

CHAPTER

8

SAS/GRAPH Statements

<i>Overview</i>	161
<i>AXIS Statement</i>	162
<i>Description</i>	162
<i>Syntax</i>	163
<i>Options</i>	164
<i>Text Description Suboptions</i>	172
<i>Using Text Description Suboptions</i>	175
<i>Tick Mark Description Suboptions</i>	175
<i>Using the AXIS Statement</i>	176
<i>Assigning AXIS Definitions</i>	177
<i>BY Statement</i>	177
<i>Description</i>	177
<i>Syntax</i>	177
<i>Required Arguments</i>	178
<i>Options</i>	178
<i>Preparing Data for BY-Group Processing</i>	178
<i>Controlling BY Lines</i>	179
<i>Suppressing the BY line</i>	179
<i>Suppressing the name of the BY variable</i>	179
<i>Controlling the appearance of the BY line</i>	179
<i>Naming the Catalog Entries</i>	179
<i>Using the BY Statement</i>	180
<i>With the GCHART Procedure</i>	180
<i>With the GMAP Procedure</i>	180
<i>With the GPLOT Procedure</i>	180
<i>With the RUN Groups</i>	181
<i>With the Annotate Facility</i>	181
<i>With TITLE, FOOTNOTE, and NOTE Statements</i>	181
<i>With PATTERN and SYMBOL Definitions</i>	181
<i>FOOTNOTE Statement</i>	182
<i>GOPTIONS Statement</i>	182
<i>Description</i>	182
<i>Syntax</i>	183
<i>Options</i>	186
<i>Using the GOPTIONS Statement</i>	186
<i>Graphics Option Processing</i>	186
<i>LEGEND Statement</i>	187
<i>Description</i>	187
<i>Syntax</i>	187
<i>Options</i>	188
<i>Text Description Suboptions</i>	194

<i>Using Text Description Suboptions</i>	197
<i>Using the LEGEND Statement</i>	197
<i>Positioning the Legend</i>	198
<i>Positioning the Legend on the Graphics Output Area</i>	198
<i>Using POSITION= and OFFSET=</i>	198
<i>Using ORIGIN=</i>	199
<i>Relating Legends to Other Graphic Elements</i>	199
<i>Interactions Between POSITION= and MODE=</i>	199
<i>Creating Drop Shadows and Block Effects</i>	199
NOTE Statement	200
ODS HTML Statement	200
Description	200
Syntax	200
Required Arguments	201
Options	204
<i>Using the ODS HTML Statement</i>	209
<i>Specifying a Destination for ODS HTML Output</i>	209
<i>Submitting Multiple ODS HTML Statements</i>	209
<i>About Anchors</i>	210
<i>How ODS Constructs Links and References</i>	210
PATTERN Statement	211
Description	212
Syntax	212
Options	213
<i>Using the PATTERN Statement</i>	219
<i>Altering or Canceling PATTERN Statements</i>	219
<i>About Default Patterns</i>	220
<i>How Default Patterns and Outlines Are Generated</i>	220
<i>Things That Affect Default Patterns</i>	221
<i>Working with PATTERN Statements</i>	221
<i>Explicitly Specifying Patterns</i>	222
<i>Generating Multiple Pattern Definitions</i>	222
<i>Selecting an Appropriate Pattern</i>	222
<i>Controlling Outline Colors</i>	222
<i>The Effect of the CPATTERN= Graphics Option</i>	223
<i>Specifying Version 6 Patterns</i>	223
<i>Specifying Device-Dependent Hardware Patterns</i>	223
GDDM Drivers	223
TEK42xx Series Terminal Drivers	224
HPLJxxxx Drivers	224
Metagraphics Drivers	224
<i>Understanding Pattern Sequences</i>	224
<i>Generating Pattern Sequences</i>	224
<i>Repeating Pattern Sequences</i>	225
SYMBOL Statement	226
Description	226
Syntax	226
Options	227
<i>Using the SYMBOL Statement</i>	243
<i>Altering or Canceling SYMBOL Statements</i>	244
<i>Controlling Consecutive SYMBOL Statements</i>	245
<i>Setting Definitions for PROC GPLOT</i>	245
<i>Specifying Plot Symbols</i>	246
<i>Specifying a Default Interpolation Method</i>	246

<i>Sorting Data with Spline Interpolation</i>	246
<i>Using Color</i>	247
<i>Specifying Colors with SYMBOL Statements</i>	247
<i>Specifying Color with CSYMBOL=</i>	248
<i>Specifying Line Types</i>	248
<i>Using Generated Symbol Sequences</i>	249
<i>Default Symbol Sequences</i>	250
<i>Symbol Sequences Generated from SYMBOL Statements</i>	250
TITLE, FOOTNOTE, and NOTE Statements	251
<i>Description</i>	252
<i>Syntax</i>	252
<i>Options</i>	253
<i>Using TITLE and FOOTNOTE Statements</i>	263
<i>Using the NOTE Statement</i>	264
<i>Using Multiple Options</i>	264
<i>Setting Defaults</i>	265
<i>Using Options That Can Reset Other Options</i>	265
<i>Substituting BY Line Values in a Text String</i>	266
<i>Example 1. Ordering Axis Tick Marks with SAS Datetime Values</i>	266
<i>Example 2. Specifying Logarithmic Axes</i>	269
<i>Example 3. Rotating Plot Symbols through the Colors List</i>	271
<i>Example 4. Creating and Modifying Box Plots</i>	273
<i>Example 5. Filling the Area between Plot Lines</i>	276
<i>Example 6. Enhancing Titles</i>	278
<i>Example 7. Using BY-group Processing to Generate a Series of Charts</i>	280
<i>Example 8. Creating a Simple Web Page with the ODS HTML Statement</i>	284
<i>Example 9. Combining Graphs and Reports in a Web Page</i>	287
<i>Example 10. Creating a Bar Chart with Drill-down for the Web</i>	294
<i>Details</i>	297
<i>Building an HREF value</i>	297
<i>Creating an image map</i>	298
<i>Referencing SAS/GRAPH output</i>	298
<i>See Also</i>	299

Overview

SAS/GRAPH programs can use some of the SAS language statements that you typically use with the base SAS procedures or with the DATA step, such as LABEL, WHERE, and FORMAT. These statements are described in the *SAS Language Reference: Dictionary*.

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

AXIS

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

FOOTNOTE

adds footnotes to graphics output. This statement is like the TITLE statement and is described in that section.

GOPTIONS

submits graphics options that control the appearance of graphics elements by specifying characteristics such as 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.

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, NOTE is like the TITLE statement and is described in that section.

PATTERN

controls the color and fill of patterns assigned to areas in charts, maps, and plots.

SYMBOL

specifies the shape and color of plot symbols as well the interpolation method for plot data. It also controls the appearance of lines in contour plots.

TITLE

add titles to graphics output. The section describing the TITLE statement includes the FOOTNOTE and NOTE statements.

These statements are described in this chapter, which also includes 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.

ODS HTML

generates one or more files written in Hyper Text Markup Language (HTML). If you use it with SAS/GRAPH procedures, you can specify one of the device drivers GIF, ACTIVEX, or JAVA (ACTIVEX and JAVA are only available with GCHART, GCONTOUR, GMAP, GPLOT, and G3D). With the GIF device driver, the graphics output is stored in GIF files. With the ACTIVEX device driver, graphics output is stored as ActiveX controls. With the JAVA device driver, graphics output is stored as Java applets. The HTML files that are generated reference the graphics output. When viewed with a Web browser, the HTML files can display graphics and non-graphics output together on the same Web page.

AXIS Statement

The AXIS statement controls the location, values, and appearance of the axes in plots and charts.

Used by:

GCHART, GCONTOUR, and GPLOT procedures

Global

Description

AXIS statements specify the characteristics of an axis, including:

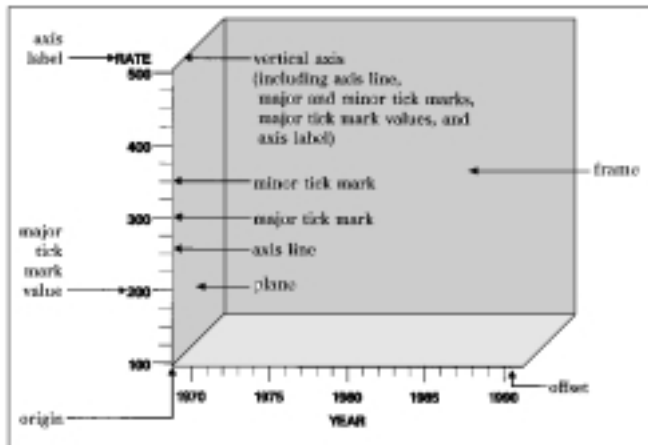
- the way the axis is scaled

- how the data values are ordered
- the location and appearance of the axis line and the tick marks
- the text and appearance of the axis label and major tick mark values.

AXIS definitions are used only when they are explicitly assigned by an option in a procedure that produces graphs with axes.

Figure 8.1 on page 163 illustrates the terms associated with the various parts of axes.

Figure 8.1 Parts of Axes



Syntax

AXIS<1...99><options>;

option(s) can be one or more options from any or all of the following categories:

- axis scale options:
 - LOGBASE=*base* | E | PI
 - LOGSTYLE=EXPAND | POWER
 - ORDER=(*value-list*)
- appearance options:
 - COLOR=*axis-color*
 - LENGTH=*axis-length* <*units*>
 - NOBRACKETS
 - NOPLANE
 - OFFSET=(<*n1*><,n2 >><*units*> | (<*n1*<*units*>><,n2<*units*>>))
 - ORIGIN=<(x<,y >><*units*> | (<x<*units*>><,y<*units*>>))
 - STYLE=*line-type*
 - WIDTH=*thickness-factor*
- tick mark options:
 - MAJOR=(*tick-mark-suboption(s)*) | NONE
 - MINOR=(*tick-mark-suboption(s)*) | NONE
- text options:
 - LABEL=(*text-argument(s)*) | NONE

REFLABEL=(*text-argument(s)*) | NONE

SPLIT="*split-char*"

VALUE=(*text-argument(s)*) | NONE

Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters
IN	inches
PCT	percentage of the graphics output area
PT	points

If you omit *units*, a unit specification is searched for in this order:

- 1 GUNIT= in a GOPTIONS statement
- 2 the default unit, CELLS.

COLOR=*axis-color*

C=*axis-color*

specifies the color for all axis components (the axis line, all tick marks, and all text) unless you include a more explicit AXIS statement color specification. Any of these color specifications override COLOR= for the specified item:

Table 8.1

Option	Items Affected
AXIS statement:	axis label
LABEL=(COLOR= <i>color</i>)	reference-line labels
REFLABEL=(COLOR= <i>color</i>)	major tick mark values
VALUE=(COLOR= <i>color</i>)	
calling procedure:	all axis text (AXIS label and major tick mark value descriptions)
CTEXT=	
CAXIS=	axis line and major and minor tick marks

If you omit all color options, the AXIS statement looks for a color specification in this order:

- 1 the CTEXT= graphics option in a GOPTIONS statement.
- 2 If CTEXT= is not used, the color of all axis components is the first color in the colors list, except for PROC GCONTOUR, which uses the second color.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266

LABEL=(*text-argument(s)*) | NONE

modifies an axis label. *Text-argument(s)* defines the appearance or the text of an axis label, or both. NONE suppresses the axis label. *Text-argument(s)* can be one or more of these:

'text-string'

provides up to 256 characters of label text. By default, the text of the axis label is either the variable name or a previously assigned variable label. Enclose each string in quotes. Separate multiple strings with blanks.

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

ANGLE=*degrees*

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height* <*units*>

JUSTIFY=LEFT | CENTER | RIGHT

ROTATE=*degrees*

See “Text Description Suboptions” on page 172 for a complete description.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266, “Example 2. Specifying Logarithmic Axes” on page 269, and “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

LENGTH=*axis length* <*units*>

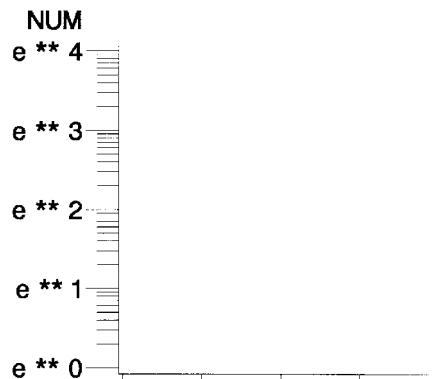
specifies the length of the axis in number of units. If you request a length that cannot fit the display, an error message is issued and no graph is drawn.

Featured in: “Example 2. Specifying Logarithmic Axes” on page 269 and “Example 9. Combining Graphs and Reports in a Web Page” on page 287

LOGBASE=*base* | E | PI

scales the axis values logarithmically according to the value specified. *Base* must be greater than 1. How the values are displayed on the axis depends on the LOGSTYLE= option. For example, LOGBASE=E with the default LOGSTYLE=EXPAND generates an axis like the one in Figure 8.2 on page 165.

Figure 8.2 Axis Generated with LOGBASE=E and LOGSTYLE=EXPAND



Featured in: “Example 2. Specifying Logarithmic Axes” on page 269

LOGSTYLE=EXPAND | POWER

specifies whether the values displayed on the logarithmic axis are the values of the base or the values of the power. LOGSTYLE= is meaningful only when you use LOGBASE=.

