variables X, Y, and Z
• labels each axis with the name of the variable or its associated label
• derives its colors from the ODS style

In addition to the Global statement options, the following Plot statement options enable you to specify the appearance of many of the plot's elements.
Changing the Surface Appearance
The XYTYPE= option specifies the direction of the lines that form the surface plot. The following plots show examples of each type of plot surface.

Figure 46.5 Surface Appearance for XYTYPE=1

Figure 46.6 Surface Appearance for XYTYPE=2
SCATTER Statement

Creates three-dimensional scatter plots using values of three numeric variables from the input data set.

Alias: SCAT

Restriction: The AXIS statement is partially supported by Java and ActiveX devices only.

Requirement: One plot request is required.

Global statements: AXIS, BY, FOOTNOTE, GOPTIONS, NOTE, TITLE

Notes: The procedure can include the FORMAT, LABEL, and WHERE statements. The SCATTER statement does not require a full grid of observations to generate a plot.

Syntax

SCATTER y*x=z <option(s)>;

Summary of Optional Arguments

Appearance options

ANNOTATE=annotate-data-set
specifies an annotate data set to annotate plots that are produced by the SCATTER statement.

COLOR="data-point-color" | data-point-color-variable
specifies a color name or a character variable in the input data set whose values are color names.

NONEEDLE
specifies that a plot has no lines that connect the shapes representing data points to the X-Y plane.

**ROTATE=angle-list**
specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis.

**SHAPE=“symbol-name” | shape-variable**
specifies a symbol name or a character variable whose values are symbol names.

**SIZE=symbol-size | size-variable**
specifies either a constant or a numeric variable, the values of which determine the size of symbol shapes on the scatter plot.

**TILT=angle-list**
specifies one or more angles at which to tilt the graph toward you.

**Axes options**

**CAXIS=axis-color**
specifies a color for axis lines, tick marks, and horizontal grid lines.

**CTEXT=text-color**
specifies a color for all text on the axes, including tick mark values and axis labels.

**GRID**
draws reference lines at the major tick marks on all axes.

**NOAXIS**
specifies that a plot has no axes, including labels, tick marks, and values.

**NOLABEL**
specifies that a plot has no axes labels or tick mark values.

**XAXIS= AXIS<1...<99>**
assigns an axis definition.

**XTICKNUM=number-of-tick-marks**
specify the number of major tick marks that are located on a plot's X axis.

**YAXIS= AXIS<1...<99>**
assigns an axis definition.

**YTICKNUM=number-of-tick-marks**
specify the number of major tick marks that are located on a plot's Y axis.

**ZAXIS= AXIS<1...<99>**
assigns an axis definition.

**ZMAX=maximum-value**
specify the maximum data value that is displayed on a plot's Z axis.

**ZMIN=minimum-value**
specify the minimum value that is displayed on a plot's Z axis.

**ZTICKNUM=number-of-tick-marks**
specify the number of major tick marks that are located on a plot's Z axis.

**Catalog entry description options**

**DESCRIPTION=“description”**
specifies the description of the plot.

**NAME="name"**
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.
Required Argument

\[ y \times x = z; \]

specifies three numeric variables from the input data set:

- **Y** specifies a horizontal variable whose values are plotted on the Y axis
- **X** specifies a horizontal variable whose values are plotted on the X axis
- **Z** specifies a vertical variable whose values are plotted on the Z axis

Optional Arguments

Options in a SCATTER statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

**ANNOTATE=** annotate-data-set

specifies an annotate data set to annotate plots that are produced by the SCATTER statement.

<table>
<thead>
<tr>
<th>Alias</th>
<th>ANNO=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>Partially supported by Java and ActiveX</td>
</tr>
</tbody>
</table>

**CAXIS=** axis-color

specifies a color for axis lines, tick marks, and horizontal grid lines.

<table>
<thead>
<tr>
<th>Style reference</th>
<th>Color attribute of the GraphAxisLines element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>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.</td>
</tr>
</tbody>
</table>

**COLOR=** "data-point-color" | data-point-color-variable

specifies a color name or a character variable in the input data set whose values are color names. These color values determine the color or colors of the shapes that represent a plot's data points. Color values must be valid color names for the device that is used.

Using a list of colors in the value of the data-point-color-variable enables you to assign different colors to the shapes to classify data.

| Style reference | Color attribute of the GraphData1 element |

**CTEXT=** text-color

specifies a color for all text on the axes, including tick mark values and axis labels. The G3D procedure uses the first color that it finds from the following list:

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:

1. for the Java and ActiveX devices the default color is black
- for all other devices, the first color in the device's color list

**Style reference**

Color attribute of the GraphValueText and GraphLabelText elements

**Notes**

If you use a BY statement in the procedure, the color of the BY variable label is controlled by the CBY= option in the GOPTIONS statement.

For Java and ActiveX only, specific text options specified in the AXIS statement override the CTEXT= option.

**DESCRIPTION=**\textit{“description”}

specifies the description of the plot. The descriptive text is displayed as follows:

- the description in the Results window
- the properties that you view from the Explorer window
- the description in the Explorer view of the catalog entry
- the Table of Contents that is generated when you use CONTENTS= in an ODS HTML statement, assuming that the G3D output is generated while the contents page is open
- the description field of the PROC GREPLAY window
- the ALT= text in the HTML file when the output destination is ODS HTML
- customized by inserting BY variable values with \#BYLINE, \#BYVAL(n), and \#BYVAR(n)

**Alias** DES=

**Default** 3D surface plot of $z$ by $x$ and $y$

**See** “Substituting BY Line Values in a Text String” on page 959

**GRID**

draws reference lines at the major tick marks on all axes.

**NAME=**\textit{name}

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to \textit{name}:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.

**For the GRSEG entry name:**

- The name is truncated to eight characters.
- The first character is always represented in uppercase, and all other characters are represented in lowercase.
- If the name begins with a number, an underscore is prepended to the name.
- If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, \textit{name}1, \textit{name}2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
• **For the graphics output filename:**
  
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See “Controlling Graphics Output for ODS LISTING” on page 118.
  
  - All characters are represented in lowercase.
  
  - If a number is added to the GRSEG name, the same number is added to the output filename. See “About Filename Indexing” on page 119.
  
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See “About Filename Indexing” on page 119.

  **Note:** Prior to SAS 9.4M2, if the name begins with a number, an underscore is prepended to the filename.
  
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default: **G3D**

**NOAXIS**

specifies that a plot has no axes, including labels, tick marks, and values. Use this option if you want to generate axes with an annotate data set.

Alias: **NOAXES**

**NOLABEL**

specifies that a plot has no axes labels or tick mark values. Use this option if you want to generate axis labels and tick mark values with an annotate data set.

**NONEEDLE**

specifies that a plot has no lines that connect the shapes representing data points to the X-Y plane.

Restriction: The NONEEDLE option has no effect when SHAPE=“PILLAR” or when SHAPE=“PRISM”

**ROTATE=angle-list**

specifies one or more angles at which to rotate the X-Y plane around the perpendicular Z axis. Specify the value in degrees. The values specified in the angle-list can be negative or positive. The value can be greater than 360 degrees. If you specify a sequence of angles, separate graphs are produced for each angle. The angles that are specified in the ROTATE= option are paired with any angles that are specified with the TILT= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The angle-list list is in one of the following forms:

- an explicit list of values: $n <...n>$
- a starting value and an ending value with an interval increment: $n$ TO $n$ <BY increment>
• a combination of both forms: $n <...n> \ TO \ n <BY \ increment > <n <...n> >$

**Default**

70 degrees

**Restriction**

Not supported by ActiveX

**SHAPE=**“symbol-name” | shape-variable

specifies a symbol name or a character variable whose values are symbol names. If you specify SHAPE=“symbol-name”, all data points are drawn in that shape.

If you specify SHAPE=shape-variable, the shape of the data point is determined by the value of the shape variable, in the input data set, for that observation. For example, the procedure uses the value of the variable CLASS for a particular observation as the shape for that data point when you specify:

`shape=class`

Using a list of values in the variable named in SHAPE=shape-variable enables you to assign different shapes to the data points, to categorize your data.

Valid values for symbol-name are as follows:

- BALLOON
- CLUB
- CROSS
- CUBE
- CYLINDER
- DIAMOND
- FLAG
- HEART
- PILLAR
- POINT
- PRISM
- PYRAMID
- SPADE
- SQUARE
- STAR
SIZE= symbol-size | size-variable

specifies either a constant or a numeric variable, the values of which determine the size of symbol shapes on the scatter plot. If you specify SIZE= symbol-size, all data points are drawn in that size.

If you specify SIZE= size-variable, the size of the data point is determined by the value of the size variable, in the input data set for that observation. For example, when you specify SIZE= CLASS, the procedure uses the value of the variable CLASS, for each observation in the input data set as the size of that data point. If you use a list of sizes as the value of the variable named in SIZE= size-variable, you can assign different sizes to the data points to categorize your data.

TILT= angle-list

specifies one or more angles at which to tilt the graph toward you. The value must be specified in degrees. The valid values specified in the angle-list are 0 through 90. To generate a sequence of graphs, specify different angles, and a graph is generated for each angle. The angles that are specified in the TILT= option are paired with any angles that are specified with the ROTATE= option. If one option contains fewer values than the other, the last value in the shorter list is paired with the remaining values in the longer list. The angle-list is in one of the following forms:

- an explicit list of values: n <...n>
- a starting value and an ending value with an interval increment: n TO n <BY increment>
- a combination of both forms: n <...n> TO n <BY increment> <n <...n>>

Default 70 degrees

XAXIS= AXIS<1...<99>

assigns an axis definition.
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 only

ZMAX=maximum-value
specify the maximum data value that is displayed on a plot's Z axis. You can use the ZMAX= option to extend the Z axis beyond the value range. The value that is specified by the ZMAX= option must be greater than that specified by the ZMIN= option. If you specify the ZMAX= option within the range of the Z variable values, the plot's data values are clipped at the level that you specified.

Default Maximum value of Z variable

ZMIN=minimum-value
specifies the minimum value that is displayed on a plot's Z axis. Defining the ZMIN= value less than the minimum value in the input data set extends the plot's Z axis. Defining the ZMIN= value greater than the minimum value in the input data set displays all Z values in the range of ZMIN-to-ZMAX, and might cause data clipping. The value of the ZMIN= option must be less than the value of the ZMAX= option.

Default The minimum value of the Z variable

ZTICKNUM=number-of-tick-marks
specify the number of major tick marks that are located on a plot's Z axis. At least two values are needed to generate the axis.

Default 4 (except ActiveX is 5)

Details

Description
The SCATTER statement specifies one plot request that identifies the three numeric variables to plot. The statement also does the following actions:

• scales the axes to include the minimum and maximum values for each of the plotted variables $X$, $Y$, and $Z$
labels each axis with the name of the plotted variable or its associated label
- uses reference lines to mark the major tick marks on the X and Y axes, creating a grid on the horizontal plane
- represents each data point with a pyramid that is connected to the horizontal plane with a needle
- derives its colors from the ODS style

In addition to the Global Statement options, the following Scatter statement options enable you to specify the appearance of many of the plot's elements.

**Changing the Appearance of the Data Points**

Use the COLOR=, SHAPE=, and SIZE= options to change the appearance of your scatter plot or to classify data using color, shape, size, or any combination of these features. Figure 46.8 on page 1402 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:

```sas
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:

```sas
scatter y*x=z / size=type;
```

---

**Examples: G3D Procedure**

**Example 1: Generating a Surface Plot**

| Features:  | PLOT statement |
| Sample library member: | GTDSURFA |
| Note: | The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com. |

This surface plot reveals the shape of a generated data set named LAKE. The axes are scaled to include all data values. Each axis is labeled with the name or label of the corresponding variable. The tick marks on the axes are divided into three even intervals. The horizontal plane is rotated 70° around the Z axis. The graph is tilted 70 degrees toward you. The colors are derived from the ODS style.
Program

goptions reset=all border;
title "Surface Plot";
proc g3d data=sashelp.lake;
  plot length*width=depth;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the title.

title "Surface Plot";

Generate the surface plot.

proc g3d data=sashelp.lake;
  plot length*width=depth;
run;
quit;

Example 2: Generating a Rotated Surface Plot

Features:
- PLOT statement
- PLOT statement options
  - CTOP=
  - GRID
  - ROTATE=
The surface plot shown in this example illustrates enhancements to the axes and the presentation. The plot illustrates a grid originating from the tick marks. A Z–axis range increase raised the plot above the horizontal X-Y plane. CTOP= green changed the top color and ROTATE= rotated the plot 45 degrees toward the viewer.

Program Description

Set the graphics environment.
Define the title.

title "Rotated Surface Plot";

Generate the surface plot. CTOP=green changes the color of the plot's top surface. The GRID option draws reference lines originating from the tick marks on all the axes. The ROTATE= option specifies a rotation angle of 45°. ZMAX=5 specifies the maximum value for the Z axis. ZMIN= –50 specifies the minimum value for the Z axis. Specifying a ZMIN= value that is below the minimum value in the input data set raises the plot above the horizontal plane. Data is not displayed if it exceeds the range specified by the ZMIN= and ZMAX= options.

proc g3d data=sashelp.lake;
  plot length*width=depth/
    ctop=green
    grid
    rotate=45
    zmax=5
    zmin=-50;
run;
quit;

Example 3: Generating a Tilted Surface Plot

Features:
- PLOT statement
- PLOT statement options
  - SIDE
  - TILT=

Data set: SASHELP.LAKE

Sample library member: GTDTILT

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

Simple modifications displayed in “Example 1: Generating a Surface Plot” on page 1404 are generated by tilting the surface plot 30 degrees toward you, and adding a side wall.
Program

goptions reset=all border;
title "Tilted Surface Plot";
proc g3d data=sashelp.lake;
   plot length*width=depth/
      side
tilt=30;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define title.

title "Tilted Surface Plot";

Generate the surface plot. The SIDE option draws a side wall for the graph. The TILT= option specifies a tilt angle of 15° for the plot. The initial rotation of 70° is not affected by the TILT= option.

   proc g3d data=sashelp.lake;
      plot length*width=depth/
         side
tilt=30;
run;
quit;
Example 4: Generating a Scatter Plot

Features:
- SCATTER statement
- SCATTER statement options
  - COLOR=
  - SIZE=

This scatter plot examines the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. The Scatter statement in this example relies on the procedure defaults to scale the axes to include all the data values, label the axes with the variable's labels, divide the axes into three even intervals, rotate the horizontal plane 70 degrees around the vertical axis, tilt the plot 70 degrees toward you, and display the plot with the default ODS style.

Program

goptions reset=all border;
title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=r "Sepal Width not Shown";
proc g3d data=sashelp.iris;
  scatter PetalLength*PetalWidth=SepalLength /
    color="light blue"
    size=1.5;
run;
quit;
Program Description

Set the graphics environment.

goptions reset=all border;

Define the titles and footnote.

title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=r "Sepal Width not Shown";

Generate the surface plot. Set the marker color to light blue and the marker size to 1.5 cells.

proc g3d data=sashelp.iris;
  scatter PetalLength*PetalWidth=SepalLength /
    color="light blue"
    size=1.5;
  run;
quit;

Example 5: Generating a Scatter Plot with Modified Shapes

Features:
- SCATTER statement
- SCATTER statement options
  - COLOR=
  - SHAPE=
  - SIZE=

Other features:
- NOTE statement
- DATA Step

This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. It uses a DATA step to add a color variable and a shape variable to the data set, shapes to distinguish iris species, colors to distinguish iris species, and a NOTE statement to simulate a legend.
Program

goptions reset=all border;

title1 "Iris Species Classification";
title2 "Physical Measurement";
title3 "Source: Fisher (1936) Iris Data";
footnote1 j=l "Sepal Width not Shown";

data iris;
   set sashelp.iris;
   length color shape $8.;
   if species="Setosa" then do; shape="club"; color="blue"; end;
   if species="Versicolor" then do; shape="diamond"; color="red"; end;
   if species="Virginica" then do; shape="spade"; color="green"; end;
run;

proc g3d data=iris;
   note j=r f="Albany AMT/bo" "Species:  " c=green "Virginica   
   j=r  c=red "Versicolor   
   j=r  c=blue "Setosa     
scatter PetalLength*PetalWidth=SepalLength/
   color=color
   shape=shape
   size=1.5;
run;
quit;

Program Description

Set the graphics environment.
    goptions reset=all border;

Define the titles and footnote.
    title1 "Iris Species Classification";
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;
rung;

Generate the surface plot.

proc g3d data=iris;
note j=r f="Albany AMT/bo" "Species: " c=green "Virginica "
    j=r c=red "Versicolor "
    j=r c=blue "Setosa ";
scatter PetalLength*PetalWidth=SepalLength/
    color=color
    shape=shape
    size=1.5;
run;
quit;

Example 6: Generating a Scatter Plot with Modified Shapes and a Grid

Features: SCATTER statement
- SCATTER statement options
  - COLOR=
  - GRID
  - NONEEDLE
  - SHAPE=
  - SIZE=

Other features: DATA Step

This scatter plot modifies the results of measuring the petal length, petal width, and sepal length for the flowers of three species of irises. It uses a DATA step to add a color variable and a shape variable to the data set, shapes to distinguish iris species, and colors to distinguish iris species. It also removes needles from data points and adds a grid.
Program

goptions reset=all border;

  title1 "Iris Species Classification";
  title2 "Physical Measurement";
  footnote1 j=r f="Albany AMT/it" "Source: Fisher (1936) Iris Data";

data iris;
  set sashelp.iris;
  length color shape $8.;
  if species="Setosa" then do; shape="club"; color="blue"; end;
  if species="Versicolor" then do; shape="diamond"; color="red"; end;
  if species="Virginica" then do; shape="spade"; color="green"; end;
run;

proc g3d data=iris;
  scatter PetalLength*PetalWidth=SepalLength/
       color=color
       shape=shape
       size=1.5
       noneedle
       grid;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the titles and footnote.

title1 "Iris Species Classification";
  title2 "Physical Measurement";
  footnote1 j=r f="Albany AMT/it" "Source: Fisher (1936) Iris Data";
Add variables to original data set.

data iris;
    set sashelp.iris;
    length color shape $8.;
    if species="Setosa" then do; shape="club"; color="blue"; end;
    if species="Versicolor" then do; shape="diamond"; color="red"; end;
    if species="Virginica" then do; shape="spade"; color="green"; end;
run;

Generate the surface plot.

proc g3d data=iris;
    scatter PetalLength*PetalWidth=SepalLength/
        color=color
        shape=shape
        size=1.5
        noneedle
        grid;
run;
quit;

Example 7: Generating a Rotated Scatter Plot with Modified Axes

Features: Scatter statement

SCATTER statement options
    CAXIS=
    COLOR=
    ROTATE=
    SHAPE=
    SIZE=
    XTTICKNUM=
    YTTICKNUM=
    ZTTICKNUM=
    ZMAX=
    ZMIN=

This scatter plot modifies the procedure defaults. It specifies a shape for the data points and classifies the data by color. For the axes, it specifies blue as the axis color. It specifies five major tick marks for the Y axis and two major tick marks for the X axis. For the Z axis, it specifies five major tick marks, zero as the minimum axis value, and one hundred as the maximum axis value. It also rotates the X-Y plane –15 degrees around the perpendicular Z axis.
Program

goptions reset=all border;

title1 "Relative Humidity in Percent";
footnote1 j=r f="Albany AMT/it"

proc g3d data=sashelp.humid;
  scatter airtemp*bulbtemp=humidity/
    shape="pillar"
    color=colorvar
    caxis=blue
    rotates=-15
    yticknum=5
    xticknum=2
    zticknum=4
    zmin=0
    zmax=100;
  run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the titles and footnote.

title1 "Relative Humidity in Percent";
footnote1 j=r f="Albany AMT/it"
Generate the surface plot.

```sas
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


Overview: G3GRID Procedure

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 47.1 on page 1418, and Figure 47.2 on page 1419 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 47.1** Scatter Plot of Data Set before G3GRID Processing (gtgdefin)

![Scatter Plot of NUMS Data Set](image)

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.
The Input Data Set

The input data set must contain at least three numeric variables:

- two horizontal variables \((x, y)\)
- one or more vertical variables, \(z\) through \(z-n\), that is interpolated or smoothed as if it were a function of the two horizontal variables

The G3GRID procedure can process multiple vertical variables for each pair of horizontal variables that you specify:

- if you specify more than one vertical variable, the G3GRID procedure performs a separate analysis, and produces interpolated or smoothed values for each vertical variable
- if more than one observation in the input data set has the same values for both horizontal variables, \(x\) and \(y\), only the first observation is used in the interpolation. A warning message is printed to the log.
- by default, the interpolation is performed after both variables are similarly scaled, because the interpolation methods assume that the scales of \(x\) and \(y\) are comparable

Multiple Vertical Variables

The GRID statement, enables you to name multiple vertical variables \((z - z-n)\), to produce a data set that contains two horizontal variables and multiple vertical variables. The resulting data set enables you to produce plots of the relationships of the two horizontal variables, to different vertical variables.
Horizontal Variables along a Nonlinear Curve

If the points that are generated by the horizontal variables tend to lie along a curve, a poor interpolation or spline can result. In such cases, the vertical variable(s), and one of the horizontal variables should be modeled as a function of the remaining horizontal variable. A scatter plot of the two horizontal variables enable you to determine the appropriate function.

If the horizontal variable points are collinear, the procedure interpolates the function as constant, along lines perpendicular to the line in the plane that is generated by the input data points.

The Output Data Set

The output data set contains:

- the two horizontal variables
- the interpolated or smoothed vertical variables
- any BY variables

G3Grid enables you to control both the number of $x$ and $y$ values in the output data set, and the values themselves. In addition, you can specify an interpolation method.

Interpolation Methods

Types of Interpolation Methods

The G3GRID procedure can use one of three interpolation methods: bivariate interpolation (the default), spline interpolation, and smoothing spline interpolation.

Bivariate Interpolation

Unless you specify the SPLINE option, the G3GRID procedure is an interpolation procedure. It calculates the $z$ values for $x$, $y$ points that are missing from the input data set. The surface that is formed by the interpolated data passes precisely through the data points in the input data set.

This method of interpolation works best for fairly smooth functions, with values given at uniformly distributed points in the plane. If the data points in the input data set are erratic, the default interpolated surface can be erratic.

This default method is a modification of that described by Akima (1978). This method consists of the following actions:

1. dividing the plane into non-overlapping triangles that use the positions of the available points
2. fitting a bivariate fifth degree polynomial within each triangle
3. calculating the interpolated values by evaluating the polynomial at each grid point that falls in the triangle

The coefficients for the polynomial are computed based on the following criteria:

- the values of the function at the vertices of the triangle
- the estimated values for the first, and second derivatives of the function at the vertices

The estimates of the first, and second derivatives are computed using the \( n \) nearest neighbors of the point, where \( n \) is the number specified in the GRID statement's NEAR= option. A Delauney triangulation is used for the default method (Ripley 1981, p. 38). The coordinates of the triangles are available in an output data set, if requested by the OUTTRI= option, in the PROC G3GRID statement. This is the default interpolation method.

**Spline Interpolation**

If you specify the SPLINE option, a method is used that produces either an interpolation or smoothing that is optimally smooth. See Harder and Desmarais (1972), Meinguet (1979), and Green and Silverman (1994). The surface that is generated can be thought of as one that would be formed if a stiff, thin metal plate were forced through, or near the given data points. For large data sets, this method is substantially more expensive than the default method.

The function \( u \), formed when you specify the SPLINE option, is determined by letting

\[
  t_j = (x_j, y_j)
\]

\[
  t = (x, y)
\]

and

\[
  |t - t_j| = \left( (x - x_j)^2 + (y - y_j)^2 \right)^{1/2}
\]

\[
  u(x, y) = \sum_{j=1}^{n} c_j E(t, t_j) + d_0 + d_1x + d_2y
\]

where

\[
  E(s, t) = |s - t| \log(|s - t|)
\]

The coefficients \( c_1, c_2, ..., c_n \), and \( d_1, d_2, d_3 \) of this polynomial are determined by the following equations:

\[
  (E + n\lambda I) c + Td = z
\]

and

\[
  T'c = 0
\]

where:

- \( E \) is the \( n \times n \) matrix \( E(t_i, t_j) \)
- \( I \) is the \( n \times n \) identity matrix
\( \lambda \) is the smoothing parameter that is specified in the SMOOTH= option

c is \((c_1, \ldots, c_n)\)

z is \((z_1, \ldots, z_n)\)

d is \((d_1, d_2, d_3)\)

\( T \) is the \( n \times 3 \) matrix whose \( i \)th row is \((1, x_i, y_i)\)

See Wahba (1990) for more detail.

**Spline Smoothing**

Using the SMOOTH= option in the GRID statement with the SPLINE option, enables you to produce a smoothing spline. See Eubank (1988) for a general discussion of spline smoothing. The value or values specified in the SMOOTH= option are substituted for \( \lambda \) in the equation that is described in “Spline Interpolation” on page 1421. A smoothing spline trades closeness to the original data points for smoothness. To find a value that produces the best balance between smoothness, and fit to the original data, several values for the SMOOTH= option can be run.

### Syntax: G3GRID Procedure

**Restriction:** This procedure is not included in SAS Viya. However, this procedure can use data that has been processed in CAS. For more information, see Chapter 12, “Plotting a Cloud Analytic Services (CAS) In-Memory Table,” on page 135.

**Requirement:** Exactly one GRID statement is required.

**Note:** The procedure can include the "BY Statement" on page 370.

```plaintext
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.

**Requirement:** An input data set is required.

**Syntax**

```plaintext
PROC G3GRID <DATA=input-data-set>
<OUT=output-data-set>
<OUTTRI=output-data-set>;
```
Optional Arguments

**DATA=** *input-data-set*

specifies the SAS data set that contains the variables to process. By default, the procedure uses the most recently created SAS data set.

See “The SAS Data Set: Your Key to the SAS System” in *Step-by-Step Programming with Base SAS*.

“About Data Set Options” in *SAS Data Set Options: Reference*

“*The Input Data Set*” on page 1419

**OUT=** *output-data-set*

specifies the output data set. The data set contains any BY variables that you specify, the interpolated or smoothed values of the vertical variables (z through z-n), and the coordinates for all grid positions on the horizontal (x-y) plane. If you specify smoothing, the output data set also contains a variable named _SMTH_, whose value is a smoothing parameter. The observations in this data set are ordered by any variables that you specify with a BY statement. By default, the output of PROC G3GRID creates WORK.DATA1.

Depending on the shape of the original data, and the options that you use, the output data set can contain values for the vertical (z through z-n) values that are outside of the range of the original values in the data set.

Example “Example 1: Using the Default Interpolation Method” on page 1428

**OUTTRI=** *output-data-set*

specifies an additional output data set that contains triangular coordinates. The data set will contain any BY variables that you specify, the two horizontal variables giving the horizontal (x-y) plane coordinates of the input points, and a variable named TRIANGLE that uses the integer values to label the triangles. The observations in this data set are ordered by any variables that you specify with a BY statement.

The data set contains three observations for each value of the variable TRIANGLE. The three observations give the coordinates of the three vertices of the triangle. Points on the convex hull of the input data set of points are also assumed to lie in degenerate triangles, whose other vertices are at infinity. The points in the convex hull can be recovered by keeping only those triangles with exactly two missing vertices.

By default, no OUTTRI= data set is produced. OUTTRI= is not valid when you specify the SPLINE option in the GRID statement.

GRID Statement

Specifies the three numeric variables for interpolation or for smoothing. Can also specify the number of observations (x and y values), in the output data set; output values for the two horizontal variables x-y; and the interpolation method for the vertical variables.

**Requirement:** Exactly one grid request is required.
Syntax

GRID grid-request /option(s);

Summary of Optional Arguments

Grid options

AXIS1=ascending-value-list
specifies a list of numeric values to assign to the first (y) variable in the grid request for the output data set.

AXIS2=ascending-value-list
specifies a list of numeric values to assign to the second (x) variable in the grid request for the output data set.

NAXIS1=n
specifies the number of values for the first (y) variable in the grid request for the output data set.

NAXIS2=n
specifies the number of values for the second (x) variable in the grid request for the output data set.

Interpolation options

JOIN
uses a linear interpolation within a set of triangular regions that are formed from the input data set.

NEAR=n
specifies the number of the nearest data points to use for computing the estimates of the first derivative, and the second derivative.

NOSCALE
specifies that the x and y variables not be scaled to the same range before interpolation.

PARTIAL
specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation.

SCALE
specifies that the x and y variables be scaled to the same range before interpolation.

SMOOTH=ascending-value-list
specifies a list of numbers for smoothing parameters.

SPLINE
specifies the use of a bivariate spline to interpolate, or to form a smoothed estimate, if you also use the SMOOTH= option.

Required Argument

grid-request
specifies three or more numeric variables from the input data set as $y^*x=z(s)$. The variables are:

$y$
is one of the variables that form the horizontal (x-y) plane

$x$
is another of the variables that form the horizontal (x-y) plane
is one or more of the vertical variables for the interpolation

Although the GRID statement can specify only two horizontal variables, it can include multiple vertical variables. Separate vertical variables with blanks:

grid x*y=z w u v;

Optional Arguments

**AXIS1=ascending-value-list**

specifies a list of numeric values to assign to the first (y) variable in the grid request for the output data set. Numbers that you specify with this option determine the number of values for y, and override a value that you specify with the NAXIS1= option. The *ascending-value-list* must be arranged in ascending order. The value list can be in any of the following forms:

- \[ n <...n> \]
- \[ n \text{ TO } n <\text{BY increment}> \]
- \[ n <...n> \text{ TO } n <\text{BY increment}> <n <...n>> \]

Examples  
“Example 1: Using the Default Interpolation Method” on page 1428  
“Example 4: Spline Interpolation” on page 1435

**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:

- \[ n <...n> \]
- \[ n \text{ TO } n <\text{BY increment}> \]
- \[ n <...n> \text{ TO } n <\text{BY increment}> <n <...n>> \]

Examples  
“Example 1: Using the Default Interpolation Method” on page 1428  
“Example 4: Spline Interpolation” on page 1435

**JOIN**

uses a linear interpolation within a set of triangular regions that are formed from the input data set. This interpolation method creates values in the range of the initial values of the vertical variable, but the resulting interpolated surface might not be smooth.

**NAXIS1=n**

specifies the number of values for the first (y) variable in the grid request for the output data set. You can determine the actual values used for y by taking the minimum and maximum values of y and dividing the range into *n*− one equal sections.

A value specified with NAXIS1= is ignored if values are also specified with AXIS1=.

Default  11
NAXIS2=n
specifies the number of values for the second (x) variable in the grid request for the output data set. You can determine the actual values that are used for x by taking the minimum value and the maximum value of x, and dividing the range into n- one equal sections.

A value specified with NAXIS2= is ignored if values are also specified with AXIS2=.

Default 11

NEAR=n
specifies the number of the nearest data points to use for computing the estimates of the first derivative, and the second derivative. As NEAR= values become larger, time and computation costs increase significantly. NEAR= is ignored if you specify SPLINE. The value of n must be greater than or equal to 3.

If the number of input data points is insufficient for the number that you specify with NEAR=, a smaller number of data points is used.

Default 3

Example “Example 3: Partial Spline Interpolation” on page 1434

NOSCALE
specifies that the x and y variables not be scaled to the same range before interpolation. By default, the interpolation is performed after both variables are similarly scaled because the interpolation methods assume that the scales of x and y are comparable.

Default SCALE

PARTIAL
specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation. A bivariate spline is fit to the nearest neighbors, and is used to estimate the needed derivatives. This option produces results that are less smooth than those produced by the SPLINE option and uses fewer computer resources. However, the results produced by PARTIAL are smoother than those that are produced by the default. If you use both the PARTIAL option and the SPLINE option, the PARTIAL option is ignored.

Example “Example 3: Partial Spline Interpolation” on page 1434

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 $\lambda$ of the smoothing parameter, a function $u(x, y)$ is formed that minimizes
\[
\frac{1}{n} \sum_{j=1}^{n} (u(x_j, y_j) - z_j)^2 + \lambda \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left( \frac{\partial^2 u}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 u}{\partial x^2} \right)^2 + \left( \frac{\partial^2 u}{\partial y^2} \right)^2 \, dx \, dy
\]
where $n$ is the number of datapoints, and the pairs $(x_j, y_j)$ are the available points, with corresponding function values $z_j$ (Wahba 1990).

The higher the value of the smoothing parameter, the smoother the resulting interpolation. The lower the smoothing parameter, the closer the resulting surface is to the original data points. A smoothing parameter of 0 produces the same results as the SPLINE option without the SMOOTH= option.

This procedure repeats for each value of the smoothing parameter. The output data set that you specify in the OUT= option contains:

- the interpolated values
- the values of the grid points
- the values of the smoothing parameter in the variable _SMTH_
- a separate grid for each value of the smoothing parameter

Example  “Example 2: Spline and Smoothing Interpolations” on page 1431

SPLINE

specifies the use of a bivariate spline to interpolate, or to form a smoothed estimate, if you also use the SMOOTH= option. The SPLINE option results in the use of an order $n^3$ algorithm, where $n$ is the number of input data points. Consequently, this method can be time-consuming. If you use more than 100 input points, the procedure can use excessive time.

See

Harder and Desmarais (1972)

Meinguet (1979)

Green and Silverman (1994)

Examples

“Example 2: Spline and Smoothing Interpolations” on page 1431

“Example 4: Spline Interpolation” on page 1435

Details

**Controlling Observations in the Output Data Set**

The G3GRID procedure produces a data set with 121 observations for combinations of eleven values for each of the horizontal variables, $x$ and $y$. To create a data set with a different number of observations, use the GRID statement's NAXIS1= option, or the NAXIS2= option to specify the number of the values of $y$ or $x$, respectively. You can use the GRID statement's AXIS1= option or the AXIS2= option to specify the actual values for $y$ or $x$, respectively.

The following table shows the number of observations that will be in the output data set if you use any of these options.

If you specify multiple smoothing parameters, the number of observations in the output data set will be the number shown in the table, multiplied by the number of smoothing
values that you specify in the SMOOTH= option. If you use BY-group processing, multiply the number in the table by the number of BY groups.

<table>
<thead>
<tr>
<th>Options Specified</th>
<th>Number of Observations in Output Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>121</td>
</tr>
<tr>
<td>AXIS1=</td>
<td>(number of values for AXIS1=) * 11</td>
</tr>
<tr>
<td>AXIS2=</td>
<td>(number of values for AXIS2=) * 11</td>
</tr>
<tr>
<td>NAXIS1=</td>
<td>(value of NAXIS1=) * 11</td>
</tr>
<tr>
<td>NAXIS2=</td>
<td>(value of NAXIS2=) * 11</td>
</tr>
<tr>
<td>AXIS1=, AXIS2=</td>
<td>(number of values for AXIS1=) * (number of values for AXIS2=)</td>
</tr>
<tr>
<td>AXIS1=, NAXIS1=</td>
<td>(number of values for AXIS1=) * 11</td>
</tr>
<tr>
<td>AXIS1=, NAXIS2=</td>
<td>(number of values for AXIS1=) * (value of NAXIS2=)</td>
</tr>
<tr>
<td>AXIS2=, NAXIS1=</td>
<td>(number of values for AXIS2=) * (value of NAXIS1=)</td>
</tr>
<tr>
<td>AXIS2=, NAXIS2=</td>
<td>(number of values for AXIS2=) * 11</td>
</tr>
<tr>
<td>NAXIS1=, NAXIS2=</td>
<td>(value of NAXIS1=) * (value of NAXIS2=)</td>
</tr>
</tbody>
</table>

Depending on the shape of the original data, and the options that you specify, the output data set can contain values for the vertical (z) values that are outside of the range of the original values in the data set.

Examples: G3GRID Procedure

Example 1: Using the Default Interpolation Method

Features: G3GRID statement option OUT=
GRID statement options AXIS1= and AXIS2=

Other features: DATA step and the G3D procedure

Sample library member: GTGDEFIN

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
This example demonstrates the default interpolation method that is used by the GRID statement. The example first generates a scatter plot of random data to show the concentration of data values before processing the data set with the G3GRID procedure. The original data does not contain enough combinations of x, y, and z values to generate a surface plot with the G3D procedure or generate a contour plot with the GCONTOUR procedure.

**Output 47.1** Scatter Plot of NUMS Data Set (gtgdefin)

The example then runs the G3GRID procedure to interpolate additional x, y, and z values. Because no interpolation method is specified, the default interpolation method is used. The resulting output data set is used as input to the G3D procedure, which generates the surface plot shown in the following output.

**Output 47.2** Surface Plot Using Interpolated Data Set (gtgdefin)
Program

goptions reset=all border;

data nums;
  keep x y z;
  do i=1 to 30;
    x=10*ranuni(33)-5;
    y=10*ranuni(33)-5;
    z=sin(sqrt(x*x+y*y));
    output;
  end;
run;

title "Scatter Plot of NUMS Data Set";

proc g3d data=nums;
  scatter y*x=z;
run;
quit;

proc g3grid data=nums out=default;
  grid y*x=z /
    axis1=-5 to 5 by .5
    axis2=-5 to 5 by .5;
run;
quit;

title "Surface Plot using Interpolated Data Set";

proc g3d data=default;
  plot y*x=z;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Create data set. NUMS uses a set of randomly sampled points to create the data used in this, and all remaining examples in this chapter.

data nums;
  keep x y z;
  do i=1 to 30;
    x=10*ranuni(33)-5;
    y=10*ranuni(33)-5;
    z=sin(sqrt(x*x+y*y));
    output;
  end;
run;

Define the title for the plot.

title "Scatter Plot of NUMS Data Set";

Generate the scatter plot with PROC G3D.
Grid the data with PROC G3GRID. The OUT= option on PROC G3GRID specifies a name for the temporary output data set. The GRID option specifies the variables Y*X=Z for the output data set. The AXIS statements define axes ranges.