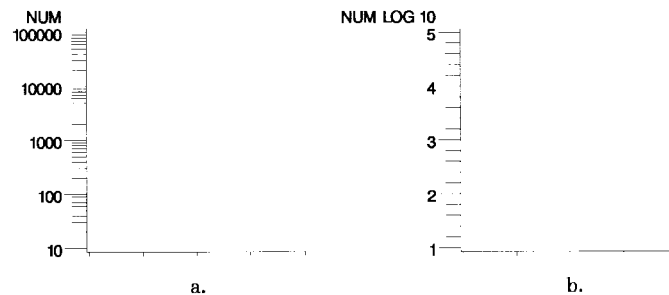
LOGSTYLE=EXPAND specifies that the values displayed are the values of the base raised to successive powers and that the minor tick marks are logarithmically placed. For example, if the base is 10, the values displayed are 10, 100, 1000, 10000, and so on. The default is LOGSTYLE=EXPAND. This statement generates an axis like the one in part (a) of Figure 8.3 on page 166:

```
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 8.3 on page 166:

```
axis logbase=10 logstyle=power;
```

Figure 8.3 Axes Generated with the LOGSTYLE=Option



If you use ORDER= with a logarithmic axis, the values specified by ORDER= must match the style specified by LOGSTYLE=. 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 ORDER= 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.

Featured in: “Example 2. Specifying Logarithmic Axes” on page 269

MAJOR=(tick-mark-suboption(s)) | NONE

modifies the major tick marks. *Tick-mark-suboption(s)* defines the color, size, and number of the major tick marks. NONE suppresses all major tick marks, although

the values represented by those tick marks are still displayed.

Tick-mark-suboption can be

COLOR=*tick-color*

HEIGHT=*tick-height* <*units*>

NUMBER=*number-of-ticks*

WIDTH=*thickness-factor*

See “Tick Mark Description Suboptions” on page 175 for complete descriptions.

List all suboptions and their values within the parentheses.

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MAJOR= because the axis does not use tick marks.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266, “Example 2. Specifying Logarithmic Axes” on page 269, and “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

MINOR=(*tick-mark-suboption(s)*) | NONE

modifies the minor tick marks that appear between major tick marks.

Tick-mark-suboption(s) defines the color, number, or size of the minor tick marks.

NONE suppresses all minor tick marks. *Tick-mark-suboption* can be

COLOR=*tick-color*

HEIGHT=*tick-height* <*units*>

NUMBER=*number-of-ticks*

WIDTH=*thickness-factor*

See “Tick Mark Description Suboptions” on page 175 for complete descriptions.

List all suboptions and their values within the parentheses.

AXIS definitions assigned to the group axis of a bar chart by the GAXIS= option ignore MINOR= because the axis does not use tick marks.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266, “Example 2. Specifying Logarithmic Axes” on page 269, and “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

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.

See also: GROUP= on page 550 and GAXIS= on page 549

NOPLANE

removes either the horizontal or vertical 3D 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 3D planes use the NOAXIS option in the procedure. To remove the backplane, use the NOFRAME option in the procedure.

Featured in: “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

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, and 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 3 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.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266

ORDER=(*value-list*)

specifies the order in which data values appear on the axis. The values specified by ORDER= are the major tick mark values. You can modify the appearance of these values with the VALUE= option.

The way you specify *value-list* depends on the type of variable:

- For numeric variables, *value-list* is either an explicit list of values or a starting and an ending value with an interval increment, or a combination of both forms:

```
n <...n>
```

```
n TO n <BY increment>
```

```
n<...n> TO n <BY increment > <n <...n > >
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

Values must be listed in either ascending or descending order. By default the increment value is 1. You can use a negative integer for *increment* to specify a value list in descending order. In all forms, multiple *n* values can be separated by blanks or commas. Here are some examples:

```
order=(2 4 6)
order=(6,4,2)
order=(2 to 10 by 2)
order=(50 to 10 by -5)
```

If the specified range is not evenly divisible by the increment value, the highest value displayed on the axis is the last incremental value below the ending value for the range. For example, this value list produces a maximum axis value of 9:

```
order=(0 to 10 by 3)
```

- For character variables, *value-list* is a list of unique character values enclosed in quotes and separated by blanks:

```
'value-1' <... 'value-n'>
```

If a character variable has an associated format, the specified values must be the *formatted* values.

Character values can be specified in any order, but the character strings must match exactly the variable values in case and spelling. For example,

```
order=('Paris' 'London' 'Tokyo')
```

Observations can be inadvertently excluded if entries in the *value-list* are misspelled or if the case does not match exactly.

- For date and time values, *value-list* can have the following forms:

```
'SAS-value'i <... 'SAS-value'i>
```

```
'SAS-value'i TO 'SAS-value'i <BY interval>
```

```
'SAS-value'i
```

is any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX. Enclose the value in quotes and specify one of the following for *i*:

D	date
T	time
DT	datetime

```
interval
```

is one of the valid arguments for the INTCK or INTNX functions. These are the default intervals:

DAY	default interval for date
SECOND	default interval for time
DTSECOND	default interval for datetime

These value lists use SAS date and time values:

```
order=('25MAY98'd '04JUL98'd '07SEP98'd)
order=('01JUL97'd to '01AUG97'd)
order=('01JUL97'd to '01JAN98'd by week)
order=('9:25't to '11:25't by minute)
order=('04JUN97:12:00:00'dt to
      '10JUN9712:00:00'dt by dtday)
```

With SAS date and time values, use a FORMAT statement so that the tick mark values have an understandable form. For more information on SAS date and time values, see *SAS Language Reference: Dictionary*.

With any type of *value-list*, specifying values that are not distributed uniformly or are not in ascending or descending order, generates a warning message in the SAS log. The specified values are spaced evenly along the axis even if the values are not distributed uniformly.

Using ORDER= to restrict the values displayed on the axis may result in clipping. For example, if the data range is 1 to 10 and you specify ORDER=(3 TO 5), only the data values from 3 to 5 appear on the plot or chart. For charts, the omitted values are still included in the statistic calculation.

Note: Values out of range do not always produce a warning message in the SAS log. Δ

CAUTION:

The **ORDER=** option does not calculate midpoint values; as a result it is not interchangeable with the **MIDPOINTS=** option in the **GCHART** procedure. Δ

You can use **ORDER=** to specify the order in which the midpoints are displayed on a chart, but do not use it to calculate midpoint values. Be sure that the values you specify match the midpoint values that are calculated either by default by the **GCHART** procedure or by the **MIDPOINTS=** option. For details, see the description of **MIDPOINTS=** for the appropriate statement in Chapter 13, “The **GCHART** Procedure,” on page 519.

ORDER= overrides the suboption **NUMBER=** described in “Tick Mark Description Suboptions” on page 175.

ORDER= is not valid with the **ASCENDING**, **DESCENDING** and **NOZEROS** options used with the bar chart statements in the **GCHART** procedure.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266, “Example 5. Filling the Area between Plot Lines” on page 276, and “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

ORIGIN=($\langle x \rangle \langle , \rangle \langle y \rangle \langle units \rangle$ | ($\langle x \rangle \langle units \rangle \langle , \rangle \langle y \rangle \langle units \rangle \langle \rangle$)

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. **ORIGIN=** 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, as in this example:

```
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.

REFLABEL=(*text-argument(s)*) | NONE

creates and defines the appearance of a reference-line label. *Text-argument(s)* defines the appearance or the text of the label, or both. NONE suppresses the reference-line label. *Text-argument(s)* can be one or more of these:

'text-string'

provides up to 256 characters of label text. By default, a reference line does not have a label. Enclose each string in quotes. Separate multiple strings with blank spaces; the strings are applied to the reference lines consecutively along the axis, from the plot origin to the end of the axis.

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

ANGLE=*degrees*
 AUTOREF
 COLOR=*text-color*
 FONT=*font* | NONE
 HEIGHT=*text-height* <*units*>
 JUSTIFY=LEFT | CENTER | RIGHT
 POSITION=TOP | MIDDLE | BOTTOM
 ROTATE=*degrees*
 T=*n*

See “Text Description Suboptions” on page 172 for a complete description.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

SPLIT="*split-char*"

specifies the split character that the AXIS statement uses to break axis values into multiple lines. *Split-char* can be any character value that can be specified in a SAS character variable. The split character must be embedded in the variable values in the data set or in an associated format. When the AXIS statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the data set contains the value **Berlin, Germany**, and you specify SPLIT=", ". The value would appear on the axis as

```
Berlin
Germany
```

Note that the split character itself is not displayed.

Axis values specified with VALUE= do not use the split character. For example, suppose you specify this statement:

```
axis1 spilt=", " value=(t1='December, 1999');
```

The value will appear on the axis on one line as **December, 1999**. However, any other axis values containing a comma would honor the split character.

Featured in: Example 7 on page 596

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.

See also: Figure 8.22 on page 249 for examples of the available line types

VALUE=(*text-argument(s)*) | NONE

modifies the major tick mark values. That is, this option modifies the text that labels the major tick marks on the axis. *Text-argument(s)* defines the appearance or the text of a major tick mark value, or both. NONE suppresses the major tick mark values, although the major tick marks are still displayed. *Text-argument(s)* can be one or more of these:

'text-string'

provides up to 256 characters of text for the major tick mark value. By default, the value is either the variable value or an associated format value. Enclose each string in quotes and separate multiple strings with blanks.

Specified text strings are assigned to major tick marks in order. If you specify only one text string, only the first tick mark value changes, and all

the other tick mark values display the default. If you specify multiple strings, the first string is the value of the first major tick mark, the second string is the value of the second major tick mark, and so forth. 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 VALUE= 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 ORDER=. To change the value of midpoints in bar charts produced with the GCHART procedure, use the MIDPOINTS= option in the procedure. Δ

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

ANGLE=*degrees*

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height* <*units*>

JUSTIFY=LEFT | CENTER | RIGHT

ROTATE=*degrees*

TICK=*n*

For a complete description, see “Text Description Suboptions” on page 172.

Place text description suboptions before the text strings they modify.

Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Featured in: “Example 2. Specifying Logarithmic Axes” on page 269, “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280, and “Example 9. Combining Graphs and Reports in a Web Page” on page 287

WIDTH=*thickness-factor*

specifies the thickness of the axis line. Thickness increases directly with the value of *thickness-factor*. By default, WIDTH=1.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266

Text Description Suboptions

Text description suboptions are used by the LABEL=, REFLABEL=, and VALUE= options to change the color, height, justification, font, and angle of either default text or specified text strings. See LABEL= on page 164, REFLABEL= on page 170, and VALUE= on page 171.

ANGLE=*degrees*

A=*degrees*

specifies the angle of the *baseline* with respect to the horizontal. A positive value for *degrees* moves the baseline counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal) unless the text is automatically

angled or rotated to avoid overlapping. For an illustration of the effect of ANGLE=, see Figure 8.24 on page 254.

See also: the ROTATE= suboption on page 174

Featured in: Example 7 on page 596

AUTOREF

automatically labels each reference line on an axis with the response value at the reference line's position. AUTOREF is only used with the REFLABEL= option. The automatic labels are applied only to reference lines that do not have specific labels assigned to them. For example, the following option uses the response-axis value as the label for every reference line except the second reference line, which is assigned the label *two*:

```
reflabel=(autoref t=2 "two")
```

See also: the T= suboption on page 175

COLOR=*text-color*

C=*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 default, the first color in the colors list.

FONT=*font* | NONE

F=*font* | NONE

specifies the font for the text. See Chapter 6, "SAS/GRAPH Fonts," on page 125 for details on specifying *font*. If you omit FONT=, a font specification is searched for in this order:

- 1 the FTEXT= option in a GOPTIONS statement
- 2 the default hardware font, NONE.

HEIGHT=*text-height* <*units*>

H=*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.

JUSTIFY=LEFT | CENTER | RIGHT

J=L | C | R

specifies the alignment of the text. The default depends on the option with which it is used and the text it applies to.

- With the LABEL= option:
 - for a left vertical axis label, the default is JUSTIFY=RIGHT
 - for a right vertical axis label, the default is JUSTIFY=LEFT
 - for a horizontal axis label, the default is JUSTIFY=CENTER.
-

With the REFLABEL= option:

- for a vertical axis, the default is JUSTIFY=CENTER. RIGHT places the text string on the right end of the line, CENTER in the middle of the line, and LEFT to the left of the line.
- for a horizontal axis label, the default is JUSTIFY=RIGHT. RIGHT places the text string just to the right of the line, CENTER is centered

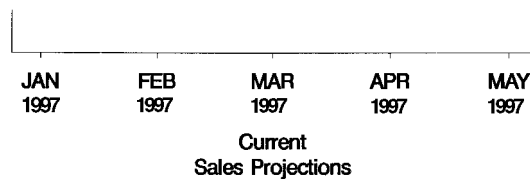
on top of the line, and LEFT places the text string just to the left of the line.

- With the VALUE= option:
 - for numeric variables on a vertical axis, the default is JUSTIFY=RIGHT
 - for character variables on a vertical axis, the default is JUSTIFY=LEFT
 - for all variables on a horizontal axis, the default is JUSTIFY=CENTER.

You can use JUSTIFY= to print multiple lines of text by repeating JUSTIFY= 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 8.4 on page 174.

```
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 8.4 The JUSTIFY= Suboption



Specify additional suboptions before any string.

See also: the suboption TICK= on page 175

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. POSITION= is only available on the REFLABEL= option.

- For horizontal reference lines, TOP places the label just above the reference line, MIDDLE places the label on the reference line, and BOTTOM places the label just under the reference line.
- For vertical reference lines, TOP places the label at the top end of the reference line, MIDDLE places the label in the middle of the line, and BOTTOM places the label at the bottom end of the line.

ROTATE=*degrees*

R=*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. For an illustration of the effect of ROTATE=, see Figure 8.31 on page 262.