```
proc g3grid data=nums out=default;
  grid y*x=z /
    axis1=-5 to 5 by .5
    axis2=-5 to 5 by .5;
run;
quit;
```

Define the title for the plot.

```
title "Surface Plot using Interpolated Data Set";
```

Generate the surface plot. The G3D procedure using the G3GRID procedure's output data set as the input data set.

```
proc g3d data=default;
  plot y*x=z;
run;
quit;
```

---

**Example 2: Spline and Smoothing Interpolations**

**Features:** GRID statement options SMOOTH= and SPLINE

**Data set:** NUMS

**Sample library member:** GTGSISS

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

This example extends “Example 1: Using the Default Interpolation Method” on page 1428 to specify the SPLINE option in the GRID statement. The output data set, when used in PROC G3D, generates a smoother surface plot.
Output 47.3  Surface Plot Using Spline Interpolation (gtgsiss)

The following plot extends “Example 1: Using the Default Interpolation Method” on page 1428 to specify the SPLINE option, and the SMOOTH= option in the GRID statement. The SMOOTH= option is set to .05 for additional smoothing. The output data set, when used in PROC G3D, generates a smoother surface plot.

Output 47.4  Surface Plot Using Spline Interpolation and .05 Smoothing (gtgsiss)

Program

goptions reset=all border;

title "Surface Plot using Spline Interpolation";

proc g3grid data=nums out=spline;
   grid y*x=z / spline
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
run;

proc g3d data=spline;
   plot y*x=z ;
run;
quit;

title "Surface Plot using Spline Interpolation and .05 Smoothing";
proc g3grid data=nums out=smoothed;
  grid y*x=z / spline
    smooth=.05
    axis1=-5 to 5 by .5
    axis2=-5 to 5 by .5;
run;
quit;

proc g3d data=smoothed;
  plot y*x=z;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the title for the plot.

title "Surface Plot using Spline Interpolation";

Process points with PROC G3GRID. The SPLINE option specifies the bivariate spline method for the data set interpolation.

proc g3grid data=nums out=spline;
  grid y*x=z / spline
    axis1=-5 to 5 by .5
    axis2=-5 to 5 by .5;
run;

Generate the surface plot.

proc g3d data=spline;
  plot y*x=z;
run;
quit;

Define the title for the plot.

title "Surface Plot using Spline Interpolation and .05 Smoothing";

Process the data with PROC G3GRID. The SMOOTH=.05 option specifies the smoothing parameter to use during spline interpolation.

proc g3grid data=nums out=smoothed;
  grid y*x=z / spline
    smooth=.05
    axis1=-5 to 5 by .5
    axis2=-5 to 5 by .5;
Example 3: Partial Spline Interpolation

**Features:** GRID statement options NEAR and PARTIAL

**Data set:** NUMS

**Sample library member:** GTGPART

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

This example specifies a partial spline interpolation in the GRID statement, using the eight nearest neighbors for computing the estimates of the first, and second derivatives. The output data set, when used in PROC G3D, generates a more smooth surface plot than the surface plot that results from the default interpolation shown in “Example 1: Using the Default Interpolation Method” on page 1428, but does not generate the smoothness of the surface plot that results from the spline interpolation shown in “Example 2: Spline and Smoothing Interpolations” on page 1431.

**Output 47.5  Surface Plot Using Partial Spline Interpolation (gtgpart)**

![Surface Plot using Partial Spline Interpolation](image)

**Program**

```sas
options reset=all border;
proc g3grid data=nums out=partial;
    grid y*x=z / partial
```
near=8
axis1=-5 to 5 by .5
axis2=-5 to 5 by .5;
run;
title "Surface Plot using Partial Spline Interpolation";
proc g3d data=partial;
   plot y*x=z;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Process data with PROC G3GRID. The PARTIAL option specifies that a spline be used to estimate the derivatives for the biquintic polynomial interpolation. The NEAR= option specifies the number of nearest neighbors to be used for computing the estimates of the first, and the second derivatives.

proc g3grid data=nums out=partial;
   grid y*x=z / partial
       near=8
       axis1=-5 to 5 by .5
       axis2=-5 to 5 by .5;
run;

Define title for the plot.

title "Surface Plot using Partial Spline Interpolation";

Generate the surface plot.

proc g3d data=partial;
   plot y*x=z;
run;
quit;

Example 4: Spline Interpolation

Features: GRID statement options AXIS1=, AXIS2=, and SPLINE
Data set: NUMS
Sample library member: GTGSPLIN

Note: The SAS Sample Library is not available in SAS Studio. If you are using SAS Studio, you can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.
This example demonstrates the default interpolation method when used by the GCONTOUR procedure to generate a contour plot from the resulting output data set.

**Output 47.6** Contour Plot Using Default Interpolation (gtgspin)

![Contour Plot using Default Interpolation](image1)

The second plot demonstrates the spline interpolation method when used by the GCONTOUR procedure to generate a contour plot from the resulting output data set.

**Output 47.7** Contour Plot Using Spline Interpolation (gtgspin)

![Contour Plot using Spline Interpolation](image2)

**Program**

```plaintext
   goptions reset=all border;
   title "Contour Plot using Default Interpolation";
```
axis1 width=3;
proc g3grid data=nums out=numdef;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
run;
proc gcontour data=numdef;
   plot y*x=z /
      haxis=axis1
      vaxis=axis1;
run;
quit;
title "Contour Plot using Spline Interpolation";
proc g3grid data=nums out=numspl;
   grid y*x=z / spline
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
run;
proc gcontour data=numspl;
   plot y*x=z /
      haxis=axis1
      vaxis=axis1;
run;
quit;

Program Description

Set the graphics environment.

goptions reset=all border;

Define the title for the plot.

title "Contour Plot using Default Interpolation";

Define the axis characteristics.

axis1 width=3;

Process data with PROC G3GRID.

proc g3grid data=nums out=numdef;
   grid y*x=z /
      axis1=-5 to 5 by .5
      axis2=-5 to 5 by .5;
run;

Generate the contour after default interpolation.

proc gcontour data=numdef;
   plot y*x=z /
      haxis=axis1
run;

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**


Appendixes

Appendix 1
Summary of ActiveX and Java Support ................................... 1441

Appendix 2
Using SAS/GRAPH Fonts ......................................................... 1493

Appendix 3
Using Device-Resident Fonts .................................................... 1505

Appendix 4
Predefined Colors ................................................................. 1511

Appendix 5
Color Utility Macro Dictionary ................................................. 1525

Appendix 6
GREPLAY Procedure Template Reference ................................. 1533

Appendix 7
Differences between Base SAS ODS Graphics and SAS/GRAPH 1545

Appendix 8
Understanding Hexadecimal Values ................................. 1547

Appendix 9
Transporting and Converting Graphics Output .................. 1551
### Appendix 1

**Summary of ActiveX and Java Support**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1442</td>
</tr>
<tr>
<td>SAS/GRAPH Statements</td>
<td>1442</td>
</tr>
<tr>
<td>AXIS Statement</td>
<td>1442</td>
</tr>
<tr>
<td>AXIS Statement Text Description Suboptions</td>
<td>1443</td>
</tr>
<tr>
<td>AXIS Statement Tick Mark Description Suboptions</td>
<td>1443</td>
</tr>
<tr>
<td>GOPTIONS Statement</td>
<td>1444</td>
</tr>
<tr>
<td>LEGEND Statement</td>
<td>1449</td>
</tr>
<tr>
<td>LEGEND Statement Text Description Suboptions</td>
<td>1450</td>
</tr>
<tr>
<td>PATTERN Statement</td>
<td>1450</td>
</tr>
<tr>
<td>SYMBOL Statement</td>
<td>1451</td>
</tr>
<tr>
<td>SYMBOL Statement POINTLABEL= Label Description Suboptions</td>
<td>1452</td>
</tr>
<tr>
<td>TITLE and FOOTNOTE Statements (NOTE Statement is not supported)</td>
<td>1453</td>
</tr>
<tr>
<td>PROC GAREABAR</td>
<td>1454</td>
</tr>
<tr>
<td>PROC GBARLINE</td>
<td>1454</td>
</tr>
<tr>
<td>PROC GCHART</td>
<td>1457</td>
</tr>
<tr>
<td>PROC GCONTOUR</td>
<td>1465</td>
</tr>
<tr>
<td>PROC GMAP</td>
<td>1467</td>
</tr>
<tr>
<td>PROC GPLOT</td>
<td>1471</td>
</tr>
<tr>
<td>PROC GRADAR</td>
<td>1477</td>
</tr>
<tr>
<td>PROC GTILE</td>
<td>1481</td>
</tr>
<tr>
<td>PROC G3D</td>
<td>1482</td>
</tr>
<tr>
<td>Annotate Functions</td>
<td>1484</td>
</tr>
<tr>
<td>ARROW Function</td>
<td>1484</td>
</tr>
<tr>
<td>BAR Function</td>
<td>1485</td>
</tr>
<tr>
<td>DRAW Function</td>
<td>1485</td>
</tr>
<tr>
<td>DRAW2TXT Function</td>
<td>1486</td>
</tr>
<tr>
<td>FRAME Function</td>
<td>1486</td>
</tr>
<tr>
<td>IMAGE Function</td>
<td>1487</td>
</tr>
<tr>
<td>LABEL Function</td>
<td>1487</td>
</tr>
<tr>
<td>MOVE Function</td>
<td>1488</td>
</tr>
<tr>
<td>PIE Function</td>
<td>1488</td>
</tr>
<tr>
<td>PIECNTR Function</td>
<td>1489</td>
</tr>
<tr>
<td>PIEXY Function</td>
<td>1490</td>
</tr>
</tbody>
</table>
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 can 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.

SAS/GRAPH Statements

AXIS Statement

Table A1.1  ActiveX and Java Support for the AXIS Statement

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR= C=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INTERVAL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LABEL=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>LENGTH=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td>LOGBASE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LOGSTYLE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MAJOR=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>MINOR=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>NOBRACKETS</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOPLANE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OFFSET=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>
### AXIS Statement Text Description Suboptions

Text description suboptions are used by the LABEL=, REFLABEL=, and VALUE= options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGIN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REFLABEL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SPLIT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STYLE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VALUE=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>WIDTH=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
</tbody>
</table>

### AXIS Statement Tick Mark Description Suboptions

Tick mark description suboptions are used by the MAJOR= and MINOR= options to change the color, height, width, and number of the tick marks to which they apply.
**Table A1.3** ActiveX and Java Support for Tick Mark Description Suboptions

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR= C=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEIGHT= H=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NUMBER= N=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WIDTH= W=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>

**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 101 for more information.

**Table A1.4** ActiveX and Java Support for the GOPTIONS Statement

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESSIBLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ASPECT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AUTOCOPY NOAUTOCOPY</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AUTOFEED NOAUTOFEED</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AUTOSIZE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BINDING=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BORDER</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CBACK=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CBY=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CELL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHARACTERS NOCHARCTERS</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CHARTYPE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CIRCLEARC</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOCIRCLEARC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLLATE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOCOLLATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLORS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CPATTERN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CS Symbol</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CPATTERN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CS Symbol</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>CTITLE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DASH NODASH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DASHSCALE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DELAY=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DEV ADDR=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DEVICE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DEVMA P=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DISPLAY NOD</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DISPOSAL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DRV INIT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DRV TERM=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D UPLEX NOD</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ERASE NOERASE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EXTENSION</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FAST TEXT NO</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FB Y=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FC ACHE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FILE CLOS E</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FILEONLY NO</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FILEONLY</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>FILL NOFILL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FILLINC=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FONTRES=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FTEXT=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>FTITLE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FTRACK=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GACCESS=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GCOPIES=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GEND=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GEPILLOG=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GOUTMODE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GPROLOG=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GPROTOCOL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GRAPHRC NOGRAPHRC</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSFLEN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSFMODE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSFNAME=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSFPROMPT NOGSFPROMPT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSIZE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GSTART=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GUNIT=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>GWAIT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HANDSHAKE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HBY=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HORIGIN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>HPOS=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HSIZE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>HTEXT=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>HTITLE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IBACK=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>IMAGEPRINT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOIMAGEPRINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAGESTYLE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>INTERLACED</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOINTERLACED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERPOL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ITERATION=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>KEYMAP=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LFACTOR=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFFSHADOW=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERDEST=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERFEED=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERLIMIT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERSIZE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERSOURCE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PAPERTYPE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PCLIP NOPCLIP</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PENMOUNTS=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PENSORT NOPENSORT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PIEFILL NOPIEFILL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POLYGONCLIP</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOPOLYGONCLIP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Summary of ActiveX and Java Support

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLYGONFILL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NOPOLYGONFILL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POSTGEPILOG</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POSTGPROLOG</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POSTGRAPH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PPDFILE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PREGEPILOG</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PREGPROLOG</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PREGRAPH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PROMPT NOPROMPT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PROMPTCHARS</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RENDER</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RENDERLIB</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REPAINT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REVERSE NOREVERSE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROTATE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROTATE NORotate</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SIMFONT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SPEED</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SWAP NOSWAP</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SWFONTRENDER</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SYMBOL NOSYMBOL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TARGETDEVICE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TRANSPARENCY NOTRANCAPRENCY</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
</tbody>
</table>
### Option Supported by ActiveX? Supported by Java?

<table>
<thead>
<tr>
<th>Option</th>
<th>ActiveX</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANTAB=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>UCC=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>USERINPUT NOUSERINPUT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VORIGIN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VPOS=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VSIZE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>V6COMP NOV6COMP</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>XMAX=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XPIXELS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>YMAX=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>YPIXELS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>

#### LEGEND Statement

**Table A1.5** *ActiveX and Java Support for the LEGEND Statement*

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACROSS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CBLOCK=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CBORDER=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CFRAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CSHADOW=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DOWN=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FRAME</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FWIDTH=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LABEL=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>MODE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OFFSET=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ORDER=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ORIGIN=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>POSITION=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>REPEAT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SHAPE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VALUE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR= C=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FONT= F=</td>
<td>Yes (partial)</td>
<td>Yes</td>
</tr>
<tr>
<td>HEIGHT= H=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JUSTIFY= J=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POSITION=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td>TICK= T=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**PATTERN Statement**

Table A1.7  ActiveX and Java Support for the PATTERN Statement

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR= C=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>
### SYMBOL Statement

**Table A1.8  ActiveX and Java Support for the SYMBOL Statement**

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>IMAGESTYLE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>REPEAT=R=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>VALUE=bar/block-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>V=bar/block-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>VALUE=map/plot-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>V=map/plot-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>VALUE=pie/star-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>V=pie/star-pattern</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>BWIDTH=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CI=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CO=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COLOR=C=</td>
<td>Yes (GAPLOT and GBARLINE) No (GCONTOUR)</td>
<td>Yes (GAPLOT and GBARLINE) No (GCONTOUR)</td>
</tr>
<tr>
<td>CV=</td>
<td>Yes (GAPLOT) No (GCONTOUR)</td>
<td>Yes (GAPLOT) No (GCONTOUR)</td>
</tr>
<tr>
<td>FONT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HEIGHT=H=</td>
<td>Yes (GAPLOT) No (GCONTOUR)</td>
<td>Yes (GAPLOT) No (GCONTOUR)</td>
</tr>
<tr>
<td>INTERPOL=BOX=I=BOX</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>INTERPOL=HILO=I=HILO</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>INTERPOL=JOIN=I=JOIN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INTERPOL=L=I=L</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### SYMBOL Statement POINTLABEL= Label Description Suboptions

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COLOR= C=</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>FONT= F=</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>HEIGHT= H=</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### TITLE and FOOTNOTE Statements (NOTE Statement is not supported)

**Table A1.10**  
ActiveX and Java Support for TITLE and FOOTNOTE Statements

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BCOLOR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BLANK=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BOX=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BSPACE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COLOR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DRAW=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FONT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEIGHT=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>JUSTIFY=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LANGLE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LINK=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LSPACE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MOVE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROTATE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>UNDERLIN=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>
### PROC GAREABAR

**Table A1.11** ActiveX and Java Support for GAREABAR

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GAREABAR</td>
<td>DATA=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HBAR and VBAR</td>
<td>CFRFRAME=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>DISCRETE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>FRAME NOFRAME</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>RSTAT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>RESPSTAT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>RESPONSESTAT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SUBGROUP=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SUMVAR=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>WSTAT= WIDTHSTAT=</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### PROC GBARLINE

**Table A1.12** ActiveX and Java Support for PROC GBARLINE

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GBARLINE</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>IMAGEMAP=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>BAR</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCENDING</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AUTOREF</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AXIS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CAUTOREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CAXIS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CERROR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFNAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFREQ</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CLIPREF</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CLM=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>COUTLINE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CPERCENT CPCT</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTEXT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCENDING</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DES=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DISCRETE</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ERRORBAR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME FR NOFR</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FREQ</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FREQ=numeric-variable</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FRONTREF</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INSIDE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAUTOREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LEGEND</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LEVELS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LREF= LR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MAXIS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MEAN</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=value-list</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=OLD</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MINOR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MISSING</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOAXIS</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOBASEREF</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOZERO</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OUTSIDE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PATTERNID=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PERCENT PCT</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RAXIS= AXIS=</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>REF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SPACE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
### PROC GCHART

**Statement Options**

*Table A1.13  ActiveX and Java Support for PROC GCHART*

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GCHART</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Summary of ActiveX and Java Support

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GOUT=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IMAGEMAP=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BLOCK</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCKMAX=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CAXIS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>COUTLINE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DES=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FREQ=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>G100</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GROUP=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value-list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=OLD</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOHEADING</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td>PATTERNID=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SUBGROUP=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SUMVAR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TYPE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>WOUTLINE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>HBAR, HBAR3D, VBAR, and VBAR3D</td>
<td>ANNOTATE=</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ASCENDING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AUTOREF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AXIS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CAUTOREF=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CAXIS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CFRAME= CFR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CERROR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CFREQ</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CFREQLABEL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CLIPREF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CLM=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CPERCENT CPCT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CPERCENTLABEL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CREF=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCENDING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DESCRIPTION= DES=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ERRORBAR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME FR NOFR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FREQ</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FREQLABEL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FREQ=numeric-variable</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRONTRREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>G100</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>GROUP=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GSPACE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IFRAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IMAGESTYLE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INSIDE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LAUTOREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LEGEND=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LREF= LR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>MEAN</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>MEANLABEL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=</td>
<td>_value-list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=</td>
<td>OLD</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MINOR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOAXIS</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOBASEREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOSTATS</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OUTSIDE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PATTERNID=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PERCENT</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PCT</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PERCENTLABEL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PERCENTSUM</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PCTSUM</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>RAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>AXIS=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SHAPE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SPACE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SUBGROUP=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>SUMLABEL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SUMVAR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TYPE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>WIDTH=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>WOUTLINE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PIE, PIE3D, and DONUT ACROSS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ANGLE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ANNOTATE= ANNO=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ASCENDING</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CFILL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CLOCKWISE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>COUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCENDING</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION= DES=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL_PERCENT</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL_RADIUS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL_SLICE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL_THRESHOLD=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DETAIL_VALUE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DONUTPCT=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DOWN=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>EXPLODE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FILL=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>FREQ=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GROUP=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INVISIBLE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>JSTYLE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LABEL=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LABEL=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>NPERCENT=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MATCHCOLOR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=value-list</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MIDPOINTS=OLD</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOGROUPHEADIN G</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOHEADING</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OTHER=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
### DONUT Statement 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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE= A=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COLOR= C=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FONT= F=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>HEIGHT= H=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JUSTIFY= J=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ROTATE= R=</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
## PROC GCONTOUR

Table A1.15  *ActiveX and Java Support for PROC GCONTOUR*

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GCONTOUR</td>
<td>ANNOTATE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>DATA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GOUT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>INCOMPLETE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PLOT</td>
<td>ANNOTATE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>AUTOHREF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>AUTOLABEL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>AUTOVREF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CAUTOHREF=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CAUTOVREF=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CAXIS=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>CFRAME= CFR=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CHERF= CH=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CLEVELS=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CVREF= CV=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DES=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GRID</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>HAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
</tbody>
</table>
## Appendix 1 • Summary of ActiveX and Java Support

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMINOR= HM=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HREVERSE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>JOIN</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LAUTOHREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAUTOVREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LHREF= LH=</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LLEVELS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LVREF= LV=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NLEVELS=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOAXIS NOAXES</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOFRAME</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PATTERN</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VMINOR= VM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREF=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>VREVERSE</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>XTICKNUM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>YTICKNUM=</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## PROC GMAP

Table A1.16  ActiveX and Java Support for PROC GMAP

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GMAP</td>
<td>MAP=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DATA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GOUT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>IMAGEMAP=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>STRETCH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UNIFORM</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>AREA</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>MIDPOINTS=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>PERCENT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RANGE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STATFMT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STATISTIC=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>UNIFORM</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>BLOCK</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AREA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>BLOCKSIZE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CBLKOUT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CDEFAULT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CEMPTY=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>MIDPOINTS=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>PERCENT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RANGE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RELZERO</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SHAPE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STATISTIC=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STRETCH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>UNIFORM</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>WOUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>XSIZE= YSIZE=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>XVIEW= YVIEW= ZVIEW=</td>
<td>Yes</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>CHORO</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CDEFAULT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CEMPTY=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>MIDPOINTS=</td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>OSM</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>PERCENT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RANGE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STATFMT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>STATISTIC=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>STRETCH</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UNIFORM</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>WOUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>XSIZE= YSIZE=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PRISM</td>
<td>Annotate=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AREA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CDEFAULT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CEMPTY=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>COUTLINE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DISCRETE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>LEGEND=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>LEVELS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>MIDPOINTS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>MISSING</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>PERCENT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RANGE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>RELZERO</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>STATFMT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GPLOT</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GOUT=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IMAGEMAP=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UNIFORM</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BUBBLE</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOHREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>AUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BCOLOR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BFILL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BFONT=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BLABEL</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BSCALE=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>BSIZE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>CAUTOHREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAUTOVREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAXIS= CA=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CFRAME= CFR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CHREF= CH=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CTEXT= C=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CVREF= CV=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION= DES=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME FR NOFR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HMINOR= HM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HREVERSE</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IFRAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>IMAGESTYLE=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LAUTOHREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LAUTOVREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LHREF= LH=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LVREF= LV=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOAXIS NOAXES</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VMINOR= VM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREVERSE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>BUBBLE2</td>
<td>ANNOTATE= ANNO=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>AUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>BCOLOR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>BFILL=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>BFONT=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>BLABEL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>BSCALE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>BSIZE=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td>CAUTOVREF=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CAXIS= CA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CFRAME= CFR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CTEXT = C</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CVREF = CV</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FR NOFR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HAXIS</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HREVERSE</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IFRAME</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LVREF = LV</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOAXIS NOAXES</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
<td></td>
</tr>
<tr>
<td>VAXIS</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VMINOR = VM</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREVERSE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PLOT</td>
<td>ANNOTATE</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AREAS</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>AUTOHREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>AUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAUTOHREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAXIS = CA</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CBASELINE= CB=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFRAME= CFR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CHREF= CH=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>COUTLINE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTEXT= C=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CVREF= CV=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION= DES=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME FR NOFR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HMINOR= HM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HREVERSE</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HTML=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>IFRAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IMAGESTYLE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAUTOHREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LAUTOVREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LEGEND=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LHREF= LH=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LVREF= LV=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>NOAXIS NOAXES</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OVERLAY</td>
<td>Yes</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>REGEQN</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SKIPMISS</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>URL=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VAXIS=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VMINOR= VM=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREVERSE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>PLOT2</td>
<td>With INTERPOL= BOX, HILO, or STD</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ANNOTATE= ANNO=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>AREAS=</td>
<td>Yes</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>AUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAUTOVREF=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CAXIS= CA=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CBASELINE= CB=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFRAME= CFR=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>COUTLINE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTEXT= C=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CVREF= CV=</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>FRAME NOFRAME FR NOFR</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>GRID</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>Yes (partial)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LAUTOVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LVREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOAXIS NOAXES</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OVERLAY</td>
<td>Yes</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>REGEQN</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SKIPMISS</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
<td></td>
</tr>
<tr>
<td>VAXIS</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td>VMINOR VM</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VREVERSE</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>VZERO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**PROC GRADAR**

*Table A1.18  ActiveX and Java Support for PROC GRADAR*

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GRADAR</td>
<td>ANNOTATE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DATA=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GOUT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CHART</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACROSS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ACROSSVAR=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANNOTATE=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ANNO=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALENDAR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CAXIS=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CAXES=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFRAME=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFR=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFRAMESIDE=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CFRAMETOP=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CSPOKES=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CSPOKE=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTARCIRCLES=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CSTARCIRCLE=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTARFILL=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CSTARS=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CSTARS=</td>
<td>CSTAR=</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTEXT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTILES=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CTILE=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DES=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOWN=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DOWNVAR=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FONT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>FREQ=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HEIGHT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HLABEL=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>HTML_LEGEND</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IFRAME</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IMAGESTYLE</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INBORDER</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INHEIGHT</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>INTERTILE=INTERCHART=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LSPOKES</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LSTARCIRCLES=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>LSTARS=LSTAR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MAXNVERT=MAXVERT=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MISSING</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NCOLS=NCOL=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NLEVELS</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOLEGEND</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NOZEROREF</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NROWS=NROW=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ORDERACROSS=</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OTHER=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>OVERLAY=OVERLAYVAR=</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SPEED</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>SPIDERWEB SPIDER</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SPKLABEL=</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SPOKESCALE=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARAXIS=</td>
<td>STARAXES=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARCIRCLES=</td>
<td>STARCIRCLE=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARFILL=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARINRADIUS=</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARLEGEND=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARLEGENDLAB=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STAROUTRADIUS=</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>STARSTART=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STARTYPE=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SUMVAR=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TILELEGEND=</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TILELABEL=</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>URL=</td>
<td></td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td>WFRAME=</td>
<td>WAXIS=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WINDROSE</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WINDROSECIRCLES=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WSPOKES=</td>
<td>WSPoke=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WSTARCIRCLES=</td>
<td>WSTARCIRCLE=</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
## PROC GTILE

**Table A1.19**  
ActiveX and Java Support for PROC GTILE

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GTILE</td>
<td>DATA=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TILEBY</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FLOW</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>BASELINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CDEFAULT=</td>
<td>Yes (partial)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>COLORRAMP=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>COLORVAR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DETAILLEVEL=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>LABELLEVEL=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOGGLE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**PROC G3D**

Table A1.20  *ActiveX and Java Support for PROC G3D*

<table>
<thead>
<tr>
<th>Statement</th>
<th>Option</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC G3D</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GOUT=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PLOT</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAXIS=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CBOTTOM=</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CTEXT=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CTOP=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DESCRIPTION=</td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DES=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRID</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>NAME=</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NOAXIS</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NOAXES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOLABEL</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROTATE=</td>
<td></td>
<td>No</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>SIDE</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TILT=</td>
<td></td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>XAXIS</td>
<td></td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>XTICKNUM=</td>
<td></td>
<td>Yes (partial)</td>
<td>No (partial)</td>
</tr>
<tr>
<td>XYTYPE</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes (partial)</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes (partial)</td>
<td>No (partial)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td>SCATTER</td>
<td>ANNOTATE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ANNO=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CAXIS=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>COLOR=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CTEXT=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DES=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>GRID</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NAME=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOAXIS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOAXES</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NOLABEL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NONEEDLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ROTATE=</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SHAPE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SIZE=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TILT=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>X TICKNUM=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>YTICKNUM=</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>ZAXIS=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ZMAX=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ZMIN=</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ZTICKNUM=</td>
<td>Yes (partial)</td>
<td>No</td>
</tr>
<tr>
<td>Statement</td>
<td>Option</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ZMAX</td>
<td></td>
<td>Yes</td>
<td>Yes (partial)</td>
</tr>
<tr>
<td>ZMIN</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

## Annotate Functions

**ARROW Function**

*Table A1.21*  ActiveX and Java Support for the ARROW Function

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**BAR Function**

*Table A1.22  ActiveX and Java Support for the BAR Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**DRAW Function**

*Table A1.23  ActiveX and Java Support for the DRAW Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### DRAW2TXT Function

**Table A1.24** ActiveX and Java Support for the DRAW2TXT Function

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### FRAME Function

**Table A1.25** ActiveX and Java Support for the FRAME Function

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes (Partial)</td>
<td>No</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### IMAGE Function

**Table A1.26  ActiveX and Java Support for the IMAGE Function**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IMGPATH</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### LABEL Function

**Table A1.27  ActiveX and Java Support for the LABEL Function**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CBORDER</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CBOX</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POSITION</td>
<td>Yes</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>ROTATE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Variable</td>
<td>Supported by ActiveX?</td>
<td>Supported by Java?</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TEXT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**MOVE Function**

*Table A1.28  ActiveX and Java Support for the MOVE Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**PIE Function**

*Table A1.29  ActiveX and Java Support for the PIE Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### PIECNTR Function

**Table A1.30**  
ActiveX and Java Support for the PIECNTR Function

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROTATE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### PIEXY Function

**Table A1.31  ActiveX and Java Support for the PIEXY Function**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### POINT Function

**Table A1.32  ActiveX and Java Support for the POINT Function**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes (Partial)</td>
</tr>
</tbody>
</table>
**POLY Function**

**Table A1.33**  
*ActiveX and Java Support for the POLY Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LINE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**POLYCONT Function**

**Table A1.34**  
*ActiveX and Java Support for the POLYCONT Function*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### SYMBOL Function

**Table A1.35  ActiveX and Java Support for the SYMBOL Function**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported by ActiveX?</th>
<th>Supported by Java?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBOX</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CBORDER</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>COLOR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBGROUP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HTML</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MIDPOINT</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIZE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STYLE</td>
<td>Yes(Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>TEXT</td>
<td>Yes (Partial)</td>
<td>Yes (Partial)</td>
</tr>
<tr>
<td>WHEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XC, YC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XSYS, YSYS, ZSYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Introduction

SAS/GRAPH fonts are the entries in the SASHELP.FONTS catalog. Information about these fonts is provided for special purposes only. For example, some specialized devices do not support system fonts. Or, you might want to use special symbols in the Marker font to display solid symbols for data points in a plot. If you specify the NOGSTYLE system option and one of the Z device drivers (see “Compatibility Device Drivers (Z Drivers)” on page 93), 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 300 and “TrueType Fonts Supplied by SAS” on page 300 for more information.

Note: The Java and ActiveX devices do not support SAS/GRAPH fonts. Also, SAS/GRAPH fonts cannot be used with template-based graphics.

Rendering Bitstream Fonts

SAS/GRAPH includes methods of storing rendered versions of Bitstream fonts, along with three graphics options to control how the fonts are rendered.

When your graphics output uses one of the Bitstream fonts that are provided with SAS/GRAPH, SAS/GRAPH must process information contained in corresponding FONTS catalog entries to determine how to draw characters of the specified size and typeface. The process of calculating the character shapes and sizes is known as
rendering the font. Bitstream fonts that are available with SAS/GRAPH include the Century, Swiss, and Zapf font families.

SAS/GRAPH can store rendered versions of the Bitstream fonts in memory or in special SAS files. Using these rendered versions of the fonts can provide a speed improvement when characters of the same size and style are used again during the SAS session. SAS/GRAPH can read the rendered version of the characters from memory or from the rendered font file instead of performing the rendering calculations each time the characters are used. If you store the rendered fonts in files in a permanent SAS data set, SAS/GRAPH can use the rendered font files again in subsequent SAS sessions.

Note: Because the rendered font files use a special utility member type, they do not appear in the list of library members that is displayed in the DIRECTORY window.

You control whether and how rendered versions of fonts are stored using the FONTRES=, RENDER=, and RENDERLIB= graphics options. See Chapter 25, “Graphics Options and Device Parameters Dictionary,” on page 515 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:

```sas
proc catalog catalog=sashelp.fonts entrytype=font;
   contents out=work.swfonts(keep=name);
run;
quit;
data work.swfonts;
   set work.swfonts;
   if name =:'HW' then delete;
run;
proc print data=work.swfonts;
run;
```

You can display these fonts with the GFONT procedure. See “Example 1: Displaying Fonts with Character Codes” on page 1093.

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)
light (thin)

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.

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

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Font Name</th>
<th>Type Sample</th>
<th>Uniform Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brush</td>
<td>BRUSH</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Century</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Bold</td>
<td>CENTB</td>
<td>ABCabc123</td>
<td>CENTBU</td>
</tr>
<tr>
<td>• Bold Empty</td>
<td>CENTBE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>• Bold Italic</td>
<td>CENTBI</td>
<td>ABCabc123</td>
<td>CENTBIU</td>
</tr>
<tr>
<td>• Bold Italic Empty</td>
<td>CENTBIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>• Expanded</td>
<td>CENTX</td>
<td>ABCabc123</td>
<td>CENTXU</td>
</tr>
<tr>
<td>• Expanded Empty</td>
<td>CENTXE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>• Expanded Italic</td>
<td>CENTXI</td>
<td>ABCabc123</td>
<td>CENTXIU</td>
</tr>
<tr>
<td>• Expanded Italic Empty</td>
<td>CENTXIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>GERMAN</td>
<td>UBCabc123</td>
<td>GERMANU</td>
</tr>
<tr>
<td>German Italic</td>
<td>GITALIC</td>
<td>UBCabc123</td>
<td>GITALICU</td>
</tr>
<tr>
<td>Hershey</td>
<td>SIMPLEX</td>
<td>ABCabc123</td>
<td>SIMPLEXU</td>
</tr>
<tr>
<td>Typeface</td>
<td>Font Name</td>
<td>Type Sample</td>
<td>Uniform Font</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Sans Serif Bold</td>
<td>DUPLEX</td>
<td>ABCabc123</td>
<td>DUPLEXU</td>
</tr>
<tr>
<td>Serif</td>
<td>COMPLEX</td>
<td>ABCabc123</td>
<td>COMPLEXU</td>
</tr>
<tr>
<td>Serif Bold</td>
<td>TRIPLEX</td>
<td>ABCabc123</td>
<td>TRIPLEXU</td>
</tr>
<tr>
<td>Serif Bold Italic</td>
<td>TITALIC</td>
<td>ABCabc123</td>
<td>TITALICU</td>
</tr>
<tr>
<td>Serif Italic</td>
<td>ITALIC</td>
<td>ABCabc123</td>
<td>ITALICU</td>
</tr>
<tr>
<td>Old English</td>
<td>OLDENG</td>
<td>ABCabc123</td>
<td>OLDENGU</td>
</tr>
<tr>
<td>Script</td>
<td>SCRIPT</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Cscript</td>
<td>CSCRIPT</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Swiss</td>
<td>SWISS</td>
<td>ABCabc123</td>
<td>SWISSU</td>
</tr>
<tr>
<td>Empty</td>
<td>SWISSE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Bold</td>
<td>SWISSB</td>
<td>ABCabc123</td>
<td>SWISSBU</td>
</tr>
<tr>
<td>Bold Empty</td>
<td>SWISSBE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Bold Italic</td>
<td>SWISSBI</td>
<td>ABCabc123</td>
<td>SWISSBIU</td>
</tr>
<tr>
<td>Bold Italic Empty</td>
<td>SWISSBIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Expanded</td>
<td>SWISSX</td>
<td>ABCabc123</td>
<td>SWISSXU</td>
</tr>
<tr>
<td>Expanded Empty</td>
<td>SWISSXE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Expanded Bold</td>
<td>SWISSXB</td>
<td>ABCabc123</td>
<td>SWISSXBU</td>
</tr>
<tr>
<td>Expanded Bold Empty</td>
<td>SWISSXBE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Italic</td>
<td>SWISSI</td>
<td>ABCabc123</td>
<td>SWISSIU</td>
</tr>
<tr>
<td>Italic Empty</td>
<td>SWISSIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>SWISSL</td>
<td>ABCabc123</td>
<td>SWISSLU</td>
</tr>
<tr>
<td>Light Empty</td>
<td>SWISSLE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Zapf</td>
<td>ZAPF</td>
<td>ABCabc123</td>
<td>ZAPFU</td>
</tr>
<tr>
<td>Typeface</td>
<td>Font Name</td>
<td>Type Sample</td>
<td>Uniform Font</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Empty</td>
<td>ZAPFE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Bold</td>
<td>ZAPFB</td>
<td>A B C a b c 1 2 3</td>
<td>ZAPFBU</td>
</tr>
<tr>
<td>Bold Empty</td>
<td>ZAPFBE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Bold Italic</td>
<td>ZAPFBI</td>
<td>ABCabc123</td>
<td>ZAPFBIU</td>
</tr>
<tr>
<td>Bold Italic Empty</td>
<td>ZAPFBIIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
<tr>
<td>Italic</td>
<td>ZAPFI</td>
<td>ABCabc123</td>
<td>ZAPFIU</td>
</tr>
<tr>
<td>Italic Empty</td>
<td>ZAPFIE</td>
<td>ABCabc123</td>
<td></td>
</tr>
</tbody>
</table>

**Table A2.2** *Non-Roman Alphabet Fonts*

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Font Name</th>
<th>Uniform Font Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>ARABIC</td>
<td></td>
</tr>
<tr>
<td>Arabic Empty</td>
<td>ARABICE</td>
<td></td>
</tr>
<tr>
<td>Cyrillic</td>
<td>CYRILLIC</td>
<td>CYRILLIU</td>
</tr>
<tr>
<td>David</td>
<td>DAVID</td>
<td></td>
</tr>
<tr>
<td>Davidb</td>
<td>DAVIDB</td>
<td></td>
</tr>
<tr>
<td>Fsong</td>
<td>FSONG</td>
<td>FSONGU</td>
</tr>
<tr>
<td>Greek</td>
<td>GREEK</td>
<td>GREEKU</td>
</tr>
<tr>
<td>Greek (serif)</td>
<td>CGREEK</td>
<td>CGREEKU</td>
</tr>
<tr>
<td>Hebrew</td>
<td>HEBREW</td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>NHEBREW*</td>
<td></td>
</tr>
<tr>
<td>Hebrewb</td>
<td>HEBREWB</td>
<td></td>
</tr>
<tr>
<td>Hebrew Empty</td>
<td>HEBREWE</td>
<td></td>
</tr>
<tr>
<td>Hei</td>
<td>HEI</td>
<td>HEIU</td>
</tr>
<tr>
<td>Hiragana</td>
<td>HIRA</td>
<td></td>
</tr>
<tr>
<td>Typeface</td>
<td>Font Name</td>
<td>Uniform Font Name</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Hiragana</td>
<td>NHIRA*</td>
<td></td>
</tr>
<tr>
<td>Kanji</td>
<td>KANJI</td>
<td></td>
</tr>
<tr>
<td>Kanji</td>
<td>KANSJIS</td>
<td></td>
</tr>
<tr>
<td>Kanji Subset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kanji 1</td>
<td>KAN1</td>
<td></td>
</tr>
<tr>
<td>Kanji 2</td>
<td>KAN2</td>
<td></td>
</tr>
<tr>
<td>Kanji 3</td>
<td>KAN3</td>
<td></td>
</tr>
<tr>
<td>Kanji 4</td>
<td>KAN4</td>
<td></td>
</tr>
<tr>
<td>Kanji 5</td>
<td>KAN5</td>
<td></td>
</tr>
<tr>
<td>Kanji 6</td>
<td>KAN6</td>
<td></td>
</tr>
<tr>
<td>Kanji 7</td>
<td>KAN7</td>
<td></td>
</tr>
<tr>
<td>Kanji 8</td>
<td>KAN8</td>
<td></td>
</tr>
<tr>
<td>Katakana</td>
<td>KATA</td>
<td></td>
</tr>
<tr>
<td>Katakana</td>
<td>NKATA*</td>
<td></td>
</tr>
<tr>
<td>Korean</td>
<td>KGOTHB1</td>
<td></td>
</tr>
<tr>
<td>Mincho</td>
<td>MINCHO</td>
<td>MINCHOE</td>
</tr>
</tbody>
</table>

*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)

\[ \text{\LaTeX}\]

<table>
<thead>
<tr>
<th>!</th>
<th>&quot;</th>
<th>$</th>
<th>%</th>
<th>&amp;</th>
<th>'</th>
<th>(</th>
<th>)</th>
<th>*</th>
<th>+</th>
<th>,</th>
<th>-</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
</tr>
<tr>
<td>/</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>&lt;</td>
</tr>
<tr>
<td>ç</td>
<td>?</td>
<td>œ</td>
<td>À</td>
<td>Á</td>
<td>Â</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
<td>Æ</td>
</tr>
<tr>
<td>&gt;</td>
<td>?</td>
<td>@</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
<td>À</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>ε</td>
<td>ψ</td>
<td>γ</td>
<td>η</td>
<td>ι</td>
<td>κ</td>
<td>λ</td>
<td>μ</td>
<td>ν</td>
<td>ο</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
</tr>
<tr>
<td>π</td>
<td>θ</td>
<td>ρ</td>
<td>σ</td>
<td>τ</td>
<td>υ</td>
<td>ψ</td>
<td>χ</td>
<td>ψ</td>
<td>ζ</td>
<td>η</td>
<td>ι</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td>}</td>
</tr>
</tbody>
</table>
**Table A2.3** Symbol Fonts

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Font Name</th>
<th>Uniform Font Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartographic</td>
<td>CARTOG</td>
<td>CARTOGU</td>
</tr>
<tr>
<td>Electronic</td>
<td>ELECTRO</td>
<td>ELECTROU</td>
</tr>
<tr>
<td>Marker</td>
<td>MARKER</td>
<td></td>
</tr>
<tr>
<td>Marker</td>
<td>MARKERE</td>
<td></td>
</tr>
<tr>
<td>Empty</td>
<td>MARKERE</td>
<td>*</td>
</tr>
<tr>
<td>Math</td>
<td>MATH</td>
<td>MATHU</td>
</tr>
<tr>
<td>Music</td>
<td>MUSIC</td>
<td>MUSICU</td>
</tr>
<tr>
<td>Special</td>
<td>SPECIAL</td>
<td>SPECIALU</td>
</tr>
<tr>
<td>Weather</td>
<td>WEATHER</td>
<td>WEATHERU</td>
</tr>
</tbody>
</table>

*MARKERE is not displayed in the figures.
**Figure A2.3** Cartographic Font

![Cartographic Font](image)

**Figure A2.4** Electronic Font

![Electronic Font](image)

*Note:* Figure A2.5 on page 1501 shows the MARKER font. The MARKERE font produces the same symbols but in empty (outline) form.

**Figure A2.5** Marker Font

![Marker Font](image)

**Figure A2.6** Math Font

![Math Font](image)
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=HWxxnnn 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 611 for more information about the SIMFONT= option.

Font Locations and the Default Search Path

SAS/GRAPH fonts are stored in catalogs. SAS/GRAPH looks only into catalogs with certain librefs and names to find fonts. By default, SAS/GRAPH searches for the font in the catalog SASHELP.FONTS, which contains SAS/GRAPH fonts, key maps, and device maps.
If you want to specify fonts that you have created locally, submit a LIBNAME statement that associates the libref GFONT0 with the location of your font catalog. If you have specified more than one libref in the sequence GFONT0 through GFONT9, SAS/GRAPH performs a sequential search of these catalogs when locating the font that you have specified.

When you specify a font name, SAS/GRAPH searches for the font in the following order:

1. If a SAS library with the libref GFONT0 exists, then SAS/GRAPH looks there for a catalog named FONTS. If GFONT0.FONTS exists, it is checked for the specified font. If the font is not there, then SAS/GRAPH looks next for a library with the libref GFONT1 and for a catalog named FONTS in that library. The search is repeated for the sequence of librefs through GFONT9.

2. SAS/GRAPH searches for the font in SASHELP.FONTS if the following situations occur.
   a. It fails to find the specified font in any FONTS catalog in the libraries GFONT0 to GFONT9.
   b. It finds a GFONTn 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 1503 for more information.

See Chapter 38, “GFONT Procedure,” on page 1067 for additional information about 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 1506
for details.

- by using the GDEVICE procedure to specify the number of the font that you want to
use as the default device-resident font. See “Using the GDEVICE Procedure to
Change the Default Device-Resident Font” on page 1507 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 1507 for details.

- by explicitly specifying a device-resident font name of the type HWxxxxnnn. See
“Specifying Alternative Device-Resident Fonts” on page 1508 for details.

There are several advantages to using device-resident fonts instead of SAS/GRAPH
fonts. Device-resident fonts often are produced faster than SAS/GRAPH fonts and
produce smaller output files. Also, some devices, such as laser printers with device-
resident fonts, might produce better quality output with device-resident fonts than with
SAS/GRAPH fonts.
Default Device-Resident Fonts

When Are Device-Resident Fonts Used?

SAS/GRAPH uses a device's default device-resident font to draw characters when both of the following conditions are true:

- No font specification is made in the SAS/GRAPH program, or FONT=NONE is specified.
- The device-resident font can be used. See “Default Fonts” on page 301 for details about when device-resident fonts cannot be used.

The Chartype Value and Default Device-Resident Fonts

Every available device-resident font for a particular device has a number associated with it. This number and the corresponding font name are listed in the Chartype window of the device entry for your device. The default device-resident font is the font whose number is entered in the Chartype field in the Parameters window of the device entry. When FONT=NONE or no font is specified, SAS/GRAPH uses the font assigned to this field.

If your device has more than one device-resident font, you can assign a different default device-resident font in two ways:

- by specifying the font with the CHARTYPE= option in a GOPTIONS statement. See “Using a GOPTIONS Statement to Change the Default Device-Resident Font” on page 1506.
- 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 1507 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 530 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 37, “GDEVICE Procedure,” on page 1046 for information about viewing device entries and changing device parameters.)

Specifying the Full Font Name

You can specify the full font name in any SAS statement where a font specification is valid. For example, you can specify the full font name in the FTEXT=font graphics option or the FONT=font specification in a TITLE statement. For the value font, specify the full font name exactly as it appears in the Chartype window of the device entry. For example, to specify the Times-Roman font in a TITLE statement when you use the PS300 device, use this code:

title font="Times-Roman" "Testing the Times-Roman font";

SAS allows up to 255 characters for the font name. The font name might contain spaces. If the font name is longer than 40 characters, PROC GDEVICE in fullscreen mode displays only the first 37 characters, followed by an ellipsis (...). To see the complete font name when the name is longer than 40 characters, use PROC GDEVICE with the NOFS (no fullscreen) option as follows:

```
proc gdevice c=sashelp.devices nofs;
   list driver-name;
run;
quit;
```
When a font is quoted, SAS first looks at the Chartype window of the device driver entry to determine whether it is a valid device-resident font. If the font is not found in the Chartype window, SAS then checks to determine whether the quoted font is a valid SAS/GRAPH font. If the font is not recognized as either a valid device-resident font or a valid SAS/GRAPH font, the SIMULATE font is used.

Specifying Alternative Device-Resident Fonts

An alternative device-resident font can be specified in any SAS statement where a font specification is valid. You can use more than one device-resident font in a single graph or even in a single statement. All of the fonts that you specify must exist on your device. If you specify a device-resident font, make sure that the font is available on the device and that there is a corresponding Chartype value for the font. If you request a device-resident font that does not have a Chartype defined, SAS/GRAPH substitutes the SIMULATE font.

These are the three ways to specify alternative device-resident fonts:

• In the font specification, explicitly assign a device-resident font using the following form:

  \texttt{HW}\_xxx\_nnn

  \texttt{HW} identifies the font as a device-resident font. The font name must begin with the characters HW.

  \texttt{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).

  \texttt{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:

  \texttt{device-resident-font-name}

  identifies the name of the device-resident font that is listed in the Chartype window of the device entry. \texttt{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 \texttt{HW}\_xxx\_nnn name. Note that in Annotate, the specified font name must be enclosed in both double quotation marks and single quotation marks. See “The Annotate Function, Variable, and Macro Dictionaries” on page 638 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 \texttt{FONT=NONE} in your SAS/GRAPH procedure or statement to specify the new default device-resident font.
When you specify FONT=HWxxnn 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 25, “Graphics Options and Device Parameters Dictionary,” on page 515 for additional information.
Appendix 4
Predefined Colors

The following table shows the colors that are predefined in the SAS Registry, arranged by name. For each color, the equivalent CX color code (RGB), the equivalent HLS color code, and a color sample are shown. For a table of the SAS registry colors arranged by hue, see Table A4.2 on page 1518.

Table A4.1 SAS Registry Colors by Name

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Sample</th>
<th>CX Color Code (RGB)</th>
<th>HLS Color Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AliceBlue</td>
<td></td>
<td>CXF0F8FF</td>
<td>H148F8FF</td>
</tr>
<tr>
<td>AntiqueWhite</td>
<td></td>
<td>CXFAEBD7</td>
<td>H09AE9C6</td>
</tr>
<tr>
<td>Aqua</td>
<td></td>
<td>CX00FFFF</td>
<td>H12C80FF</td>
</tr>
<tr>
<td>Aquamarine</td>
<td></td>
<td>CX7FFDD4</td>
<td>H118BEF7</td>
</tr>
<tr>
<td>Azure</td>
<td></td>
<td>CXF0FFFF</td>
<td>H12CF8FF</td>
</tr>
<tr>
<td>Beige</td>
<td></td>
<td>CXF5F5DC</td>
<td>H0B4E98E</td>
</tr>
<tr>
<td>Bisque</td>
<td></td>
<td>CXFFE4C4</td>
<td>H098E2FF</td>
</tr>
<tr>
<td>Black</td>
<td></td>
<td>CX000000</td>
<td>H0000000</td>
</tr>
<tr>
<td>BlanchedAlmond</td>
<td></td>
<td>CXFFEBCD</td>
<td>H09CE6FF</td>
</tr>
<tr>
<td>Blue</td>
<td></td>
<td>CX0000FF</td>
<td>H00080FF</td>
</tr>
<tr>
<td>BlueViolet</td>
<td></td>
<td>CX8A2BE2</td>
<td>H01F87C2</td>
</tr>
<tr>
<td>Brown</td>
<td></td>
<td>CXA52A2A</td>
<td>H0786898</td>
</tr>
<tr>
<td>Burlywood</td>
<td></td>
<td>CXDEB887</td>
<td>H099B391</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code*</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>CadetBlue</td>
<td></td>
<td>CX5F9EA0</td>
<td>H12D8041</td>
</tr>
<tr>
<td>Chartreuse</td>
<td></td>
<td>CX7FFF00</td>
<td>H0D280FF</td>
</tr>
<tr>
<td>Chocolate</td>
<td></td>
<td>CXD2691E</td>
<td>H09178BF</td>
</tr>
<tr>
<td>Coral</td>
<td></td>
<td>CXFF7F50</td>
<td>H088A8FF</td>
</tr>
<tr>
<td>CornflowerBlue</td>
<td></td>
<td>CX6495ED</td>
<td>H152A9CA</td>
</tr>
<tr>
<td>Cornsilk</td>
<td></td>
<td>CXFFF8DC</td>
<td>H0A7EEFF</td>
</tr>
<tr>
<td>Crimson</td>
<td></td>
<td>CXDC143C</td>
<td>H06C78D5</td>
</tr>
<tr>
<td>Cyan</td>
<td></td>
<td>CX00FFFF</td>
<td>H12C80FF</td>
</tr>
<tr>
<td>DarkBlue</td>
<td></td>
<td>CX00008B</td>
<td>H00046FF</td>
</tr>
<tr>
<td>DarkCyan</td>
<td></td>
<td>CX008B8B</td>
<td>H12C46FF</td>
</tr>
<tr>
<td>DarkGoldenrod</td>
<td></td>
<td>CXB8860B</td>
<td>H0A262E2</td>
</tr>
<tr>
<td>DarkGray</td>
<td></td>
<td>CXA9A9A9</td>
<td>H000A900</td>
</tr>
<tr>
<td>DarkGreen</td>
<td></td>
<td>CX006400</td>
<td>H0F032FF</td>
</tr>
<tr>
<td>DarkKhaki</td>
<td></td>
<td>CXBDB76B</td>
<td>H0AF9462</td>
</tr>
<tr>
<td>DarkMagenta</td>
<td></td>
<td>CX8B008B</td>
<td>H03C46FF</td>
</tr>
<tr>
<td>DarkOliveGreen</td>
<td></td>
<td>CX556B2F</td>
<td>H0CA4D63</td>
</tr>
<tr>
<td>DarkOrange</td>
<td></td>
<td>CXFF8C00</td>
<td>H09880FF</td>
</tr>
<tr>
<td>DarkOrchid</td>
<td></td>
<td>CX9932CC</td>
<td>H0287F9B</td>
</tr>
<tr>
<td>DarkRed</td>
<td></td>
<td>CX8B0000</td>
<td>H07846FF</td>
</tr>
<tr>
<td>DarkSalmon</td>
<td></td>
<td>CXE9967A</td>
<td>H087B2B7</td>
</tr>
<tr>
<td>DarkSeaGreen</td>
<td></td>
<td>CX8FBC8F</td>
<td>H0F0A640</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>DarkSlateBlue</td>
<td></td>
<td>CX483D8B</td>
<td>H0086463</td>
</tr>
<tr>
<td>DarkSlateGray</td>
<td></td>
<td>CX2F4F4F</td>
<td>H12C3F41</td>
</tr>
<tr>
<td>DarkSlateGrey</td>
<td></td>
<td>CX2F2F2F</td>
<td>H0002F00</td>
</tr>
<tr>
<td>DarkTurquoise</td>
<td></td>
<td>CX00CED1</td>
<td>H12C69FF</td>
</tr>
<tr>
<td>DarkViolet</td>
<td></td>
<td>CX9400D3</td>
<td>H02A6AFF</td>
</tr>
<tr>
<td>DeepPink</td>
<td></td>
<td>CXFF1493</td>
<td>H0578AFF</td>
</tr>
<tr>
<td>DeepSkyBlue</td>
<td></td>
<td>CX00BFFF</td>
<td>H13B80FF</td>
</tr>
<tr>
<td>DimGray</td>
<td></td>
<td>CX696969</td>
<td>H0006900</td>
</tr>
<tr>
<td>DodgerBlue</td>
<td></td>
<td>CX1E90FF</td>
<td>H1498FFF</td>
</tr>
<tr>
<td>Firebrick</td>
<td></td>
<td>CXB22222</td>
<td>H0786AAD</td>
</tr>
<tr>
<td>FloralWhite</td>
<td></td>
<td>CXFFF0FF</td>
<td>H0A0F8FF</td>
</tr>
<tr>
<td>ForestGreen</td>
<td></td>
<td>CX228B22</td>
<td>H0F0579B</td>
</tr>
<tr>
<td>Fuchsia</td>
<td></td>
<td>CXFF00FF</td>
<td>H03C80FF</td>
</tr>
<tr>
<td>Gainsboro</td>
<td></td>
<td>CXDCDCDC</td>
<td>H000DC00</td>
</tr>
<tr>
<td>GhostWhite</td>
<td></td>
<td>CXF8F8FF</td>
<td>H000FCFF</td>
</tr>
<tr>
<td>Gold</td>
<td></td>
<td>CXFFD700</td>
<td>H0AA80FF</td>
</tr>
<tr>
<td>Goldenrod</td>
<td></td>
<td>CXDA520</td>
<td>H0A27DBE</td>
</tr>
<tr>
<td>Gray or Grey</td>
<td></td>
<td>CX808080</td>
<td>H0008000</td>
</tr>
<tr>
<td>Green</td>
<td></td>
<td>CX008000</td>
<td>H0F040FF</td>
</tr>
<tr>
<td>GreenYellow</td>
<td></td>
<td>CXADFF2F</td>
<td>H0CB97FF</td>
</tr>
<tr>
<td>Honeydew</td>
<td></td>
<td>CXF0FF00</td>
<td>H0F0F8FF</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code*</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>HotPink</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFF69B4</td>
<td>H05AB4FF</td>
</tr>
<tr>
<td>IndianRed</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXCD5C5C</td>
<td>H0789587</td>
</tr>
<tr>
<td>Indigo</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX4B0082</td>
<td>H02241FF</td>
</tr>
<tr>
<td>Ivory</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFFFFFF0</td>
<td>H0BF4F8FF</td>
</tr>
<tr>
<td>Khaki</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXF0E68C</td>
<td>H0AEBC4</td>
</tr>
<tr>
<td>Lavender</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXE6E6FA</td>
<td>H000F0AA</td>
</tr>
<tr>
<td>LavenderBlush</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFFFF05</td>
<td>H064F8FF</td>
</tr>
<tr>
<td>LawnGreen</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX7FC00</td>
<td>H0D27EFF</td>
</tr>
<tr>
<td>LemonChiffon</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFFFFACD</td>
<td>H0AE6FF</td>
</tr>
<tr>
<td>LightBlue</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXADD8E6</td>
<td>H13ACA88</td>
</tr>
<tr>
<td>LightCoral</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXF08080</td>
<td>H078B8C9</td>
</tr>
<tr>
<td>LightCyan</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXE0FFFF</td>
<td>H12CF0FF</td>
</tr>
<tr>
<td>LightGoldenrodYellow</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFAFAD2</td>
<td>H0B4E6CC</td>
</tr>
<tr>
<td>LightGray</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXD3D3D3</td>
<td>H000D300</td>
</tr>
<tr>
<td>LightGreen</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX90E90</td>
<td>H0F0BFBB</td>
</tr>
<tr>
<td>LightPink</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFFB6C1</td>
<td>H06EDBFF</td>
</tr>
<tr>
<td>LightSalmon</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXFFA07A</td>
<td>H089BDF</td>
</tr>
<tr>
<td>LightSeaGreen</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX20B2AA</td>
<td>H12869B1</td>
</tr>
<tr>
<td>LightSkyBlue</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX87CEFA</td>
<td>H142C1EB</td>
</tr>
<tr>
<td>LightSlateGray</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CX778899</td>
<td>H14A8824</td>
</tr>
<tr>
<td>LightSteelBlue</td>
<td><img src="image" alt="Color Sample" /></td>
<td>CXB0C4DE</td>
<td>H14DC769</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code*</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>LightYellow</td>
<td></td>
<td>CXFFFFE0</td>
<td>H0B4F0FF</td>
</tr>
<tr>
<td>Lime</td>
<td></td>
<td>CX00FF00</td>
<td>H0F080FF</td>
</tr>
<tr>
<td>LimeGreen</td>
<td></td>
<td>CX32CD32</td>
<td>H0F0809B</td>
</tr>
<tr>
<td>Linen</td>
<td></td>
<td>CXFAF0E6</td>
<td>H095F0AA</td>
</tr>
<tr>
<td>Magenta</td>
<td></td>
<td>CXFF00FF</td>
<td>H0C80FF</td>
</tr>
<tr>
<td>Maroon</td>
<td></td>
<td>CX800000</td>
<td>H07840FF</td>
</tr>
<tr>
<td>MediumAquamarine</td>
<td></td>
<td>CX66CDAA</td>
<td>H1179A81</td>
</tr>
<tr>
<td>MediumBlue</td>
<td></td>
<td>CX0000CD</td>
<td>H00067FF</td>
</tr>
<tr>
<td>MediumOrchid</td>
<td></td>
<td>CXBA55D3</td>
<td>H0309496</td>
</tr>
<tr>
<td>MediumPurple</td>
<td></td>
<td>CX9370DB</td>
<td>H013A698</td>
</tr>
<tr>
<td>MediumSeaGreen</td>
<td></td>
<td>CX3CB371</td>
<td>H10A787F</td>
</tr>
<tr>
<td>MediumSlateBlue</td>
<td></td>
<td>CX7B68EE</td>
<td>H008ABCB</td>
</tr>
<tr>
<td>MediumSpringGreen</td>
<td></td>
<td>CX00FA9A</td>
<td>H1147DFF</td>
</tr>
<tr>
<td>MediumTurquoise</td>
<td></td>
<td>CX48D1CC</td>
<td>H1298D99</td>
</tr>
<tr>
<td>MediumVioletRed</td>
<td></td>
<td>CXC71585</td>
<td>H0526ECE</td>
</tr>
<tr>
<td>MidnightBlue</td>
<td></td>
<td>CX191970</td>
<td>H00045A2</td>
</tr>
<tr>
<td>MintCream</td>
<td></td>
<td>CXF5FFFA</td>
<td>H10EFAFF</td>
</tr>
<tr>
<td>MistyRose</td>
<td></td>
<td>CXFFE4E1</td>
<td>H07E0FF</td>
</tr>
<tr>
<td>Moccasin</td>
<td></td>
<td>CXFFE4B5</td>
<td>H09EDAFF</td>
</tr>
<tr>
<td>NavajoWhite</td>
<td></td>
<td>CXFFDEAD</td>
<td>H09BD6FF</td>
</tr>
<tr>
<td>Navy</td>
<td></td>
<td>CX000080</td>
<td>H00040FF</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code*</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>OldLace</td>
<td></td>
<td>CXFDF5E6</td>
<td>H09FF2D9</td>
</tr>
<tr>
<td>Olive</td>
<td></td>
<td>CX808000</td>
<td>H0B440FF</td>
</tr>
<tr>
<td>OliveDrab</td>
<td></td>
<td>CX6B8E23</td>
<td>H0C7599A</td>
</tr>
<tr>
<td>Orange</td>
<td></td>
<td>CXFFA500</td>
<td>H09E80FF</td>
</tr>
<tr>
<td>OrangeRed</td>
<td></td>
<td>CXFF4500</td>
<td>H08880FF</td>
</tr>
<tr>
<td>Orchid</td>
<td></td>
<td>CXDA70D6</td>
<td>H03EA596</td>
</tr>
<tr>
<td>PaleGoldenrod</td>
<td></td>
<td>CXEEE8AA</td>
<td>H0AECCAA</td>
</tr>
<tr>
<td>PaleGreen</td>
<td></td>
<td>CX98FB98</td>
<td>H0F0CAEC</td>
</tr>
<tr>
<td>PaleTurquoise</td>
<td></td>
<td>CXAFEEEE</td>
<td>H12CCFA6</td>
</tr>
<tr>
<td>PaleVioletRed</td>
<td></td>
<td>CXDB7093</td>
<td>H064A698</td>
</tr>
<tr>
<td>PapayaWhip</td>
<td></td>
<td>CXFFEFD5</td>
<td>H09DEAFFF</td>
</tr>
<tr>
<td>Peachpuff</td>
<td></td>
<td>CXFFDAB9</td>
<td>H094DCFFF</td>
</tr>
<tr>
<td>Peru</td>
<td></td>
<td>CXCD853F</td>
<td>H0958696</td>
</tr>
<tr>
<td>Pink</td>
<td></td>
<td>CXFFC0CB</td>
<td>H06DE0FF</td>
</tr>
<tr>
<td>Plum</td>
<td></td>
<td>CXDDA0DD</td>
<td>H03CBF79</td>
</tr>
<tr>
<td>PowderBlue</td>
<td></td>
<td>CXB0E0E6</td>
<td>H132CB84</td>
</tr>
<tr>
<td>Purple</td>
<td></td>
<td>CX800080</td>
<td>H03C40FF</td>
</tr>
<tr>
<td>Red</td>
<td></td>
<td>CXFF0000</td>
<td>H07880FF</td>
</tr>
<tr>
<td>RosyBrown</td>
<td></td>
<td>CXBC8F8F</td>
<td>H078A640</td>
</tr>
<tr>
<td>RoyalBlue</td>
<td></td>
<td>CX4169E1</td>
<td>H15991B9</td>
</tr>
<tr>
<td>SaddleBrown</td>
<td></td>
<td>CX8B4513</td>
<td>H0914FC2</td>
</tr>
<tr>
<td>Color Name</td>
<td>Color Sample</td>
<td>CX Color Code (RGB)</td>
<td>HLS Color Code*</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Salmon</td>
<td></td>
<td>CXFA8072</td>
<td>H07EB6EE</td>
</tr>
<tr>
<td>SandyBrown</td>
<td></td>
<td>CXF4A460</td>
<td>H093AADE</td>
</tr>
<tr>
<td>SeaGreen</td>
<td></td>
<td>CX2E8B57</td>
<td>H10A5D80</td>
</tr>
<tr>
<td>Seashell</td>
<td></td>
<td>CXFFF5EE</td>
<td>H090F7FF</td>
</tr>
<tr>
<td>Sienna</td>
<td></td>
<td>CXA0522D</td>
<td>H08B678F</td>
</tr>
<tr>
<td>Silver</td>
<td></td>
<td>CXC0C0C0</td>
<td>H000C000</td>
</tr>
<tr>
<td>SkyBlue</td>
<td></td>
<td>CX87CEEB</td>
<td>H13DB9B6</td>
</tr>
<tr>
<td>SlateBlue</td>
<td></td>
<td>CX6A5ACD</td>
<td>H0089488</td>
</tr>
<tr>
<td>SlateGray</td>
<td></td>
<td>CX708090</td>
<td>H14A8020</td>
</tr>
<tr>
<td>Snow</td>
<td></td>
<td>CXFFFFFAFA</td>
<td>H078FDFF</td>
</tr>
<tr>
<td>SpringGreen</td>
<td></td>
<td>CX00FF7F</td>
<td>H10D80FF</td>
</tr>
<tr>
<td>SteelBlue</td>
<td></td>
<td>CX4682B4</td>
<td>H1477D70</td>
</tr>
<tr>
<td>Tan</td>
<td></td>
<td>CXD2B48C</td>
<td>H09AAF70</td>
</tr>
<tr>
<td>Teal</td>
<td></td>
<td>CX008080</td>
<td>H12C40FF</td>
</tr>
<tr>
<td>Thistle</td>
<td></td>
<td>CXD8BFD8</td>
<td>H03CCC3E</td>
</tr>
<tr>
<td>Tomato</td>
<td></td>
<td>CXFF6347</td>
<td>H081A3FF</td>
</tr>
<tr>
<td>Turquoise</td>
<td></td>
<td>CX40E0D0</td>
<td>H12690B8</td>
</tr>
<tr>
<td>Violet</td>
<td></td>
<td>CXEE82EE</td>
<td>H03CB8C2</td>
</tr>
<tr>
<td>Wheat</td>
<td></td>
<td>CXF5DEB3</td>
<td>H09FD4C4</td>
</tr>
<tr>
<td>White</td>
<td></td>
<td>CXFFFFFF</td>
<td>H000FF00</td>
</tr>
<tr>
<td>WhiteSmoke</td>
<td></td>
<td>CXF5F5F5</td>
<td>H000F500</td>
</tr>
</tbody>
</table>
The HLS color naming scheme in SAS follows the Tektronix Color Standard, which places blue at 0 degrees on the hue coordinate. For more information, see “HLS Color Codes” on page 321.

The following table shows the colors that are predefined in the SAS Registry, arranged by hue according to the Tektronix model. For each color, a color sample, and the equivalent RGB (CX) and HLS color codes are shown.

Table A4.2  SAS Registry Colors by Hue

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Sample</th>
<th>CX Color Code (RGB)</th>
<th>HLS Color Code*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow</td>
<td></td>
<td>CXXFFFF00</td>
<td>H0B480FF</td>
</tr>
<tr>
<td>YellowGreen</td>
<td></td>
<td>CX9ACD32</td>
<td>H0C7809B</td>
</tr>
</tbody>
</table>

* The HLS color naming scheme in SAS follows the Tektronix Color Standard, which places blue at 0 degrees on the hue coordinate. For more information, see “HLS Color Codes” on page 321.
<table>
<thead>
<tr>
<th>Color Sample</th>
<th>Color Name</th>
<th>CX Code (RGB)</th>
<th>HLS Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Blue</td>
<td>CX0000FF</td>
<td>H0080FF</td>
</tr>
<tr>
<td>GhostWhite</td>
<td>GhostWhite</td>
<td>CXF8F8FF</td>
<td>H000FCFF</td>
</tr>
<tr>
<td>SlateBlue</td>
<td>SlateBlue</td>
<td>CX6A5ACD</td>
<td>H0089488</td>
</tr>
<tr>
<td>DarkSlateBlue</td>
<td>DarkSlateBlue</td>
<td>CX483D8B</td>
<td>H0086463</td>
</tr>
<tr>
<td>MediumSlateBlue</td>
<td>MediumSlateBlue</td>
<td>CX7B68EE</td>
<td>H008ABCB</td>
</tr>
<tr>
<td>MediumPurple</td>
<td>MediumPurple</td>
<td>CX9370DB</td>
<td>H013A698</td>
</tr>
<tr>
<td>BlueViolet</td>
<td>BlueViolet</td>
<td>CX8A2BE2</td>
<td>H01F87C2</td>
</tr>
<tr>
<td>Indigo</td>
<td>Indigo</td>
<td>CX4B0082</td>
<td>H02241FF</td>
</tr>
<tr>
<td>DarkOrchid</td>
<td>DarkOrchid</td>
<td>CX9932CC</td>
<td>H0287F9B</td>
</tr>
<tr>
<td>DarkViolet</td>
<td>DarkViolet</td>
<td>CX9400D3</td>
<td>H02A6AFF</td>
</tr>
<tr>
<td>MediumOrchid</td>
<td>MediumOrchid</td>
<td>CXBA55D3</td>
<td>H0309496</td>
</tr>
<tr>
<td>Thistle</td>
<td>Thistle</td>
<td>CXD8BFD8</td>
<td>H03CCC3E</td>
</tr>
<tr>
<td>Plum</td>
<td>Plum</td>
<td>CXDDA0DD</td>
<td>H03CBF79</td>
</tr>
<tr>
<td>Violet</td>
<td>Violet</td>
<td>CXEE82EE</td>
<td>H03CB8C2</td>
</tr>
<tr>
<td>Purple</td>
<td>Purple</td>
<td>CX800080</td>
<td>H03C40FF</td>
</tr>
<tr>
<td>DarkMagenta</td>
<td>DarkMagenta</td>
<td>CX8B008B</td>
<td>H03C46FF</td>
</tr>
<tr>
<td>Fuchsia</td>
<td>Fuchsia</td>
<td>CXFF00FF</td>
<td>H03C80FF</td>
</tr>
<tr>
<td>Magenta</td>
<td>Magenta</td>
<td>CXFF00FF</td>
<td>H03C80FF</td>
</tr>
<tr>
<td>Orchid</td>
<td>Orchid</td>
<td>CXDA70D6</td>
<td>H03EA596</td>
</tr>
<tr>
<td>MediumVioletRed</td>
<td>MediumVioletRed</td>
<td>CXC71585</td>
<td>H0526ECE</td>
</tr>
<tr>
<td>DeepPink</td>
<td>DeepPink</td>
<td>CXFF1493</td>
<td>H0578AFF</td>
</tr>
<tr>
<td>Color Sample</td>
<td>Color Name</td>
<td>CX Code (RGB)</td>
<td>HLS Code</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>HotPink</td>
<td>CXFF69B4</td>
<td>H05AB4FF</td>
</tr>
<tr>
<td></td>
<td>LavenderBlush</td>
<td>CXFFF0F5</td>
<td>H064F8FF</td>
</tr>
<tr>
<td></td>
<td>PaleVioletRed</td>
<td>CXDB7093</td>
<td>H064A698</td>
</tr>
<tr>
<td></td>
<td>Crimson</td>
<td>CXDC143C</td>
<td>H06C78D5</td>
</tr>
<tr>
<td></td>
<td>Pink</td>
<td>CXFFC0CB</td>
<td>H06DE0FF</td>
</tr>
<tr>
<td></td>
<td>LightPink</td>
<td>CXFFB6C1</td>
<td>H06EDBFF</td>
</tr>
<tr>
<td></td>
<td>RosyBrown</td>
<td>CXBC8F8F</td>
<td>H078A640</td>
</tr>
<tr>
<td></td>
<td>IndianRed</td>
<td>CXCD5C5C</td>
<td>H0789587</td>
</tr>
<tr>
<td></td>
<td>Brown</td>
<td>CXA52A2A</td>
<td>H0786898</td>
</tr>
<tr>
<td></td>
<td>Firebrick</td>
<td>CXB22222</td>
<td>H0786AAD</td>
</tr>
<tr>
<td></td>
<td>LightCoral</td>
<td>CXF08080</td>
<td>H078B8C9</td>
</tr>
<tr>
<td></td>
<td>Maroon</td>
<td>CX800000</td>
<td>H07840FF</td>
</tr>
<tr>
<td></td>
<td>DarkRed</td>
<td>CX8B0000</td>
<td>H07846FF</td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>CXFF0000</td>
<td>H07880FF</td>
</tr>
<tr>
<td></td>
<td>Snow</td>
<td>CXFFFAFA</td>
<td>H078FDFF</td>
</tr>
<tr>
<td></td>
<td>MistyRose</td>
<td>CXXFE4E1</td>
<td>H07EF0FF</td>
</tr>
<tr>
<td></td>
<td>Salmon</td>
<td>CXFA8072</td>
<td>H07EB6EE</td>
</tr>
<tr>
<td></td>
<td>Tomato</td>
<td>CXFF6347</td>
<td>H081A3FF</td>
</tr>
<tr>
<td></td>
<td>DarkSalmon</td>
<td>CXE9967A</td>
<td>H087B2B7</td>
</tr>
<tr>
<td></td>
<td>Coral</td>
<td>CXFF7F50</td>
<td>H088A8FF</td>
</tr>
<tr>
<td></td>
<td>OrangeRed</td>
<td>CXFF4500</td>
<td>H08880FF</td>
</tr>
<tr>
<td>Color Sample</td>
<td>Color Name</td>
<td>CX Code (RGB)</td>
<td>HLS Code</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>☜</td>
<td>LightSalmon</td>
<td>CXXFA07A</td>
<td>H089BDFF</td>
</tr>
<tr>
<td>☩</td>
<td>Sienna</td>
<td>CXXA0522D</td>
<td>H08B678F</td>
</tr>
<tr>
<td>☜</td>
<td>Seashell</td>
<td>CXXFFF5EE</td>
<td>H090F7FF</td>
</tr>
<tr>
<td>☩</td>
<td>Chocolate</td>
<td>CXXD2691E</td>
<td>H09178BF</td>
</tr>
<tr>
<td>☜</td>
<td>SaddleBrown</td>
<td>CXX8B4513</td>
<td>H0914FC2</td>
</tr>
<tr>
<td>☩</td>
<td>SandyBrown</td>
<td>CXXF4A460</td>
<td>H093A9DE</td>
</tr>
<tr>
<td>☜</td>
<td>Peachpuff</td>
<td>CXXFDAB9</td>
<td>H094DCFF</td>
</tr>
<tr>
<td>☩</td>
<td>Peru</td>
<td>CXXCD853F</td>
<td>H0958696</td>
</tr>
<tr>
<td>☜</td>
<td>Linen</td>
<td>CXXFAF0E6</td>
<td>H095F0AA</td>
</tr>
<tr>
<td>☩</td>
<td>Bisque</td>
<td>CXXFFE4C4</td>
<td>H098E2FF</td>
</tr>
<tr>
<td>☜</td>
<td>DarkOrange</td>
<td>CXXF8C00</td>
<td>H09880FF</td>
</tr>
<tr>
<td>☩</td>
<td>Burlywood</td>
<td>CXXDEB887</td>
<td>H099B391</td>
</tr>
<tr>
<td>☜</td>
<td>Tan</td>
<td>CXXD2B48C</td>
<td>H09AA7FF</td>
</tr>
<tr>
<td>☩</td>
<td>AntiqueWhite</td>
<td>CXXFAEBD7</td>
<td>H09AE9C6</td>
</tr>
<tr>
<td>☜</td>
<td>NavajoWhite</td>
<td>CXXFDEAD</td>
<td>H09BD6FF</td>
</tr>
<tr>
<td>☩</td>
<td>BlanchedAlmond</td>
<td>CXXFEB4D</td>
<td>H09CE6FF</td>
</tr>
<tr>
<td>☜</td>
<td>PapayaWhip</td>
<td>CXXFEFD5</td>
<td>H09DEAFF</td>
</tr>
<tr>
<td>☩</td>
<td>Moccasin</td>
<td>CXXFE4B5</td>
<td>H09EDAFF</td>
</tr>
<tr>
<td>☜</td>
<td>Orange</td>
<td>CXXFA500</td>
<td>H09E80FF</td>
</tr>
<tr>
<td>☩</td>
<td>Wheat</td>
<td>CXXF5DEB3</td>
<td>H09FD4C4</td>
</tr>
<tr>
<td>☜</td>
<td>OldLace</td>
<td>CXXDFE5E6</td>
<td>H09FF2D9</td>
</tr>
<tr>
<td>Color Sample</td>
<td>Color Name</td>
<td>CX Code (RGB)</td>
<td>HLS Code</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>FloralWhite</td>
<td>CXFFFAF0</td>
<td>H0A0F8FF</td>
</tr>
<tr>
<td></td>
<td>DarkGoldenrod</td>
<td>CXB8860B</td>
<td>H0A262E2</td>
</tr>
<tr>
<td></td>
<td>Goldenrod</td>
<td>CXDAA520</td>
<td>H0A27DBE</td>
</tr>
<tr>
<td></td>
<td>Cornsilk</td>
<td>CXFFF8DC</td>
<td>H0A7EEFF</td>
</tr>
<tr>
<td></td>
<td>Gold</td>
<td>CXFFD700</td>
<td>H0AA80FF</td>
</tr>
<tr>
<td></td>
<td>Khaki</td>
<td>CXF0E68C</td>
<td>H0AEBEC4</td>
</tr>
<tr>
<td></td>
<td>LemonChiffon</td>
<td>CXFFFACD</td>
<td>H0AEE6FF</td>
</tr>
<tr>
<td></td>
<td>PaleGoldenrod</td>
<td>CXEEE8AA</td>
<td>H0AECCAA</td>
</tr>
<tr>
<td></td>
<td>DarkKhaki</td>
<td>CXBDB76B</td>
<td>H0AF9462</td>
</tr>
<tr>
<td></td>
<td>Beige</td>
<td>CXF5F5DC</td>
<td>H0B4E98E</td>
</tr>
<tr>
<td></td>
<td>LightGoldenrodYellow</td>
<td>CXFAFAD2</td>
<td>H0B4E6CC</td>
</tr>
<tr>
<td></td>
<td>Olive</td>
<td>CX808000</td>
<td>H0B440FF</td>
</tr>
<tr>
<td></td>
<td>Yellow</td>
<td>CXFFFF00</td>
<td>H0B480FF</td>
</tr>
<tr>
<td></td>
<td>LightYellow</td>
<td>CXFFFFE0</td>
<td>H0B4F0FF</td>
</tr>
<tr>
<td></td>
<td>Ivory</td>
<td>CXFFFFF0</td>
<td>H0B4F8FF</td>
</tr>
<tr>
<td></td>
<td>OliveDrab</td>
<td>CX6B8E23</td>
<td>H0C7599A</td>
</tr>
<tr>
<td></td>
<td>YellowGreen</td>
<td>CX9ACD32</td>
<td>H0C7809B</td>
</tr>
<tr>
<td></td>
<td>DarkOliveGreen</td>
<td>CX556B2F</td>
<td>H0CA4D63</td>
</tr>
<tr>
<td></td>
<td>GreenYellow</td>
<td>CXADFF2F</td>
<td>H0CB97FF</td>
</tr>
<tr>
<td></td>
<td>Chartreuse</td>
<td>CX7FF00</td>
<td>H0D280FF</td>
</tr>
<tr>
<td></td>
<td>LawnGreen</td>
<td>CX7CFC00</td>
<td>H0D27EFF</td>
</tr>
<tr>
<td>Color Sample</td>
<td>Color Name</td>
<td>CX Code (RGB)</td>
<td>HLS Code *</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>DarkSeaGreen</td>
<td>CX8FBC8F</td>
<td>H0F0A640</td>
<td></td>
</tr>
<tr>
<td>ForestGreen</td>
<td>CX228B22</td>
<td>H0F0579B</td>
<td></td>
</tr>
<tr>
<td>LimeGreen</td>
<td>CX32CD32</td>
<td>H0F0809B</td>
<td></td>
</tr>
<tr>
<td>LightGreen</td>
<td>CX90EE90</td>
<td>H0F0BFBB</td>
<td></td>
</tr>
<tr>
<td>PaleGreen</td>
<td>CX98FB98</td>
<td>H0F0CAEC</td>
<td></td>
</tr>
<tr>
<td>DarkGreen</td>
<td>CX006400</td>
<td>H0F032FF</td>
<td></td>
</tr>
<tr>
<td>Green</td>
<td>CX008000</td>
<td>H0F040FF</td>
<td></td>
</tr>
<tr>
<td>Lime</td>
<td>CX00FF00</td>
<td>H0F080FF</td>
<td></td>
</tr>
<tr>
<td>Honeydew</td>
<td>CXF0FFF0</td>
<td>H0F0F8FF</td>
<td></td>
</tr>
<tr>
<td>SeaGreen</td>
<td>CX2E8B57</td>
<td>H10A5D80</td>
<td></td>
</tr>
<tr>
<td>MediumSeaGreen</td>
<td>CX3CB371</td>
<td>H10A787F</td>
<td></td>
</tr>
<tr>
<td>SpringGreen</td>
<td>CX00FF7F</td>
<td>H10D80FF</td>
<td></td>
</tr>
<tr>
<td>MintCream</td>
<td>CXF5FFFF</td>
<td>H10EFAFF</td>
<td></td>
</tr>
<tr>
<td>MediumSpringGreen</td>
<td>CX00FA9A</td>
<td>H1147DFF</td>
<td></td>
</tr>
<tr>
<td>MediumAquamarine</td>
<td>CX66CDAA</td>
<td>H1179A81</td>
<td></td>
</tr>
<tr>
<td>Aquamarine</td>
<td>CX7FFDD4</td>
<td>H118BEF7</td>
<td></td>
</tr>
<tr>
<td>Turquoise</td>
<td>CX40E0D0</td>
<td>H12690B8</td>
<td></td>
</tr>
<tr>
<td>LightSeaGreen</td>
<td>CX20B2AA</td>
<td>H12869B1</td>
<td></td>
</tr>
<tr>
<td>MediumTurquoise</td>
<td>CX48D1CC</td>
<td>H1298D99</td>
<td></td>
</tr>
<tr>
<td>DarkSlateGray</td>
<td>CX2F4F4F</td>
<td>H12C3F41</td>
<td></td>
</tr>
<tr>
<td>PaleTurquoise</td>
<td>CXAFEEEE</td>
<td>H12CCFA6</td>
<td></td>
</tr>
<tr>
<td>Color Name</td>
<td>CX Code (RGB)</td>
<td>HLS Code*</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Teal</td>
<td>CX008080</td>
<td>H12C40FF</td>
<td></td>
</tr>
<tr>
<td>DarkCyan</td>
<td>CX008B8B</td>
<td>H12C46FF</td>
<td></td>
</tr>
<tr>
<td>Aqua</td>
<td>CX00FFFF</td>
<td>H12C80FF</td>
<td></td>
</tr>
<tr>
<td>Cyan</td>
<td>CX00FFFF</td>
<td>H12C80FF</td>
<td></td>
</tr>
<tr>
<td>LightCyan</td>
<td>CXE0FFFF</td>
<td>H12CF0FF</td>
<td></td>
</tr>
<tr>
<td>Azure</td>
<td>CXF0FFFF</td>
<td>H12CF8FF</td>
<td></td>
</tr>
<tr>
<td>DarkTurquoise</td>
<td>CX00CED1</td>
<td>H12C69FF</td>
<td></td>
</tr>
<tr>
<td>CadetBlue</td>
<td>CX5F9EA0</td>
<td>H12D8041</td>
<td></td>
</tr>
<tr>
<td>PowderBlue</td>
<td>CXB0E0E6</td>
<td>H132CB84</td>
<td></td>
</tr>
<tr>
<td>LightBlue</td>
<td>CXADD8E6</td>
<td>H13ACA88</td>
<td></td>
</tr>
<tr>
<td>DeepSkyBlue</td>
<td>CX00BFFF</td>
<td>H13B80FF</td>
<td></td>
</tr>
<tr>
<td>SkyBlue</td>
<td>CX87CEEB</td>
<td>H13DB9B6</td>
<td></td>
</tr>
<tr>
<td>LightSkyBlue</td>
<td>CX87CEFA</td>
<td>H142C1EB</td>
<td></td>
</tr>
<tr>
<td>SteelBlue</td>
<td>CX4682B4</td>
<td>H1477D70</td>
<td></td>
</tr>
<tr>
<td>AliceBlue</td>
<td>CXF0F8FF</td>
<td>H148F8FF</td>
<td></td>
</tr>
<tr>
<td>DodgerBlue</td>
<td>CX1E90FF</td>
<td>H1498FFF</td>
<td></td>
</tr>
<tr>
<td>SlateGray</td>
<td>CX708090</td>
<td>H14A8020</td>
<td></td>
</tr>
<tr>
<td>LightSlateGray</td>
<td>CX778899</td>
<td>H14A8824</td>
<td></td>
</tr>
<tr>
<td>LightSteelBlue</td>
<td>CXB0C4DE</td>
<td>H14DC769</td>
<td></td>
</tr>
<tr>
<td>CornflowerBlue</td>
<td>CX6495ED</td>
<td>H152A9CA</td>
<td></td>
</tr>
<tr>
<td>RoyalBlue</td>
<td>CX4169E1</td>
<td>H15991B9</td>
<td></td>
</tr>
</tbody>
</table>

* The HLS color naming scheme in SAS follows the Tektronix Color Standard, which places blue at 0 degrees on the hue coordinate. For more information, see "HLS Color Codes" on page 321.
Dictionary

%COLORMAC Macro
creates the color utility macros in the current SAS session.

Note: You must run the %COLORMAC macro in your current SAS session before you can use the other color utility macros. After you run the %COLORMAC macro, the color utility macros remain available for the duration of the SAS session.

Syntax

%COLORMAC;

%HELPCLR Macro
writes Help information about the color utility macros to the SAS log.

Requirement: You must run the %COLORMAC macro in the current SAS session before you can use this macro.
Syntax

%HELPCLR<(ALL | macro-name)>;

**Without Arguments**
The %HELPCLR macro writes a list of the color utility macros for which Help information is available to the SAS log.

**Optional Arguments**

**ALL**
writes a short description and an example for each of the color utility macros to the SAS log.

**macro-name**
writes a short description and an example for the specified color utility macro to the SAS log. Do not include the percent sign (%) in macro-name.

**Example**
The following example writes the Help information about the %CMY macro to the SAS log.