See also: the ANGLE= suboption on page 172

TICK=*n*

T=*n*

specifies the *n* reference line or tick mark value. Used only with REFLABEL= or with VALUE=

- With REFLABEL=, T= specifies the *n*th reference line. It is used to limit modifications to individual reference lines when there are multiple reference lines on an axis. For example, the following option changes the color of only the third reference line's label and leaves all other reference-line labels unchanged:

```
reflabel=(autoref t=3 color=red)
```

Suboptions that *precede* T= affect all the reference-line labels on an axis. Suboptions that *follow* T= affect only the specified line's label. For example, the following option assigns the color green to all the reference-line labels on an axis, but left-justifies only the third reference line's label:

```
reflabel(c=green "one" "two" t=3 j=left "three")
```

For the options to be applied to a text string, they must precede the quoted string. In the following option, the j=left is ignored because it follows the string:

```
reflabel(c=green "one" "two" t=3 "three" j=left)
```

- With VALUE=, TICK= specifies the *n*th major tick mark value. It is used to designate the tick mark value whose text and appearance you want to modify. For example, the following option changes the color of only the third tick mark value and leaves all others unchanged:

```
value=(tick=3 color=red)
```

Suboptions that *precede* TICK= affect all the major tick mark values. Suboptions that *follow* TICK= affect only the specified value. For example, the following option makes all the major tick mark values 4 units high and colors all of them blue except for the third one, which is red:

```
value=(height=4 color=blue tick=3 color=red)
```

Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=swiss height=4 'Weight'
        justify=right height=3 '(in tons)')
```

FONT=SWISS applies to both **Weight** and **(in tons)**. HEIGHT=4 affects **Weight**, but is respecified as HEIGHT=3 for **(in tons)**. JUSTIFY=RIGHT affects only **(in tons)**.

Tick Mark Description Suboptions

Tick mark description suboptions are used by MAJOR= and MINOR= to change the color, height, width, and number of the tick marks to which they apply. See MAJOR= and MINOR=.

COLOR=tick-mark-color

C=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 default, the first color in the colors list.

HEIGHT=tick-height <units>

H=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 *MAJOR=* the default height .5 CELLS.
- With *MINOR=* the default height .25 CELLS.

If you specify a negative number, tick marks are drawn inside the axis.

NUMBER=number-of-ticks

N=number-of-ticks

specifies the number of tick marks to be drawn. With *MAJOR=*, *number-of-ticks* must be greater than 1. With *MINOR=*, *number-of-ticks* must be greater than 0.

With *MAJOR=*, the *NUMBER=* suboption can be overridden by a major tick mark specification in the procedure, which in turn can be overridden by *ORDER=*.

With *MINOR=*, the *NUMBER=* suboption can be overridden by a minor tick mark specification in the procedure.

NUMBER= is not valid with logarithmic axes.

WIDTH=thickness-factor

W=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*.

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 one replaces the previously defined statement of the same number. An *AXIS* statement without a number is treated as an *AXIS1* statement.

Cancel individual *AXIS* statements by defining an *AXIS* statement of the same number without options (a null statement):

```
axis4;
```

Canceling one *AXIS* statement does not affect any other *AXIS* definitions. To cancel all current *AXIS* statements, use the *RESET=* option in a *GOPTIONS* statement:

```
goptions reset=axis;
```

Specifying *RESET=GLOBAL* or *RESET=ALL* cancels all current *AXIS* definitions as well as other settings.

To display a list of current *AXIS* definitions in the LOG window, use the *GOPTIONS* procedure with the *AXIS* option:

```
proc goptions axis nolist;
run;
```

Assigning AXIS Definitions

AXIS definitions must always be explicitly assigned by the appropriate option in the statement that generates the graph. The following table lists the procedures and statements that generate axes, the type of axis, and the statement option that assigns an AXIS definitions to that axis:

Procedure	Statement that generates an axis	Type of axis	Option that assigns an AXIS definition
GCHART	HBAR VBAR	group axis midpoint axis response axis	GAXIS= MAXIS= RAXIS=
GCONTOUR	PLOT	horizontal axis vertical axis	HAXIS= VAXIS=
GPLOT	PLOT	horizontal axis vertical axis	HAXIS= VAXIS=

Some types of axes cannot use certain AXIS statement options:

- group and midpoint axes ignore LOGBASE=, MAJOR=, and MINOR=
- midpoint, horizontal and vertical axes ignore NOBRACKETS.

BY Statement

The BY statement processes data and orders output according to the BY group.

Used by:

GCHART, GCONTOUR, GMAP, GPLOT, GREDUCE, G3D, G3GRID procedures

Description

The BY statement divides the observations from an input data set into groups for processing. Each set of contiguous observations with the same value for a specified variable is called a *BY group*. A variable that defines BY groups is called a *BY variable* and is the variable that is specified in the BY statement. When you use a BY statement, the graphics procedure

- 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, but the effect on the output is different when it is used with SAS/GRAPH procedures. Δ

Syntax

BY<DESCENDING>*variable*

```
<...<DESCENDING>variable-n>
<NOTSORTED>;
```

Required Arguments

variable

specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. By default, the procedure expects observations in the data set to be sorted in ascending order by all the variables that you specify or to be indexed appropriately.

Options

DESCENDING

indicates that the data set is sorted in descending order by the specified variable. The option affects only the variable that immediately follows the option name, and must be repeated before every variable that is not sorted in ascending order. For example, this BY statement indicates that observations in the input data set are arranged in descending order of VAR1 values and ascending order of VAR2 values:

```
by descending var1 var2;
```

This BY statement indicates that the input data set is sorted in descending order of both VAR1 and VAR2 values:

```
by descending var1 descending var2;
```

NOTSORTED

specifies that observations with the same BY value are grouped together, but are not necessarily sorted in alphabetical or numeric order. The observations can be grouped in another way, for example, in chronological order.

NOTSORTED can appear anywhere in the BY statement and affects all variables specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

The requirement for ordering or indexing observations according to the values of BY variables is suspended when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. For NOTSORTED, the procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same value for the BY variables are not contiguous, the procedure treats each new value it encounters as the first observation in a new BY group and will create a graph for that value, even if it is only one observation.

Preparing Data for BY-Group Processing

Unless you specify the NOTSORTED option, observations in the input data set must be in ascending numeric or alphabetic order. To prepare the data set, either sort it with the SORT procedure using the same BY statement that you plan to use in the target SAS/GRAPH procedure or create an appropriate index on the BY variables.

If the procedure encounters an observation is out of the proper order, it issues an error message.

If you need to group data in some other order, such as chronological 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 280.

Controlling BY Lines

By default, the BY statement prints a BY line above each graph that contains the variable name followed by an equal sign and the variable value. For example, if you specify BY SITE in the procedure, the default heading when the value of SITE is **London** would be SITE=London.

Suppressing the BY line

To suppress the entire BY line, use the NOBYLINE option in an OPTION statement or specify HBY=0 in the GOPTIONS statement. See “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280.

Suppressing the name of the BY variable

To suppress the variable name and the equal sign in the heading and leave only the BY value, use the LABEL statement to assign a null label ('00'X) to the BY variable. For example, this statement assigns a null label to the SITE variable:

```
label site='00'x;
```

See also Example 12 on page 618.

Controlling the appearance of the BY line

To control the color, font, and height of the BY lines, use the following graphics options in a GOPTIONS statement:

CBY=BY-line-color
specifies the color for BY lines.

FBY=font
specifies the font for BY lines.

HBY=n<units>
specifies the height for BY lines.

See Chapter 9, “Graphics Options and Device Parameters Dictionary,” on page 301 for a complete description of each option.

Naming the Catalog Entries

The catalog entries generated with BY-group processing always use incremental naming. This means that the first entry created by the procedure uses the base name and subsequent entries increment that name. The base name is either the default entry name for the procedure (for example, GPLOT) or the name specified with the NAME= option in the action statement. Incrementing the base name automatically appends a number to each subsequent entry (for example, GPLOT1, GPLOT2, and so forth). See also “Names and Descriptions of Catalog Entries” on page 51 and “Using the default output name” on page 59. For an example of incremented catalog names, see “Example 9. Combining Graphs and Reports in a Web Page” on page 287.

Using the BY Statement

This section describes the following:

- the effect of BY-group processing on the GCHART, GMAP, and GPLOT procedures
- the interaction between BY-group and RUN-group processing
- the requirements for using BY-group processing with the Annotate facility
- how to include BY information in titles, notes, and footnotes
- how patterns and symbols are assigned to BY-groups
- the effect of using BY-group processing with the ODS HTML statement.

For additional information on any of these topics, refer to the appropriate chapter.

With the GCHART Procedure

When you use BY-group processing with the GCHART procedure, you can do the following:

- 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.
- With the BLOCK statement, you can use the BLOCKMAX= option to produce the same block-height scaling in all block charts in a BY group.
- With the HBAR or VBAR statement, you can use the RAXIS= option to produce the same response axis scaling in all horizontal or vertical bar charts in a BY group.

With the PIE and STAR statements, the effect of a BY statement is similar to that of the GROUP= option, except that the GROUP= option allows you to put more than one graph on a single page while the BY statement does not. Do not use a BY variable as the group variable in STAR or PIE statements.

With the GMAP Procedure

By default, BY-group processing affects both the map data set and the response data set. This means that you get separate, individual output for each map area common to both data sets. For example, if the map data set REGION contains six states and the response data set contains the same six states, and you specify BY STATE in the GMAP procedure, you get six graphs with one state on each graph.

If you use the ALL option in the PROC GMAP statement and you also use the BY statement, you get one output for each map area in the response data set, but that output displays all the map areas in the map data set. Only one map area per output contains response data information; the others are empty. For example, if you create a block map using the data sets REGION and SALES, specify BY STATE, and include the ALL option in the PROC GMAP statement, you get six graphs with six states on each graph. One state per graph has a block; the remaining five are empty.

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 may vary from graph to graph, but UNIFORM forces the scaling to be the same for all graphs generated by the procedure.

With the RUN Groups

If you use the BY statement with a procedure that processes data and supports RUN-group processing (the GCHART, GMAP, and GPLOT procedures), then each time you submit an action statement or a RUN statement you get a separate graph for each value of the BY variable. For example, each of these two RUN-groups produces a separate plot for every value of the BY variable SITE:

```
/* first run group*/
proc gplot data=sales;
  title1 'Sales Summary';
  by site;
  plot sales*model_a;
run;

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

The BY statement stays in effect for every subsequent RUN group until you submit another BY statement or exit the procedure. Variables in subsequent BY statements replace any previous BY variables.

You can also turn off BY-group processing by submitting a null BY statement (BY;) in a RUN group, but when you do this, the null BY statement turns off BY-group processing *and* the RUN group generates a graph.

For more information, see “RUN-Group Processing” on page 28.

With the Annotate Facility

If a procedure that is using BY-group processing also specifies annotation with the ANNOTATE= option in the PROC statement, the same annotation is applied to every graph generated by the procedure.

If you specify annotation with the ANNOTATE= option in the action statements for a procedure, the BY-group processing is applied to the Annotate data set. In this way, you can customize the annotation for the output from each BY group by including the BY variable in the Annotate data set and by using each BY-variable value as a condition for the annotation to be applied to the output for that value.

With TITLE, FOOTNOTE, and NOTE Statements

TITLE, FOOTNOTE, and NOTE statements can automatically include the BY variable name, BY variable values, or BY lines in the text they produce. To insert BY variable information into the text strings used by these statements, use the #BYVAR, #BYVAL, and #BYLINE substitution options. For details, see the description of the text-string argument on page 262. For an example, see “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280.

With PATTERN and SYMBOL Definitions

Procedures that use SYMBOL or PATTERN definitions, assign 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:

```
PATTERN1 is assigned to East,
PATTERN2 is assigned to North,
```

PATTERN3 is assigned to **South**,
 PATTERN4 is assigned to **West**.

If you create sets of graphs from several data sets containing the variable REGION, and if you want the same pattern assigned to the same region each time, you must be sure that REGION always has the same four values. Otherwise, the patterns may not be the same across graphs. For example, if the value **North** is missing from the data, the patterns are assigned as follows:

PATTERN1 is assigned to **East**,
 PATTERN2 is assigned to **South**,
 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

See “TITLE, FOOTNOTE, and NOTE Statements” on page 251.

GOPTIONS Statement

The 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

Global

Description

The GOPTIONS statement specifies values for *graphics options*. Graphics options control characteristics of the graph, such as size, colors, type fonts, fill patterns, and symbols. 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 allows 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

- override default values for graphics options that control either graphics attributes or device parameters for a single graph or for an entire SAS session
- reset individual graphics options or all graphics options to their default values
- cancel definitions for AXIS, FOOTNOTE, PATTERN, SYMBOL, and TITLE statements.

To change device parameters permanently, you must use the GDEVICE procedure to modify the appropriate device entry or to create a new one. See Chapter 15, “The GDEVICE Procedure,” on page 651 for details.

To review the current settings of all graphics options, use the GOPTIONS procedure. See Chapter 20, “The GOPTIONS Procedure,” on page 795 for details.

Syntax

GOPTIONS< *options-list*>;

options-list can be one or more options from any or all of the following categories:

□ reset option

RESET=ALL | GLOBAL | *statement-name* | (*statement-name(s)*)

□ options that affect the appearance of the display area and the graphics output

ASPECT=*scaling-factor*

AUTOSIZE=ON | OFF | DEFAULT

BORDER | NOBORDER

CELL | NOCELL

GSIZE=*lines*

HORIGIN=*horizontal-offset* <IN | CM>

HPOS=*columns*

HSIZE=*horizontal-size* <IN | CM>

ROTATE=LANDSCAPE | PORTRAIT

ROTATE | NOROTATE

SWAP | NOSWAP

TARGETDEVICE=*target-device-entry*

VORIGIN=*vertical-offset* <IN | CM>

VPOS=*rows*

VSIZE=*vertical-size* <IN | CM>

XMAX=*width* <IN | CM>

XPIXELS=*width-in-pixels*

YMAX=*height* <IN | CM>

YPIXELS=*height-in-pixels*

□ options that affect color

CBACK=*background-color*

CBY=*BY-line-color*

COLORS=<(colors-list | NONE)>

CPATTERN=*pattern-color*

CSYMBOL=*symbol-color*

CTEXT=*text-color*

CTITLE=*title-color*

PENMOUNTS=*active-pen-mounts*

PENSORT | NOPENSORT

□ options that control font selection or text appearance

CHARTYPE=*hardware-font-chartype*

FASTTEXT | NOFASTTEXT

FBY=*BY-line-font*

FCACHE=*number-fonts-open*

FONTRIS=NORMAL | PRESENTATION

FTEXT=*text-font*

FTITLE=*title-font*

FTRACK=LOOSE | NONE | NORMAL | TIGHT | TOUCH | V5

HBY=*BY-line-height* <units>
 HTEXT=*text-height* <units>
 HTITLE=*title-height* <units>
 RENDER=APPEND | DISK | MEMORY | NONE | READ
 RENDERLIB=*libref*
 SIMFONT=*software-font*

- options that set defaults for procedures and global statements
 - GUNIT=*units*
 - INTERPOL=*interpolation-method*
 - OFFSHADOW=(*x* <units>, *y* <units> | (*x,y*) <units>
 - V6COMP | NOV6COMP
- image animation options
 - DELAY=*delay-time*
 - DISPOSAL=NONE | BACKGROUND | PREVIOUS | UNSPECIFIED
 - INTERLACED | NONINTERLACED
 - ITERATION=*iteration-count*
 - TRANSPARENCY | NOTRANSPARENCY
- options that affect how your SAS/GRAPH program runs
 - DISPLAY | NODISPLAY
 - ERASE | NOERASE
 - GWAIT=*seconds*
 - GRAPHRC | NOGRAPHRC
 - PCLIP | NOPCLIP
 - POLYGONCLIP | NOPOLYGONCLIP
- options that control how output is sent to devices or files
 - ADMGDF | NOADMGDF
 - DEVADDR=*device-address*
 - DEVICE=*device-entry*
 - DEVMAP=*device-map-name* | NONE
 - EXTENSION=*'file-type'*
 - FILECLOSE=DRIVERTERM | GRAPHEND
 - FILEONLY | NOFILEONLY
 - GACCESS=*output-format* | '*output-format* > *destination*'
 - GEND=*'string'* <...'*string-n*'>
 - GPILOG=*'string'* <...'*string-n*'>
 - GOUTMODE=APPEND | REPLACE
 - GPROLOG=*'string'* <...'*string-n*'>
 - GPROTOCOL=*module-name*
 - GSFLEN=*record-length*
 - GSFMODE=APPEND | PORT | REPLACE
 - GSFNAME=*fileref*
 - GSFPROMPT | NOGSFPROMPT
 - GSTART=*'string'* <...'*string-n*'>
 - HANDSHAKE=HARDWARE | NONE | SOFTWARE | XONXOFF
 - KEYMAP=*map-name* | NONE