```sas
%HELPCLR (CMY);
```

---

**%CMY Macro**

creates an RGB color name from numeric CMY component values.

**Requirement:** You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

**Tip:** Run the %HELPCLR(CYM) macro to write the %CYM macro Help information to the SAS log.

**Syntax**

%CMY(cyan, magenta, yellow);

**Required Arguments**

**cyan, magenta, yellow**
specifies the percentage of cyan, magenta, and yellow to include the RGB color value.

**Range**
0–100 for each value

**Example**
The following code writes the value CX00FF00, which is green, to the SAS log.

```sas
%COLORMAC;
data _null_; put "%%CMY(100,0,100)"; run;
```
%CMYK Macro

creates a CMYK color name from numeric CMYK component values.

**Requirement:** You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

**Tip:** Run the %HELPCLR(CMYK) macro to write the %CMYK macro Help information to the SAS log.

---

**Syntax**

%CMYK(cyan, magenta, yellow, black);

**Required Arguments**

cyan, magenta, yellow, black

specifies the percentage of cyan, magenta, yellow, and black.

**Range** 0–100 for each value

**See** “CMYK Color Codes” on page 320 for more information about the color value produced by using this macro.

**Example** The following code writes the CMYK value 0075294F, which is purple, to the SAS log. In the PUT statement, notice that the %CMYK macro is not placed in quotation marks.

```
%COLORMAC;
data _null_
  put %CMYK(0,46,16,31);
run;
```

---

%CNS Macro

creates a SAS/GRAPH HLS color name from a color-naming scheme (CNS) color name.

**Requirement:** You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

**Tip:** Run the %HELPCLR(CNS) macro to write the %CNS macro Help information to the SAS log.

---

**Syntax**

%CNS(CNS-color-name);

**Required Argument**

CNS-color-name

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

See “HLS Color Codes” on page 321 for more information about HLS color values.

See “Color Naming System (CNS) Values” on page 323 or the %CNS macro Help for more information about valid color-naming scheme color names.

Example The following code writes the HLS value H04B8040, which is grayish reddish purple, to the SAS log.

```sas
%COLORMAC;
data _null_; put "%CNS(GRAYISH REDDISH PURPLE)"; run;
```

%HLS Macro

creates an HLS color name from numeric HLS component values.

**Requirement:** You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

**Tip:** Run the %HELPCLR(HLS) macro to write the %HLS macro Help information to the SAS log.

**Syntax**

```
%HLS(hue, lightness, saturation);
```

**Required Arguments**

**hue, lightness, saturation**

specifies the hue and the percentage of lightness and saturation to be included in the HLS value.

**Ranges**

- **hue**: 0–360
- **lightness** and **saturation**: 0–100 for each value

See “HLS Color Codes” on page 321 for more information.

Example The following example writes the HLS value H00080FF, which is blue, to the SAS log.

```sas
%COLORMAC;
data _null_; put "%HLS(0,50,100)"; run;
```

%HSV Macro

creates an HLS color name from numeric HSV component values.
%HVSMacro

Requirement: You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(HSV) macro to write the %HSV macro Help information to the SAS log.

Syntax

%HSV(hue, saturation, value);

Required Arguments

hue, saturation, value

specifies the hue and the percentage of saturation and brightness (value) to be included in the HLS value.

Ranges

hue: 0–360

saturation and value: 0–100 for each value

See

“HSV (or HSB) Color Codes” on page 322 and “HLS Color Codes” on page 321 for more information.

Example

The following example writes the HSV value V000FFBF, which is dark red, to the SAS log.

%COLORMAC;
data _null_;  
put "*%HSV(0,100,75)";  
run;

%RGB Macro

creates an RBG color name from numeric RGB component values.

Requirement: You must run the %COLORMAC on page 1525 macro in the current SAS session before you can use this macro.

Tip: Run the %HELPCLR(RGB) macro to write the %RGB macro Help information to the SAS log.

Syntax

%RGB(red, green, blue);

Required Arguments

red, green, blue

specifies the percentage of red, green, and blue to be included in the RGB value.

Range

0–100 for each value

See

“RGB Color Codes” on page 318 for more information.
Example

The following example writes the RGB value CXFFFF00, which is yellow, to the SAS log.

```sas
%COLORMAC;
data _null_; put "&RGB(100,100,0)"; run;
```

%HLS2RGB Macro

creates an RGB color name from an HLS color name.

**Requirement:**

You must run the `%COLORMAC` macro in the current SAS session before you can use this macro.

**Note:**

Round-trip conversions using the HLS2RGB and RGB2HLS macros might produce ultimate output values that differ from the initial input values. For example, converting CXABCDEF (a light blue) using %RGB2HLS produces H14ACDAD. Converting this value back to RGB using %HLS2RGB returns CXAACCEE. While not identical, the colors are very similar on the display, and when printed.

**Tip:**

Run the `%HELPCLR(HLS2RGB)` macro to write the %HLS2RGB macro Help information to the SAS log.

**Syntax**

```sas
%HLS2RGB(HLS-color-name);
```

**Required Argument**

**HLS-color-name**

specifies the HLS color name that is to be converted to an RGB color name.

**See**

“HLS Color Codes” on page 321 and “RGB Color Codes” on page 318 for more information.

**Example**

The following example converts HLS name H04B8040 to RGB name CX9F5F8F, which is grayish reddish purple, and writes it to the SAS log.

```sas
%COLORMAC;
data _null_; put "&HLS2RGB(H04B8040)"; run;
```

%RGB2HLS Macro

creates an HLS color name from an RGB color name.

**Requirement:**

You must run the `%COLORMAC` macro in the current SAS session before you can use this macro.

**Note:**

Round-trip conversions using the RGB2HLS and HLS2RGB macros might produce ultimate output values that differ from the initial input values. For example, converting CXABCDEF (a light blue) using %RGB2HLS produces H14ACDAD. Converting this...
value back to RGB using %HLS2RGB returns CXAACCEE. While not identical, the colors are very similar on the display, and when printed.

**Tip:** Run the %HELPCLR(RGB2HLS) macro to write the %RGB2HLS macro Help information to the SAS log.

---

### Syntax

```
%RGB2HLS(RGB-color-name);
```

### Required Argument

*RGB-color-name*

specifies the RGB color name that is to be converted to an HLS color name.

<table>
<thead>
<tr>
<th>See</th>
<th>“RGB Color Codes” on page 318 and “HLS Color Codes” on page 321 for more information.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>The following example converts RGB name CX9F5F8F to HLS name H04C7F40, which is grayish reddish purple, and writes it to the SAS log.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%COLORMAC;</td>
</tr>
<tr>
<td></td>
<td>data <em>null</em>;</td>
</tr>
<tr>
<td></td>
<td>put &quot;%%RGB2HLS(CX9F5F8F)&quot;;</td>
</tr>
<tr>
<td></td>
<td>run;</td>
</tr>
</tbody>
</table>
Appendix 6
GREPLAY Procedure
Template Reference

Overview

This SAS/GRAPH code can be used to re-create the templates stored in SASHELP.TEMPLT. You can modify the code to create custom templates. For detailed
information about using, creating, and modifying templates, refer to Chapter 43, “GREPLAY Procedure,” on page 1280.

**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.

```plaintext
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=0
         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.

```plaintext
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=0
         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.

```plaintext
proc greplay tc=tempcat nofs;
```
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=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=0
    color=black
  3/llx=66.6 lly=0
    ulx=66.6 uly=100
    urx=100 ury=100
    lrx=100 lry=0
    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
```

```
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)"
    1llx=0  lly=0
        ulx=0  uly=100
        urx=22 lry=100
        lrx=22 lry=0
        color=black
    2/llx=26 lly=0
        ulx=26  uly=100
        urx=48 lry=100
        lrx=48 lry=0
        color=black
    3/llx=52 lly=0
        ulx=52  uly=100
        urx=74 lry=100
        lrx=74 lry=0
        color=black
    4/llx=78 lly=0
        ulx=78  uly=100
        urx=100 lry=100
        lrx=100 lry=0
        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=0
            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=0
            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.

```plaintext
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=0
              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.

```plaintext
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=0
              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.

```plaintext
proc greplay tc=tempcat nofs;
  tdef L2R2 des="2 BOXES LEFT, 2 BOXES RIGHT"
    1/llx=0 lly=50
       ulx=0 uly=100
       urx=50 ury=100
       lrx=50 lry=50
       color=black
    2/llx=0 lly=0
       ulx=0 uly=50
       urx=50 ury=50
       lrx=50 lry=0
       color=black
    3/llx=50 lly=50
       ulx=50 uly=100
       urx=100 ury=100
       lrx=100 lry=50
       color=black
    4/llx=50 lly=0
       ulx=50 uly=50
       urx=100 ury=50
       lrx=100 lry=0
       color=black;
quit;
```

L2R2S: Two Boxes Left and Two Boxes Right with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```plaintext
proc greplay tc=tempcat nofs;
  tdef L2RS des="2 BOXES LEFT, 2 BOXES RIGHT (WITH SPACE)"
    1/llx=0 lly=52
       ulx=0 uly=100
       urx=48 ury=100
       lrx=48 lry=52
       color=black
    2/llx=0 lly=0
       ulx=0 uly=48
       urx=48 ury=48
       lrx=48 lry=0
       color=black
    3/llx=52 lly=52
```
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.

```plaintext
proc greplay tc=tempcat nofs;
tdef U1D2 des= "1 BOX UP, 2 BOXES DOWN"
  1/llx=0  lly=50
    ulx=0   uly=100
    urx=100 ury=100
    lrx=100 lry=50
    color=black
  2/llx=0  lly=0
    ulx=0   uly=50
    urx=50  ury=50
    lrx=50  lry=0
    color=black
  3/llx=50 lly=0
    ulx=50  uly=50
    urx=100 ury=50
    lrx=100 lry=0
    color=black;
quit;
```

U1D2S: One Box Up and Two Boxes Down with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```plaintext
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
```

Appendix 6 • GREPLAY Procedure Template Reference
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.

```plaintext
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.

```plaintext
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
```

U2D1S: Two Boxes Up and One Box Down with Space
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.

```plaintext
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=00
      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.

```plaintext
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=00
      color=black;
  quit;
```
V3: Three Boxes Vertically

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
  tdef V3 des= "3 BOXES VERTICALLY"
  /* define panel 1 */
  1/llx=0   lly=66.6
  ulx=0 uly=100
  urx=100 ury=100
  lrx=100 lry=66.6
  color=black
  2/llx=0   lly=33.3
  ulx=0 uly=66.6
  urx=100 ury=66.6
  lrx=100 lry=33.3
  color=black
  3/llx=0   lly=0
  ulx=0 uly=33.3
  urx=100 ury=33.3
  lrx=100 lry=00
  color=black;
quit;
```

V3S: Three Boxes Vertically with Space

Start the GREPLAY procedure in line mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```
proc greplay tc=tempcat nofs;
  tdef V3S des= "3 BOXES VERTICALLY (WITH SPACE)"
  1/llx=0   lly=70
  ulx=0 uly=100
  urx=100 ury=100
  lrx=100 lry=70
  color=black
  2/llx=00  lly=35
  ulx=00 uly=65
  urx=100 ury=65
  lrx=100 lry=35
  color=black
  3/llx=00  lly=00
  ulx=00 uly=30
  urx=100 ury=30
  lrx=100 lry=00
  color=black;
quit;
```
WHOLE: Entire Screen Template

Start the GREPLAY procedure in line-mode. The TC statement specifies the catalog where the template is stored. The TDEF statement defines the name and description of each catalog entry.

```plaintext
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 7

Differences between Base SAS ODS Graphics and SAS/GRAPH

SAS provides two distinct systems for generating graphics output. The following table lists the major differences between these two systems.

Table A7.1 Differences between ODS Graphics and SAS/GRAPH

<table>
<thead>
<tr>
<th>Graphics System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS Graphics</td>
<td>ODS Graphics is a template-based graphics system that is delivered with Base SAS. In ODS Graphics, output is produced from a compiled ODS template of type STATGRAPH. Predefined templates that are supplied by SAS are stored in SASHELP.TMPLMST. User-defined templates are stored in SASUSER.TMPLMST. The Base SAS procedures that produce template-based graphics are the SGPLOT, SG PANEL, SGSCATTER, SG DESIGN, and SGRENDER. Many SAS/STAT, SAS/ETS, and SAS/QC procedures also produce template-based graphics automatically by default. Template-based graphics are always produced as files and never as GRSEG catalog entries. For template-based graphics, the ODS GRAPHICS statement is used to control the graphical environment. Template-based graphics are referred to as ODS Graphics.</td>
</tr>
<tr>
<td>SAS/GRAPH</td>
<td>SAS/GRAPH is a device-based graphics system that is a separate product. It must be licensed and installed at your site. In SAS/GRAPH, output is produced by graphics devices, which determine the type of output that is generated. Examples of graphics devices are SVG, PNG, GIF, ACTIVEX, and SASPRTC. The graphics devices that are supplied by SAS are stored in the SASHELP.DEVICES catalog. Most procedures that produce device-based graphics also produce GRSEG catalog entries in addition to any image files, vector files, or displayed output that are produced. SAS/GRAPH procedures that produce device-based graphics and GRSEG catalog entries include the GCHART, GPLOT, GMAP, GBARLINE, GCONTOUR, and G3D procedures. The device-based procedures that do not produce GRSEG catalog entries are the GAREABAR, GKPI, and GTILE procedures. For device-based graphics, the GOPTIONS statement is used to control the graphical environment.</td>
</tr>
</tbody>
</table>

Although ODS Graphics and SAS/GRAPH are distinct systems, the following SAS statements apply to both:

• ODS destination statements such as ODS HTML, ODS PDF, and so on, specify the output document format for both ODS Graphics and SAS/GRAPH.
Global statements TITLE, FOOTNOTE, and NOTE add text to output from both ODS Graphics and SAS/GRAPH.

Conversely, the following features of each are not compatible:

- The annotate facility for ODS Graphics and SAS/GRAPH are not interchangeable.
- The SAS/GRAPH global statements AXIS, LEGEND, PATTERN, and SYMBOL do not apply to ODS Graphics.
Appendix 8
Understanding Hexadecimal Values

This section provides an overview of the hexadecimal numbering system. The hexadecimal system is a base 16 numbering system where each digit represents one of the values shown in the following table.

<table>
<thead>
<tr>
<th>Hexadecimal Value</th>
<th>Decimal Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–9</td>
<td>0–9</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
</tr>
</tbody>
</table>

The maximum decimal value that a hexadecimal value can represent is $16^n - 1$, where $n$ is the number of digits in the hexadecimal value. A two-digit hexadecimal value can represent a maximum decimal value of 255, and a four-digit hexadecimal value can represent a maximum decimal value of 65,535. To convert a hexadecimal number to decimal, sum the product of each hexadecimal character and its base power, $16^n$, where $n$ is the digit’s significance. For example, to convert hexadecimal value C8A4 to decimal manually, do the following:

\[
(12 \times 16^3) + (8 \times 16^2) + (10 \times 16) + 4 = 51364
\]

To convert from decimal to hexadecimal manually, iteratively divide the decimal value by 16. In each iteration, multiply the remainder by 16 to get the hexadecimal character for that iteration, and then use the quotient in the next iteration until the quotient is zero. This method generates the hexadecimal value from the least-significant digit to the most significant digit. The following table demonstrates how to convert decimal value 51,364 back to its hexadecimal value.

<table>
<thead>
<tr>
<th>Divide by 16</th>
<th>Quotient</th>
<th>Remainder</th>
<th>16 x Remainder</th>
<th>Hexadecimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>51364 / 16</td>
<td>3210</td>
<td>0.25</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Since the hexadecimal characters are generated from the least-significant digit to the most significant, the result is C8A4.

Apart from the base and the value representation, the methods for manipulating hexadecimal values, such as addition, subtraction, and multiplication, are the same as those that are used in the base 10 system. Here are some examples.

<table>
<thead>
<tr>
<th>Dividing by 16</th>
<th>Quotient</th>
<th>Remainder</th>
<th>16 x Remainder</th>
<th>Hexadecimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3210 / 16</td>
<td>200</td>
<td>0.625</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>200 / 16</td>
<td>12</td>
<td>0.5</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>12 / 16</td>
<td>0</td>
<td>0.75</td>
<td>12</td>
<td>C</td>
</tr>
</tbody>
</table>

Many pocket calculators and calculator programs enable you to manipulate hexadecimal values and convert values between hexadecimal and decimal. You can also use a DATA step in SAS to manipulate and convert hexadecimal values. Here is a simple macro that uses the INPUT statement in a DATA step to convert a hexadecimal value to its equivalent decimal value.

```sas
%macro hex2dec(hex);
  data _null_
  msg=cat("#", %upcase("&hex"), ", " , input("&hex", hex.));
  put msg;
  run;
%mend hex2dec;
```

The following example shows how to use this macro to convert the hexadecimal value C8A4 to its decimal equivalent.

```sas
%hex2dec(c8a4);
#C8A4 = 51364
```

Here is a simple macro that uses the PUT statement in a DATA step to convert a decimal value to its equivalent hexadecimal value.

```sas
%macro dec2hex(dec);
  data _null_
  msg=cat("&dec = ", put(&dec, hex4.));
  put msg;
  run;
%mend dec2hex;
```

The following example shows how to use this macro to convert the decimal value 51,364 back to its hexadecimal equivalent.

```sas
%dec2hex(51364);
51364 = #C8A4
```
For information about macros, see *SAS Macro Language: Reference*. For information about the INPUT and PUT DATA step functions, see *SAS Functions and CALL Routines: Reference*.

When you use the hexadecimal system to define your own colors, the exact value for the color is not as significant as the relationship of the digits in the value to each other, and the relationship of the value of this color to other colors.
Appendix 9

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

Transporting Catalogs Using CPORT and CIMPORT

Use the CPORT and CIMPORT procedures to transport catalogs and catalog entries from one machine to another machine running in a different operating environment. In addition to graphics output stored in GRSEG catalog entries, SAS/GRAPH software produces several other files that you can transport from host environment to host environment. These other files include...
To transport catalog entries that contain graphics output (catalog entries of type GRSEG), follow these steps:

1. Use the CPORT procedure to create a transport file from the catalog entries in the current host environment. A transport file is a sequential file that contains the catalog in SAS transport format. To create a transport file, you must specify a catalog to be converted and a fileref for the transport file.

   To retain the original order of the GRSEG entries in the catalog, use SELECT= in the PROC CPORT statement to export individual graphs in the order in which they were created. Otherwise, when you use the GREPLAY procedure to list the graphics entries in the imported catalog, the procedure will list the entries in alphabetical order, rather than the order in which they were created.

   Note: Only the GREPLAY procedure can list catalog entries in the order in which they were created. All other procedures list entries in alphabetical order.

   To export a catalog that contains groups of entries created using the GREPLAY procedure, you must use SELECT= in the PROC CPORT statement to select the names of the groups, rather than the names of individual graphs, to be included in the transport file. If you export the entire catalog without using SELECT=, the groups are not maintained in the catalog created when you import the transport file in the new host environment.

   When you use the CPORT procedure, messages in the SAS log identify the catalog entries that have been placed in the transport file. If the catalog entry was created by replaying several graphs into a template, the log messages list the names of all of the entries that contributed to the templated graph.

2. Move the transport file to the target machine, if necessary. You must move the transport file in binary format. If you do not move the transport file in binary format, the CIMPORT procedure cannot read the file that you create.

3. Once you have moved the transport file to the target machine, import the transport file into a catalog in the new host environment using the CIMPORT procedure. The entries are imported in the order specified in SELECT= in the PROC CPORT statement used to create the transport file.

   The SELECT= option in the PROC CIMPORT statement does not affect the order of the imported entries.

   Note: You must use the CIMPORT procedure from the current version of 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 about using the CPORT and CIMPORT procedures, see the Base SAS Procedures Guide.

Example of Transporting GRSEGs

This example shows how to port three entries from the catalog MYLIB.GRAPHS.
First, the CPORT procedure writes selected graphs from MYLIB.GRAPHS to the transport file TRANFILE. The SELECT option names the graphs to be ported.

```sas
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.

```sas
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:

```sas
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:

```sas
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:

```sas
filename tranfile "external-file";
libname gfont0 "SAS-data-library";
proc cport file=tranfile
catalog=gfont0.fonts
 select=(figures.font);
```
After moving the transport file to the new host environment, you can import the font using the following statements:

```plaintext
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:

```plaintext
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:

```plaintext
filename tranfile "external-file";
libname gdevice0 "SAS-data-library";
proc cimport infile=tranfile catalog=gdevice0.devices;
run;
```

**Converting Catalogs to a Different Version of SAS**

To convert catalogs to a different version of SAS, use the LIBNAME statement and the CATALOG procedure.