- POSTGPEILOG=*'string'*
- POSTGPROLOG=*'string'*
- PREGPEILOG=*'string'*
- PREGPROLOG=*'string'*
- PROMPTCHARS=*'prompt-chars-hex-string'X*
- options that specify hardware capabilities of the device
 - CHARACTERS | NOCHARACTERS
 - CIRCLEARC | NOCIRCLEARC
 - DASH | NODASH
 - DASHSCALE=*scaling-factor*
 - FILL | NOFILL
 - FILLINC=0...9999
 - LFACTOR=*line-thickness-factor*
 - PIEFILL | NOPIEFILL
 - POLYGONFILL | NOPOLYGONFILL
 - SYMBOL | NOSYMBOL
- options that control printer hardware features
 - AUTOCOPY | NOAUTOCOPY
 - AUTOFEED | NOAUTOFEED
 - BINDING=DEFAULTEDGE | LONGEDGE | SHORTEGE
 - COLLATE | NOCOLLATE
 - DUPLEX | NODUPLEX
 - GCOPIES=(*<current-copies><,max-copies>*)
 - PAPERDEST=*bin*
 - PAPERFEED=*feed-increment* <IN | CM>
 - PAPERLIMIT=*width* <IN | CM>
 - PAPERSIZE=*'size-name'* | (*width,height*)
 - PAPERSOURCE=*tray*
 - PAPERTYPE=*'type-name'*
 - PPDFILE=*fileref* | *'external-file'*
 - REPAINT=*redraw-factor*
 - REVERSE | NOREVERSE
 - SPEED=*pen-speed*
 - UCC=*'control-characters-hex-string'X*
- options that interact with the operating environment
 - DRVINIT=*'system-command(s)'*
 - DRVTERM=*'system-command(s)'*
 - PREGRAPH=*'system-command(s)'*
 - POSTGRAPH=*'system-command(s)'*
 - PROMPT | NOPROMPT
- options for mainframe systems
 - GCLASS=*SYSOUT-class*
 - GDDMCOPY=FSCOPY | GSCOPY
 - GDDMNICKNAME=*nickname*
 - GDDMTOKEN=*token*

GDEST=*destination*
 GFORMS=*'forms-code'*
 GWRITER=*'writer-name'*
 TRANTAB=*table* | *user-defined-table*

Options

See Chapter 9, “Graphics Options and Device Parameters Dictionary,” on page 301 for a complete description of all graphics options used by the GOPTIONS statement.

Using the GOPTIONS Statement

GOPTIONS statements are global and can be located anywhere in your SAS program. However, for the graphics options to affect the output from a procedure, the GOPTIONS statement must execute before the procedure.

With the exception of RESET=, graphics options can be listed in any order in a GOPTIONS statement. RESET= should be the first option in the GOPTIONS statement.

A graphics option remains in effect until you either specify the option in another GOPTIONS statement, or use RESET= to reset the values, or end the SAS session. When a session ends, the values of the graphics options return to their default values.

Graphics options are additive; that is, the value of a graphics option remains in effect until the graphics option is explicitly changed or reset or until you end your SAS session. Graphics options remain in effect even after you submit additional GOPTIONS statements specifying different options.

To reset an individual option to its default value, submit the option without a value (a null graphics option.) You can use a comma (but it is not required) to separate a null graphics option from the next one. For example, this GOPTIONS statement sets the values for background color, text height, and text font:

```
goptions cback=blue htext=6 pct ftext=zapf;
```

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. Δ

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

The LEGEND statement controls the location and appearance of legends on two-dimensional plots, contour plots, maps, and charts.

Used by:

GCHART, GCONTOUR, GMAP, and GPLOT procedures

Global

Description

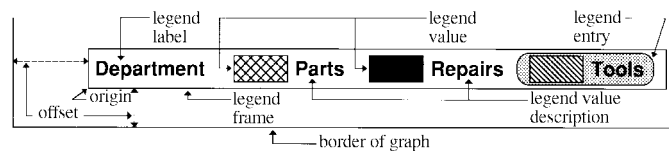
LEGEND statements specify the characteristics of a legend but do not create legends. These characteristics are

- 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.

illustrates the terms associated with the various parts of a legend.

Figure 8.5 Parts of a Legend



Syntax

LEGEND<1...99><options>;

option(s) can be one or more options from any or all of the following categories:

- appearance options
 - ACROSS=*number-of-columns*
 - CBLOCK=*block-color*
 - CBORDER=*frame-color*
 - CFRAME=*background-color*
 - CSHADOW=*shadow-color*
 - DOWN=*number-of-rows*

FRAME

FWIDTH=*thickness-factor*

SHAPE=BAR(*width,height*) <units> | LINE(*length*) <units> |
SYMBOL(*width,height*) <units>

□ position-options

MODE=PROTECT | RESERVE | SHARE

OFFSET=(<*x*><,y >><units > | (<*x* <units >><,y <units >>)

ORIGIN=(<*x*><,y >><units > | (<*x* <units >><,y <units >>)

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>
<INSIDE | OUTSIDE>)

□ text-options

LABEL=(*text-argument(s)*) | NONE

ORDER=(*value-list*)

VALUE=(*text-argument(s)*) | NONE

Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters
IN	inches
PT	points
PCT	percentage of the graphics output area

If you omit *units*, a unit specification is searched for in this order:

- 1 GUNIT= in a GOPTIONS statement
- 2 the default unit, CELLS.

ACROSS=*number-of-columns*

specifies the number of columns to use for legend entries.

Featured in: “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284

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*).

CBLOCK= and CSHADOW= are mutually exclusive. If both are present, SAS/GRAPH software uses the last one specified. CBLOCK= is usually used in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See also: “Creating Drop Shadows and Block Effects” on page 199 and the OFFSHADOW=“OFFSHADOW” on page 364 graphics option

CBORDER=*frame-color*

draws a colored frame around the legend. This option overrides the FRAME option. CBORDER= can be used in conjunction with the CFRAME= option.

CFRAME=*background-color*

specifies the background color of the legend. This option overrides the FRAME option. If both CFRAME= and FRAME= are specified, only the solid background

produced by CFRAME= is displayed. CFRAME= 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*).

CSHADOW= and CBLOCK= are mutually exclusive. If both are present, SAS/GRAPH uses the last one specified. CSHADOW= is usually specified in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See also: “Creating Drop Shadows and Block Effects” on page 199 and the OFFSHADOW=“OFFSHADOW” on page 364 graphics option

DOWN=*number-of-rows*

specifies the number of rows to use for legend entries.

FRAME

draws a frame around the legend. The color of the frame is the first color in the colors list.

FWIDTH=*thickness-factor*

specifies the thickness of the frame, where *thickness-factor* is a number. The thickness of the line increases directly with *thickness-factor*. By default, FWIDTH=1.

LABEL=(*text-argument(s)*) | NONE

modifies a legend label. *Text-argument(s)* defines the appearance or the text of a legend label, or both. NONE suppresses the legend label. By default, the text of the legend label is either the variable name or a previously assigned variable label. *Text-argument(s)* can be one or more of these:

'text-string'

provides up to 256 characters of label text. Enclose each string in quotes. Separate multiple strings with blanks.

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height* <*units*>

JUSTIFY=LEFT | CENTER | RIGHT

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)

See “Text Description Suboptions” on page 194 for complete descriptions.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Featured in: “Example 3. Rotating Plot Symbols through the Colors List” on page 271 and “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284

MODE=PROTECT | RESERVE | SHARE

specifies whether or not the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements. MODE= can take one of these values:

PROTECT

draws the legend in the procedure output area, but a *blanking area* surrounds the legend, preventing other graphics elements from being displayed in the legend. (A blanking area is a

protected area in which no other graphics elements are displayed.)

- RESERVE** takes space for the legend from the procedure output area, thereby reducing the amount of space available for the graph. If `MODE=RESERVE` is specified in conjunction with `OFFSET=`, the legend may 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=` 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 which case the default changes to `MODE=PROTECT`.

See also: “Positioning the Legend” on page 198

Featured in: “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284

`OFFSET=($\langle x \rangle$, $\langle y \rangle$) $\langle units \rangle$ | ($\langle x \langle units \rangle \rangle$, $\langle y \langle units \rangle \rangle$)`

specifies the distance to move the entire legend; x is the number of units to move the legend right (positive numbers) or left (negative numbers), and y is the number of units to move the legend up (positive numbers) or down (negative numbers).

To set only the x offset, specify one value, with or without a following comma:

```
offset=(4 cm,)
```

To set both the x and y offset, specify two values, with or without a comma separating them:

```
offset=(2 pct, 4 pct)
```

To set only the y offset, specify one value preceded by a comma:

```
offset=(,-3 pct)
```

`OFFSET=` is usually used in conjunction with `POSITION=` to adjust the position of the legend. Moves are relative to the location specified by `POSITION=`, with `OFFSET=(0,0)` representing the initial position. You can also apply `OFFSET=` to the default legend position.

`OFFSET=` is unnecessary with `ORIGIN=` since `ORIGIN=` explicitly positions the legend and requires no further adjustment. However, if you specify both options, the values of `OFFSET=` are added to the values of `ORIGIN=`, and the `LEGEND` is positioned accordingly.

See also: “Positioning the Legend” on page 198 and the option `POSITION=` on page 191

ORDER=(*value-list*)

selects or orders the legend values that appear in the legend. The way you specify *value-list* depends on the type of variable that generates the legend:

- For numeric variables, *value-list* is either an explicit list of values, or a starting and an ending value with an interval increment, or a combination of both forms:

n <...*n*>

n TO *n* <BY *increment*>

n <...*n*> TO *n* <BY *increment*> <*n* <...*n*>>

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

- For character variables, *value-list* is a list of unique character values enclosed in quotes and separated by blanks:

'*value-1*' <... '*value-n*'>

If a character variable has an associated format, the specified values must be the *formatted* values.

For a complete description of *value-list*, see the ORDER= on page 168 option in the AXIS statement.

Even though ORDER= controls whether a legend value is displayed and where it appears, the VALUE= option controls the text that the legend value displays.

ORIGIN=(<*x*><*y*><*units*> | (<*x* <*units*>><*y* <*units*>>)

specifies the *x* coordinate and the *y* coordinate of the lower-left corner of the legend box. ORIGIN= explicitly positions the legend anywhere on the graphics output area. It is possible to run a legend off the page or overlay the graph.

To set only the *x* coordinate, specify one value, with or without a following comma:

origin=(4 cm,)

To set both the *x* and *y* coordinates, specify two values, with or without a comma separating them:

origin=(2 pct, 4 pct)

To set only the *y* coordinate, specify one value preceded by a comma:

origin=(,3 pct)

ORIGIN= overrides the POSITION= option if both are used. Although using the OFFSET= option with the ORIGIN= option is unnecessary, if OFFSET= is also specified, it is applied after the ORIGIN= request has been processed.

See also: "Positioning the Legend" on page 198

Featured in: "Example 8. Creating a Simple Web Page with the ODS HTML Statement" on page 284

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>
<OUTSIDE | INSIDE>)

positions the legend on the graph. Value for POSITION= are

OUTSIDE or specifies the location of the legend in relation to the axis area.
INSIDE

BOTTOM or specifies the vertical position.
MIDDLE or
TOP

LEFT or specifies the horizontal position.
 CENTER or
 RIGHT

By default, POSITION=(BOTTOM CENTER OUTSIDE). You can change one or more settings. If you supply only one value the parentheses are not required. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Once you assign the initial legend position, you can adjust it with the OFFSET= option.

The ORIGIN= options overrides POSITION=. The value of the MODE= option can affect the behavior of POSITION=.

See also: “Positioning the Legend” on page 198 and the OFFSET= option on page 190 and the MODE= option on page 189

SHAPE=BAR(*width*<units>,<i>height<units>) <units> | LINE(*length*) <units> |
 SYMBOL(*width*<units>,<i>height<units>) <units>

specifies the size and shape of the legend values displayed in each legend entry. The value you specify for SHAPE= depends on which procedure generates the legend.

BAR(*width,height*)<units>

is used with the GCHART and GMAP procedures, the GPLOT procedure if you use the AREAS= option, and the GCONTOUR procedure if you use the PATTERN option. Each legend value is a bar of the specified width and height. By default, *width* is 5, *height* is 0.8, and *units* are CELLS. You can specify *units* for the *width,height* pair or for the individual coordinates.

Featured in: “Example 3. Rotating Plot Symbols through the Colors List” on page 271 and “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284

LINE(*length*) <units>

is used with the GPLOT and GCONTOUR procedures. Each legend value is a line of the length you specify. Plotting symbols are omitted from the legend values. By default, *length* is 5 and *units* are CELLS. You can specify *units* for *length*.

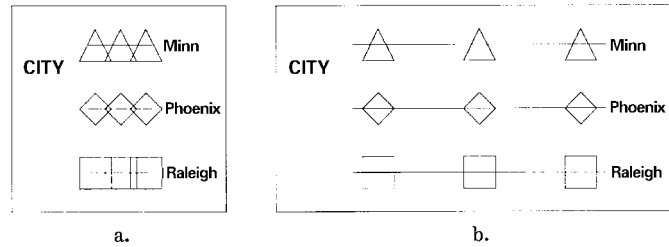
SYMBOL(*width*<units>,<i>height<units>) <units>

is used with the GPLOT procedure. Each legend value (*not* each symbol) is the width and height you specify. For example, this specification produces legend values like the ones in Figure 8.6 on page 193(a):

```
shape=symbol(.5,.5)
```

This specification produces legend values like the ones in Figure 8.6 on page 193(b):

```
shape=symbol(2,.5)
```

Figure 8.6 Legend Values Produced with SHAPE= SYMBOL

By default, *width* is 5, *height* is 1, and *units* are CELLS. You can specify *units* for the *width,height* pair or for the individual coordinates.

Featured in: “Example 3. Rotating Plot Symbols through the Colors List” on page 271

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.

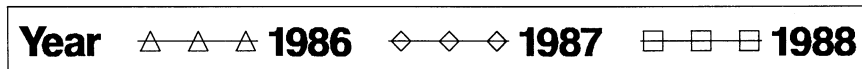
NONE suppresses the value descriptions although the legend values (bars, lines, and so on) are still displayed. *Text-argument(s)* can be one or more of these:

'text-string'

provides up to 256 characters of text for the value description. Enclose each string in quotes. Separate multiple strings with blanks.

Specified text strings are assigned to the legend values in order. If you submit only one string, only the first legend entry uses the value of that string. If you specify multiple strings, the first string is the text for the first entry; the second string is the text for the second entry; and so forth. For example, this specification produces legend entries like those shown in Figure 8.7 on page 193:

```
value=('1986' '1987' '1988')
```

Figure 8.7 Specifying Value Descriptions with the VALUE= Option

text-description-suboption

modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be

COLOR=*text-color*

FONT=*font* | NONE

HEIGHT=*text-height* <*units*>

JUSTIFY=LEFT | CENTER | RIGHT

TICK=*n*

See “Text Description Suboptions” on page 194 for complete descriptions.

Place text description suboptions before the text strings they modify.

Suboptions not followed by a text string affect the default values. To specify

and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

To order or select legend entries, use the ORDER= option.

See also: “Text Description Suboptions” on page 194 and the option ORDER= on page 191

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= on page 189 and VALUE= on page 193.

COLOR=*text-color*

C=*text-color*

specifies the color of the text. If you omit COLOR=, 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 default, the first color in the colors list.

FONT=*font* | NONE

F=*font* | NONE

specifies the font for the text. See Chapter 6, “SAS/GRAPH Fonts,” on page 125 for details on specifying *font*. If you omit FONT=, a font specification is searched for in this order:

- 1 the FTEXT= option in a GOPTIONS statement
- 2 the default hardware font, NONE.

HEIGHT=*text-height* <*units*>

H=*text-height* <*units*>

specifies the height of the text characters in the 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.

JUSTIFY=LEFT | CENTER | RIGHT

J=L | C | R

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 JUSTIFY= to print multiple lines of text by repeating JUSTIFY= before the text string for each line. For example, this statement produces a legend label and value descriptions like those shown in Figure 8.8 on page 195:

```
legend label=(justify=c 'Distribution'
              justify=c 'Centers')
value=(tick=1 justify=c 'Portland,'
       justify=c 'Maine'
       tick=2 justify=c 'Paris,'
       justify=c 'France'
       tick=3 justify=c 'Sydney,'
       justify=c 'Australia');
```

Figure 8.8 Specifying Multiple Lines of Text with the JUSTIFY= Suboption

Specify additional suboptions before any string.

See also: the TICK= suboption on page 196

POSITION=(\langle BOTTOM | MIDDLE | TOP \rangle \langle LEFT | CENTER | RIGHT \rangle)

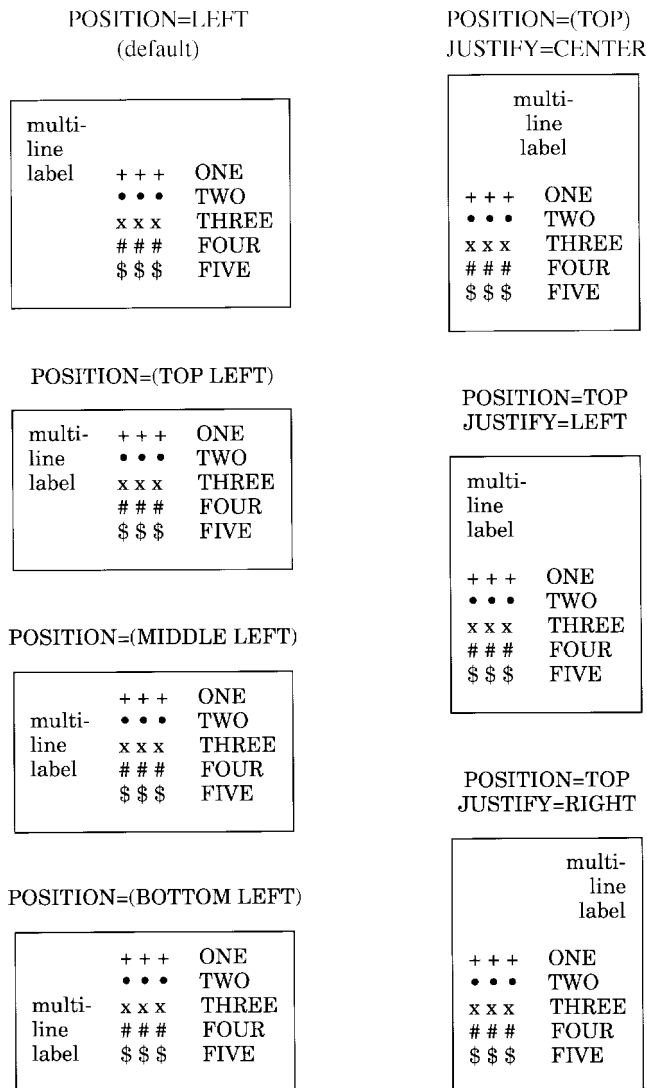
places the legend label in relation to the legend entries. The POSITION= suboption is used only with the LABEL= option. By default, POSITION=LEFT.

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 8.9 on page 196 shows some of the ways POSITION= affects a multi-line legend label in which the entries are stacked in a column (ACROSS=1). This figure uses a label specification such as

```
label=('multi-'  
      justify=left 'line'  
      justify=left 'label'  
      position=left)
```

In this specification, POSITION= specifies the default value, LEFT, which is represented by the first legend in the figure. The value of POSITION= is indicated above each legend. The default justification is used unless you also use the JUSTIFY= suboption.

Figure 8.9 Using the POSITION= Suboption with Multi-line Legend Labels

In addition, specifying POSITION=RIGHT mirrors the effect of POSITION=LEFT, and specifying POSITION=BOTTOM mirrors the effect of POSITION=TOP.

TICK=*n*

T=*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

```
value=(tick=3 'Minneapolis')
```

The characteristics of all other value descriptions remain unchanged.

If you use TICK= when you designate text for one legend entry, you must also use it when you designate text for any additional legend entries. For example, this option changes the text of both the second and third legend entries:

```
value=(tick=2 'Paris' tick=3 'Sydney')
```

If you omitted TICK=3, the text of the second legend entry would be **ParisSydney**.

Text description suboptions that *precede* TICK= 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* TICK= affect only the specified legend entry. For example, suppose you specify this option for a legend with three entries:

```
value=(color=red font=swiss tick=2 color=blue)
```

The text of all three entries would use the Swiss font; the first and third entries would be red and only the second entry would be blue.

Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=swiss height=4 'Weight'
        justify=right height=3 '(in tons)')
```

FONT=SWISS applies to both **Weight** and **(in tons)**. HEIGHT=4 affects **Weight**, but is respecified as HEIGHT=3 for **(in tons)**. JUSTIFY=RIGHT affects only **(in tons)**.

Using the 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 one replaces the previously defined statement of the same number. A LEGEND statement without a number is treated as a LEGEND1 statement.

Cancel individual LEGEND statements by defining a LEGEND statement of the same number without options (a null statement):

```
legend4;
```

Canceling one LEGEND statement does not affect any other LEGEND definitions. To cancel all current LEGEND statements, use RESET= in a GOPTIONS statement:

```
goptions reset=legend;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current LEGEND definitions as well as other settings.

To display a list of current LEGEND definitions in the LOG window, use the GOPTIONS procedure with the LEGEND option:

```
proc goptions legend nolist;
run;
```

Positioning the Legend

By default, the legend shares the procedure output area with the procedure output, such as a map or bar chart. (See “Placement of Graphic Elements in the Graphics Output Area” on page 34.) However, several LEGEND statement options allow you to position a legend anywhere on the graphics output area and even to overlay the procedure output. This section describes these options and their effect on each other.

Positioning the Legend on the Graphics Output Area

There are two ways you can position the legend on the graphics output area:

- Describe the general location of the legend with the POSITION= option. If necessary, fine-tune the position with the OFFSET= option.
- Position the legend explicitly with the ORIGIN=option.

Using POSITION= and OFFSET=

The values of the POSITION= option affect the legend in two ways:

- OUTSIDE and INSIDE determine whether the legend is located outside or inside the axis area.
- BOTTOM or MIDDLE or TOP (vertical position) and LEFT or CENTER or RIGHT (horizontal position) determine where the legend is located in relation to its OUTSIDE or INSIDE position.

Figure 8.10 on page 198 shows the legend positions inside the axis area.

Figure 8.10 Legend Positions Inside the Axis Area

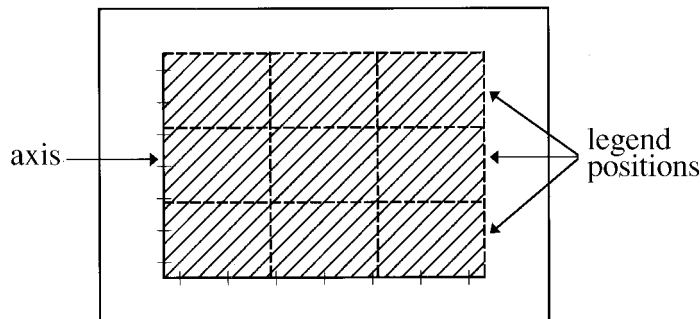
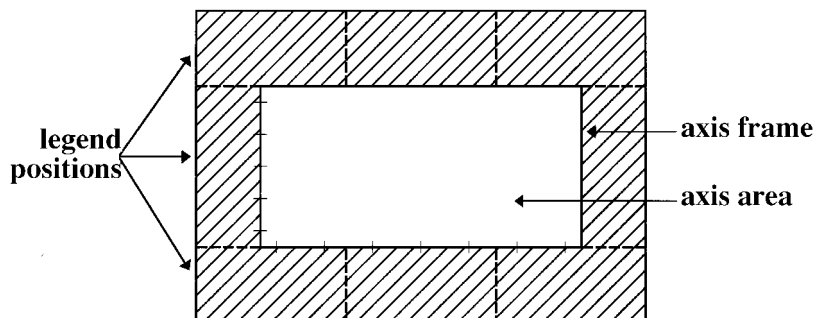


Figure 8.11 on page 198 shows legend positions outside the axis area.

Figure 8.11 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 POSITION=. The x value shifts the legend either left or right and the y value shifts the legend either up or down.

The offset values are always applied *after* the POSITION= request. For example, if POSITION=(TOP RIGHT OUTSIDE), the legend is located in the upper right corner of the graphics output area. If OFFSET=(0,0) is specified, the legend does not move. If OFFSET=(-5,-8)CM, the legend moves 5 centimeters to the left and 8 centimeters down.

Using ORIGIN=

Use ORIGIN=(x,y) to specify the coordinates of the exact location of the lower left corner of the legend box. Because ORIGIN=(0,0) is the lower left corner of the graphics output area, the values of x and y must be positive. If you specify negative values, a warning is issued and the default value is used.

Relating Legends to Other Graphic Elements

By default, the legend is inside the procedure output area and the space it occupies reduces the size of the graph itself. To control the way the legend relates to the other elements of the graph, use the MODE= option. These are values for MODE=:

- RESERVE reserve space for the legend outside the axis area and move the graph to make room for the legend. This is the default setting and is valid only when POSITION=OUTSIDE.
- PROTECT prevents the legend from being overwritten by the procedure output. PROTECT blanks out graphics elements, allowing only legend elements to be displayed in the legend's space.
- SHARE displays both graphics elements and legend elements in the same space. This setting is usually used when the legend is positioned inside the axis area. SHARE is useful when the graph has a space that the legend can fit into. For an example, see "Example 8. Creating a Simple Web Page with the ODS HTML Statement" on page 284.

Interactions Between POSITION= and MODE=

You cannot specify both POSITION=INSIDE and MODE=RESERVE because MODE=RESERVE assumes 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 MODE= to SHARE or PROTECT. Otherwise, SAS/GRAPH issues a warning and automatically changes the value of MODE= to PROTECT.

Creating Drop Shadows and Block Effects

To produce a drop shadow or a three-dimensional block effect behind the legend use the CSHADOW= or CBLOCK= option in the LEGEND statement in conjunction with the graphics option OFFSHADOW=(x,y).

The value of x determines how far the shadow or block extends to the right (positive numbers) or to the left (negative numbers) of the legend. The value of y determines how far the shadow or block extends above (positive numbers) or below (negative numbers) the legend. If OFFSHADOW=(0,0) is specified, the shadow or block is not visible.

By default, OFFSHADOW=(0.0625, -0.0625) IN; that is, the shadow or block extends 1/16th of an inch to the right and 1/16th of an inch below the legend.

NOTE Statement

See “TITLE, FOOTNOTE, and NOTE Statements” on page 251.

ODS HTML Statement

The ODS HTML statement opens, manages, or closes the HTML destination. If the destination is open, it produces output that is written in Hyper Text Markup Language (HTML). If you use it with SAS/GRAPH procedures, you can specify one of the device drivers GIF, ACTIVEX, or JAVA (ACTIVEX and JAVA are only available with GCHART, GCONTOUR, GMAP, GPLOT, and G3D). With the GIF device driver, the graphics output is stored in GIF files. With the ACTIVEX device driver, graphics output is stored as ActiveX controls. With the JAVA device driver, graphics output is stored as Java applets. The HTML files that are generated reference the graphics output. When viewed with a Web browser, the HTML files can display graphics and non-graphics output together on the same Web page.

Used by:

GANNO, GCHART, GCONTOUR, GFONT, GIMPORT, GMAP, GPLOT, GPRINT, GREPLAY, GTESTIT, GSLIDE, G3D, and G3GRID procedures

Requirements:

If the HTML destination is open, the BODY= argument is required.

Operating Environment Information: On mainframes, you must also use PATH= or GPATH= to direct the graphics output. In the CMS operating environment, any file specification for the HTML output files must use the URL suboption in order to form valid URLs for a Web browser. Δ

Description

This section describes the ODS HTML statement as it relates to SAS/GRAPH procedures.