*Note:* You will not be able to use your old catalogs without transporting them first.

Before using PROC CATALOG, you must assign librefs to both catalogs and specify the Version 6 Compatibility Engine (saseb) on the input catalog LIBNAME. Then use PROC CATALOG with a COPY statement to convert a catalog from Version 6 to Version 7 or later. For details about using the CATALOG procedure, see the Base SAS Procedures Guide.

For example, the following statements can be submitted from Version 8 to assign the Version 6 Compatibility Engine and convert a catalog from Version 6 to Version 8.

```plaintext
libname v6lib saseb "SAS-data-library";
libname v8lib "SAS-data-library";
proc catalog catalog=v6lib.v6cat;
copy out=v8lib.v8cat;
```
run;
Here is the recommended reading list for this title:

- *SAS/GRAPH®: Beyond the Basics*
- *Output Delivery System: The Basics and Beyond*
- *SAS/GRAPH and Base SAS: Mapping Reference*

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books
absolute coordinate
a coordinate that is measured from the origin of a coordinate system.

ActiveX
a technology developed by Microsoft that is used to add interactivity to web pages.

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 bar has both a height 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.

axis area
an area bounded by axes, which might be enclosed by an axis line.

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.

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
See character cell.

center point
the location in the GRAPH window that, in conjunction with a radius point, defines
the placement and shape of an ellipse or a pie.

CGM
See computer graphics metafile.

character cell (cell)
in device-based SAS/GRAPH procedures, a unit of measure whose size and shape
are determined by both the size of the graphics output area and by the number of
rows and columns in the graphics output area.

character up vector
the angle at which a character is positioned. The character up vector has two
components, x and y, which determine the angle.

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 vertex
a point on a radar chart where a statistical value intersects the spokes.

choropleth map
a two-dimensional map that uses color and fill pattern combinations to represent
different categories or levels of magnitude.

class variable
See classification variable.

classification variable (class variable)
a variable whose values are used to classify the observations in a data set into
different groups that are meaningful for analysis. A classification variable can have
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
the list of foreground colors that are available for 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.
The colors are derived from either the main color scheme models (NAME, RGB, RGBA, HLS), or from the secondary color scheme models (CMYK, GRAY, HSV).

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.

computer graphics metafile (CGM)
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.

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, 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.

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 of type DEV that stores the values of device parameters (or the characteristics) that are used with a particular output device.
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.

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 explore data and access information by moving from summary information to more detailed data from which the summary is derived. For example, you could click folders in a hierarchy from the top downward to find a specific file. Drilling down provides a method of exploring multidimensional data by moving from one level of detail to the next.

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 typeface with a specific character shape, spacing, weight, and size. The characters in a font can be figures, symbols, or alphanumeric.

font family
a set of one or more typefaces that share common design characteristics such as serifs, proportional or uniform spacing, or special symbols. For example, Helvetica, Arial, and Albany AMT are members of a sans-serif, proportional font family.

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 unit
in the GFONT procedure, a unit within a range that is defined by 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.
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.

geo-variable
in a feature table, the $GEOREF formatted variable that stores the spatial information as a geometry object.

geocoding
the process of assigning geographic coordinates (often expressed as latitude and longitude) to other geographic data such as street addresses, or postal codes.

global statement
a SAS statement that you can specify anywhere in a SAS program.

graphics device
See graphics output device.

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. See also device-independent catalog entry, 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, graphics output device.

graphics output device (graphics device, hard-copy 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 (GSF)
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.

graphics template
See ODS template.

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 that is used to categorize chart variable values into
groups. A group variable enables the data for each distinct group value to be
rendered in a visually different manner. For example, a grouped scatter plot displays
a distinct marker and color for each group value.

GRSEG
a SAS catalog entry that contains graphic output in a generic, rather than device-
specific, format.

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.

hardcopy device
See graphics output device.

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.

hardware 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.
**hatch**  
a fill pattern consisting of parallel lines at any specified angle.

**HLS color model**  
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.

**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 model**  
a color-coding scheme that specifies a color in terms of its hue, saturation, and value components. Hue is the color. Saturation is the aspect of a color that determines its relative strength and departure from gray. And value, or 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 graphics elements with HTML links to implement drill-down functionality. The graphics elements are represented by sets of coordinates. See also data tip.

**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 CPOR procedure.

**interactive 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**  
See JPEG.
JPEG (Joint Photographic Experts Group)
the name of an industry-standard file format for compressed images. Saving an image in JPEG format typically provides 10:1 compression with little perceptible loss in image quality.

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/GRAF internal character encoding. See also device map.

latitude
used with maps, the angular measure between the equator and the circle of parallel on which a point lies.

library reference
See libref.

libref (library reference)
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

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 mark
one of a series of points on an axis that mark the major divisions of the axis scale.

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, key map.

map area
See unit area.

map data set
a data set provided by SAS that contains variables whose values are coordinates that define the boundaries of map areas, such as a state or country.

mapping
the process of displaying data values on a map.

marker
a symbol such as a diamond, a circle, or a triangle that is used to indicate the location of, or annotate, a data point in 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 mark**
one of a series of points that fall between major tick marks on an axis scale.

**needle plot**
a plot in which data points are connected by a vertical line that 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/link diagram, 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.

**ODS template (graphics template)**
a description of how output should appear when it is formatted. ODS templates are stored as compiled entries in a template store (item store). Common template types include STATGRAPH, STYLE, CROSSTABS, TAGSET, and TABLE.

**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.

**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.
**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 visual representation of data such as a scatter plot, needle plot, or contour plot.

**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 (PNG)**

A file format that returns the graphical output in separate files and that produces a non-interactive image. This format is similar to the GIF format, but has additional features, such as support for true-color images and better compression.

**PostScript**

A device-independent page description language for printing high-resolution integrated text and graphics.

**predefined color**

One of the set of colors for which SAS/GRAPH software defines and recognizes names (for example, BLACK, BLUE, and CYAN).

**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**, **data area**.

**projection**

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, between computers and peripheral devices, and between software applications. TCP/IP, FTP, and HTTP are examples of protocols.

**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 appearance of a radar screen. Variations of the radar chart have names based on what they look like; these include star charts, spider charts, wind rose charts, and calendar charts.

**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.

**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 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 value**
any value 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**
in the GMAP procedure, a variable in the response data set that contains data values that are associated with a map area. See also **chart variable**, **response data set**, **response levels**.
**RGB color model**

a color-coding scheme that specifies a color in terms of amounts of red, green, and blue components.

**RGBA color space**

da color-coding scheme that specifies a color in terms of amounts of red, green, and blue components, along with an alpha channel that controls the color opacity. See also RGB color model.

**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 output files. 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 table rows. The coordinates of each point in the plot correspond to the data values for a single table row (observation).

**snap**
in the graphics editor, to automatically place graphics objects in the grid display area with precision.

**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**
a method of interpolation in which a smooth line or surface connects data points.

**spoke**
any of a number of lines that radiate from the center of a radar chart or a 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 type of radar chart that features lines drawn to connect the chart statistics for each spoke, resulting in a star-like appearance.

**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 is defined in ODS with a reserved name and value. Style attributes are collectively referenced by a style element within a style template.
**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 values of a vertical Z variable based on gridded 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.

**Tagged Image File Format**
*See TIFF.*

**template**
a specification of an area or areas on a page. A GREPLAY template defines a layout in which you can display one or more graphs on a single page.

**template panel**
in the GREPLAY procedure, a part of the template in which one or more graphics can be displayed. A template can contain one or more panels.

**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**
*See treemap.*

**tilt angle**
the measure in degrees from the horizontal axis to the major axis of an object.

**tool palette**
the collection of icons that represent functions in the interface.

**tooltip**
descriptive text that appears when a cursor is placed over certain elements of a graphical user interface, such as the tool icons in a toolbar.

**translate**
to change the location of a graphics object.
treemap (tile chart)  
a graph that represents the relative values of data by using nested rectangular areas.  
The color or pattern of each area represents the value of one measure in the query.  
The size of each area represents the value of another measure in the query.

typeface  
a set of characters with a common design, represented by one or more fonts that  
differ in weight, orientation, width, size, and spacing. For example, Arial, Arial Bold,  
Arial Italic, and Arial Bold Italic share the same typeface (Arial), but differ in their  
orientation and weight.

unit  
a single quantity of measurement. In SAS/GRAPH software, units can represent any  
of the following: centimeters, percentages, points, inches, or cells.

unit area (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.  
See also identification variable.

view  
a definition of a virtual data set that is named and stored for later use. A view  
contains no data; it merely describes or defines data that is stored elsewhere.

web server  
a computer program that delivers (serves) content, such as web pages, over the  
World Wide Web. It can also refer to the computer or virtual machine that runs the  
program.

wind rose chart  
a type of radar chart that depicts how wind speed and direction are typically  
distributed at a particular location. The cardinal directions or subdirections with the  
longest spokes indicate the wind direction that has the greatest frequency. See also  
radar chart.

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

Special Characters
_ALL_ option
   LIST statement (GDEVICE) 1053
_NEXT_ option
   LIST statement (GDEVICE) 1053
_PREV_ option
   LIST statement (GDEVICE) 1053
? statement
   GREPLAY procedure 1290
# (number sign), variables as plot point labels 430
#BYLINE option, text string specifications 461
#BYVAL option, text string specifications 348, 357, 360, 369, 380, 387, 395, 461, 465, 946, 959
#BYVAR option, text string specifications 348, 357, 360, 369, 380, 388, 395, 461, 465, 946, 959
%ANNOMAC macro, Annotate facility 752
%ARROW macro, Annotate facility 752
%BAR, %BAR2 macro, Annotate facility 753
%CENTROID macro
   Base SAS 754
%CIRCLE macro, Annotate facility 755
%CMY macro 1526
%CMYK macro 1527
%CNS macro 1527
%COMMENT macro, Annotate facility 756
%DCLANNO macro, Annotate facility 757
%DRAW macro, Annotate facility 757
%DRAW2TXT macro, Annotate facility 758
%FRAME macro, Annotate facility 758
%HELPCLR macro 1525
%HLS macro 1528
%HLS2RGB macro 1530
%HSV macro 1528
%LABEL macro, Annotate facility 759
%LINE macro, Annotate facility 760
%MAPLABEL macro, Annotate facility 761
%MOVE macro, Annotate facility 762
%PIEXY macro, Annotate facility 762
%POLY, %POLY2 macro, Annotate facility 763
%POLYCONT macro, Annotate facility 764
%POP macro, Annotate facility 765
%PUSH macro, Annotate facility 765
%RECT macro, Annotate facility 765
%RGB macro 1529
%RGB2HLS macro 1530
%SCALE macro, Annotate facility 766
%SCALET macro, Annotate facility 768
%SEQUENCE macro, Annotate facility 769
%SHORTCUT macro 94
%SLICE macro, Annotate facility 770
%SWAP macro, Annotate facility 771
%SYSTEM macro, Annotate facility 772
%TXT2CNTL macro, Annotate facility 773

Numbers
3-D charts
3-D bar charts 352
3-D vertical bar charts
   subgrouping 982

A
A option
   AXIS option, PROC GOPTIONS statement 1127
A= option
   AXIS statement options 361
   LEGEND statement options 389
   TITLE, FOOTNOTE, and NOTE statements 449
   access permissions, browsers 267
   ACROSS= option
   CHART statement (GRADAR) 1251
LEGEND statement options 378
PIE, PIE3D, DONUT statements (GCHART) 940
STAR statement (GCHART) 963
ACROSSVAR= option
CHART statement (GRADAR) 1251
active color lists 1106
ACTIVECOLORS= option
GKPI procedure 1112
ActiveX Control
authentication 267
ACTIVEX device driver
data tips for 192
drill-down links in output 198
ActiveX devices 87
ActiveX support 1442
Annotate functions 1484
AXIS statement 1442
G3D procedure 1482
GAREABAR procedure 1454
GBARLINE procedure 1454
GCHART procedure 1457
GCONTOUR procedure 1457
GMAP procedure 1457
GOPTIONS statement 1444
G PLOT procedure 1471
GRADAR procedure 1477
LEGEND statement 1449
PATTERN statement 1450
SYMBOL statement 1451
TITLE and FOOTNOTE statements 1453
ACTUAL= argument
GKPI procedure 1111
ACTXIMG device driver 112
chart descriptions for 190
data tips for 192
drill-down links in output 198
sample programs for non-interactive images 160
SVG, PNG, GIF, and TIFF vs. 156
web presentations, developing 157
ADD statement
GDEVICE procedure 1050
additional fonts 300
AFONT= option
GKPI procedure 1112
AFTER argument
MOVE statement (GREPLAY) 1299
alignment
axis labels 348, 353, 356, 364
axis values 364
character cells 527
legend labels 380
legend text 390
legend values 387
 legends 384, 393
plot point labels 429
text in graphics output 455
ALTDESC option 522
alternative device-resident fonts 1508
anchors 398, 513
angle rotation
donut chart labels 954, 956
ANGLE variable, Annotate facility 706
angle, rotation
axis labels 348, 353, 356, 361
hardware text rotation 590
landscape orientation of graphics output
area 581, 582, 609
portrait orientation of graphics output
area 591, 604, 609
print orientation 609
printing orientation 609
text in graphics output 449, 456, 459
ANGLE= option
AXIS statement 361
LEGEND statement options 389
PIE, PIE3D, DONUT statements (GCHART) 940
STAR statement (GCHART) 963
TITLE, FOOTNOTE, and NOTE statements 449
ANGLE= suboption
LABEL= option, DONUT statement (GCHART) 954
angles
legend text 389
ANIMATE system option 171
animated GIF images 176, 178
creating with BY-group processing 176
creating with GIF graphics output device 178, 181
creating with GREPLAY procedure 181
creating with RUN-group processing 178
creating with transparent background 178
sample programs 176
Animated GIF images
creating with GIF graphics output device 167
animated GIF or SVG images
starting and stopping output in SAS programs 171
animated GIF presentations
drill-down in 198
animation 167
annotation labels 173
delay between graphs 538
repeating as loop 579
SVG graphics 167
ANNO= option
GSLIDE procedure 1337, 1342
Annotate data sets 638, 650
  applying to web output 201
  missing values 651
observation and structure of producing graphics output 651
producing multiple graphs 790
scaling data-dependent output 784
Annotate DATA step 44
Annotate facility 636
ActiveX and Java support for coordinates 646
drill-down links, generating 202
drill-downs, generating 201
error messages, list of 656
examples 648
functions 643, 667
graphic elements and formatting images, displaying 338
  in text slides 1334, 1342
internal coordinates 706
macros for 750
macros, how to use 750
processing details 652
scaling graphs 783
variables 640, 649, 652, 706
Annotate graphics storing 787
Annotate macros 651
ANNOTATE= argument
PROC GANNO statement 781
ANNOTATE= option 201, 652
BAR statement (GBARLINE) 832
BLOCK statement (GCHART) 890
BBUBBLE statement (GPLOT) 1146
BBUBBLE2 statement (GPLOT) 1160
CHART statement (GRADAR) 1251
G3D procedure 1387
GSLIDE procedure 1337, 1342
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 906
PIE, PIE3D, DONUT statements (GCHART) 940
PLOT statement (G3D) 1387
PLOT statement (GCONTOUR) 1014
PLOT statement (GPLOT) 1173
PLOT2 statement (GPLOT) 1192
PROC GBARLINE statement 828
PROC GCHART statement 888
PROC GCONTOUR statement 1011
PROC GPLOT statement 1142
PROC GRADAR statement 1247
PROC statement 374
SCATTER statement (G3D) 1396
STAR statement (GCHART) 963
appearance differences between devices 268
appearance of graphs 274
graphical style element reference for device-based graphics 287
modifying styles 284
overriding style attributes 280
precedence of appearance option specifications 281
specifying styles 279
style attributes versus device entry parameters 274
style templates 275
turning off styles 298
viewing list of styles provided by SAS 284
arc-drawing capability, device 530
archivable PDF file, creating 243
arcs
drawing with Annotate facility 692, 693
area bar charts 797
ActiveX and Java support for example, generating 803
  with numeric chart variable 805
  with subgroups 808
  with subgroups and variable percentages 810
AREA element (HTML) 513
areas, Annotate graphics 646
AREAS= option
PROC GANNO statement 781
ANNOTATE= option 201, 652
BAR statement (GBARLINE) 832
BLOCK statement (GCHART) 890
BUBBLE statement (GPLOT) 1146
BUBBLE2 statement (GPLOT) 1160
CHART statement (GRADAR) 1251
G3D procedure 1387
GSLIDE procedure 1337, 1342
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 906
PIE, PIE3D, DONUT statements (GCHART) 940
PLOT statement (G3D) 1387
PLOT statement (GCONTOUR) 1014
PLOT statement (GPLOT) 1173
PLOT2 statement (GPLOT) 1192
PROC GBARLINE statement 828
PROC GCHART statement 888
PROC GCONTOUR statement 1011
PROC GPLOT statement 1142
PROC GRADAR statement 1247
PROC statement 374
SCATTER statement (G3D) 1396
BUBBLE statement (G PLOT) 1147
PLOT statement (G CONTOUR) 1015
PLOT statement (G PLOT) 1174
AUTOLABEL option
PLOT statement (G CONTOUR) 1015
automatic paper feed  524, 586
AUTOREF option
BAR statement (G BARLINE) 832
HBAR, HBAR3D, VBAR, VBAR3D
statements (GCHART) 907
PLOT statement (G BARLINE) 852
AUTOREF option, AXIS statement
options  362
AUTOSIZE  525
AUTOSIZE= graphics option  525
AUTOVREF option
BUBBLE statement (G PLOT) 1147
BUBBLE2 statement (G PLOT) 1160
PLOT statement (G CONTOUR) 1015
PLOT statement (G PLOT) 1174
PLOT2 statement (G PLOT) 1194
AVALUE option
GKPI procedure  1112
axes  367
bar-line charts  818
chart statistic and response axis  849
color of  826, 833, 853
color of, bar charts  936
color of, block charts  900
colors  1015, 1149, 1175, 1194, 1389,
1398
contour plots  1028
fill color  833
frame around axis area  837
labels  347, 356
line type  359
logarithmic  349, 469, 849, 933, 1141
major tick mark values  845, 857
midpoint axis  840
modifying, radar chartsaxi  1272
offset  353
origins  356
plots with two vertical axes  1137, 1205,
1211
suppressing  843, 856, 923, 1022, 1155,
1184, 1199
suppressing, NOAXES option for  1392,
1400
suppressing, NOAXIS option for  1392,
1400
surface and scatter plots  1386
tick marks  351
tick marks, ordering  465
AXIS  367
AXIS definitions
displaying values of  1125
AXIS option
PROC GOPTIONS statement  1127
AXIS statement  43, 44, 345, 367
ActiveX and Java support for  1442
assigning AXIS definitions  369
logarithmic axes  469
ordering datetime tick marks  465
text description suboptions  368
using  368
AXIS= option
BAR statement (G BARLINE) 845
HBAR, HBAR3D, VBAR, VBAR3D
statements (GCHART) 907, 927
PLOT statement (G BARLINE) 857
AXIS1= option
GRID statement (G3GRID) 1425
AXIS2= option
GRID statement (G3GRID) 1425
B
B= option
BASELINE= option, PROC GFONT
statement  1077
background color
graphics output area  526
image transparency  617
legends  379
text in graphics output  450
background images  333, 576, 577
backplane images  334
bar  671
bar charts  873
See also area bar charts
See also bar-line charts
3-D plane  352
adding images  936
drill-down functionality in  204
group brackets on axis  352
horizontal  873, 932
horizontal, error bars in  989
horizontal, midpoints and statistics in
985
images on bars of  336
links in  931
ordering and selecting midpoints  935
outlines  935
patterns  935
patterns, outlines, and colors  398
specifying sum statistic  980
subgroup labels  661
subgrouping a 3-D vertical chart  982
terminology  877
vertical  873, 932
with web drill-down (example)  506
BAR function, Annotate facility 671, 1485
BAR statement
GBARLINE procedure 829
BAR statement, GBARLINE procedure 848, 1454
chart statistic and response axis 849
displaying statistics 849
logarithmic axes 849
options 832
ordering and selecting midpoints 850
required arguments 832
syntax 832
bar-line charts 816
ActiveX and Java support for 1454
adding images to 826
axes 818
creating 861
displaying statistics in 849
links in 847
parts of 817
plot overlays 860
vertical 848
bars
order of 832, 835
outline width 848
subgrouping 846, 866
width of 848
Base SAS language statements 46
baseline 1068
baseline, text in graphics output 449, 456
rotating characters from 459
underlining 461
BASELINE= option
FLOW, TILE, TOGGLE statements (GTILE) 1356
PROC GFONT statement 1077
basic mode
GKPI procedure 1109
BC= option
TITLE, FOOTNOTE, and NOTE statements 450
BCOLOR= option
BUBBLE statement (GPLOT) 1147
BUBBLE2 statement (GPLOT) 1160
TITLE, FOOTNOTE, and NOTE statements 450
BEFORE argument
MOVE statement (GREPLAY) 1299
BFILL= option
BUBBLE statement (GPLOT) 1147
BUBBLE2 statement (GPLOT) 1161
BFONT= option
BUBBLE statement (GPLOT) 1147
GKPI procedure 1112
BFONTR= option
BUBBLE2 statement (GPLOT) 1161
BINDING 526
BINDING= option 525, 526
Bitstream fonts, rendering 554, 606, 1493
spacing between letters 556
bivariate interpolation 1420
BL= option
TITLE, FOOTNOTE, and NOTE statements 451
BLABEL option
BUBBLE statement (GPLOT) 1148
BUBBLE2 statement (GPLOT) 1161
black and white, reversing 612
BLANK= option, TITLE, FOOTNOTE, and NOTE statements 451
block charts 872
creating 899
grouping and subgrouping 977
links in 899
negative or zero values 901
patterns, outlines, and colors 900
sum statistic in 975
text 901
block effects for legends 378, 395
BLOCK statement
GCHART procedure 888, 899
BLOCK statement (GCHART)
URL= option 899
BLOCK statement, GCHART procedure
ActiveX and Java support for 1457
BLOCK statement, GMAP procedure
ActiveX and Java support for 1467
BLOCKMAX= option
BLOCK statement (GCHART) 891
BLOCKMAX= option, BLOCK statement 373
BO= option
TITLE, FOOTNOTE, and NOTE statements 451
bookmarks
PDF graphs 244, 246
BORDER 526
BORDER option 526
BORDER option, GSLIDE procedure 1337, 1339
BORDER= graphics option 526
borders
Annotate facility to draw 678
click through 526, 1339
legends 379
boundary values
GKPI procedure 1103
BOUNDS= argument
GKPI procedure 1111
BOX function
BAR function, Annotate facility 671
Index

box plots 421
creating and modifying (example) 476
line width 436
width 414
BOX= option, TITLE, FOOTNOTE, and NOTE statements 451
boxes around graphics output text 451
brackets, bar charts 352
Browse mode
  GDEVICE procedure 1049
BROWSE option
  PROC GDEVICE statement 1049
browser permissions 267
browsers
  supporting SVG 149
BS= option
  TITLE, FOOTNOTE, and NOTE statements 452
BSCALE= option
  BUBBLE statement (GPILOT) 1148
  BUBBLE2 statement (GPILOT) 1162
BSIZING= option
  BUBBLE statement (GPILOT) 1148
  BUBBLE2 statement (GPILOT) 1162
BSPACE= option, TITLE, FOOTNOTE, and NOTE statements 452
bubble plots 1136
  adding right vertical axis (example) 1211
  controlling bubble display 1157
  creating 0
  generating simple bubble plots (example) 1206
  labeling and sizing plot bubbles (example) 1208
  links in 1155
BUBBLE statement
  GPLOT procedure 1143
BUBBLE statement (GPILOT)
  URL=option 1155
BUBBLE statement, GPLOT procedure
  ActiveX and Java support for 1471
  adding right vertical axis (example) 1211
  coordinating with BUBBLE statements 1167
  options 1166
  required arguments 1160
  syntax 0
  bullet graph KPI charts 1100
  gray scale 1118
  BVALUE option
    GKPI procedure 1113
  BWIDTH= option, SYMBOL statement 414
BY 371
BY line values
  substituting in text strings 369, 395, 465, 959
BY lines 372
BY statement 370, 371
  Annotate facility with 653
  color of BY lines 527
  fonts FOR BY lines 550
  generating chart series (example) 488
  height of BY lines 571
  RUN-group processing with 68
  using 372
BY-group processing
  creating animated GIF images with 176
  creating multiple-page PDF files 255
BYLINE option
  PROC GREPLAY statement 1287
BYLINE statement
  GREPLAY procedure 1291

C
C=
  CTEXT= procedure option 1149, 1162, 1177, 1196
C= option
  AXIS statement 346
  AXIS statement options 362, 366
  CAPLINE= option, PROC GFONT statement 1077
  LEGEND statement options 389
  PATTERN statement options 398
  SYMBOL statement 415, 441
  SYMBOL statement, POINTLABEL= option 428
  TITLE, FOOTNOTE, and NOTE statements 453
CA=
CAXIS= procedure option 1149, 1162, 1175, 1194, 1251
calendar charts 1276
CALENDAR option
CHART statement (GRADAR) 1251
capline 1068
CAPLINE= option
PROC GFONT statement 1077
carriage return at record ends 559
Cartographic font 1500
catalog entries 1281
BY line 372
copying or duplicating 1307
description of 835
duplicate entry names 1283
managing 1315
replaying 1315
CATALOG= option
PROC GDEVICE statement 1049
catalogs
adding device entries to 1050
deleting device entries from 1052
device catalogs 1046
managing 1314
CAUTOHREF= option
BUBBLE statement (GPLOT) 1148
PLOT statement (GCONTOUR) 1015
PLOT statement (GPLOT) 1175
CAUTOREF= option
BAR statement (GBARLINE) 833
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 907
PLOT statement (GBARLINE) 853
CAUTOVREF= option
BUBBLE statement (GPLOT) 1148
BUBBLE2 statement (GPLOT) 1162
PLOT statement (GCONTOUR) 1015
PLOT statement (GPLOT) 1175
PLOT2 statement (GPLOT) 1194
CAXIS= option
BAR statement (GBARLINE) 833
BLOCK statement (GCHART) 891
BUBBLE statement (GPLOT) 1149
BUBBLE2 statement (GPLOT) 1162
CHART statement (GRADAR) 1251
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 907
PLOT statement (G3D) 1389
PLOT statement (GBARLINE) 853
PLOT statement (GCONTOUR) 1015
PLOT statement (GPLOT) 1175
PLOT2 statement (GPLOT) 1194
SCATTER statement (G3D) 1398
CB=
CBASELINE= procedure option 1175, 1195
CBACK 526
CBACK= option 526
CBASELINE= option
PLOT statement (GPLOT) 1175
PLOT2 statement (GPLOT) 1195
CBORDER= option, LEGEND statement
378, 395
CBORDER variable, Annotate facility 707
CBORDER= option, LEGEND statement options 379
CBOTTOM= option
PLOT statement (G3D) 1389
CBOX variable, Annotate facility 708
CBY 527
CBY= graphics option 527
CC argument
? statement (GREPLAY) 1290
LIST statement (GREPLAY) 1297
CC statement
GREPLAY procedure 1291
CC= option
PROC GREPLAY statement 1287
CCOPY statement
GREPLAY procedure 1291
CDEF statement
GREPLAY procedure 1292
CDEFAULT= option
FLOW, TILE, TOGGLE statements (GTILE) 1356
CDELETE statement
GREPLAY procedure 1293
CELL option 527
CELL= option 527
cells
changing size of 73
in device display area 72
in graphics output area 72
CENTIMETERS option
PROC GOPTIONS statement 1128
CERROR= option
BAR statement (GBARLINE) 833
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 908
CFILL= option
PIE, PIE3D, DONUT statements (GCHART) 940
STAR statement (GCHART) 963
CFRAME= option
BAR statement (GBARLINE) 833
BUBBLE statement (GPLOT) 1149
BUBBLE2 statement (GPLOT) 1162
CHART statement (GRADAR) 1252
GSLIDE procedure 1337, 1340
HBAR, HBAR3D, VBAR, VBAR3D statements 799
HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART) 908
  LEGEND statement options 379
PLOT statement (GCONTOUR) 1016
PLOT statement (GPLOT) 1175
PLOT2 statement (GPLOT) 1195
CFRAMESIDe= option
  CHART statement (GRADAR) 1252
CFRAME= option
  CHART statement (GRADAR) 1252
Cfreq option
  BAR statement (GBARLINE) 833
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART) 908
PLOT statement (GBARLINE) 853
CFREQLabel= option
  HBAR, HBAR3D statements
  (GCHART) 909
CGROUPREF option
  HBAR, VBAR statements (GCHART) 909
CH= CHREF= procedure option 1016, 1149,
  1176
  character cells 72
    alignment 527
    size of 525
  character chart variables 880
  character codes 304
    displaying 1093
  character midpoints 819
  characters
    as axis values 353
    as legend values 383
    HTML entities 267
    prefixing output records 567
    prompts 602
    special plot symbols 432
CHARACTERS option 528
  CHARACTERS= option 528
  CHARREC 529
  CHARREC= option, GDEVICE
    procedure 529
  CHARSPACETYPE= option
    PROC GFONT statement 1077
  chart descriptions
    for web presentations 189
    support in graphics output devices 188
CHART statement
  GRADAR procedure 1247
CHAR= CHREF= option
  PLOT statement (GCONTOUR) 1016
  PLOT statement (GPLOT) 1176
  CI= option
    SYMBOL statement 414
    CI= option, SYMBOL statement 441
CIMPORT procedure 1551
  circle of stars, drawing (example) 664
  circle-drawing capability, device 530
  CIRCLE= option
    CIRCLE= option 530
    classification variables
      multiple, in radar charts 1270
    classification variables, plotting 1135
CLASSPATH environmental variables
  267
CLEVELS= option
  PLOT statement (GCONTOUR) 1016
CLIP option
  TDEF statement (GREPLAY) 1304
  clipped polygons 590, 594
CLIPREF option
  BAR statement (GBARLINE) 834
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART) 909
  PLOT statement (GBARLINE) 853
CLM= option
  BAR statement (GBARLINE) 834
cumulative percentage 822
  frequency 821
  GBARLINE procedure 821
  GCHART procedure 877, 883
  mean 822
  percentage 822
  response axis and 849, 933
  specifying 847
  sum 822
  weighted statistics 822
chart variables
  bar-line charts 819
  character 880
  GAREABAR procedure 796
  GCHART procedure 877, 879
  GTILE procedure 1346
  midpoints and 819
  numeric 805
charts
  See also GCHART procedure
  calendar charts 1276
  wind rose charts 1265, 1275
CHARTYPE 530
  Chartype window (GDEVICE) 1058
  CHARTYPE= graphics option 530, 1506
  CHORO statement, GMAP procedure
    ActiveX and Java support for 1467
  CI= option
    BUBBLE statement (GPLOT) 1149
    PLOT statement (GCONTOUR) 1016
    PLOT statement (GPLOT) 1176
  CLASSPATH environmental variables
    267
CLEVELS= option
  PLOT statement (GCONTOUR) 1016
CLIP option
  TDEF statement (GREPLAY) 1304
clipped polygons 590, 594
CLIPREF option
  BAR statement (GBARLINE) 834
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART) 909
  PLOT statement (GBARLINE) 853
CLM= option
  BAR statement (GBARLINE) 834
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 909
CLOCKWISE option
PIE, PIE3D, DONUT statements (GCHART) 941
closed destinations, ODS 98
closing
GSF (graphics stream file) 551
CMAP 531
CMAP argument
? statement (GREPLAY) 1290
LIST statement (GREPLAY) 1297
CMAP entries 1282
CMAP statement
GREPLAY procedure 1293
CMAP= option
PROC GREPLAY statement 1288
CMAP= option, GREPLAY procedure 531
CMIDPOINTREF option
HBAR, VBAR statements (GCHART) 909
CMYK color codes 320
CNS (SAS Color Naming Scheme) 323
cntl2txt 673
CNTL2TXT function, Annotate facility 673
CO=
COUQUEST= procedure option 799, 834, 891, 909, 941, 964, 1016, 1176, 1195
CO= option
SYMBOL statement 414
CO= option, SYMBOL statement 441
CODELENN= option
PROC GFONT statement 1078
COLLATE 531
COLLATE option 531
collating printed output 531
COLMAJOR option
LEGEND statement options 385
COLOR= option
COLOR= suboption
LABEL= option, DONUT statement (GCHART) 955
Colorormap window (GDEVICE) 1058
COLORRAMP= option
FLOW, TILE, TOGGLE statements (GTILE) 1358, 1367
colors 313
active, number of (plotters) 592
assigning with GTILE procedure 1351
axes 346, 362, 826, 833, 936, 1015, 1149, 1175, 1194
axes, CAXIS= option for 1389, 1398
axis area fill 833
axis area frame 853
axis labels 346, 348, 353, 356, 362
axis tick marks 346, 351, 366
axis values 362
block charts 900
borders 535, 536
bubble plots 1147
BY lines 372, 527
CMYK codes 320
Color Naming System values 323
default, specifying 314, 532
donut chart labels 955
error bars 833
GBARLINE procedure 824
graphics output area 526
gray-scale codes 322
HLS codes 321
HSB codes 322
HSV codes 322
image quality across devices and 75
image transparency 617
KPI ranges 1104
legend label 380
legend text 389
legend values 387
specifying/assigning 531
transporting 1553
color ramp
specifying a custom color ramp 1352
COLOR variable, Annotate facility 709
COLOR= option
AXIS statement options 346, 362, 366
GKPI procedure 1116
LEGEND statement options 389
PATTERN statement 398
SCATTER statement (G3D) 1398
SYMBOL statement 415, 441
SYMBOL statement, POINTLABEL= option 428
TDEF statement (GREPLAY) 1304
TITLE, FOOTNOTE, and NOTE statements 453
COLOR= suboption
LABEL= option, DONUT statement (GCHART) 955
Colorormap window (GDEVICE) 1058
COLORRAMP= option
FLOW, TILE, TOGGLE statements (GTILE) 1358, 1367
colors 313
active, number of (plotters) 592
assigning with GTILE procedure 1351
axes 346, 362, 826, 833, 936, 1015, 1149, 1175, 1194
axes, CAXIS= option for 1389, 1398
axis area fill 833
axis area frame 853
axis labels 346, 348, 353, 356, 362
axis tick marks 346, 351, 366
axis values 362
block charts 900
borders 535, 536
bubble plots 1147
BY lines 372, 527
CMYK codes 320
Color Naming System values 323
default, specifying 314, 532
donut chart labels 955
error bars 833
GBARLINE procedure 824
graphics output area 526
gray-scale codes 322
HLS codes 321
HSB codes 322
HSV codes 322
image quality across devices and 75
image transparency 617
KPI ranges 1104
legend label 380
legend text 389
legend values 387
legends 379
maximum display at once 582
modifying when specified by styles 285
naming schemes 317
outlines 834
patterns 534
pie and donut chart slices 958
plot point labels 428
plot symbols 414, 441, 473, 535
plotting in order of 593
precedence of specifications 281
processing limitations 328
reference lines 833, 834, 853, 1016, 1017, 1149, 1150, 1176, 1177, 1196
reversing black and white 612
RGB codes 318
RGBA codes 319
SAS color names and RGB values 324
specifying in SAS/GRAPH programs 314
text 835, 853
text in graphics output 453
tick marks 853
titles, footnotes, and notes 102, 535, 536
utility macros for 327
COLORS 532
COLORS= graphics option 408, 532
COLORS= option
GKPI procedure 1113
COLORTBL option 533
COLORTYPE 533
COLORTYPE= option, GDEVICE procedure 533
COLORVAR= option
FLOW, TILE, TOGGLE statements
(GTILE) 1360
GTILE procedure 1349
COLS 534
columns in graphics output area 534, 573, 581, 591
columns, legends 378
commands
GDEVICE window commands 1055
comment 675
COMMENT function, Annotate facility 675
comments 539
communications ports, how output is written to 565
compatibility with previous releases appearance of output 298
device drivers 93
compression level
for PDF files 246
confidence intervals 912
for error bars 834, 836
confidence limits 423
continuous numeric midpoints 819
continuous numeric variables 881
CONTINUOUS option
HBAR, HBAR3D, VBAR, VBAR3D statements 799
continuous output stream 557
continuous paper feed 524, 586
continuous variables 880
contour labels, size of 416
contour lines
colors for 414, 441
distance between labels 431
fonts 416
labeling 1033
size of 436
type of 427, 442
contour plots 1007
ActiveX and Java support for 1465
axis order 1028
contour levels 1019, 1026, 1036
labeling contour lines 1033
modifying contour lines and labels with SYMBOL statement 1030
modifying horizontal axis 1033
modifying legend 1033
PATTERN statement, GCONTOUR procedure 405
patterns 401
patterns and joins 1038
simple 1031
terminology 1008
text for contour labels 1034
control characters, device 619
controlling placement of text
TITLE, FOOTNOTE, and NOTE statements 463
controlling placement of text in titles, footnotes, and notes 458
converting
graphics output 1551, 1554
coordinates and coordinate systems
Annotate facility 646
data-dependent, GSLIDE with 1340
COPY statement
GDEVICE procedure 1051
GREPLAY procedure 1294
COPY= option
TDEF statement (GREPLAY) 1305
copying
catalog entries 1307
numbers of print copies 558
COUTLINE option
HBAR, HBAR3D, VBAR, VBAR3D statements 799, 964
COUTLINE= option
BAR statement (GBARLINE) 834
BLOCK statement (GCHART) 891
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 909
patterns 409
PIE, PIE3D, DONUT statements (GCHART) 941
PIE, PIE3D, DONUT statements (GCHART) 941
PLOT statement (G3D) 1390
PLOT statement (GBARLINE) 834
PLOT statement (GPLOT) 1177
PLOT2 statement (GPLOT) 1196
SCATTER statement (G3D) 1398
STAR statement (GCHART) 964
CTEXT= options, GOPTIONS statement 464, 535
CTILES= option
CHAR statement (GRADAR) 1254
CTOP= option
PLOT statement (G3D) 1390
cumulative frequency statistic 822, 833, 883
cumulative percentage statistic 822, 834, 853, 883
current catalog 1046
curve-drawing capability, device 530
curves, nonlinear 1420
custom color ramp specifying 1352
CV= option
SYMBOL statement 415, 441
CVREF= option
BUBBLE statement (GPLOT) 1150, 1177
BUBBLE2 statement (GPLOT) 1163
PLOT statement (GCONTOUR) 1017
PLOT statement (GPLOT) 1177
PLOT2 statement (GPLOT) 1196
D= option
TITLE, FOOTNOTE, and NOTE statements 453
DASH 537
DASH option 537
DASH= option 537
dashed lines
  hardware-generated 537
  lengths of dashes, scaling 538
DASHLINE 537
DASHLINE= option, GDEVICE procedure 537
DASHSCALE 538
DASHSCALE= graphics option 538
data library for rendered fonts 607
data sets
  font data set 1081
  kern data sets 1089
  space data sets 1091
DATA sets
  Annotate DATA set 44
DATA step
  Annotate data sets 650
data tips
  for web presentations 191
  GBARLINE procedure 866
  support in graphics output devices 188
  SVG devices 159
data tips, custom
  for web presentations 191
data-dependent coordinates with GSLIDE procedure 1340
DATA= argument
  PROC GFONT statement 1073
DATA= option
  PROC G3D statement 1387
  PROC GAREABAR statement 797
  PROC GBARLINE statement 828
  PROC GCHART statement 887
  PROC GCONTOUR statement 1011
  PROC GPLOT statement 1142
  PROC GRADAR statement 1247
  PROC GTILE statement 1355
DATAORDER= option
  BUBBLE statement (GPLOT) 1150
  BUBBLE2 statement (GPLOT) 1163
DATASYS option
  PROC GANNO statement 781, 783
  scaling graphs 783
date-time information
  as axis values, ordering 353
  ordering axis tick marks (example) 465
debug 675
DEBUG function, Annotate facility 675
debugging
  Annotate facility 654
DEF option
  TDEF statement (GREPLAY) 1305
default colors
  GBARLINE procedure 824
default fonts 301
default lines
  GBARLINE procedure 824
default outlines
  GBARLINE procedure 824
default patterns
  GBARLINE procedure 824
default symbols
  GBARLINE procedure 824
DEL option
  TDEF statement (GREPLAY) 1305
DELAY 538
  delay between displayed graphs 538, 569
  DELAY= option 538
DELETE option
  TDEF statement (GREPLAY) 1305
DELETE statement
  GDEVICE procedure 1052
  GREPLAY procedure 1294
deleting
  graphics output, after display 545, 548
  polygon overlap 590, 594
DES= option
  CDEF statement (GREPLAY) 1293
  GSLIDE procedure 1337
  TDEF statement (GREPLAY) 1305
  TREPLAY statement (GREPLAY) 1308
DESCENDING option
  BAR statement (GBARLINE) 835
  BY statement 370
  HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 911
  PIE, PIE3D, DONUT statements (GCHART) 942
  PLOT statement (GBARLINE) 854
  STAR statement (GCHART) 965
DESCRIPTION 539
DESCRIPTION= option
  BAR statement (GBARLINE) 835
  BLOCK statement (GCHART) 892
  BUBBLE statement (GPLOT) 1150
  CHART statement (GRADAR) 1255
  FLOW, TILE, TOGGLE statements (GTILE) 1361
  GDEVICE procedure 539
  GKPI procedure 1113
  GSLIDE procedure 1337
  HBAR, HBAR3D, VBAR, VBAR3D statements (GAREABAR) 800
  HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 911
  PIE, PIE3D, DONUT statements (GCHART) 942
  PLOT statement (G3D) 1390
PLOT statement (GCONTOUR) 1017
PLOT statement (G PLOT) 1177
PROC GANNO statement 781
SCATTER statement (G3D) 1399
STAR statement (GCHART) 965
destinations 50
  controlling graphics output format 61
destinations, ODS 97
detail pie charts 875
  creating 1004
Detail window (GDEVICE) 1056
DETAIL_PERCENT= option
  PIE, DONUT statements (GCHART) 943
DETAIL_RADIUS= option
  PIE, DONUT statements (GCHART) 943
DETAIL_SLICE= option
  PIE, DONUT statements (GCHART) 943
DETAIL_THRESHOLD= option
  PIE, DONUT statements (GCHART) 943
DETAIL_VALUE= option
  PIE, DONUT statements (GCHART) 943
DETAIL= option
  PIE, DONUT statements (GCHART) 943
DETAILLEVEL= option
  FLOW, TILE, TOGGLE statements (GTILE) 1361, 1367
DEVADDR 539
DEVADDR= option
  GOPTIONS statement 539
DEVICE 540, 629
DEVICE argument
  ? statement (GREPLAY) 1290
device catalogs 1046
  current catalog 1046
  search order of 1047
device display area 70
  cells 72
  dimensions 70
  image quality across devices 75
  resolution 72
  size 72
  sizing errors 77
  units 72
device drivers 80
  color list 315
  comparisons between 156
  compatibility 93
  Z device drivers 93
device entries 80, 1046
  adding to catalogs 1050
changing values in 1053
  copying 1051
  creating or modifying 1060
  creating with program statements 1062
  deleting from current catalog 1052
  listing parameters of 1052
  modifying parameters 92
  parameters versus style attributes 274
  renaming 1054
  transporting 1554
  viewing and modifying 91
  viewing contents of 92
device maps
  specifying 541
device parameters 518
  complete list of, alphabetical 518
DEVICE statement
  GREPLAY procedure 1295
device-generated graphics
  circles and arcs 531
  dashed lines 537
  line thickness 581
  pie filling 593
  plot symbols 614
  polygon-fill 595
  rectangle-fill 605
  vertices, maximum drawn 583
device-resident fonts 300, 1505
  alternative 1508
  default 1506
DEVICE= graphics option 540
  controlling graphics output format 61
  specifying 62
  SVG, PNG, GIF, and TIFF graphs 153
DEVICE= system option 629
devices 79, 80
  ActiveX 87
  appearance differences among graphs 268
  capabilities of, listing 541
  categories of 87
  commonly used 80
  controlling graphics output format 61
  creating 93
  default 62
  defaults for ODS destinations 82
  generating reports for 84
  how output is written to 565
  identifying type of 544
  image quality across devices 75
  interface devices 88
  introduction to 49
  Java 87
  location of, for output 539
  model numbers 583
  modifying default output attributes 87
overriding 86
related topics 94
Scalable Vector Graphics 141
selecting 85
sending strings to 560, 561
specifying type of 618
specifying with multiple open destinations 64
SVG output 142
traditional SAS/GRAPH 87
Universal Printer shortcut 87, 89
user input, enabling 620
viewing list of all available devices 83
DEVMAP 541
DEVMAP= graphics option 541
DEVMAP= option, GDEVICE procedure 541
DEVOPTS 541
DEVOPTS= 541
DEVOPTS= option, GDEVICE procedure 541
DEVTYPE 544
DEVTYPE= option, GDEVICE procedure 544
diagnostic messages, Annotate facility 656
dial KPI charts 1101, 1119
DIAL statement
  GKPI procedure 1111
DIRECTORY window 1311
DIRECTORY window (GDEVICE) 1056
discrete numeric chart variables 836
discrete numeric variables 881
  in star charts 1001
DISCRETE option
  BAR statement (GBARLINE) 836
  BLOCK statement (GCHART) 892
  HBAR, HBAR3D, VBAR, VBAR3D statements 800
  HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 912
  PIE, PIE3D, DONUT statements (GCHART) 943
  STAR statement (GCHART) 965
discrete variables 880
DISPLAY 545
DISPLAY option 545
display size (lines) 567
DISPOSAL 545
DISPOSAL= option 545
document files 106
document formats
  versus graphics formats 233
DOCUMENT procedure
  replaying output 128
donut charts 875
creating 956
labels 954
outlines 958
selecting and positioning slice labels 956
slice patterns and colors 958
statistic and group headings 950, 959
subgrouping 994
text description suboptions 954
DONUT statement
  GCHART procedure 936, 956
  DONUT statement, GCHART procedure
    ActiveX and Java support for 1457
    DONUTPCT= option
    DONUT statement (GCHART) 943
  DOWN
    DOWNVAR= procedure option 1255
    DOWN= option
    LEGEND statement options 379
    PIE, PIE3D, DONUT statements
      (GCHART) 943
    STAR statement (GCHART) 966
    DOWNVAR= option
    CHART statement (GRADAR) 1255
draw 675
DRAW function, Annotate facility 675, 1485
DRAW= option, TITLE, FOOTNOTE, and NOTE statements 453
DRAW2TXT function, Annotate facility 677, 1486
drawing areas, Annotate graphics 646
drill-down
  ACTIVEX output with 198
  ACTXIMG and JAVAIMG output with 198
  adding to PDF documents 245, 251, 255
  adding to web presentations 159
  animated GIF presentations with 198
  GIF output with 163
  JAVA output with 198
  link enhancement variables 193
  Metaview Applet presentations with 198
  support in graphics output devices 188
  SVG device drivers, additional enhancements available with 198
  SVG, PNG, and GIF output with 197
drill-down functionality
  Annotate facility for 202
  bar charts with (example) 204, 506
  creating plots with (example) 1235
  drill-down links
  SVG devices 159
  drill-down URLs 866
drill–downs
  Annotate facility for 201
  driver modules 583
driver termination 547
drivers, initializing 546
drop shadows, legends 379, 395
DROPCOLLISIONS option
  SYMBOL statement 429
DRVINIT 546
DRVINIT1 546
DRVINIT1= and DRVINIT2= options,
  GDEVICE procedure 546
DRVINIT2= and DRVINIT2= options,
  GOPTIONS statement 546
DRVINIT2 546
DRVQRY 546
DRVQRY= option, GDEVICE procedure,
  executing before driver initialization 546
DRVTERM 547
DRVTERM1 547
DRVTERM1= and DRVTERM2= options, GDEVICE procedure 547
DRVTERM1= and DRVTERM2= options, GOPTIONS statement 547
DRVTERM2 547
DUMP option
  LIST statement (GDEVICE) 1053
DUPLEX option 547
duplex printing 526, 547

E
EBCDIC-to-ASCII translation 618
editable output 557
Electronic font 1501
embedding SVG graphics 144
EMF transparency limitation
  Microsoft Office 238
EMPTY variable, Annotate facility 731
encoding
  Unicode 304
END statement
  GREPLAY procedure 1300
QUIT statement, GREPLAY procedure 1300
enhancement variables
  in web presentations 192
ERASE 548
ERASE= graphics option 548
ERASE= option, GDEVICE procedure 548
erasing
  graphics output, after display 545, 548
error bars
  color of 833
  confidence intervals for 834, 836
  in horizontal bar charts 989
ERRORBAR= option
  BAR statement (GBARLINE) 836
HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART) 912
errors
  sizing errors 77
errors and error messages, Annotate facility 656
examples 35
  sample programs 35
  examples on the web 37
executable driver modules 583
EXPLODE= option
  PIE, PIE3D, DONUT statements
    (GCHART) 943
exporting graphics output 133
exporting graphs
  color depth 234
  comparison of output 235
  editing graphs 235
  EMF transparency limitation 238
  enhancing graphs 238
  graphics formats versus document formats 233
  image resolution and size 234
  multiple-image graphics files 235
to Microsoft Office 233
  vector versus raster formats 235
EXTENSION 549
EXTENSION= graphics option 549
external files
  file extensions for 549

F
F option
  FILLED option, PROC GFONT statement 1078
  FOOTNOTE option, PROC GOPTIONS statement 1128
F= option
  AXIS statement options 363
  LEGEND statement options 389
  SYMBOL statement 416
  SYMBOL statement, POINTLABEL= option 429
  TITLE, FOOTNOTE, and NOTE statements 454
FACHE= graphics option 550
FASTTEXT option 549
FASTTEXT= graphics option 549
FBY 550
FBY= graphics option 550
FCACHE 550
file extensions 549
  graphics output files 112
FILECLOSE 551
FILECLOSE= graphics option 551
FILECLOSE= option, GDEVICE procedure 551
FILENAME 573
filename indexing 119
FILENAME statement 46, 47
  storing in device entry 573
filenames
  output 123
FILEONLY 552
FILEONLY graphics option 552
files
  image file types 331
  sending output to 54
  sending strings to 560, 561
  storing graphics output as 552
filling images 334
FILL 552
FILL= graphics option 552
FILL= option
  PIE, PIE3D, DONUT statements (GCHART) 944
  STAR statement (GCHART) 966
FILL= option, GDEVICE procedure 552
filled fonts 1069
FILLED option
  PROC GFONT statement 1078
FILLINC 553
FILLINC= graphics option 553
FILLINC= option, GDEVICE procedure 553
fixed-length output records 557
flow control, device 570
FLOW statement
  GTILE procedure 1355, 1365
font data sets 1081
  creating 1089
  variables 1082
font maximum 1068
font minimum 1068
font modifiers 303
FONT NAME 553
FONT= option 389
AXIS statement options 363
CHART statement (GRADAR) 1255
GKPI procedure 1116
SYMBOL statement 416
SYMBOL statement, POINTLABEL= option 429
TITLE, FOOTNOTE, and NOTE statements 454
FONT= suboption
LABEL= option, DONUT statement (GCHART) 955
FONTRES 554
FONTRES= graphics option 554
fonts 299, 1068
  additional 300
  axis labels 348, 356, 363
  axis values 363
  baseline 1068
  bubble plots 1147
  BY lines 372, 550
capline 1068
changing specifications used by styles 310
complete list of 1494
creating 1068, 1073, 1077, 1081
creating figures for symbol font 1094
default 301, 528, 530, 1506
determining available fonts 300
device-resident fonts 300
displaying 1068, 1073, 1074
displaying with character codes 1093
donut chart labels 955
filled 1069
full names for 1507
GFONT procedure 1067
GKPI procedure 1116
graphics output text 555
international characters 304
kern data sets 1089
legend label 380
legend text 389
legend values 387
line segments 1069
methods for specifying 308
modifying when specified by styles 285
open at one time 550
outline 1069
parts of 1068
PDF files 242
plot point label 429
plot symbols 416
polygon 1069
precedence of specifications 310
proportional 1068
registry subkeys 303
rendering 549, 556, 606, 613, 1493
rendering, data library for 607
resolution 554
SAS/GRAPH fonts 300
scaling in graphics output 534, 611
space data sets 1091
special characters 304
specifying 303
specifying in GraphFonts style element 289
specifying modifiers 303
specifying with global statement options 309
specifying with GOPTIONS statement 309
stroked 1069
system fonts 300
terminology 1068
text in graphics output 454
titles and footnotes 102, 556
transporting 1553
troubleshooting 268
TrueType 300
TrueType fonts supplied by SAS 300
types of 300, 1068
uniform 1068
user-created 1069
viewing specifications in SAS registry 302
where stored 1494, 1503
FOOTNOTE definitions displaying values of 1125
FOOTNOTE option PROC GOPTIONS statement 1128
FOOTNOTE statement 44, 375, 447, 462
ActiveX and Java support for 1452
BY statement with 374
displaying with GOPTIONS procedure 1129
footnotes 462
angle of rotation 449, 456, 459
boxes around 451
colors for 453, 535, 536
default characteristics, setting 463
defining text of 460
fonts for 454
hyperlinks for 448, 458
justification 455
ODS output 101, 102
placement in graphics output area 76
positioning 458
size of 454, 575
spacing around 458
text breaks 455
underlining 461
foreground colors default, defining 532
reversing black and white 612
FORMAT 555
FORMAT statement 46
FORMAT= option GKPI procedure 1113
FORMAT= option, GDEVICE procedure 555
FORMAT= suboption
REFLABEL= option, AXIS statement options 363
formats
axis labels 363
axis values 363
formatting
axis labels 348, 353, 356, 361
axis tick marks 351, 365
axis values 361
BY lines 372
legend label 380
legend text 389
legend values 387, 391
legends 376
FRAME function, Annotate facility 678, 1486
FRAME option
BAR statement (GBARLINE) 837
BUBBLE statement (GPLOT) 1151
BUBBLE2 statement (GPLOT) 1163
CHART statement (GRADAR) 1256
GSLIDE procedure 1338
HBAR, HBAR3D, VBAR, VBAR3D statements 800
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 913
PLOT statement (GBARLINE) 854
FREQ option
BAR statement (GBARLINE) 837
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 914
PLOT statement (GCHART) 914
PLOT2 statement (GPLOT) 1197
frame, legend 380
FRAME= option, LEGEND statement options 380
frames around axis area 837
backplane images 334
images on 334, 917, 1153, 1181, 1338
frames, drawing 1339
FREQ option
BAR statement (GBARLINE) 837
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 914
PLOT statement (GCHART) 914
FREQ= option
BAR statement (GBARLINE) 838
BLOCK statement (GCHART) 893
CHART statement (GRADAR) 1256
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 914
PIE, PIE3D, DONUT statements (GCHART) 944
PLOT statement (GBARLINE) 854
FREQLABEL= option
HBAR, HBAR3D statements (GCHART) 914
frequency statistic 821, 837, 854, 883
frequency variable
for plot statistic 854
FRONTREF option 1151
BUBBLE2 statement (G PLOT) 1164
H BAR, H BAR3D, V BAR, V BAR3D statements (G CHART) 915
PLOT statement (G PLOT) 1178
PLOT2 statement (G PLOT) 1197
FS option
PROC GREPLAY statement 1288
FS statement
GREPLAY procedure 1295
FTEXT 555
FTEXT= option
OPTIONS statement 464, 555
FTITLE 556
FTITLE= graphics option 464, 556
FTRACK 556
FTRACK= graphics option 556
FUNCTION variable, Annotate facility 645, 710
functions, Annotate 643, 667
FWIDTH= option, LEGEND statement options 380

G
G100 option
BLOCK statement (GCHART) 893
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 915
G3D procedure 1381
ActiveX and Java support for 1482
axes 1386
input data set 1384
input data sets 1384
PLOT statement 1394
PROC G3D statement 1386
rotating and tilting plots 1385
scatter plots 1382
SCATTER statement 1403
surface plots 1381, 1404, 1409, 1410, 1412, 1414
surface plots, rotated 1405
surface plots, tilted 1407
syntax 1386
terminology 1383
G3GRID 1422
G3GRID procedure 1417
controlling observations in output data set 1427
default interpolation method 1428
GRID statement 1422
horizontal variables along nonlinear curve 1420
input data set 1419
interpolation methods 1420
multiple vertical variables 1419
output data set 1420
partial spline interpolation 1434
PROC G3GRID statement 1422
spline and smoothing interpolations 1431
spline interpolation 1435
syntax 1422
GACCESS 557
GACCESS= graphics option 557
GACCESS= option, GDEVICE procedure 557
GANNO 779
syntax 780
GANNO procedure 652, 779
compared with GSLIDE procedure 779
drift–down links, generating 201
overview 779
PROC GANNO statement 780
producing multiple graphs 790
scaling data-dependent output 784
scaling graphs 783
storing Annotate graphics 787
syntax 780
GAREABAR 795
syntax 796
GAREABAR procedure 795
ActiveX and Java support for 1454
area bar charts with numeric chart variable 805
area bar charts with subgroups 808, 810
concepts 796
description 802
examples 803
HBAR and HBAR3D statements 798
overview 795
PROC GAREABAR statement 0
syntax 796
VBAR and VBAR3D statements 798
GAXIS= option
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 915
GBARLINE 816
GBARLINE procedure 816
ActiveX and Java support for 1454
BAR statement 829, 848
calculating weighted statistics 863
chart statistics 821
chart variables 819
concepts 817
creating bar-line charts 861
data tips 866
drift-down URLs 866
midpoints 819
missing values 823
multiple plots 866
patterns, outlines, colors, and images  824
PLOT statement  850, 860
plot variable values out of range  824
PROC GBARLINE statement  827
response variables  821
subgroups  866
syntax  827
GCHART  872
GCHART procedure  872
3-D vertical bar charts, subgrouping  982
ActiveX and Java support for  1457
bar charts  873
bar charts, sum statistic in  980
block charts  872
block charts, grouping and subgrouping  977
block charts, sum statistic in  975
BLOCK statement  888, 899
BY statement  373
BY-group processing with (example)  488
chart statistics  883
chart variables  879
concepts  877
detail pie charts  875, 1004
donut charts  875
donut charts, subgrouping  994
DONUT statement  936, 956
drill-down functionality in bar chart (example)  204
examples  975
HBAR statement  901
HBAR, HBAR3D statements  932
HBAR3D statement  901
horizontal bar charts, error bars in  989
horizontal bar charts, midpoints and
to statistics in  985
midpoints  880
missing values  880
PATTERN statement  405
patterns  400, 885
patterns and outlines  398
pie charts  875
pie charts, grouping and arranging  998
pie charts, ordering and labeling slices  996
pie charts, subgrouping  994
pie charts, sum statistic in  991
PIE statement  936
PIE, PIE3D statements  956
PIE3D statement  936
PROC GCHART statement  887
star charts  876
star charts, discrete numeric variable in  1001
star charts, sum statistic in  999
STAR statement  960, 972
subgroup labels (example)  661
syntax  886
VBAR statement  901
VBAR, VBAR3D statements  932
VBAR3D statement  901
GCLASS= graphics option  558
Gcolors window (GDEVICE)  1057
GCONTOUR  1007
overview  1007
syntax  1010
GCONTOUR procedure  1007
ActiveX and Java support for  1465
contour levels  1026, 1036
contour plots  1008
contour plots, modifying  1033
contour plots, simple  1031
data ranges  1009
input data  1009
interpolating data  1010
missing values  1010
modifying contour lines and labels with
SYMBOL statement  1030
overview  1007
PATTERN statement  405
patterns  401
patterns and joins  1038
PLOT statement  1026
PROC GCONTOUR statement  0
syntax  1010
GCOPIES  558
GCOPIES= graphics option  558
GCOPIES= option, GDEVICE procedure  558
GDEVICE  1046
GDEVICE procedure  1046
ADD statement  1050
Browse mode  1049
COPY statement  1051
creating device entries with program statements  1062
creating or modifying device entries  1060
default device-resident fonts  1507
DELETE statement  1052
device catalogs  1046
exiting  1048
LIST statement  1052
MODIFY statement  1053
PROC GDEVICE statement  1049
program mode  1047, 1048, 1050
RENAME statement  1054
syntax  1049
windowing mode 1047, 1048
GDEVICE windows 519, 1055
commands 1055
GEND 559
Gend window (GDEVICE) 1059
GEND= graphics option 559
GEND= option, GDEVICE procedure 559
GEPILOG 560
Gepilog field, device entries 595, 598
Gepilog window (GDEVICE) 1059
GEPILOG= graphics option 560
GEPILOG= option, GDEVICE procedure 560
Geprolog field, device entries 596
GFONT procedure 1067
creating figures for symbol font 1094
creating fonts 1068, 1081
displaying fonts 1068
displaying fonts and character codes 1093
font terminology and characteristics 1068
options for creating fonts 1077
options for displaying fonts 1074
required arguments for creating fonts 1073
required arguments for displaying fonts 1073
storing user-created fonts 1069
GFONT0 libref 1069
GFOOTNOTE= option, ODS HTML statement 101
GIF device driver
ACTXIMG, JAVAIMG vs. 156
chart descriptions for 190
data tips for 191
developing web presentations 156
drill-down links in output 163, 197
GIF graphics output device
creating animated GIF images with 167, 178, 181
OPTIONS for presentations 171
GIF images
creating animated GIF images with BY-group processing 176
creating animated GIF images with RUN-group processing 178
GKPI 1109
GKPI procedure 1099
actual KPI values 1102
basic, raised, or modern mode 1109
boundary values 1103
bullet graph charts 1100
bullet graph charts, gray scale 1118
default colors as active colors 1117
device for 1100
dial charts 1101, 1119
DIAL statement 1111
display types 1099
examples 1117
fonts 1116
HBULLET statement 1111
HSLIDER statement 1111
HTRAFFICLIGHT statement 1111
PROC GKPI statement 1109
range boundaries 1102
range colors 1104
slider charts 1100
speedometer charts 1101, 1121, 1122
SPEEDOMETER statement 1111
syntax 1108
tick mark values 1103
traffic light charts 1102, 1123
V BULLET statement 1111
VSLIDER statement 1111
VTRAFFICLIGHT statement 1111
global statement options
specifying fonts with 309
global statements 43, 343
RUN-group processing and 67
GMAP procedure
ActiveX and Java support for 1467
GOPTIONS 375, 1125
GOPTIONS procedure 1125
compared with GOPTIONS statement 1125
displaying graphics options without descriptions 1130
displaying TITLE and FOOTNOTE statements 1129
syntax 1127
GOPTIONS statement 44, 375, 518, 545
ActiveX and Java support for 1444
building color lists 316
compared with GOPTIONS procedure 1125
graphics option processing 377
resetting options 608
specifying fonts with 309
using 376
GOUT argument
? statement (GREPLAY) 1290
GOUT statement
GREPLAY procedure 1295
GOUT= option
GSLIDE procedure 1338
PROC G3D statement 1387
PROC GANNO statement 782
PROC GCHART statement 888
PROC GCONTOUR statement 1011
PROC GFONT statement 1074
PROC GLOT statement 1142
PROC GRADAR statement 1247
PROC GREPLAY statement 1288
GOUTMODE 561
GOUTMODE= graphics option 561
GPLOT procedure 405, 1134
ActiveX and Java support for 1471
adding right vertical axis (example) 1211
BUBBLE statement 1143, 0
BUBBLE2 statement 1158, 1167
BY statement 373
connecting plot data points (example) 1217
different scales of values (example) 1231
examples 1206
filling areas in overlay plot (example) 1223
generating overlay plot (example) 1219
generating simple bubble plots (example) 1206
input data set 1140
labeling and sizing plot bubbles (example) 1208
PATTERN definitions 1206
plot basics 1138
PLOT statement 1168, 0
PLOT2 statement 1189, 1203
plots with drill-down for web (example) 1235
plotting three variables (example) 1226
plotting two variables (example) 1214
PROC GLOT statement 1141, 1142
SYMBOL definitions 1189, 1206
SYMBOL statement 439
syntax 1141
GPROLOG 561
Gprolog window (GDEVICE) 1059
GPROLOG= graphics option 561
GPROLOG= option, GDEVICE procedure 561
GPROTOCOL 562
GPROTOCOL= graphics option 562
GPROTOCOL= option, GDEVICE procedure 562
GRADAR 1243
GRADAR procedure 1243
ActiveX and Java support for 1477
calculating weighted statistics 1266
CHART statement 1247
creating calendar charts 1276
creating radar charts 1267
creating wind rose charts 1275
modifying appearance of radar charts 1271
multiple classification variables in radar charts 1270
overlaying radar charts 1268
PROC GRADAR statement 1246
syntax 1246
tiling radar charts 1269
GRAPH window
displaying SAS/GRAPH output in 631
sending output to 53
GraphColors
modifying style elements 286
GraphColors style element 287
GraphFonts
modifying style elements 286
GraphFonts style element 288
font specifications in 289
graphic symbol TrueType fonts 300
graphics catalogs
converting 1554
duplicate entry names 1283
graphics devices
See devices
graphics elements 69, 106
placement in graphics output area 76
graphics files
multiple-image 235
graphics formats 106
versus document formats 233
graphics options 375, 518
complete list of, alphabetical 518
displaying without descriptions 1130
listing 1125, 1127
ODS output with 101
resetting 608
storing multiple graphs in one output file 126
graphics output 69, 106, 268
Annotate data sets 651
Annotate graphics with 652
appending strings to records 559
appending to or replacing catalogs 561
background images 576, 577
catalog name and entry name for GRSEGs 120
comparison of, for Microsoft Office 235
controlling format with DEVICE= option 61
default destinations for 552
destination for 566
device variants to set size of resolution 116
display size, in lines 567
displaying images in 577
erasing after display 545, 548
examples, using different styles 277
Index

exporting 133
GRSEG names 123
GRSEGs 108
how written, specifying 565
name and location of ODS output 117
output filenames 123
output types 107
prefixing records 567
previewing 131
previewing as if on different device 616
printing 131
process of 112
protocol module, specifying 562
queuing for log messages 604
replaying 128
replaying in templates 1317
resolution 114, 116
reversing black and white 612
saving and printing 132
sending directing to printer 132
size of graph 113, 116
storage of 116
supported graphics formats 106
suppressing display of 545
terminology 106
transporting and converting 1551
what you can do with 109
graphics output area 70
Annotate facility 648
border around 526
cells 72
columns in 534, 581, 591
dimensions 70
display area size 72
image quality across devices 75
maximum colors allowed 582
offset between graphs and 572, 620
placement of graphics elements in 76
resolution 72
rows in 582, 604, 611, 621
size of 574, 622, 623, 624, 625, 626
sizing errors 77
units 72
graphics output devices 112
graphics output files and 110
graphics output files 106
file extensions 112
filename indexing 119
name and location of 118
name of 842
ODS and 109
output devices and 110
replacing with GSFMODE= graphics option 125
specifying type of 109
storing multiple graphs in one file 126
graphics output text
colors for 536
fonts 555
size of 575
GRAPHRC 564
GRAPHRC graphics option 564
graphs
See also appearance of graphs
appearance differences among devices 268
background images 576
box plots 417, 421, 476
BY lines 372
displaying in timed series 569
ing editing 235
enhancing for Microsoft Office 238
exporting to Microsoft Office 233
importing into Microsoft Office 239
placement in graphics output area 76
producing multiple (GANNO) 790
redrawing (overdrawing) 607
replaying into templates 1323
scaling 783
suppressing display of 545
writing to PDF files 241
gray-scale color codes 322
greyscale bullet graph KPI charts 1118
GREPLAY 1280
GREPLAY procedure 1280
? statement 1290
BYLINE statement 1291
catalog entries 1281
catalog entries, managing 1315
CC statement 1291
CCOPY statement 1291
CDEF statement 1292
CDELETE statement 1293
CMAP statement 1293
code-based statements 1283
COPY statement 1294
creating animated GIF images 181
creating color maps 1318, 1330
creating multiple-page PDF files 258
creating templates 1316, 1318
DELETE statement 1294
DEVICE statement 1295
FS statement 1295
GOUT statement 1295
GROUP statement 1296
IGOUT statement 1296
invoking 1289
LIST statement 1297
managing catalogs, color maps, and templates 1314
MODIFY statement 1297
MOVE statement 1299
replaying catalog entries
replaying graphics output in templates
replaying graphs into templates
replaying GSLIDE procedure output in a template
replaying output
running with code-based statements
sizing and naming graphs for replay
storing multiple graphs in one output file
syntax
template code
ways to use
window commands
windowing environment
windows
GREPLAY procedure windows using
grid of values
See G3GRID procedure
GRID option
BBUBBLE statement (GPLOT) 1151
BBUBBLE2 statement (GPLOT) 1164
PLOT statement (G3D) 1391
PLOT statement (GCONTOUR) 1018
PLOT statement (GPLOT) 1178
PLOT2 statement (GPLOT) 1197
SCATTER statement (G3D) 1399
GRID statement
G3GRID procedure 1422, 1423
group brackets, bar charts 352
group headings
pie and donut charts 950, 959
star charts 970, 974
GROUP statement
GREPLAY procedure 1296
GROUP variable, Annotate facility 711
GROUP= option
BLOCK statement (GCHART) 893
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 915
PIE, PIE3D, DONUT statements (GCHART) 945
STAR statement (GCHART) 967
grouping
block charts 977
pie charts 998
grouping abbreviations 371
GROUPREF option
HBAR, VBAR statements (GCHART) 916
GRSEG catalog entries 1281
name of 842
ODS styles in graphs stored as 630
GRSEGs 108
catalog name and entry name for 120
names for 123
storage with multiple ODs destinations 122
GSF (graphics stream file)
closing 551
how output is written to 565
output format and destination 557
prompt messages to 567
record length 564
where written, specifying 566
GSFLEN 564
GSFLEN= graphics option 564
GSFMODE 565
GSFMODE= graphics option 565
replacing graphics output files 125
GSFMODE= option, GDEVICE procedure 565
GSFNAME 566
GSFNAME= graphics option 566
GSFNAME= option, GDEVICE procedure 566
GSFPROMPT 567
GSFPROMPT graphics option 567
GSPACE= option 567
GSIZE 567
GSPACE= graphics option 567
GSLIDE 1336
GSLIDE procedure 652
Annotate graphics, displaying 1334, 1342
compared with GANNO procedure 779
data-dependent coordinates 1340
producing text slides (example) 1340
replaying output in a template 1321
syntax and options 1336
GSPACE= option
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 916
GSTART 567
Gstart window (GDEVICE) 1059
GSTART= graphics option 567
GSTART= option, GDEVICE procedure 567
GSTYLE system option 274, 630
GTILE 1353
GTILE procedure
assigning colors 1351
chart variables 1346
creating tile charts 1365
FLOW statement 1365
missing values 1350
negative values 1350
PROC GTILE statement 1355
syntax 1353
TILE statement 1365
TOGGLE statement 1365
zero values 1350
GTITLE= option, ODS HTML statement 101
GUNIT 568
GUNIT= graphics option 568
GWAIT 569
GWAIT= graphics option 569
GWINDOW system option 631