Syntax

```
ODS HTML HTML-file-specification(s) | action
  <ANCHOR=string>
  <ARCHIVE=string>
  <BASE=base-text>
  <GFOOTNOTE | NOGFOOTNOTE>
  <GPATH=graphics-location <(URL=Uniform-Resource-Locator | NONE)>>
  <GTITLE | NOGTITLE>
  <HEADTEXT=HTML-for-document-head>
  <METATEXT=HTML-for-document-head>
  <NEWFILE=starting-point>
  <PARAMETERS=(parameter-name1'=parameter-value-1' ...
  parameter-name-n'=parameter-value-n')>
  <PATH=file-location <(URL=Uniform-Resource-Locator | NONE)>>
  <RECORD_SEPARATOR=string | NONE>
  <STYLE=style-definition>
  <TRANTAB=translation-table>;
```

- *action* can be one of

CLOSE
EXCLUDE
SELECT
SHOW

Note: For information on EXCLUDE, SELECT, and SHOW, see *The Complete Guide to the SAS Output Delivery System*. △

- *HTML-file-specification(s)* can be one or more of

BODY=*file-specification*
CONTENTS=*file-specification*
FRAME=*file-specification*
PAGE=*file-specification*

Note: BODY= is required. If you use FRAME=, you must also use CONTENTS= or PAGE=. △

Required Arguments

One of these arguments is required.

CLOSE

closes the HTML destination and closes any HTML files that are currently open.

Featured in: “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284

EXCLUDE

excludes output objects from the HTML destination.

SELECT

selects output objects to send to the HTML destination.

SHOW

writes to the SAS log the current selection or exclusion list for the HTML destination.

HTML-file-specification

opens the HTML destination and specifies the HTML file or files to write to. You can open up to four HTML files; the file designated by BODY= is required.

Whenever you open one of these files, it remains open until you either

- close the HTML destination with ODS HTML CLOSE
- open a different file for the same HTML file specification.

HTML-file-specification can be one or more of the following arguments. Values for *file-specification* follow the arguments.

BODY=*file-specification*

FILE=*file-specification*

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=GIF, the graphs are stored in separate GIF files. When you view the body file on a browser, the graphs are automatically displayed.

Featured in: “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284 and “Example 10. Creating a Bar Chart with Drill-down for the Web” on page 294

CONTENTS=*file-specification*

identifies the file that contains a table of contents to the ODS output that is produced while the HTML destination is open. The contents file contains links to the body file(s).

The text of links to graphics output is taken from the description field of the GRSEG catalog entry. Use the DESCRIPTION= option in the procedure to change the link text.

You can display a contents file alone or in conjunction with a frame file. If you display a contents file directly (without using a frame file), selecting a link opens the associated body file, and the contents file is no longer displayed. If you display a contents file with a frame file, the contents file always remains available in the left frame, and selecting a link opens the associated body file in the right frame.

FRAME=*file-specification*

identifies a file that points to the body file and to either the table of contents file or the page file, or both. If you specify FRAME=, you must also specify either CONTENTS= or PAGE= or both.

When you open the frame file in the browser, it displays the Table of Contents or the Table of Pages or both in the left frame, and the body file in the right frame.

PAGE=*file-specification*

identifies the file that contains a table of pages to the ODS output that is produced while the HTML destination is open. The pages file contains links to the body file(s). ODS produces a new page of output whenever a procedure explicitly specifies for a new page. The SAS system option PAGESIZE= has no effect on pages in HTML output.

File-specification identifies the file or SAS catalog to write to and can be one of the following:

- *fileref* (<URL='Uniform-Resource-Locator'> <NO_BOTTOM_MATTER> <NO_TOP_MATTER> <DYNAMIC>)
- *external-file* (<URL='Uniform-Resource-Locator'> <NO_BOTTOM_MATTER> <NO_TOP_MATTER> <DYNAMIC>)
- *entry.HTML* (<URL='Uniform-Resource-Locator'> <NO_BOTTOM_MATTER> <NO_TOP_MATTER> <DYNAMIC>)

where

external-file

is the physical name of an external file to write to. For information on specifying external files, see the SAS companion for your operating environment.

fileref

is a fileref that has been assigned to an external file. The fileref must refer to a single file; it cannot point to an aggregate file storage location. Use a FILENAME statement to assign a fileref. See also "FILENAME Statement" on page 24.

entry.HTML

specifies an entry in a SAS catalog to write to. You must also specify a library and catalog. See the discussion on PATH= on page 207.

URL='Uniform-Resource-Locator'

provides a URL for *file-specification*. ODS uses this URL instead of the file name in all the links and references that it creates that point to the file.

Operating Environment Information: In the CMS operating environment, you must use the URL= suboption in the HTML-file specifications for the body, contents, and page files because CMS file names do not form valid URLs. △

This option is useful for building HTML files that may be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), they work as long as the contents, page, and body files are all in the same location.

NO_BOTTOM_MATTER

NOBOT

omits the bottom matter for the file. By default, when you close a file that was open for HTML output of any kind, ODS writes some HTML to the bottom of the file. This HTML ends the file so that it can be viewed cleanly in a browser.

If you wish to leave a file in a state that you can append to, use NO_BOTTOM_MATTER on the BODY= option on the ODS HTML statement that opens the file. This option, in conjunction with NO_TOP_MATTER makes it possible for you to add output to a file that already exists and to put your own HTML code in the file between pieces of output.

To use NO_BOTTOM_MATTER, you must use a fileref for *file-specification*. The FILENAME statement that defines the fileref must include the host-specific option that opens the file for appending.

When you are opening a file that ODS has previously written to, you must use ANCHOR= to specify a new base name for the anchors to avoid duplicating anchors that already exist in the file. (See the discussion of ANCHOR= on page 204.)

NO_TOP_MATTER

NOTOP

omits the opening matter for the file. By default, when you open a file for HTML output of any kind, ODS writes some HTML to the top of the file.

If you wish to append ODS output to an existing file, you must open the file with NO_TOP_MATTER on the BODY= option on the ODS HTML statement that opens the file. This option, in conjunction with NO_BOTTOM_MATTER makes it possible for you to add output to a file that already exists and to put your own HTML code in the file between pieces of output.

To use NO_TOP_MATTER, you must use a fileref for *file-specification*. The FILENAME statement that defines the fileref must include the host-specific option that opens the file for appending.

When you are opening a file that ODS has previously written to, you must use ANCHOR= to specify a new base name for the anchors to avoid duplicating anchors that already exist in the file. (See the discussion of ANCHOR= on page 204.)

DYNAMIC

enables you to send HTML output directly to a web server instead of writing it to a file. This option sets the value of the HTMLContentType= attribute.

By default, if you do not specify DYNAMIC, ODS sets the value of HTMLContentType= for writing to a file.

Note: If you specify the DYNAMIC suboption with any file specification in the ODS HTML statement, you must specify it for all the file specifications in the statement. △

Note: Regardless of how you specify the file, you may need to include the extension .HTML or .HTM on the file name. Some browsers require one of these extensions in order to read the file. Δ

Options

ANCHOR='*anchor-name*'

specifies the base name of the HTML anchor that identifies each piece of output in the body files. An anchor is the name of a link target. Each output object must have an anchor tag for the contents, page, and frame files to link to or to reference. These links and references, which are automatically created by ODS, point to the name of an anchor. Therefore, each anchor name in a file must be unique.

By default, the first anchor name is `IDX`, and all subsequent anchors generated while the HTML destinations remain open increment that name, for example, `IDX1`, `IDX2`, and so forth. Because anchor names continue to increment while the HTML destinations remain open, it can be difficult to predict what the name of a link anchor will be. `ANCHOR=` enables you to control these names.

You can change anchor names as often as you like by submitting the `ANCHOR=` option in an ODS HTML statement anywhere in your program. Once you have specified an anchor name, it remains in effect until you specify a new one. Like the default, specified anchor values increment while the ODS HTML destination remains open or until you assign a new value. For example, if you specify `ANCHOR='SALESCHART'` and submit a procedure that generates three graphs, the anchor names in the body file are `SALESCHART`, `SALESCHART1`, and `SALESCHART2`.

If you open an HTML file to append to it, be sure to specify a new anchor so that you don't write the same anchors to the file again. ODS cannot know about anchors that are already in a file when it opens the file.

See also: the `HTML=` and `HTML_LEGEND=` options in the `GCHART`, `GPLOT`, and `GMAP` procedures "About Anchors" on page 210

ARCHIVE='*string*'

writes to an `<APPLET>` or `<OBJECT>` tag the attribute that specifies the location of the executable files that dynamic SAS/GRAPH output requires. Use `ARCHIVE=` in conjunction with SAS/GRAPH procedures and the `DEVICE=JAVA` or `DEVICE=ACTIVEX` option in the `GOPTIONS` statement.

If you specify `DEVICE=JAVA` in a `GOPTIONS` statement, `ARCHIVE=` provides the argument for the `ARCHIVE` attribute in an `<APPLET>` tag. In this case, *string* specifies the location of the archive file that contains the class files that are necessary to run the applet. (Class files are binary Java files that the browser runs.) If you specify `DEVICE=ACTIVEX` in a `GOPTIONS` statement, `ARCHIVE=` provides the argument for the `CODEBASE` attribute in the `<OBJECT>` tag. This attribute should point to an executable file that installs the ActiveX control on your machine. This attribute is not necessary for generating dynamic SAS/GRAPH output if the ActiveX control is already installed on your machine. However, if you wish to publish your output on the Web or to mail the HTML results to someone else, it is wise to include the `ARCHIVE=` option so that the ActiveX control can, if necessary, be installed on the machine that is reading the results.

The string must be one that the browser can interpret. For instance, if the archive file is local to the machine that you are running SAS on, you can use the `FILE` protocol to identify the file. If you want to point to an archive file that is on a web server, use the `HTTP` protocol.

If you do not specify `ARCHIVE=` and you are using the `JAVA` device driver, ODS uses the value of the SAS system option `APPLETOC=`. This value points to the

location of the Java archive files that ship with the SAS system. To find out what the value of this option is, you can either look in the Options window in the Files folder under Environment Control, or you can submit the following procedure step:

```
proc options option=appletloc;
run;
```

There is no default if you are using the ACTIVEX device driver.

The ARCHIVE attribute is a feature of Java 1.1. Therefore, if you are using the Java device driver, your browser must support this version of Java. Both Internet Explorer 4.01 and Netscape 4.05 support Java 1.1.

BASE=*Uniform-Resource-Locator*

specifies a URL to use as the first part of all links that ODS creates in the HTML files. Consider this specification:

```
BASE='http://www.your-company.com/local-url/'
```

In this case, ODS creates links that begin with the string **http://www.your-company.com/local-url/**. The appropriate *anchor-name* completes the link.

GFOOTNOTE | **NOGFOOTNOTE**

GFOOT | **NOGFOOT**

controls whether the HTML file displays all the titles that are defined by TITLE statements in the SAS/GRAPH program that generates the HTML output. Available only with DEVICE=GIF. GTITLE is the default and displays all currently defined titles within the SAS/GRAPH output (the GIF file) that is displayed by the HTML. NOGTITLE suppresses all TITLE statements in the GIF files that are created by SAS/GRAPH but displays them as part of the HTML (this is the only option with the ACTIVEX and JAVA device drivers).

Footnotes that are displayed by ODS HTML support most SAS/GRAPH FOOTNOTE statement options. The font must be valid for the browser. Options that ODS cannot handle, such as height and text angle specifications, are ignored. For details, see “TITLE, FOOTNOTE, and NOTE Statements” on page 251.

GPATH=*graphics-destination* <(URL=*Uniform-Resource-Locator* | NONE)>

specifies the destination for all graphics output generated by the ODS HTML statement. By default, the output is stored in the location that is specified on PATH=. If you do not specify PATH=, the output is stored using the GSFNAME location. If GSFNAME is not specified, the output is stored in the current directory.

graphics-destination

must be an aggregate storage location, such as directory or PDSE, as specified by one of the following:

fileref

is a fileref that has been assigned to an aggregate file storage location. Use a FILENAME statement to assign a fileref. For more information, see “FILENAME Statement” on page 24.

If you use a fileref in the GPATH= option, ODS does not use information from GPATH= when it constructs links. (See “How ODS Constructs Links and References” on page 210).

storage-location

is the physical name of an aggregate file storage location. For information on specifying a storage location, see the SAS companion for your operating environment.

libref.catalog

specifies a SAS catalog to write to.

Uniform-Resource-Locator

provides a URL for *graphics-destination*. ODS uses this URL instead of the file name in all the links and references that it creates to the file. If you specify the keyword NONE, no information from the GPATH= option appears in the links or references.

Operating Environment Information: In the CMS operating environment, you must use the URL= suboption in the GPATH= option because CMS file names do not form valid URLs. Δ

This option is useful for building HTML files that may be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), they work as long as the contents, page, and body files are all in the same location.

Each output GIF file is named automatically using the catalog entry name as the base name and incrementing the name as necessary. For more information on naming catalog entries and external files, see “Exporting SAS/GRAPH Output with Program Statements” on page 58.

Featured in: “Example 10. Creating a Bar Chart with Drill-down for the Web” on page 294

GTITLE | NOGTITLE

controls whether the HTML file displays all the titles that are defined by TITLE statements in the SAS/GRAPH program that generates the HTML output. Available only with DEVICE=GIF. GTITLE is the default and displays all currently defined titles within the SAS/GRAPH output (the GIF file) that is displayed by the HTML. NOGTITLE suppresses all TITLE statements in the GIF files that are created by SAS/GRAPH but displays them as part of the HTML (this is the only option with the ACTIVEX and JAVA device drivers).

Titles that are displayed by ODS HTML support most SAS/GRAPH TITLE statement options. The font must be valid for the browser. Options that ODS cannot handle, such as height and text angle specifications, are ignored. For details, see “TITLE, FOOTNOTE, and NOTE Statements” on page 251.

Featured in: “Example 8. Creating a Simple Web Page with the ODS HTML Statement” on page 284 and “Example 10. Creating a Bar Chart with Drill-down for the Web” on page 294

HEADTEXT=*HTML-for-document-head*

specifies HTML to place between the <HEAD> and </HEAD> tags in all the HTML files that the HTML destination writes to. Use HEADTEXT= to define programs (for example, java scripts) that you can use later in the file.

Note: ODS cannot parse the HTML that you supply. It should be well-formed HTML that makes sense in the context of the <HEAD> and </HEAD> tags. For information on HTML, refer to one of the many reference books that are available on the subject. Δ

METATEXT=*HTML-for-document-head*

specifies HTML to use as the <META> tag inside the <HEAD> and </HEAD> tags of all the HTML files that the HTML destination writes to. This HTML provides the browser with information about the document that it is loading. This information could include things like the content-type and the character set to use. For information on HTML, refer to one of the many reference books that are available on the subject.

If you do not specify METATEXT=, ODS writes a simple <META> tag, which includes the content-type of the document and the character set to use, to all the HTML files that it creates. It takes the value of the character set from the SAS

registry. If the registry does not specify a character set, ODS writes the HTML only to the contents, frame, and page files.

Note: ODS cannot parse the HTML that you supply. It should be well-formed HTML that makes sense in the context of the <HEAD> and </HEAD> tags. If you are using METATEXT= as it is intended, your HTML should look like this:

```
<META your-metatext-which-could-be-very-long>
```

△

NEWFILE=*starting-point*

creates a new body file at the specified *starting-point*. ODS automatically names new files by incrementing the name of the file specified by BODY=. For example, if you specify BODY='REPORT.HTML', ODS names additional body files REPORT1.HTML, REPORT2.HTML, and so forth. If you end the file name with a digit, ODS begins incrementing with that number. For instance, if you specify BODY='MAY5.HTML', ODS names the first body file MAY5.HTML. Additional files are named MAY6.HTML, MAY7.HTML, and so forth. *Starting-point* can be

NONE	writes all output to the body file that is currently open. This is the default.
PAGE	starts a new body file for each page of output. A page break occurs when a procedure explicitly starts a new page (not because the pagesize was exceeded) or when you start a new procedure.
PROC	starts a new body file each time that you start a new procedure.
OUTPUT	starts a new body file for each piece of output. For SAS/GRAPH this means a new file for each GRSEG or GIF file that the program generates. <i>Alias:</i> TABLE.

PARAMETERS=(*parameter-name1*'=*parameter-value-1*' ...
'*parameter-name-n*'=*parameter-value-n*')

writes the specified parameters between the <APPLET> and </APPLET> tags that generate dynamic graphics output. Use PARAMETERS= in conjunction with SAS/GRAPH procedures and DEVICE=JAVA in the GOPTIONS statement. Valid parameters are listed in the SAS Help facility discussion for the PARAMETERS= option on the ODS HTML statement.

PATH=*file-location*<(URL='Uniform-Resource-Locator' | NONE)>

specifies the location (external storage location or a SAS catalog) for all HTML files. If GPATH= is not specified, also specifies the location for all GIF files.

file-location

identifies the storage location (for example, directory or PDSE) or SAS catalog to write to.

file-location can be one of the following:

fileref

is a fileref that has been assigned to an external storage location. Use the FILENAME statement to assign a fileref. If you use a fileref in the PATH= option, ODS does not use information from PATH= when it constructs links. (See "How ODS Constructs Links and References" on page 210).

libname.catalog

specifies a SAS catalog to write to.

URL=*Uniform-Resource-Locator* | NONE

provides a URL for *file-location*. ODS uses this URL instead of the file name in all the links and references that it creates to the file. If you specify the keyword NONE, no information from the PATH= option appears in the links or references.

Operating Environment Information: In the CMS operating environment, you must use the URL= suboption in the PATH= option because CMS file names do not form valid URLs. Δ

This option is useful for building HTML files that may be moved from one location to another. If the links from the contents and page files are constructed with a simple URL (one name), they work as long as the contents, page, and body files are all in the same location.

RECORD_SEPARATOR=*string* | NONE

RECSEP=*string* | NONE

RS=*string* | NONE

specifies an alternate record separator for the HTML files. A record separator is the character or string that separates lines in the HTML files.

Different operating environments use different separator characters. If you don't specify a record separator, the HTML files are formatted for the environment that you run the SAS job in. However, if you are generating files in one operating environment for viewing in another operating environment that uses a different separator character, you can specify a record separator that is appropriate for the target environment.

string

is the hexadecimal representation of one or more characters. For example, the following option specifies a record-separator of a carriage-return character and a linefeed character (on an ASCII file system):

```
RECORD_SEPARATOR='0D0A' x
```

NONE

produces HTML that is appropriate for the environment that you run the SAS job in. In many operating environments, using a value of NONE is the same as omitting the RECORD_SEPARATOR option. However, in mainframe operating environments, it is not.

Operating Environment Information: In a mainframe environment, by default, ODS produces a binary file that contains embedded record-separator characters. While this approach means that the file is not restricted by the line-length restrictions on ASCII files, it also means that if you view the file in an editor, the lines all run together.

If you want to format the HTML files so that you can read them with an editor, use RECORD_SEPARATOR=NONE. In this case, ODS writes one line of HTML at a time to the file. When you use a value of NONE, the logical record length of the file that you are writing to must be at least as long as the longest line that ODS produces. If it isn't, the HTML may wrap to another line at an inappropriate place. Δ

STYLE=*style-definition*

specifies a style for the HTML output. Styles are created with the Template procedure and contain color, font, and stylistic attributes for the ODS HTML output. For more information on style definitions, see *The Complete Guide to the SAS Output Delivery System*.

TRANTAB=*translation-table*

translate the HTML files to the requested representation. This option is particularly useful if you are creating files – for example, in an EBCDIC-based operating environment and if the files are destined for an ASCII-based web server. If you use TRANTAB=, you can transfer the files in binary mode as they are already in the appropriate representation. (SAS facilities for changing translation tables are much more flexible than those of FTP, for example).

translation-table can be any translation table that SAS provides or any user-defined translation table.

See also: For information on translation tables, see the documentation of the TRANTAB= system option in the chapter on system options in *SAS Language Reference: Dictionary*.

For information on creating, editing, and displaying customized translation tables, see “The TRANTAB Procedure” in *SAS Procedures Guide*. You can also use PROC TRANTAB to view and modify translation tables that are supplied by SAS Institute.

Using the ODS HTML Statement

While the ODS HTML destination is open, you can submit as many ODS HTML statements as you like, and you can place them anywhere in your SAS/GRAPH program. This enables you to open new files, change anchor names, or specify a new location for graphics output whenever you like. At the end of your ODS HTML processing step, submit ODS HTML CLOSE to close the destination and all open files.

Specifying a Destination for ODS HTML Output

When you use the ODS HTML statement, you must do the following:

- assign a body file with the ODS HTML BODY= option
- specify DEVICE=GIF (or TARGET=GIF) in a GOPTIONS statement to create the GIF files. By default with ODS processing, the GIF files are stored in the current directory. To specify a destination for all the HTML and GIF files, use the PATH= option. To store the GIF files in a different location than the HTML files, use the GPATH= option to specify a location for the GIF files, and PATH= to specify the location of the HTML files. In both cases, the destination must be an aggregate storage location. With procedures GCHART, GCONTOUR, GMAP, GPLOT, and G3D, you can also use DEVICE=ACTIVEEX to create graphs as ActiveX controls, or DEVICE=JAVA to create graphs as Java applets. The controls or applets are defined within the body file.

For more information about the output files generated for use with the Web, see “About the Output Files Generated for the Web” on page 73.

Submitting Multiple ODS HTML Statements

Typically you submit one or more ODS HTML statements to open files and assign option values. ODS HTML statements are additive. That is, any files that you open or any options that you specify remain in effect until you change the option value or close the ODS HTML destination.

When you are using 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, the QUIT statement will cause the selection list to clear, and you will be less likely to encounter unexpected results.

About 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. The following generated HTML output shows how the default anchor name **IDX** is associated with the graph stored in the file named **totals.gif**:

```
<A NAME="IDX"></A>
<HR size=3><P>
<CENTER>
<IMG SRC="totals.gif">
```

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.

By default, the first anchor is named **IDX** and all subsequent anchors generated while the HTML destinations remain open increment that name. Anchor values increment while the ODS HTML destinations remain open unless you use ANCHOR= to assign a new value. Anchor values continue to increment when you open new body files, start new procedures, or produce different types of output.

Controlling the anchor name is useful when you create a graph with drill-down capability. In this case, you must create a variable that contains the names of the anchors that are the targets for the different areas of the graph that the user may click on.

For a complete description, see the ANCHOR= option on page 204.

How ODS Constructs Links and References

Several options in the ODS HTML statement affect how ODS constructs the links and references that point from the frame to the table of contents, table of pages, and body file and from the table of contents or table of pages to the body file. Links are made as HREF= attributes on A (anchor) tags inside the HTML files. Each HREF= attribute points to the NAME= attribute on another A (anchor) tag. HREF= must identify both the file that contains the target and the name of the anchor within that file. The value of HREF= must be a valid target in a valid URL. It uses the following form:

```
<A HREF="URL#anchor-name">
```

ODS constructs the value of an HREF= attribute based on information that you provide in the ODS HTML statement.

Note: HTML references to files use other tags, but the logic for creating the string that identifies the file is the same as the logic for creating an HREF= attribute. Δ

The *URL* in an HREF= attribute is composed of information from three options in the ODS HTML statement: the BASE= option, the GPATH= or the PATH= option, and the BODY=, the CONTENTS=, or the PAGE= option.

- 1 If you specify BASE=, the value of that option is the first part of the URL for every HREF= attribute that ODS writes.
- 2 If you specify GPATH= or PATH=, the next part of the URL in an HREF= attribute comes from that option.

If the file that you are linking to is a graphic, ODS uses information from the GPATH= option as the next part of HREF=. If the file that you are linking to is not a graphic, or if GPATH= is not specified, ODS uses information from the

PATH= option as the next part of HREF=. (For information on these options, see the discussion of GPATH= on page 205 and the discussion of PATH= on page 207.) Table 8.2 on page 211 shows how ODS uses information from a GPATH= or PATH= option in the URL in HREF= attributes.

Table 8.2 Building an HREF= Attribute from the GPATH= or PATH= Option

If the <i>file-location</i> in GPATH= or PATH= is a	And the URL= suboption is*	ODS uses this information in the second part of the URL in the HREF= attribute#
<i>external-location</i> or <i>libref.catalog</i>	not specified	the specified path
<i>external-location</i> or <i>libref.catalog</i>	specified, but not NONE	the value of the URL= suboption
<i>external-location</i> or <i>libref.catalog</i>	NONE	No information from GPATH= or PATH=
<i>fileref</i>	specified or not specified	No information from GPATH= or PATH=

* In the CMS operating environment, you must use the URL= suboption in the GPATH= or PATH= option because CMS file names do not form valid URLs.

For both GPATH= and PATH=, if you do not specify the option, ODS does not use a default or substitute value for that option in creating the HREF= attribute.

Note: If you use a fileref as the file specification in the BODY=, CONTENTS=, or PAGE= option in the ODS HTML statement, and you do not use the URL= suboption in that option, ODS does not use information from GPATH= or PATH= when it creates the complete URL for any corresponding HREF= attributes. Δ

- 3 The last part of the URL that is used in an HREF= attribute is, by default, the name of the file that contains the target. ODS determines the name of the file from the *file-specification* that you use in the BODY=, CONTENTS=, or PAGE= option. (ODS does not create links or references to frame files.) For more information on these options, see the discussion of HTML-file-specification on page 201.)

If you specify the URL= suboption in one of these options, ODS uses the string that you specify instead of the file name.

Note: If you use a fileref as the file specification and do not use the URL= suboption, ODS does not use information from GPATH= or PATH= when it creates the complete URL for the HREF= attribute. Δ

Operating Environment Information: In the CMS operating environment, you must specify the URL= suboption with the BODY=, CONTENTS=, or PAGE= option because CMS file names do not form valid URLs. Δ

The *anchor-name* comes from the value of the ANCHOR= option.

PATTERN Statement

The PATTERN statement defines the characteristics of patterns used in graphs.

Used by:

GCHART, GCONTOUR, GMAP, and GPLOT procedures; SYMBOL statement;
Annotate facility

Global

Assigned by default

Description

PATTERN statements create PATTERN definitions that define the color and type of area fill for patterns used in graphs. These are the procedures and the graphics areas that they create that use PATTERN definitions:

GCHART	2D and 3D bars in bar charts, blocks in block charts, 2D and 3D pie slices in pie charts, and star slices in star charts
GCONTOUR	contour levels in contour plots
GMAP	map areas in choropleth, block, and prism maps; blocks in block maps
GPLOT	areas beneath or between plotted lines.

In addition, the SYMBOL statement and certain Annotate facility functions and macros can use pattern specifications. For details see “SYMBOL Statement” on page 226 and Chapter 10, “The Annotate Data Set,” on page 403.

You can use the PATTERN statement to control the fill and color of a pattern, and whether the pattern is repeated. There are three types of patterns:

- bar and block patterns
- map and plot patterns
- pie and star patterns.

Pattern fills can be solid or empty, or composed of parallel or crosshatched lines. In addition, you can specify device-dependent hardware patterns for rectangle, polygon, and pie fills on devices that support hardware patterns.

If you do not create PATTERN definitions, SAS/GRAPH software generates them as needed and assigns them to your graphs by default. Generally, the default behavior is to rotate a solid pattern through the current colors list. For details, see “About Default Patterns” on page 220.

Syntax

```
PATTERN<1...99>
  <COLOR=pattern-color>
  <REPEAT=number-of-times>
  <VALUE=bar/block-pattern
  | map/plot-pattern
  | pie/star-pattern
  | hardware-pattern>;
```

- bar/block-pattern* can be one of these:
 - EMPTY
 - SOLID
 - style* <*density*>
- map/plot-pattern* can be one of these:
 - MEMPTY

MSOLID

Mdensity <style <angle>>

- pie/star-pattern* can be one of these:

PEMPTY

PSOLID

Pdensity <style <angle>>

- hardware-pattern* has this form:

HWxxxxnnn

Options

COLOR=*pattern-color*

C=*pattern-color*

specifies the color of the fill. *Pattern-color* is any SAS/GRAPH color name. See Chapter 7, “SAS/GRAPH Colors,” on page 139 for more information on specifying colors.

Using COLOR= with a null value cancels the color specified in a previous PATTERN statement of the same number without affecting the values of other options.

COLOR= overrides the CPATTERN= graphics option.

The CFILL= option in the PIE and STAR statements overrides COLOR=. For details, see “Controlling Slice Patterns and Colors” on page 571.

CAUTION:

Omitting COLOR= in a PATTERN statement may cause the PATTERN statement to generate multiple PATTERN definitions. Δ

If no color is specified for a PATTERN statement, that is, if neither COLOR= nor CPATTERN= is used, the PATTERN statement rotates the specified fill through each color in the colors list before the next PATTERN statement is used. For details, see “Understanding Pattern Sequences” on page 224.

See also: “Working with PATTERN Statements” on page 221

Featured in: “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280

REPEAT=*number-of-times*

R=*number-of-times*

specifies the number of times that a PATTERN definition is applied before the next PATTERN definition is used. By default, REPEAT=1.