H
H= option
AXIS statement options 363, 366
HEIGHT= option, PROC GFONT statement 1074
LEGEND statement options 389
SYMBOL statement 417
SYMBOL statement, POINTLABEL= option 429
TITLE, FOOTNOTE, and NOTE statements 454
HANDSHAKE 570
HANDSHAKE= graphics option 570
HANDSHAKE= option, GDEVICE procedure 570
handshaking 570, 584
hardware fonts
default 528, 530
device map, specifying 541
scaling in graphics output 534, 611
specifying for device 529, 553
when not found 611
hardware-generated graphics
circles and arcs 531
dashed lines 537
line thickness 581
pie filling 593
plot symbols 614
polygon-fill 595
rectangle-fill 552, 605
vertices, maximum drawn 583
HAXIS= option
BUBBLE statement (G PLOT) 1151
PLOT statement (GCONTOUR) 1018
PLOT statement (G PLOT) 1178
HBAR and HBAR3D statements
GAREABAR procedure 798
GCHART procedure 1457
URL= option 931
HBAR statement
GCHART procedure 901, 932
HBAR3D statement
GCHART procedure 901, 932
HBULLET statement
GKPI procedure 1111
HBY 571
HBY= graphics option 571
HEADER 571
HEADER records 571
HEADER= option, GDEVICE procedure 571
HEADERFILE 572
HEADERFILE= option, GDEVICE procedure 572
HEIGHT= option
AXIS statement 363, 366
CHART statement (GRADAR) 1256
GKPI procedure 1116
LEGEND statement options 389
PROC GFONT statement 1074
SYMBOL statement 417
SYMBOL statement, POINTLABEL= option 429
TITLE, FOOTNOTE, and NOTE statements 454
HEIGHT= suboption
LABEL= option, DONUT statement (G CHART) 955
HEX option
ROMHEX option, PROC GFONT statement 1076, 1080
hexadecimal values
specifying special characters 304
high-low plots 419
HITEXT= graphics option 464
HLABEL=
HEIGHT= procedure option 1256
HLS color codes 321
HM=
HMINOR= procedure option 1018, 1152, 1179
HMINOR= option
BUBBLE statement (G PLOT) 1152
PLOT statement (GCONTOUR) 1018
PLOT statement (G PLOT) 1179
HORIGIN 572
HORIGIN= graphics option 572
HORIGIN= option, GDEVICE procedure 572

horizontal area bar charts
  GAREABAR procedure 798

horizontal axis
  contour plots 1033

horizontal bar charts 873
  creating 932
  error bars in 989
  midpoints and statistics in 985
  statistics in 933
  horizontal variables
    along nonlinear curve 1420
    G3GRID procedure 1420

Host Commands window (GDEVICE) 1060

host commands, executing
  after driver initialization 547
  after graph production 596
  before graph production 599

Host File Options window (GDEVICE) 1060

HOSTSPEC 573

HOSTSPEC= option, GDEVICE procedure 573

HPOS 573

HPOS= graphics option 573

HREF attribute 513

HREF= option
  BUBBLE statement (G PLOT) 1152
  PLOT statement (GCONTOUR) 1018
  PLOT statement (GPLOT) 1180

HREVERSE option
  BUBBLE statement (GPLOT) 1152
  PLOT statement (GCONTOUR) 1018
  PLOT statement (GPLOT) 1180

HSB color codes 322

HSIZE 574

HSIZE= graphics option 574
  setting size of graphics area 113

HSIZE= option, GDEVICE procedure 574

HSLIDER statement
  GKPI procedure 1111

HSV color codes 322

HSYS variable, Annotate facility 714

HTEXT 575

HTEXT= graphics option 575

HTITLE 575

HTITLE= graphics option 464, 575

HTML character entities 267

HTML destination, ODS 97

HTML pages
  bar chart with drill-down (example) 506

  combining graphs and reports (example) 495

  HTML variable, Annotate facility 715

  HTML_LEGEND= option
    BAR statement (GBARLINE) 838
    BLOCK statement (GCHART) 894
    CHART statement (GRADAR) 1256
    drop-down links 192
    HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 917
    PIE, PIE3D, DONUT statements (GCHART) 945
    PLOT statement (GBARLINE) 855
    PLOT statement (GPLOT) 1180
    PLOT2 statement (GPLOT) 1198
    STAR statement (GCHART) 967

  HTML= option
    adding data tips 191
    adding pop-up images to SVG documents 198
    BAR statement (GBARLINE) 838
    BLOCK statement (GCHART) 893
    BUBBLE statement (GPLOT) 1152
    BUBBLE2 statement (GPLOT) 1164
    drop-down links 192
    GCHART procedure 513
    HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 916
    ONCLICK= HTML attribute, using with SVG device 200
    ONMOUSEOVER= HTML attribute, using with SVG device 199
    PIE, PIE3D, DONUT statements (GCHART) 945
    PLOT statement (GBARLINE) 855
    PLOT statement (GPLOT) 1180
    PLOT2 statement (GPLOT) 1197
    STAR statement (GCHART) 967
    HTRAFFICLIGHT statement
      GKPI procedure 1111

  hyperlinks
    titles and footnotes as 448, 458

HzERO option
  BUBBLE statement (GPLOT) 1152
  PLOT statement (GPLOT) 1181

I

I= option
  SYMBOL statement 417, 419, 420, 421, 422, 423, 424, 425, 427

IBACK 576

IBACK= graphics option 333, 576

ID 577

ID statement
  GMAP procedure 1467
Index

ID= option, GDEVICE procedure 577
IFRAME= option 334
BUBBLE statement (GPLOT) 1153
CHART statement (GRADAR) 1257
GSLIDE procedure 1338
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 917
PLOT statement (GPLOT) 1181
IGOUT argument
? statement (GREPLAY) 1290
LIST statement (GREPLAY) 1297
IGOUT statement
GREPLAY procedure 1296
IGOUT= option
PROC GREPLAY statement 1288
image files 106
IMAGE function, Annotate facility 679, 1487
image maps 513
image quality across devices 75
image resolution and size exporting graphs to Microsoft Office 234
IMAGE= option, PATTERN statement 336, 399
IMAGEMAP= option
PROC GANNO statement 782
PROC GBARLINE statement 828
PROC GCHART statement 888
PROC GPLOT statement 1142
PROC GREPLAY 1288
PROC GSLIDE statement 1338
IMAGEPRINT 577
IMAGEPRINT GOPTIONS statement 577
images 331
adding to bar charts 936
adding to bar-line charts 826
Annotate facility to draw 679
as graph background 576, 577
as pattern fills 399
background 333
backplane 334
disabling as output 577
displaying with Annotate facility 338
file types, list of 331
GBARLINE procedure 824
in text slides 1339
interlacing 579
on chart bars 336
transparent 617
images, background SVG graphics 145
IMAGESTYLE 577
IMAGESTYLE= graphics option 334, 577
IMAGESTYLE= option
BUBBLE statement (GPLOT) 1153
CHART statement (GRADAR) 1257
GSLIDE procedure 1338
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 917
PATTERN statement 399
PLOT statement (GPLOT) 1181
IMPATH statement (GREPLAY) 1290
importing graphs into Microsoft Office 239
inactive color lists 1106
INBORDER option
CHART statement (GRADAR) 1257
INCOMPLETE option
PROC GCONTOUR statement 1011
indexing filename 119
INHEIGHT= option
CHART statement (GRADAR) 1257
initializing drivers, executing before 546
input (user), enabling 620
input data sets
G3D procedure 1384
G3GRID procedure 1419
INSIDE= option
BAR statement (GBARLINE) 839
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 917
integer-based font rendering 549
INTERACTIVE 578
interactive Metagraphics output 617
character rotation angle 610
description string 577
hardware text rotation angle 590
interactivity of 578
negative handshaking response 584
TRAILER records 616
translating metafile into device commands 599
user-written part, files for 600
INTERACTIVE= option, GDEVICE procedure 578
INTERCHART= INTERTILE= procedure option 1257
interface devices 88
INTERLACED 579
INTERLACED GDEVICE procedure 579
INTERLACED GOPTIONS statement 579
interlacing images 579
internal coordinates, Annotate facility 648, 706
international characters 304
INTERPOL 579
INTERPOL= graphics option 579
INTERPOL= option
SYMBOL statement 417, 419, 420, 421, 422, 423, 424, 425, 427
interpolation
box plots 417, 476
connecting data points with straight lines 420
data value inclusion 428
default method, specifying 440
default value for 579
high-low plots 419
Lagrange 420
needle plots 422
partial spline 1434
regression analysis 423
regression analysis plots 423
smoothing data 424
smoothing plot lines 420
spline interpolation 425, 440
step plots 427
suppression 422
interpolation methods
bivariate 1420
default method 1428
G3GRID procedure 1420
plot overlays 861
plots 1138
spline 1421, 1435
spline smoothing 1422, 1431
INTERTILE= option
CHART statement (GRADAR) 1257
INTERVAL= option, AXIS statement options 347
INVISIBLE= option
PIE, PIE3D, DONUT statements (GCHART) 946
JUSTIFICATION= option
JOIN option
GRID statement (G3GRID) 1425
PLOT statement (GCONTOUR) 1018
joins
contour plots 1038
JUSTIFY= option
AXIS statement options 364
LEGEND statement options 390
SYMBOL statement, POINTLABEL= option 429
TITLE, FOOTNOTE, and NOTE statements 455
Java applets
authentication 267
CLASSPATH environmental variables 267
JAVA device driver
data tips for 192
drill-down links in output 198
Java devices 87
Java support 1442
Annotate functions 1484
G3D procedure 1482
GAREABAR procedure 1454
GBARLINE procedure 1454
GCHART procedure 1457
GCONTOUR procedure 1465
GMAP procedure 1467
GOPTIONS statement 1444
G PLOT procedure 1471
GRADAR procedure 1477
LEGEND statement 1449
PATTERN statement 1450
SYMBOL statement 1451
TITLE and FOOTNOTE statements 1453
JAVAIMG device driver 112
chart descriptions for 190
data tips for 192
drill-down links in output 198
SVG, PNG, GIF, and TIFF vs. 156
web presentations, developing 157
JAVAMETA device driver
data tips for 191
JOIN option
GRID statement (G3GRID) 1425
PLOT statement (GCONTOUR) 1018
justification
axis labels 348, 353, 356
donut chart labels 955
legend label 380
legend text 390
legend values 387
plot print labels 429
text in graphics output 455
JUSTIFICATION= option
G KPI procedure 1117
JUSTIFY= option
AXIS statement options 364
LEGEND statement options 390
SYMBOL statement, POINTLABEL= option 429
TITLE, FOOTNOTE, and NOTE statements 455
JUSTIFY= suboption
LABEL= option, DONUT statement (GCHART) 955
Kern data sets 1089
  creating 1090
  variables 1089
KERNDATA= option
  PROC GFONT statement 1078
kerning 1089
key performance indicators
  See KPIs
KEYMAP 580
KEYMAP= graphics option 580
KPIs 1099
  actual values 1102
  boundary values 1103
  bullet graph charts 1100, 1118
  chart types 1099
  color lists 1106
  dial charts 1101, 1119
  fonts 1116
  range boundaries 1102
  range colors 1104
  slider charts 1100
  speedometer charts 1101, 1121, 1122
  tick mark values 1103
  traffic light charts 1102, 1123

L
L option
  LEGEND option, PROC GOPTIONS
    statement 1128
L= option
  SYMBOL statement 427
L= option, SYMBOL statement 442
LA= option
  TITLE, FOOTNOTE, and NOTE
    statements 456
LABEL function, Annotate facility 681, 1487
LABEL= option
  AXIS statement 1443
  AXIS statement options 348
  DONUT statement 1464
  DONUT statement (GCHART) 946
  GKPI procedure 1114
  LEGEND statement 1450
  LEGEND statement options 380
LABELLEVEL= option
  FLOW, TILE, TOGGLE statements
    (GTILE) 1363
labels
  animating Annotate labels 683
  axes 348, 356
  bubbles in bubble plots 1147
  BY lines 372
contour lines 431, 1033
contour plots 1030, 1031
donut charts 954
legends 380
pie chart slices 956, 996
plot bubbles (example) 1208
plot points 428
star charts 973
Lagrange, interpolation 420
landscape orientation 581, 582, 609
LANGLE= option, TITLE, FOOTNOTE,
  and NOTE statements 456
language elements
  used by programs 41
Latin TrueType fonts 300
LAUTOHREF= option
  BUBBLE statement (GPLOT) 1153
  PLOT statement (GCONTOUR) 1019
  PLOT statement (GPLOT) 1181
LAUTOVREF= option
  BAR statement (GBARLINE) 839
  HBAR, HBAR3D, VBAR, VBAR3D
    statements (GCHART) 918
  PLOT statement (GBARLINE) 855
LCOLS 581
LCOLS= option, GDEVICE procedure 581
LEFTMARGIN GDEVICE procedure 574
LEFTMARGIN GOPTIONS statement 574
LEGEND 392
LEGEND definitions
  displaying values of 1125
LEGEND option
  PLOT statement (GPLOT) 1182
  PLOT2 statement (GPLOT) 1198
  PROC GOPTIONS statement 1128
LEGEND statement 44, 377, 392, 576
  ActiveX and Java support for
  filling areas in overlay plot (example) 1449
  generating overlay plot (example) 1219
  using 393
LEGEND= option
  BAR statement (GBARLINE) 839
  BLOCK statement (GCHART) 894
  HBAR, HBAR3D, VBAR, VBAR3D
    statements 801
LOGBASE= option, AXIS statement  
349, 469
LOGSTYLE= option, AXIS statement  
350
LOGSTYLE= option, TITLE,  
FOOTNOTE, and NOTE statements  
469
looping animation  579
LOWBOUNDARY option  
GKPI procedure  1115
LREF= option  
BAR statement (GBARLINE)  840  
HBAR, HBAR3D, VBAR, VBAR3D  
statements (GCHART)  920
PLOT statement (GBARLINE)  856
LROWS  582
LROWS= option, GDEVICE procedure  
582
LRX= option  
TDEF statement (GREPLAY)  1305
LRY= option  
TDEF statement (GREPLAY)  1305
LS= option  
TITLE, FOOTNOTE, and NOTE  
statements  458
LSPACE= option, TITLE, FOOTNOTE,  
and NOTE statements  458
text break and  458
LSPKES= option  
CHART statement (GRADAR)  1257
LSTARCIRCLES= option  
CHART statement (GRADAR)  1258
LSTARS= option  
CHART statement (GRADAR)  1258
LV=  
LVREF= procedure option  1020, 1154,  
1164, 1182, 1199
LVREF= option  
BUBBLE statement (G PLOT)  1154
BUBBLE2 statement (G PLOT)  1164
PLOT statement (GCONTOUR)  1020
PLOT statement (GPLOT)  1182
PLOT2 statement (GPLOT)  1199

M
M= option  
TITLE, FOOTNOTE, and NOTE  
statements  458
macros  
%SHORTCUT  94
color utility macros  327
major tick marks  351, 365
formatting  365
offset  353
scatter plots  1403
suboptions, list of  1443
surface plots  1393
with datetime values (example)  465
MAJOR= option, AXIS statement  351,  
1443
MAP element (HTML)  513
mapping macro  
%CENTROID  754
maps  
creating color maps  1318
Marker font  1501
marker symbol  
suppressing  857
MATCHCOLOR option  
PIE, PIE3D, DONUT statements  
(GCHART)  948
STAR statement (GCHART)  968
Math font  1497
MAXCOLORS  582
MAXCOLORS= option, GDEVICE  
procedure  582
MAXIS= option  
BAR statement (GBARLINE)  840  
HBAR, HBAR3D, VBAR, VBAR3D  
statements (GCHART)  920
MAXNVERT= option  
CHART statement (GRADAR)  1258
MAXPOLY  583
MAXPOLY= option, GDEVICE  
procedure  583
MAXVERT=  
MAXNVERT= procedure option  1258
MEAN option  
BAR statement (GBARLINE)  840  
HBAR, HBAR3D, VBAR, VBAR3D  
statements (GCHART)  920
PLOT statement (GBARLINE)  856
mean statistic  822, 840, 856, 884
numeric variable for  846
MEANLABEL= option  
HBAR, HBAR3D statements  
(GCHART)  921
memory, open software fonts  550
message queuing  604
metadata  
adding to PDF files  244, 246
metagraphics device drivers  
translation command  599
Metagraphics device drivers  
color space specification  533
header generation  571
metacode file format  555
Metagraphics output, interactive  617
character rotation angle  610
description string  577
hardware text rotation angle  590
interactivity of 578
negative handshaking response 584
TRAILER records 616
translating metafile into device commands 599
user-written part, files for 600
Metagraphics window (GDEVICE) 1059
Metaview Applet presentations drill-down in 198
Microsoft Office
exporting graphs to 233
importing graphs into 239
Microsoft Word
sending output to RTF files 56
midpoint axes 840
MIDPOINT variable, Annotate facility 722
MIDPOINTREF option
HBAR, VBAR statements (GCHART) 921
midpoints
character 819
continuous numeric 819
for numeric chart variable 840
GBARLINE procedure 819
GCHART procedure 877, 880
in horizontal bar charts 985
missing values and 842
ordering and selecting 820, 850, 882
ordering and selecting for bar charts 935
suppressing 843
values for bars 841
MIDPOINTS= option
BAR statement (GBARLINE) 841
BLOCK statement (GCHART) 894
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 921
PIE, PIE3D, DONUT statements (GCHART) 948
STAR statement (GCHART) 968
MIDPOINTS=OLD option
BLOCK statement (GCHART) 895
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 922
PIE, PIE3D, DONUT statements (GCHART) 949
STAR statement (GCHART) 969
minor tick marks 351, 365, 842
formatting 365
plot overlays 856
suboptions, list of 1443
with datetime values (example) 465
MINOR= option
AXIS statement 351, 1443
BAR statement (GBARLINE) 842
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 922
PLOT statement (GBARLINE) 856
MISSING option
BAR statement (GBARLINE) 842
BLOCK statement (GCHART) 895
CHART statement (GRADAR) 1258
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 922
PIE, PIE3D, DONUT statements (GCHART) 949
STAR statement (GCHART) 969
missing values
GBARLINE procedure 823
GCHART procedure 880
GCONTOUR procedure 1010
G TILE procedure 1350
in Annotate data sets 651
midpoints and 842
plot data sets 1140, 1185, 1200
MODE= option
CHART statement (GRADAR) 1258
LEGEND statement 382, 394
PROC GKPI statement 1109
SYMBOL statement 428
MODEL 583
model number, output device 583
MODEL= option, GDEVICE procedure 583
modern mode
GKPI procedure 1109
MODIFY statement
GDEVICE procedure 1053
GREPLAY procedure 1297
MODULE 583
MODULE= option, GDEVICE procedure 583
monolingual Asian TrueType fonts 300
MOVE function, Annotate facility 688, 1488
MOVE statement
GREPLAY procedure 1299
MOVE= option, TITLE, FOOTNOTE, and NOTE statements 458
Mozilla Firefox
font mapping 150
SVG files and 150
multiline
axis values 359
legend labels 390
multilingual Unicode TrueType fonts 300
Music font 1501
MWIDTH= option
PROC GFONT statement 1078
N option
   NOLIST option, PROC GOPTIONS
   statement 1128
N= argument
   NAME= argument, PROC GFONT
   statement 1073

N= option
   AXIS statement options 367
N= argument
   NAME= argument, PROC GFONT
   statement 1073
NAME= argument
   PROC GFONT

N= option
   NAXIS1= option
   GRID statement (G3GRID) 1426
   needle plots 422
   negative values
      block charts 901
      GTILE procedure 1350
   NLEVELS= option
      CHART statement (GRADAR) 1260
      PLOT statement (GCONTOUR) 1021
   no frame, legend 380
   NOALTDESC option 522
   noautocopy 523
   NOAUTOCOPY option 523
   NOAUTOFEED option 524
   NOAVALUE option
      GKPI procedure 1112
   NOAXES
      NOAXIS procedure option 1155, 1165,
      1184, 1199
   NOAXES option
      NOAXIS option, PLOT statement
      (G3D) 1392
      NOAXIS option, SCATTER statement
      (G3D) 1400
   NOAXIS option
      BAR statement (GBARLINE) 843
      BUBBLE statement (GPLOT) 1155
      BUBBLE2 statement (GPLOT) 1165
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBASEREF option
      BAR statement (GBARLINE) 843
      HBAR, HBAR3D, VBAR, VBAR3D
      statements (GCHART) 923
   PLOT statement (G3D) 1392
   PLOT statement (GBARLINE) 856
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   SCATTER statement (G3D) 1400
   NOBRACKETS option, AXIS statement
   options 352
   NOBUILD argument
      PROC GFONT statement 1073
   NOBUILD argument, PROC GFONT
   statement 1073
NOCHARACTERS option 528
NOCIRCLEARC option 530
NOCOLLATE option 531
NOCONNECT option
   STAR statement (GCHART) 970
NODASH option 537
NODISPLAY 545
NODISPLAY option 545
   PROC GFONT statement 1079
NODROPCOLLISIONS option
SYMBOL statement 429
NODRVQRY= option, GDEVICE procedure, executing before driver initialization 546
NODUPLEX option 547
NOERASE graphics option 548
NOERASE option, GDEVICE procedure 548
NOFASTTEXT option 549
NOFASTTEXT= graphics option 549
NOFILEONLY graphics option 552
NOFILL graphics option 552
NOFILL option, GDEVICE procedure 552
NOFRAME option
   BAR statement (GBARLINE) 837
   BUBBLE statement (GPLOT) 1151
   CHART statement (GRADAR) 1256
   HBAR, HBAR3D, VBAR, VBAR3D statements 800
   HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 913
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1178
   PLOT2 statement (GPLOT) 1197
NOFRAME= option, LEGEND statement options 380
NOFS option
   PROC GDEVICE statement 1050
   PROC GREPLAY statement 1289
NOGFOOTNOTE= option, ODS HTML statement 101
NOGRAPHRC graphics option 564
NOGROUPHEADING option
   PIE, PIE3D, DONUT statements (GCHART) 950
   STAR statement (GCHART) 970
NOSFPROMPT option 567
NOSTITLE= option, ODS HTML statement 101
NOHEADING option
   BLOCK statement (GCHART) 896
   PIE, PIE3D, DONUT statements (GCHART) 950
   STAR statement (GCHART) 970
NOHEX option
   NOROMHEX option, PROC GFONT statement 1079
NOIMAGEPRINT GOPTIONS statement 340, 577
NOIMAGEPRINT graphics option 340, 577
NOKEYMAP option
   PROC GFONT statement 1074, 1079
NOLABEL option
   PLOT statement (G3D) 1392
   SCATTER statement (G3D) 1400
NOLASTAREA option
   PLOT statement (GPLOT) 1184
   PLOT2 statement (GPLOT) 1199
NOLASTFILL
   NOLASTAREA procedure options 1184, 1199
NOLEGEND
   NOSYMBOL procedure option 924
NOLEGEND option
   BAR statement (GBARLINE) 843
   BLOCK statement (GCHART) 896
   CHART statement (GRADAR) 1260
   FLOW, TILE, TOGGLE statements (GTILE) 1364
   HBAR, HBAR3D, VBAR, VBAR3D statements 801
   HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 924
   PIE, PIE3D, DONUT statements (GCHART) 950
   PLOT statement (GCONTOUR) 1022
   PLOT statement (GPLOT) 1184
   PLOT2 statement (GPLOT) 1199
   STAR statement (GCHART) 970
NOLINE option
   PLOT statement (GBARLINE) 857
NOLIST option
   PROC GOPTIONS statement 1128
NOLOG option
   PROC GOPTIONS statement 1128
NOLOWBOUNDARY option
   GKPI procedure 1115
NOMARKER option
   PLOT statement (GBARLINE) 857
non-interactive graphs
   adding drill-down to web presentations 159
creating with ODS 154
   developing presentations with SVG, PNG, and GIF 156
presentations developed with ACTXIMG, JAVA IMG 157
   sample programs for 159
non-roman alphabet fonts 1497
NONEEDLE option
SCATTER statement (G3D) 1400
NONINTERLACED GDEVICE
procedure 579
NONINTERLACED GOPTIONS
statement 579
nonlinear curves
horizontal variables along 1420
NOPCLIP 590
NOPIEFILL GDEVICE procedure 593
NOPIEFILL GOPTIONS statement 593
NOPLANE option, AXIS statement
options 352
NOPOLYGONCLIP graphics option 594
NOPOLYGONFILL graphics option 595
NOQMSG option 617
NOREVERSE graphics option 608
NOROMAN option
PROC GFONT statement 1075
NOROMHEX option
PROC GFONT statement 1075
NOSCALE option
GRID statement (G3GRID) 1426
NOTSTATS option
HBAR, HBAR3D statements
(GCHART) 924
NOSYMBOL option
HBAR, HBAR3D, VBAR, VBAR3D
statements (GCHART) 924
NOTE statement 44, 396, 447, 463
BY statement with 374
notes 463
angle of rotation 449, 456, 459
boxes around 451
colors for 453, 535, 536
default characteristics, setting 463
defining text of 460
fonts for 454
justification 455
on top of graph 396
positioning 458
size of 454, 575
spacing around 458
text breaks 455
underlining 461
NOTRANSPARENCY GOPTIONS
statement 617
NOTSORTED= option, BY statement
371
NOV6COMP 622
NOV6COMP graphics option 622
NOZERO option
BAR statement (GBARLINE) 843
HBAR, HBAR3D, VBAR, VBAR3D
statements (GCHART) 924
NOZEROREF option
CHART statement (GRADAR) 1260
NR option
NOROMAN option, PROC GFONT
statement 1075
NROWS= option
CHART statement (GRADAR) 1260
number sign #, variables as plot point
labels 430
NUMBER= option, AXIS statement
options 367
numeric chart variables 805, 880
discrete 881
numeric values
continuous 881
O
observations
in Annotate data sets 639
in output data set (G3GRID) 1427
ODS (Output Delivery System)
graphics output files and 109
ODS destinations 50, 97
changing current style with STYLE= option 279
closing to save system resources 64
default 62
default devices for 82
default styles and 275
introduction to 49
LISTING destination 51
opening and closing 50
sending output to multiple open
destinations 64
specifying devices and styles with 64
storing GRSEGs with multiple
destinations 122
ODS documents
replaying 129
ODS HTML 397
ODS HTML statement 397
bar chart with drill-down (example) 506
multiple graphs and reports in web page
(example) 495
ODS output
graphic options with 101
name and location of 117
non-interactive graphs 154
RUN-group processing 100
titles and footnotes, controlling 101
ODS statement 46
ODS statements 45
ODS USEGOPT statement 101
ODS styles 50, 51
default 62
in graphs stored as GRSEG catalog entries 630
introduction to 49
specifying 52
specifying with multiple open destinations 64
offset
angle of rotation 459
axes 353
between Bitstream font letters 556
between display area and graphic 620
between displayed area and graph 572
between fill lines 553
between graphs and display 572, 620
legend 382, 394
legends 382, 394
text in graphics output 452, 458
OFFSET= option
AXIS statement options 353
LEGEND statement options 382, 394
OFFSHADOW 584
OFFSHADOW= graphics option 584
ONCLICK= HTML attribute
color for 834
colors 409
default 407
donut charts 958
GBARLINE procedure 824
GCHART procedure 885
pie charts 958
slice colors and patterns 958
star charts 973
output
See also graphics output
displaying SAS/GRAPH output in GRAPH window 631
generating with procedures 53
sending to a file 54
sending to GRAPH window 53
sending to multiple ODS destinations 64
sending to PDF files 57
sending to RTF files 56
sending to web pages 55
SVG 142
output data sets
controlling observations in (G3GRID) 1427
G3GRID procedure 1420
output filenames 123
output formats
exporting graphs to Microsoft Office 233
output printer bins 585
OUTSIDE= option
BAR statement (GBARLINE) 843
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 924
OUTTRI= option
PROC G3GRID statement 1423
overdrawing graphs 607
OVERLAY option
PLOT statement (GPLOT) 1184
PLOT2 statement (GPLOT) 1200
overlay plots 1219, 1223
OVERLAY=" OVERLAYVAR= procedure option 1260
OVERLAY= option
PROC G3GRID statement 1423
OVERLAY= option
PLOT statement, GPLOT procedure 1219, 1223
out-of-range plot variables 824, 1140
OUT= option
PROC G3GRID statement 1423
OUTCOLOR= option
PROC G3GRID statement 1423
OUTCOLOR= option
PIE, PIE3D, DONUT statements (GCHART) 950
OUTCOLOR= option
PIE, PIE3D, DONUT statements (GCHART) 950
OUTCOLOR= option
PIE, PIE3D, DONUT statements (GCHART) 951
outlines
bar charts 935
block charts 900
color for 834
colors 409
default 407
donut charts 958
GBARLINE procedure 824
GCHART procedure 885
pie charts 958
slice colors and patterns 958
star charts 973
overlaying graphics, Annotate facility
   652
overlaying radar charts  1268
OVERLAYVAR= option
   CHART statement (GRADAR)  1260
overriding devices  86

P
P option
   PATTERN option, PROC GOPTIONS
      statement  1128
page layout in PDF output  244
page orientation
   PDF files  246
page orientation in PDF output  244
paints, plot  420
paper feed  524, 586
paper size  586, 587
paper type, specifying  589
PAPERDEST  585
PAPERDEST= graphics option  585
PAPERFEED  586
PAPERFEED= graphics option  586
PAPERFEED= option, GDEVICE
      procedure  586
PAPERLIMIT  586
PAPERLIMIT= graphics option  586
PAPERSIZE  587
PAPERSIZE= graphics option  587
PAPERSOURCE  588
PAPERSOURCE= graphics option  588
PAPERTYPE  589
PAPERTYPE= graphics option  589
Parameters window (GDEVICE)  1057
parametric language interpolation  420
PARTIAL option
   GRID statement (G3GRID)  1426
   partial spline interpolation  1434
PATH  590
PATH= option, GDEVICE procedure  590
PATTERN  405
PATTERN definitions  1206
   displaying values of  1125
PATTERN definitions, BY statement with
      374
PATTERN option
   PLOT statement (GCONTOUR)  1022
   PROC GOPTIONS statement  1128
PATTERN statement  44, 398, 405
   ActiveX and Java support for
      altering/canceling  406
   images on bar chart bars  336
      using  406
PATTERNID= option
   BAR statement (GBARLINE)  844
   BLOCK statement (GCHART)  897
   BY line  373
HBAR, HBAR3D, VBAR, VBAR3D
   statements (GCHART)  925
patterns
   assigning  844
   bar charts  935
   block charts  900
   changing  826
   contour plots  1038
   GBARLINE procedure  824
   GCHART procedure  885
   pie and donut chart slices  958
   plots  1206
   star charts  973
   user-defined  886, 900, 935
   when patterns change  900, 935
patterns and fills  405
   bar charts  400
   built-in pie-fill capability  593
   built-in polygon-fill capability  595
   built-in rectangle capability  605
   built-in rectangle-fill capability, device
      552
   color for  534
   contour plots  401
   default  407
   fill color  398
   filling area between plot lines (example)  
      481
   for symbol plots  421
   images as fill elements  399
   images on bar chart bars  336
   outline colors  409
   pattern sequences  410
   pie and star charts  403
   spacing between fill lines  553
PCLIP  590
PCLIP= graphics option  590
PCOLS  591
PCOLS= option, GDEVICE procedure  
      591
PDF device drivers
   features and capabilities  242
PDF files
   adding bookmarks  244
   adding metadata to  244
   archivable  243
   changing page layout  244
   default compression level  246
   examples  246
   fonts  242
   multipage, with bookmarks and
      metadata  246
   multiple-page, using BY-group
      processing  255
multiple-page, using GREPLAY procedure 258
page orientation 244, 255
paper size 244
PDF/A-1b compliant, with multiple graphs per page 250
sending output to 57
writing drill-down graphs to 245, 251, 255
writing graphs to 241
PE option
PATTERN statement, VALUE= option 403
PEMPTY variable, Annotate facility 731
pen speed, plotters 612
PENMOUNTS 592
PENMOUNTS= graphics option 328
pens, active 592
PENSORT 593
PENSORT= graphics option 593
PENSORT= option, GDEVICE procedure 593
PERCENT option
BAR statement (GBARLINE) 844
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 925
PLOT statement (GBARLINE) 857
PERCENT= option
PIE, PIE3D, DONUT statements (GCHART) 951
STAR statement (GCHART) 970
percentage statistic 822, 883
percentiles, box plots 417, 476
PERCENTLABEL= option
HBAR, HBAR3D statements (GCHART) 926
PERCENTSUM option
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 926
pie and donut charts
links in 954
PIE and PIE3D statements
GCHART procedure 956
PIE and PIE3D statements, GCHART procedure
ActiveX and Java support for 1457
pie charts 875
creating 956
detail pie charts 875, 1004
grouping and arranging 998
ordering and labeling slices 996
outlines 958
patterns 403
selecting and positioning slice labels 956
slice patterns and colors 958
statistic and group headings 950, 959
subgrouping 994
sum statistic in 991
terminology 879
PIE function, Annotate facility
ActiveX and Java support for 1488
pie slices, drawing with Annotation facility 689
PIE statement
GCHART procedure 936
PIE statement, BY statement with 373
pie-fill capability, device 593
PIE, PIE3D, DONUT statements (GCHART)
URL= option 954
PIE3D statement
GCHART procedure 936
PIE毫无 function, Annotate facility 692, 1489
PIEFILL 593
PIEFILL GDEVICE procedure 593
PIEFILL OPTIONS statement 593
PIEXY function, Annotate facility 693, 1490
PLABEL= option
PIE, PIE3D, DONUT statements (GCHART) 951
plot data sets 1140
plot lines 427, 442
filling area between plot lines (example) 481
type of 427
plot overlays
bar-line charts 860
interpolation methods 861
legends for 855
multiple plots 866
SYMBOL definitions 850
variable to plot 858
PLOT statement
G3D procedure 1387, 1394
GBARLINE procedure 850
GCONTOUR procedure 1011
GPLOT procedure 1168
PLOT statement (GBARLINE)
SYMBOL definitions 850
PLOT statement (GPLOT)
URL= option 1185
PLOT statement, G3D procedure
ActiveX and Java support for 1482
PLOT statement, GBARLINE procedure 860
ActiveX and Java support for 1454
interpolation methods 861
options 852
syntax 860
PLOT statement, GCONTOUR procedure 1026
ActiveX and Java support for axis order 1028
contour levels 1026
options 1014
required arguments 1014
syntax 0
PLOT statement, GPLOT procedure
ActiveX and Java support for axis order 1465
connecting plot data points (example) 1217
different scales of values (example) 1231
filling areas in overlay plot (example) 1223
generating overlay plot (example) 1219
matching plot requests 1203
multiple plot requests 1204
options 0
plots with drill-down for web (example) 1235
plotting three variables (example) 1226
plotting two variables (example) 1214
required arguments 1171
syntax 0
plot statistic
frequency variable for 854
specifying 858
plot symbols 1189, 1206
altering or canceling 437
built-in drawing capability 614
colors for 414, 441, 535
colors for, rotating through (example) 473
default 444
fonts of 416
GBARLINE procedure 860
in Annotate graphics output 701
interpolation 579
scatter plots 1401
size of 417, 428
specifying for plot points 439
suppressing the line connecting 857
plot variables
out-of-range 824
PLOT2 statement
PLOT2 procedure 1189
PLOT2 statement (GPLOT)
URL= option 1200
PLOT2 statement, GPLOT procedure 1203
ActiveX and Java support for different scales of values (example) 1231
matching plot requests 1203
multiple plot requests 1204
options 1202
required arguments 1192
syntax 1192
plots
See also G3D procedure
basics of 1138
box plots 417, 419, 476
classification variables with 1135
connecting plot data points (example) 1217
different scales of values in (example) 1231
drill-down functionality (example) 1235
filling areas in overlay plot (example) 1223
generating overlay plot (example) 1219
generating simple bubble plots (example) 1206
high-low plots 419
interpolation methods 1138
labeling and sizing plot bubbles (example) 1208
legends in 855
links in 855, 859, 1185, 1200
matching plot requests 1203
missing values 1140, 1185, 1200
multiple plot requests 1204
needle plots 422
order of points 852, 854
overlay plots 1219, 1223
patterns 1206
plotting three variables (example) 1226
plotting two variables (example) 1214
regression analysis 423
regression analysis plots 423
right vertical axis to bubble plot (example) 1211
standard deviations 425
step plots 427
symbols in 1189, 1206
three variables and legend 1204
two variables 1134
two vertical axes 1137, 1205, 1211
with multiple variables 1188
plotters
active pens or colors 592
drawing elements in color order 593
paper size 586
pen speed 612
PNG device driver
ACTXIMG, JAVAIMG vs. 156
chart descriptions for 190
data tips for 191
developing web presentations 156
drill-down links in output 197
HTML files, generating 118
sample programs for non-interactive images 161
POINT function, Annotate facility 695, 1490
POINTLABEL= option
SYMBOL statement 428
POINTLABEL= option, SYMBOL statement 1452
points, drawing with Annotate facility 695
points, plot
labels for 428
specifying for plot points 432
symbols for, specifying 432, 439
POLY function, Annotate facility 696, 1491
POLYCONT function, Annotate facility 698, 1491
POLYFILL= option, GDEVICE procedure 595
polygon fonts 1069
polygon-fill capability, device 595
POLYGONCLIP 594
POLYGONCLIP graphics option 594
POLYGONFILL 595
POLYGONFILL graphics option 595
polygons
clipped (intersecting) 590, 594
drawing with Annotate facility 696
vertices, maximum drawn 583
POP function, Annotate facility 700
pop-up images
adding to SVG documents 198
portrait orientation 591, 604, 609
ports, how output is written to 565
POSITION variable, Annotate facility 724
POSITION= option
AXIS statement options 365
LEGEND statement options 384, 390, 393, 395
SYMBOL statement, POINTLABEL= option 430
positioning
Annotate graphics 646, 647
axis labels 348, 353, 356, 365
BY lines 372
legend label 380
legend text 390
legends 384, 393
notes on top of graph 396
plot point labels 430
text in graphics output 458
titles and footnotes, ODS output 101
POSTGEPLOG 595
POSTGEPLOG= graphics option 595
POSTGPROLOG 596
POSTGPROLOG= option 596
POSTGRAPH= option, GDEVICE procedure 596
POSTGRAPH1 596
POSTGRAPH1= graphics option 596
POSTGRAPH2 596
POSTGRAPH2= graphics option 596
PPD file, location of 597
PPDFILE 597
PPDFILE= graphics option 597
PPERCENT= option
PIE, PIE3D, DONUT statements (GCHART) 951
PREGEPLOG 598
PREGEPLOG= graphics option 598
PREGPROLOG 598
PREGPROLOG= graphics option 598
PREGRAP= graphics option 599
PREGRAP= option, GDEVICE procedure 599
PREGRAPH1 599
PREGRAPH1= option 599
PREGRAPH2 599
PREGRAPH2= option 599
PRESENTATION option
PROC GREPLAY statement 1289
PRESENTATION window 1311
PREVIEW statement
GREPLAY procedure 1300
previewing device output 616
previewing output 131
printers
sending graphs directly to 132
printing 131
automatic 523
collating output 531
copies to print 558
duplex 547
duplex, binding edge for 526
flow control 570
graph orientation 609
output bin, specifying 585
paper feed 524, 586
paper size 586, 587
paper tray, specifying 588
paper type, specifying 589
PPD file, location of 597
previewing output 616
protocol module, specifying 562
redrawing (overdrawing) graphs 607
reverse printing 608
saving and printing graphs 132
PRISM statement, GMAP procedure
ActiveX and Java support for 1467
PROC G3D statement 1387
PROC G3GRID statement 1422
PROC GANNO statement 780
PROC GAREABAR statement 797
PROC GBARLINE
UNIFORMAXES option 828
PROC GBARLINE statement
GBARLINE procedure 827
PROC GCHART statement 887
PROC GCONTOUR statement 1010
PROC GDEVICE statement 1049
PROC GKPI statement 1109
PROC GOPTIONS statement 1127
PROC GPLOT statement 1142
GPLOT procedure 1141
PROC GREPLAY statement 1286
PROC GREPLAY window 1310
PROC GSLIDE statement 1336
PROC GTILE statement 1354
PROC steps 42
procedure output area
Annotate facility 648
placement of graphics in 76
procedure statements 43
procedure termination, step code at 564
    generating output with 53
    PROC steps 42
    statements 43
    subordinate statements 43
PROCESS 599
PROCESS= option 599
PROCESSINPUT 600
PROCESSINPUT= option, GDEVICE procedure 600
PROCESSOUTPUT 600
PROCESSOUTPUT= option, GDEVICE procedure 600
product codes 35
program mode
    GDEVICE procedure 1047
    programs 41
        a typical program 41
Annotate DATA set 44
Base SAS language statements 46
global statements 43
language elements used by 41
ODS statements 45
other statements and options 43
PROC steps 42
procedure statements 43
RUN-group processing 67
subordinate statements 43
PROMPT 601
prompt messages to GSF 567
PROMPT= graphics option 601
PROMPTCHARS 602
PROMPTCHARS= graphics option 602
PROMPTCHARS= option, GDEVICE procedure 602
prompts
    characters for, specifying 602
    specifying if used 601
proportion
    image quality across devices and 75
    proportional fonts 1068
protocol module, specifying 562
PROWS 604
PROWS= option, GDEVICE procedure 604
PS option
    PATTERN statement, VALUE= option 403
PUSH function, Annotate facility 701

Q
QDEVICE procedure
    generating reports for devices 84
QMSG 604
QMSG option 604
QMSG= option, GDEVICE procedure 604
quality of images
    across devices 75
QUIT statement 46
GREPLAY procedure 1300
RUN-group processing 100

R
R= option
    AXIS statement options 365
    LEGEND statement options 391
    PATTERN statement 411
    PATTERN statement options 400
    RESOL= option, PROC GFONT statement 1079
    SYMBOL statement 445
    SYMBOL statement, REPEAT= option 431
    TITLE, FOOTNOTE, and NOTE statements 459
radar charts 1243
    axes, modifying 1272
    creating 1246
    links in 1265
    modifying appearance of 1271
    multiple classification variables in 1270
    overlaying 1268
tiling 1269
radar charts (star charts)
  ActiveX and Java support for 1477
RADIUS= option
  PIE, PIE3D, DONUT statements
  (GCHART)  952
raised mode
  GKPI procedure  1109
range boundaries  1102
range colors  1104
range of values  844
RANGE option
  BAR statement (GBARLINE)  844
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART)  926
raster formats  235
RAXIS= option
  BAR statement (GBARLINE)  845
  HBAR statement  373
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART)  927
  PLOT statement (GBARLINE)  857
  VBAR statement  373
RC= option
  ROMCOL= option, PROC GFONT
  statement  1075
record length, GSF, origins  564
rectangle-fill capability, device  552, 605
rectangles, drawing with Annotate facility
  671
RECTFILL  605
RECTFILL= option, GDEVICE
  procedure  605
redrawing graphs  607
REF= option
  BAR statement (GBARLINE)  845
  HBAR, HBAR3D, VBAR, VBAR3D
  statements (GCHART)  927
  PLOT statement (GBARLINE)  857
  VBAR statement  373
REFCOL= option
  PROC GFONT statement  1075
reference lines  845
  at major tick marks  832, 852
  clipping at bars  834, 853
  color of  833, 834, 853
  colors  1016, 1017, 1149, 1150, 1176, 1177, 1196
  for plot overlays  857
  line type for  839, 840, 855, 856
  suppressing  843
  width of  847, 848, 859
  reference-line labels, axis  356, 362
REFLABEL= option, AXIS statement
  options  356, 1443
  FORMAT= suboption  363
REFLINES option
  PROC GFONT statement  1075
REGEQN option
  PLOT statement (G PLOT)  1185
  PLOT2 statement (G PLOT)  1200
registry subkeys
  specifying fonts  303
regression analysis plots  423
RENAME statement
  GDEVICE procedure  1054
RENDER  606
RENDER= graphics option  606
rendering fonts  549, 1493
  Bitstream fonts  556, 606
  resolution, setting  554
  software fonts  613
  storing font files  607
RENDERLIB  607
RENDERLIB= graphics option  607
REPAINT  607
REPAINT= graphics option  607
REPAINT= option, GDEVICE procedure
  607
REPEAT= option
  LEGEND statement  385
REPEAT= option, PATTERN statement
  400, 411
REPEAT= option, SYMBOL statement
  445
  repeating animation loops  579
REPLAY statement
  GREPLAY procedure  1300
replaying graphs
  considerations for graphs with borders
  1284
  sizing and naming  1283
replaying output  128
  DOCUMENT procedure  128
  GREPLAY procedure  128
  ODS documents  129
RESET  608
RESET= graphics option  608
resetting graphics options  376, 608
RESOL= option
  PROC GFONT statement  1079
resolution  72, 114, 116
  display device  623, 624, 625, 626
exporting graphs to Microsoft Office
  234
  image interlacing  579
  software fonts  554
  SVG graphics  151
response axis
  chart statistic and  849, 933
response variables
  GBARLINE procedure  821
RETA I N statement  650
return characters at record ends 559
return codes 564
REVERSE 608
REVERSE graphics option 608
reversing black and white 612
RF= option
  ROMFONT= option, PROC GFONT statement 1075
RGB color codes 318
  SAS color names 324
RGBA color codes 319
RH= option
  ROMHT= option, PROC GFONT statement 1076
RIGHTMARGIN= option, GDEVICE procedure 574
RIGHTMARGIN= option, GOPTIONS statement 574
ROMCOL= option
  PROC GFONT statement 1075
ROMFONT= option
  PROC GFONT statement 1075
ROMHEX option
  PROC GFONT statement 1076, 1080
ROMHT= option
  PROC GFONT statement 1076
ROTATE 609
ROTATE variable, Annotate facility 727, 728
ROTATE= graphics option 609
ROTATE= option
  AXIS statement options 365
  GDEVICE procedure 609
  PLOT statement (G3D) 1392
SCATTER statement (G3D) 1400
TDEF statement (GREPLAY) 1305
TITLE, FOOTNOTE, and NOTE statements 459
ROTATE= option, LEGEND statement options 391
ROTATE= suboption
  LABEL= option, DONUT statement (GCHART) 956
rotating graphs for printing 609
rotating plot symbols through colors (example) 473
rotating plots 1385, 1405
rotation
  legend values 391
ROTATION 610
ROTATION= option, GDEVICE procedure 610
ROWMAJOR option
  LEGEND statement options 385
rows
  in graphics output area 582, 611, 621
  legends 379
ROWS 611
ROWS= option, GDEVICE procedure 611
RSTAT= option, HBAR and VBAR statements 810
RTF files
  sending output to 56
RUN statement 46, 373
RUN-group processing 67
  BY statements with 68
  creating animated GIF images with 178
  global and local statements with 67
  GSLIDE procedure 1340
ODS and 100
WHERE statement with 68
running programs
  RUN-group processing and 67
S
S option
  SYMBOL option, PROC GOPTIONS statement 1129
sample programs 35
  product codes for 35
SAS Color Naming Scheme (CNS) 323
SAS output 106
SAS registry
  changing default style in 279
  viewing font specifications in 302
SAS/GRAPH
  displaying output in GRAPH window 631
  ODS styles in graphs stored as GRSEG catalog entries 630
  specifying a terminal device driver 629
  using system options 629
SAS/GRAPH
  introduction to devices 49
  introduction to ODS destinations 49
  introduction to ODS styles 49
  product codes 35
  terminal device driver, specifying 629
SAS/GRAPH device 49
SAS/GRAPH fonts 300
SAS/GRAPH programs
  See programs
SAS/GRAPH templates
  listing 1282
SCALABLE 611
Scalable Vector Graphics
  See SVG graphics
SCALABLE= option, GDEVICE procedure 611
SCALE option
GRID statement (G3GRID) 1426
SCALEX= option
TDEF statement (GREPLAY) 1306
SCALEY= option
TDEF statement (GREPLAY) 1306
scaling
dash length in lines 538
data-dependent output 784
graphs 783
hardware fonts 534, 611
scatter plots 1382
appearance of points 1404
axes, controlling 1386
connecting plot data points (example) 1217
data ranges 1385
input data sets 1385
plotting three variables (example) 1226
plotting two variables (example) 1214
rotating and tilting 1385
three-dimensional, syntax for 1403
SCATTER statement
G3D procedure 1396, 1403
SCATTER statement, G3D procedure
ActiveX and Java support for 1482
search order
of device catalogs 1047
shadow color, legends 379
shadowing, legend frames 584
shape, legend values 385
SHAPE= option
HBAR3D, VBAR3D statements (GCHART) 927
LEGEND statement options 385
SCATTER statement (G3D) 1401
shapes in scatter plots 1401
SHORT option
PROC GOPTIONS statement 1129
SHOWALL option
PROC GFONT statement 1076
SHOWROMAN option
PROC GFONT statement 1076, 1080
SIDE option
PROC GOPTIONS statement (G3D) 1392
SIMFONT 611
SIMFONT= graphics option 611
SIMULATE font 1503
SINGULAR= option, POINTLABEL= specification 431
singularities, checking for 431
size
aspect ratio 523
axis labels 348, 353, 356, 363
axis tick marks 351, 366, 367
axis values 363
boxes in box plots 449
BY lines 372, 571
character cells 525
contour labels 417
contour lines 436
dash length in lines 538
display, in lines 567
fonts 529
graphics output text 575
legend frame 380
legend frame drop shadows 584
legend label 380
legend values 385, 387
line thickness, default 581
paper 586, 587
paper feed increments 586
plot bubbles (example) 1208
plot point labels 429
plot symbols 417, 428
record length, to GSF 564
scatter plot points 1402, 1404
text in graphics output 454
titles and footnotes 102, 575
units of measurement 568
SIZE variable, Annotate facility 728
size, graphics
SVG graphics 152
size, graphics output area 574, 622, 623, 624, 625, 626
columns in 534, 573, 581, 591
rows in 582, 611, 621
SIZE= option
SCATTER statement (G3D) 1402
sizing
bubbles in bubble plots 1148
sizing errors 77
sizing graphs for replay
best practices 1283
SKIPMISS option
PLOT statement (GPLOT) 1185
PLOT2 statement (GPLOT) 1200
slice labels 973
SLICE= option
PIE, PIE3D, DONUT statements (GCHART) 952
STAR statement (GCHART) 970
slices
ordering and labeling 996
slider KPI charts 1100
smooth line fit 425
SMOOTH option
PLOT statement (GCONTOUR) 1022
SMOOTH= option
GRID statement (G3GRID) 1426
smoothing
spline smoothing 1422
smoothing plot lines 420
software fonts 611
open at one time 550
rendering 613
resolution 554
where stored 1494
sorting
grouped observations 371
plot data set observations 1140
space
between bars 845
space data sets 1091
creating 1091
variables 1091
SPACE= option
BAR statement (GBARLINE) 845
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 928
LEGEND statement options 386
SPACEDATA= option
PROC GFONT statement 1080
spacing
angle of rotation 459
between Bitstream font letters 556
between display area and graphic 620
between displayed area and graph 572
between fill lines 553
legends 382, 394
text in graphics output 452, 458
spacing, legend values 386
special characters 304
HTML entities 267
Special font 1502
special plot symbols 432
SPEED 612
speed of plotter pens 612
SPEED= graphics option 612
SPEED= option
CHART statement (GRADAR) 1261
SPEED= option, GDEVICE procedure 612
speedometer KPI charts 1101, 1121
reversed colors 1122
SPEEDOMETER statement
GKPI procedure 1111
SPIDERWEB option
CHART statement (GRADAR) 1261
spine labels 973
SPKLABEL= option
CHART statement (GRADAR) 1261
spline interpolation 425, 440, 1421, 1435
partial 1434
SPLINE option
GRID statement (G3GRID) 1427
spline smoothing 1422, 1431
SPLIT= option
LEGEND statement options 386
SPLIT= option, AXIS statement options 358
splitting, legend value descriptions 386
SPOKESCALE= option
CHART statement (GRADAR) 1261
SR option
SHOWROMAN option, PROC GFONT statement 1076, 1080
STAGGER option
AXIS statement 359
standard deviations 359
star charts 876
See also radar charts
creating 972
discrete numeric variables in 1001
links in 971
patterns 403
patterns and outlines 973
spine and slice labels 973
statistic and group headings 970, 974
sum statistic in 999
STAR statement
GCHART procedure 960, 972
STAR statement (GCHART)
URL= option 971
STAR statement, GCHART procedure
BY statement with 373
STARAXES=
STARAXIS= procedure option 1261
STARAXIS= option
CHART statement (GRADAR) 1261
STARCIRCLES= option
CHART statement (GRADAR) 1262
STARFILL= option
CHART statement (GRADAR) 1262
STARINRADIUS= option
CHART statement (GRADAR) 1263
STARLEGEND= option
CHART statement (GRADAR) 1263
STARLEGENDLAB= option
CHART statement (GRADAR) 1263
STARLEGENDMAX= option
CHART statement (GRADAR) 1263
STARLEGENDMIN= option
STAR statement (GCHART) 970
STARMIN= option
STAR statement (GCHART) 970
STAROUTRADIUS= option
CHART statement (GRADAR) 1263
stars, drawing circle of (example) 664
STARSTART= option
CHART statement (GRADAR) 1263
STARTYPE= option
CHART statement (GRADAR) 1264
statement options
overriding style attributes with 280
statements 343
Base SAS language statements 46
global 43
ODS 45, 46
STATFMT option
BAR statement (GBARLINE) 845
BLOCK statement (GCHART) 897
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 928
PIE, PIE3D, and DONUT statements (GCHART) 952
PLOT statement (GBARLINE) 858
STAR statement (GCHART) 970
statistic headings
pie and donut charts 950, 959
star charts 970, 974
statistics
See also chart statistics
displaying above bars 843
displaying inside bars 839
in bar-line charts 849
in horizontal bar charts 933, 985
in vertical bar charts 934
weighted 884
weighted (GBARLINE) 863
weighted (GRADAR) 1266
step codes 564
step plots 427
STEP= option, SYMBOL statement 431
stock market high, low, close data 419
STOP statement
QUIT statement, GREPLAY procedure 1300
storing
Annotate graphics 787
clipped polygons 590
fonts 1503
graphics catalogs 1554
graphics output as files 552
PPD file 597
rendered font files 606, 607
strings
appending to graphics data records 559
prefixing output records 567
sending to devices or files 560, 561
stroked fonts 1069
style attribute
versus device entry parameters 274
style attributes 275
overriding with statement options 280
style elements 275
for device-based output 289
GraphColors 287
GraphFonts 288
modifying 286
style templates 275
STYLE variable, Annotate facility 729, 730, 731
STYLE= option
ODS destination statements 279
STYLE= option, AXIS statement options 359
styles 51, 274
See also appearance of graphs
changing current style with STYLE= option 279
changing default style in SAS registry 279
changing font specifications used by 310
equivalent examples of output using different styles 277
modifying 284
modifying fonts or colors 285
modifying GraphFonts and GraphColors style elements 286
ODS destinations and default styles 275
precedence of appearance option specifications 281
recommended 276
specifying 279
turning off 298
viewing list of styles provided by SAS 284
SUBGROUP variable, Annotate facility 733
SUBGROUP= option
BAR statement (GBARLINE) 846
BLOCK statement (GCHART) 898
HBAR and VBAR statements 810
HBAR, HBAR3D, VBAR, VBAR3D statements 801
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 928
PIE, PIE3D, DONUT statements (GCHART) 953
subgrouping
3-D vertical bar charts 982
area bar charts 808, 810
block charts 977
donut or pie charts 994
subordinate statements 43
SUM option
BAR statement (GBARLINE) 846
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 929
PLOT statement (GBARLINE) 858
sum statistic 822, 846, 858, 884
in bar charts 980
in block charts 975
in pie charts 991
in star charts 999
numeric variable for 846
SUMLABEL= option
HBAR, HBAR3D statements (GCHART) 930
summarization of tiles by GTILE procedure 1347
SUMVAR= option
BAR statement (GBARLINE) 846
BLOCK statement (GCHART) 898
CHART statement (GRADAR) 1264
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 930
PIE, PIE3D, DONUT statements (GCHART) 953
PLOT statement (GBARLINE) 858
STAR statement (GCHART) 971
suppressing axes 1022
BUBBLE statement 1155
PLOT statement, G3D procedure 1392
PLOT statement, G3D procedure 1184
PLOT2 statement, G3D procedure 1199
SCATTER statement, G3D procedure 1400
surface maps
axes, controlling 1386
rotating and tilting 1385
surface plots 1381, 1404, 1409, 1410, 1412, 1414
appearance of surface 1395
data ranges 1385
input data sets 1384
rotating 1405
three-dimensional, syntax for 1394
tilting 1407
SVG device
OPTIONS for presentations 171
SVG device driver
ACTXIMG, JAVAIMG vs. 156 chart descriptions for 190
data tips for 191
developing web presentations 156
drill-down links in output 197
drill-down links, additional enhancements 198
SVG devices
SVG universal printers, differences 142
SVG documents
adding pop-up images to 198
SVG files
browser support for 149
Mozilla Firefox and 150
multiple pages, example 147
output from devices 142
system options for 152
terminology 143
SVG graphics
animating 167
animating annotation labels 173
background images, adding 145
definition 141
embedding 144
graph size 152
mime content type 149
multiple images per file, example 147
reasons for creating 142
resolution 151
SVG graphs
data tips 159
drill-down links 159
SVG, PNG, GIF, and TIFF graphs 153
SWAP 612
SWAP function, Annotate facility 701
SWAP= graphics option 612
SWAP= option, GDEVICE procedure 612
SWFONTTRENDER 613
SWFONTTRENDER= graphics option 613
SYMBOL 436, 614
SYMBOL definitions 1189, 1206
displaying values of 1125
GBARLINE procedure 860
plot requests assigning 1189
SYMBOL definitions, BY statement with 374
symbol fonts
creating figures for 1094
SYMBOL function, Annotate facility 701, 1492
SYMBOL option
LEGEND statement options 385
PROC GOPTIONS statement 1129
SYMBOL statement 44, 412, 436
ActiveX and Java support for 1451
altering or canceling 437
box plots, modifying (example) 476
contour plots and 1030
rotating plot symbols through colors
(example) 473
using 437
SYMBOL= graphics option 614
SYMBOL= option, GDEVICE procedure 614
SYMBOLS 614
SYMBOLS= option, GDEVICE procedure 614
system fonts 300
system options
DEVICE= 629
for GIF presentations 171
for SVG device presentations 171
GSTYLE 630
replaying GSLIDE procedure output in
style templates 275
transporting 1553
Templates window
viewing list of styles provided by SAS 284
templates, SAS/GRAPH
listing 1282
terminal device driver 629
terminating drivers 547
terminology
fonts 1068
graphics output 106
text
angle of 590
animating Annotate labels 683
as axis values 353, 358, 359
as legend values 383, 387
axis text, formatting 361
block charts 901
BY lines 372
color of 835, 853
donut chart labels 955
for contour labels 1031
HTML entities 267
in Annotate graphics output 681
legend text, formatting 389, 391
text color 535
text description suboptions
AXIS statement 368
text slides for presentations 1333, 1336
Annotate graphics, displaying 1334, 1342
producing (example) 1340
TEMPLATE DESIGN window 1311
template entries 1281
template procedure
modifying styles 284
viewing list of styles provided by SAS 284
tEMPLATE statement
replaying graphics output in 1323
template procedure
GREPLAY procedure 1307
TEMPLATE= option
PROC GREPLAY statement 1289
templates 1281
assigning 1307
copying or duplicating 1301
creating 1316, 1318
defining or modifying in catalogs 1302
deleting 1307
GREPLAY template code 1533
managing 1314
TICK= option, AXIS statement options 365
tick mark values
GREPLAY procedure 1103
tick marks
color of 853
minor 842, 856
tick marks, axes 351, 365
formatting 365
offset 353
scatter plots 1403
suboptions, list of 1443
surface plots 1393
with datetime values (example) 465
TILE statement
GTILE procedure 1355, 1365
tile summarization
by GTILE procedure 1347
TILELEGEND= option
CHART statement (GRADAR) 1264
TILELEGEND= option
CHART statement (GRADAR) 1264
tiling radar charts 1269
TILT= option
PLOT statement (G3D) 1392
SCATTER statement (G3D) 1402
tilting
surface and scatter plots 1385
surface plots 1407
TITLE definitions
displaying values of 1125
TITLE option
PROC GOPTIONS statement 1129
TITLE statement 44, 447, 462
ActiveX and Java support for 1453
BY statement with 374
displaying with GOPTIONS procedure 1129
enhancing titles (example) 485
titles 462
angle of rotation 449, 456
boxes around 451
colors for 453, 535, 536
default characteristics, setting 463
defining text of 460
enhancing (example) 485
fonts 556
fonts for 454
fonts, color, and size (ODS output) 102
hyperlinks for 448, 458
justification 455
ODS output 101
placement in graphics output area 76
positioning 458
size of 454, 575
spacing around 458
text breaks 455
underlining 461
TOGGLE statement
GTILE procedure 1355, 1365
traditional SAS/GRAPH devices 87
traffic light KPI charts 1102, 1123
TRAILER 616
TRAILER records 616, 617
TRAILER= option, GDEVICE procedure 616
TRAILERFILE 617
TRAILERFILE= option, GDEVICE procedure 617
translation table, ASCII-to-EBCDIC 618
TRANSPARENCY 617
TRANSPARENCY GOPTIONS statement 617
transparency, image 617
transparency, specifying 319
transporting and converting graphics
output 1551
TRANTAB 618
TRANTAB= graphics option 618
TRANTAB= option, GDEVICE procedure 618
tray, paper 588
TREPLAY statement
GREPLAY procedure 1307
troubleshooting
Annotate data sets 654
Web output 263
TrueType fonts
supplied by SAS 300
two-dimensional bar charts 336
two-sided printing 526, 547
TXT2CNTL function, Annotate facility 704
TYPE 618
TYPE= option
BAR statement (GBARLINE) 847
BLOCK statement (GCHART) 898
GDEVICE procedure 618
GKPI procedure 1116
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 930
PIE, PIE3D, DONUT statements (GCHART) 953
PLOT statement (GBARLINE) 858
STAR statement (GCHART) 971
U
U option
UNIFORM option, PROC GFONT statement 1080
U= option
TITLE, FOOTNOTE, and NOTE statements 461
UCC 619
UCC values 619
UCC= graphics option 619
UCC= option, GDEVICE procedure 619
ULX= option
TDEF statement (GREPLAY) 1306
ULY= option
TDEF statement (GREPLAY) 1306
unclipped polygons, storing 590
UNDERLIN= option, TITLE, FOOTNOTE, and NOTE statements 461
underlining in titles, footnotes, and notes 461
Unicode code points 304
Unicode encoding 304
uniform fonts 1068
UNIFORM option
   GPLOT procedure 373
   PROC GFONT statement 1080
   PROC GPLOT statement 1143
UNIFORMAXES option
   PROC GBARLINE statement 828
units of measurement 568
   Annotate graphics 647
Universal Printer shortcut devices 87, 89
universal printers, SVG 142
URL= option
   BAR statement (GBARLINE) 847
   BLOCK statement (GCHART) 899
   CHART statement (GRADAR) 1265
   HBAR, HBAR3D, VBAR, and VBAR3D statements (GCHART) 931
   PIE, PIE3D, DONUT statements (GCHART) 954
   PLOT statement (GBARLINE) 859
   PLOT statement (GPLOT) 1185
   PLOT2 statement (GPLOT) 1200
   STAR statement (GCHART) 972
URLs
   drill-down 866
URX= option
   TDEF statement (GREPLAY) 1306
URY= option
   TDEF statement (GREPLAY) 1306
user input, enabling 620
user-defined fonts
   storing 1069
user-defined colors
   GBARLINE procedure 826
user-defined control characters, device
   storing 619
user-defined lines
   GBARLINE procedure 826
user-defined outlines
   GBARLINE procedure 826
user-defined patterns 886, 900, 935
   GBARLINE procedure 826
user-defined symbols
   GBARLINE procedure 826
USERINPUT 620

V
V= option
   SYMBOL statement 432
V= option, SYMBOL statement 439
V= options
   PATTERN statement options 403
V6COMP 622
V6COMP graphics option 622
value descriptions on legends
   splitting of 386
value-spacing
   legend values 386
value-splitting
   legend value descriptions 386
VALUE= option
   AXIS statement 359, 1443
   LEGEND statement 1450
   LEGEND statement options 387
   PATTERN statement 400
   PIE, PIE3D, DONUT statements (GCHART) 954
   STAR statement (GCHART) 972
   SYMBOL statement 432, 439
values on axes
   order of 359
   splitting (multiline) 358
values on legends
   order of 383
   size and shape of 385
   spacing of 386
variables
   Annotate facility 640, 649, 652, 706
classification, plotting 1135
declaring as plot point labels 430
link and enhancement variables in presentations 192
plotting three variables (example) 1226
plotting two variables (example) 1214
variance 425
VAXIS= option
   BUBBLE statement (GPLOT) 1155
   BUBBLE2 statement (GPLOT) 1165
   PLOT statement (GCONTOUR) 1023
   PLOT statement (GPLOT) 1186
   PLOT2 statement (GPLOT) 1201
   VBAR and VBAR3D statements
      drill-down functionality in bar chart (example) 204
   GAREABAR procedure 798
   GCHART procedure 1457
   URL= option 931
   VBAR statement
      GCHART procedure 901, 932
   VBAR3D statement
      GCHART procedure 901, 932
   VBULLET statement
GKPI procedure 1111
vector formats 235
vector graphics files, rendering software
fonts 613
Version 6, SAS/GRAPH
defaults for programs 622
vertical area bar charts
GAREABAR procedure 798
vertical axes
multiple in plots 1205
vertical axes, multiple in plots 1137, 1211
vertical bar charts
BAR statement, GBARLINE procedure 1454
creating 932
statistics in 934
subgroup labels (example) 661
subgrouping 3-D charts 982
vertical bar-line charts 848
vertical variables
G3GRID procedure 1419
vertices, maximum drawn 583
VM=
  VMINOR= procedure option 1023, 1156, 1165, 1186, 1201
  VMINOR = option
  BUBBLE2 statement (G PLOT) 1165
  VMINOR= option
  BUBBLE statement (G PLOT) 1156
  PLOT statement (GCONTOUR) 1023
  PLOT statement (GPLOT) 1186
  PLOT2 statement (GPLOT) 1201
VORIGIN 620
VORIGIN= graphics option 620
VPOS 621
VPOS= option 621
VREF = option
  BUBBLE2 statement (G PLOT) 1165
  VREF= option
  BUBBLE statement (G PLOT) 1156
  PLOT statement (GCONTOUR) 1023
  PLOT statement (GPLOT) 1186
  PLOT2 statement (GPLOT) 1201
VREVERSE option
  BUBBLE statement (G PLOT) 1156
  BUBBLE2 statement (G PLOT) 1166
  PLOT statement (GCONTOUR) 1023
  PLOT statement (GPLOT) 1186
  PLOT2 statement (GPLOT) 1201
VSIZE 622
VSIZE= graphics option 622
setting size of graphics area 113
VSIZE= option, GDEVICE procedure 622
VSLIDER statement
GKPI procedure 1111
VTRAFFICLIGHT statement
GKPI procedure 1111
VZERO option
  BUBBLE statement (G PLOT) 1156
  BUBBLE2 statement (G PLOT) 1166
  PLOT statement (G PLOT) 1186
  PLOT2 statement (G PLOT) 1202
W
W= option
  AXIS statement options 367
  SYMBOL statement 436
WAUTOHREF= option
  BUBBLE statement (G PLOT) 1156
  PLOT statement (GCONTOUR) 1023
  PLOT statement (GPLOT) 1187
WAUTOREF= option
  BAR statement (GBARLINE) 847
  HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 931
  PLOT statement (GBARLINE) 859
WAUTOVREF= option
  BUBBLE statement (G PLOT) 1156
  BUBBLE2 statement (G PLOT) 1166
  PLOT statement (GCONTOUR) 1023
  PLOT statement (GPLOT) 1187
  PLOT2 statement (GPLOT) 1202
WAXIS=
  WFRAME= procedure option 1265
Weather font 1502
web output
  Annotate variables for 652
developing with ACTXIMG and
  JAVAIMG drivers 157
  enhancing with SVG, PNG, and GIF
  drivers 156
  HTML files, generating with SVG,
  PNG, GIF, and TIFF 118
  SVG, PNG, GIF, and TIFF graphs 153
troubleshooting 263
web pages
  bar chart with drill-down (example) 506
  combining graphs and reports (example) 495
  sending output to 55
web presentations
  adding drill-down links to 159
  creating animated images for 167
weighted statistics
  calculating (GBARLINE) 822, 838, 854
calculating (GRADAR) 1266
GBARLINE procedure 863
GCHART procedure 884
Index 1623

WFRAME= option
CHART statement (GRADAR) 1265
GSLIDE procedure 1339, 1340
WGROUPREF= option
HBAR, VBAR statements (GCHART) 931
WHEN variable, Annotate facility 653, 736
WHERE statement 46
RUN-group processing with 68
white and black, reversing 612
WHREF= option
BBUBBLE statement (GPOINT) 1156
PLOT statement (GCONTOUR) 1024
PLOT statement (GPOINT) 1187
PLOT statement (GPLOT) 1202
WIDTH variable, Annotate facility 737
width variables
GAREABAR procedure 796
WIDTH= option
AXIS statement 361
AXIS statement options 367
BAR statement (GBARLINE) 848
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 932
SYMBOL statement 436
WIDTHSTAT= option
HBAR, HBAR3D, VBAR, VBAR3D statements 802
wind rose charts 1265, 1275
wind speed and direction charts 1275
windowing mode
GDEVICE procedure 1047, 1048
WINDROSE option
CHART statement (GRADAR) 1265
WINDROSECIRCLES= option
CHART statement (GRADAR) 1265
WMIDPOINTREF= option
HBAR, VBAR statements (GCHART) 931
workaround for note behind graph 396
WOUTLINE= option
BAR statement (GBARLINE) 848
BLOCK statement (GCHART) 899
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 932
PIE, PIE3D, DONUT statements (GCHART) 954
STAR statement (GCHART) 972
WREF= option
BAR statement (GBARLINE) 848
HBAR, HBAR3D, VBAR, VBAR3D statements (GCHART) 932
PLOT statement (GPOINT) 859
WSPOKES= option
CHART statement (GRADAR) 1265
WSTARCIRCLES= option
CHART statement (GRADAR) 1265
WSTARS= option
CHART statement (GRADAR) 1265
WSTAT= option, HBAR and VBAR statements
chart with subgroups and variable percentages 810
WVREF= option
BBUBBLE statement (GPOINT) 1157
BBUBBLE2 statement (GPOINT) 1166
PLOT statement (GCONTOUR) 1024
PLOT statement (GPOINT) 1187
PLOT2 statement (GPOINT) 1202

X
X variable, Annotate facility 738
XAXIS= option
PLOT statement (G3D) 1393
SCATTER statement (G3D) 1402
XC variable, Annotate facility 739
XLAST variable, Annotate facility 740
XLATEN= option
TDEF statement (GREPLAY) 1306
XLATENY= option
TDEF statement (GREPLAY) 1306
XLASTT variable, Annotate facility 741
XMAX 623
XMAX= graphics option 623
setting resolution 115
XMAX= option, GDEVICE procedure 623
XPXELS 624
XPXELS= graphics option 624
setting resolution 115
setting size of graph 113
XPXELS= option, GDEVICE procedure 624
XSYS variable, Annotate facility 741
XTICKNUM= option
PLOT statement (G3D) 1393
PLOT statement (GCONTOUR) 1024
SCATTER statement (G3D) 1403
XYTYPE= option
PLOT statement (G3D) 1393

Y
Y variable, Annotate facility 744
YAXIS= option
PLOT statement (G3D) 1393
SCATTER statement (G3D) 1403
YC variable, Annotate facility 745
YLAST variable, Annotate facility 740
YLSTT variable, Annotate facility 741
YMAX 625
YMAX= graphics option 625
setting resolution 115
YMAX= option, GDEVICE procedure 625
YPixels 626
YPixels= graphics option 625, 626
setting resolution 115
setting size of graph 113
YPixels= option, GDEVICE procedure 625, 626
YSYS variable, Annotate facility 746
YTICKNUM= option
PLOT statement (G3D) 1393
PLOT statement (GCONTour) 1024
SCATTER statement (G3D) 1403

Z
Z device drivers 93
Z variable, Annotate facility 747

z/OS
JAVAIMG driver in 158
ZAxis= option
PLOT statement (G3D) 1393
SCATTER statement (G3D) 1403
zero values
block charts 901
GTLE procedure 1350
ZMAX= option
PLOT statement (G3D) 1394
SCATTER statement, G3D procedure 1403
ZMIN= option
PLOT statement (G3D) 1394
SCATTER statement (G3D) 1403
ZSYS variable, Annotate facility 747
ZTickNum= option
PLOT statement (G3D) 1394
SCATTER statement (G3D) 1403
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brands and product names are trademarks of their respective companies. © 2015 SAS Institute Inc. All rights reserved. 20170718SUS3813