The behavior of REPEAT= depends on the color specification:

- If you use both COLOR= and REPEAT= 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 VALUE=.
- If you use CPATTERN= in a GOPTIONS statement to specify a single pattern color, and use REPEAT= either alone or with VALUE= in a PATTERN statement, the resulting hatch pattern is repeated the specified number of times.
- If you omit both COLOR= and CPATTERN=, and use REPEAT= either alone (generates default solids) or with VALUE= in a PATTERN statement, the resulting pattern is rotated through each color in the colors list, and then the entire group generated by this cycle is repeated the number of times specified

in REPEAT=. Thus, the total number of patterns produced depends on the number of colors in the current colors 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.

See also: “Understanding Pattern Sequences” on page 224

VALUE=*bar/block-pattern*

V=*bar/block-pattern*

specifies patterns for:

- bar charts produced by the HBAR, HBAR3D, VBAR, and VBAR3D statements in the GCHART procedure including 2D and 3D bar shapes.
- the front surface of blocks in block charts produced by the BLOCK statement in the GCHART procedure.
- the blocks in block maps produced by the BLOCK statement in the GMAP procedure. (The map area from which the block rises takes a map pattern as described on the VALUE= on page 215 option). See also “About Block Maps and Patterns” on page 748.

Values for *bar/block-pattern* are

EMPTY an empty pattern.

E

SOLID a solid pattern.

S

style<density> a shaded pattern.

Style specifies the direction of the lines:

L left-slanting lines.

R right-slanting lines.

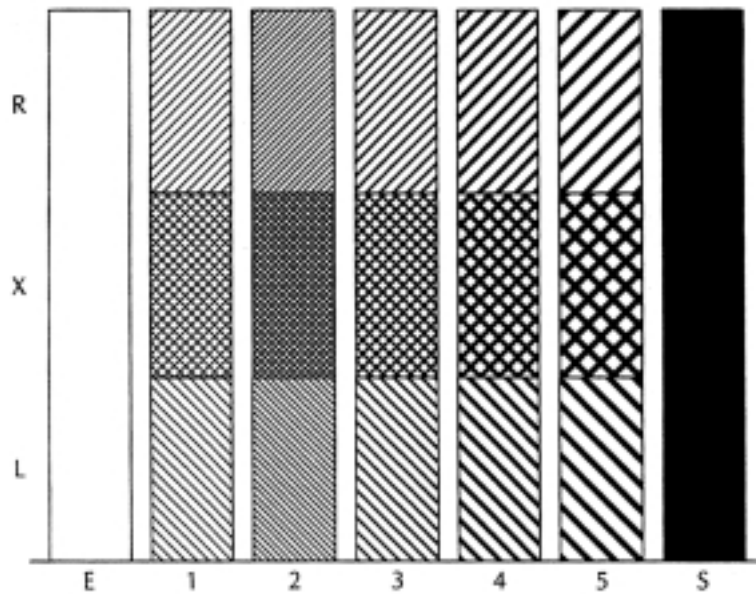
X crosshatched lines.

Density specifies the density of the pattern’s shading:

1...5 1 produces the lightest shading and 5 produces the heaviest shading.

Figure 8.12 on page 215 shows all of the patterns available for bars and blocks.

Figure 8.12 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 colors list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Note: If the V6COMP graphics option is in effect, or if color is limited to a single color with the CPATTERN= or COLORS= graphics options, the order is X1- X5, L1- L5, R1- R5, S, and E. Δ

VALUE=*map/plot-pattern*

V=*map/plot-pattern*

specifies patterns for:

- contour levels in contour plots produced by the GCONTOUR procedure
- map area surfaces in block, choropleth, and prism maps produced by the BLOCK, CHORO, AND PRISM statements in the GMAP procedure.
- areas under curves in plots produced by the AREAS= option in the PLOT statement in the GPLOT procedure.

Values for *map/plot-pattern* are

MEMPTY an empty pattern. EMPTY or E are also valid aliases, except
ME when used with the map areas in block maps created by the
 GMAP procedure.

MSOLID a solid pattern. SOLID or S are also valid aliases, except when
MS used with the map areas in block maps created by the GMAP
 procedure.

$Mdensity<style<angle>>$ a shaded pattern.

Density specifies the density of the pattern's shading:

1...5 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N parallel lines (the default).

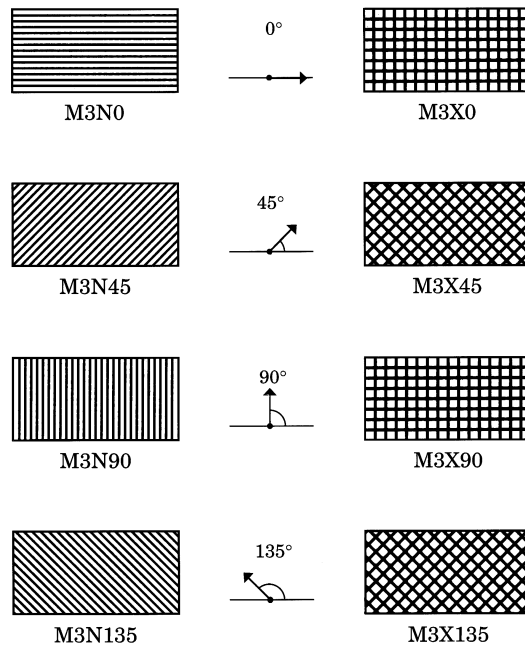
X crosshatched lines.

Angle specifies the angle of the pattern lines:

0...360 the degrees at which the parallel lines are drawn, measured from the horizontal. By default, *angle* is 0 (lines are horizontal).

Figure 8.13 on page 216 shows some typical map and plot patterns.

Figure 8.13 Map and Plot Patterns



If no valid patterns are available, default map and plot fill patterns are selected in this order:

- 1 MSOLID
- 2 M2N0
- 3 M2N90
- 4 M2X45
- 5 M4N0
- 6 M4N90
- 7 M4X90

Each fill is used once with every color in the colors list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Note: If the V6COMP graphics option is in effect, or if color is limited to a single color with the CPATTERN= or COLORS= graphics options, MSOLID is not used and the default fill list starts with M2N0. Δ

VALUE=*pie/star-pattern*

V=*pie/star-pattern*

specifies patterns for pie and star charts produced by the PIE and STAR statements in the GCHART procedure. Values for *pie/star-pattern* are

PEMPTY an empty pattern. EMPTY or E are also valid aliases.

PE

PSOLID a solid pattern. SOLID or S are also valid aliases.

PS

P*density*<*style*<*angle*>> a shaded pattern.

Density specifies the density of the pattern's shading:

1...5 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N parallel lines (the default).

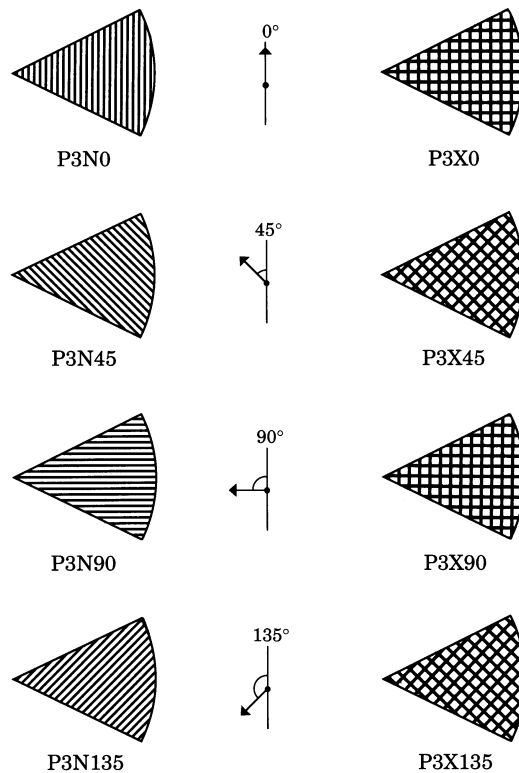
X crosshatched lines.

Angle specifies the angle of the pattern lines:

0...360 the angle of the lines, measured in degrees from perpendicular to the radius of the slice. By default, *angle* is 0.

The FILL= option in the PIE and STAR statements in the GCHART procedure overrides VALUE=.

Figure 8.14 on page 218 shows some typical pie and star patterns.

Figure 8.14 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 colors list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Note: If the V6COMP graphics option is in effect, or if color is limited to a single color with the CPATTERN= or COLORS= graphic options, PSOLID is not used and the default fill list starts with P2N0. Δ

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. Δ

VALUE=HWxxxnnn

specifies a hardware pattern where

HW identifies the pattern as a hardware pattern. The pattern name must begin with the characters HW.

xxx the last two or three characters of the module name in the *Module* field in the Detail window of your device entry. If the module name has eight characters (SASGDPSL, for example), use the last three characters (PSL). If the module name has only seven characters (SASGDVT, for example), use the last two characters (VT).

nnn the number the driver uses to identify the device-dependent pattern. Patterns and associated pattern numbers vary from device to device. See the documentation for your device for valid pattern numbers. For a brief description of some valid values for various devices, see “Specifying Device-Dependent Hardware Patterns” on page 223.

If you specify a hardware pattern for a device that does not support hardware patterns, or if you specify an invalid pattern number, a solid rectangle, polygon, or pie fill is substituted. A solid fill will also be used in place of a hardware pattern in certain types of clipped polygons. See the PCLIP and POLYGONCLIP options in Chapter 9, “Graphics Options and Device Parameters Dictionary,” on page 301 for more information on using hardware patterns with clipped polygons.

See also: “Specifying Device-Dependent Hardware Patterns” on page 223

Using the PATTERN Statement

PATTERN statements can be located anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session.

You can define up to 99 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= and VALUE= options are used. For information on PATTERN definitions, see “Working with PATTERN Statements” on page 221, as well as the description of COLOR= on page 213 and VALUE= on page 215.

PATTERN definitions are generated in the order in which the statements are numbered, regardless of gaps in the numbering or the statement’s position in the program. Although it is common practice, you do not have to start with PATTERN1, and you do not have to use sequential statement numbers.

PATTERN definitions are applied automatically to all areas of the graphics output that require patterns. When assigning PATTERN definitions, SAS/GRAPH starts with the lowest-numbered definition with an appropriate fill specification or with no fill specification. It continues to use the specified patterns until all valid PATTERN definitions have been used. Then, if more patterns are required, SAS/GRAPH returns to the default pattern rotation, but continues to outline the areas in the same color as the fill.

Altering or Canceling PATTERN Statements

PATTERN statements are additive. If you define a PATTERN statement and later submit another PATTERN statement with the same number, the new PATTERN statement redefines or cancels only the options that are included in the new statement. Options not included in the new statement are not changed and remain in effect. For example, assume you define PATTERN4 as

```
pattern4 value=x3 color=red repeat=2;
```

This statement cancels only REPEAT= without affecting the rest of the definition:

```
pattern4 repeat=;
```

Add or change options in the same way. This statement changes the color of the pattern from red to blue:

```
pattern4 color=blue;
```

After all these modifications, PATTERN4 has these characteristics:

```
pattern4 value=x3 color=blue;
```

Cancel individual PATTERN statements by defining a PATTERN statement of the same number without options (a null statement):

```
pattern4;
```

Canceling one PATTERN statement does not affect any other PATTERN definitions. To cancel all current PATTERN statements, use the RESET= option in a GOPTIONS statement:

```
goptions reset=pattern;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current PATTERN definitions as well as other settings.

To display a list of current PATTERN definitions in the LOG window, use the GOPTIONS procedure with the PATTERN option:

```
proc goptions pattern nolist;
run;
```

About Default Patterns

When a procedure produces a graph that needs one or more patterns, SAS/GRAPH either

- automatically generates the appropriate default patterns and outlines to fill the areas, or
- uses patterns, colors, and outlines that are defined by PATTERN statements, graphics options, and procedure options.

In order to understand how SAS/GRAPH generates and assigns patterns defined with PATTERN statements it is helpful to understand how it generates and assigns default patterns. The following sections describe the default pattern behavior for all procedures. See “Working with PATTERN Statements” on page 221 for details about defining patterns.

How Default Patterns and Outlines Are Generated

In general, SAS/GRAPH uses default patterns when no PATTERN statements are defined. Typically, the default pattern that SAS/GRAPH uses is a solid fill that it rotates once through the colors list, skipping the foreground color. By default, SAS/GRAPH also outlines all areas in the foreground color. (Typically, the foreground color is the first color in the device’s colors list.)

Specifically, SAS/GRAPH uses default patterns and outlines when you

- do not specify *any* PATTERN statements, and
- do not use the CPATTERN= graphics option, and
- do not use the COLORS= graphics options (that is, you use the device’s default colors list and it has more than one color), and
- do not use the COUTLINE= option in the action statement.

If all of these conditions are true, then SAS/GRAPH

- selects the first default fill for the appropriate pattern, which is always solid, and rotates it once through the colors list, generating one solid pattern for each color. If the first color in the device's colors list is black (or white), SAS/GRAPH skips that color and begins generating patterns with the next color.

Note: The one exception to the default solid pattern is the map area pattern in a block map produced by the GMAP procedure, which uses a hatch fill by default. By default the map areas and their outlines use the first color in the colors list, regardless of whether the list is the default device list or one specified with `COLORS=` in the `GOPTIONS` statement. △

- uses the foreground color to outline every patterned area.

For example, the default colors list for the PSCOLOR device contains BLACK, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, and GRAY. Therefore, for this device, the first five default patterns are solid red, solid green, solid blue, solid cyan, and solid magenta. These patterns are all outlined in black, the first color in the colors list.

If a procedure needs additional patterns, SAS/GRAPH selects the next default pattern fill appropriate to the graph and rotates it through the colors 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 may change or override the default behavior:

- If you specify a colors 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. The default outline color remains the foreground color.
- Whenever you specify `PATTERN` statements, whether or not the procedure can use them, the default outline color for all patterns changes from foreground to SAME. Therefore, when a procedure runs out of specified patterns and returns to the default pattern rotation, the outline color is SAME, not foreground.

For a description of these graphics options, see Chapter 9, "Graphics Options and Device Parameters Dictionary," on page 301.

Working with PATTERN Statements

With `PATTERN` statements, you can specify

- the type of fill (`VALUE=`)
- the color of the fill (`COLOR=`)
- how many times to apply the statement before using the next one (`REPEAT=`).

You can also use procedure options to specify the pattern outline color and the `CPATTERN=` graphics option to specify a default color for all patterns.

Whether you use `PATTERN` statement options alone or with each other affects the number and kind of patterns your `PATTERN` statements generate. Depending on the options you use, you can explicitly specify every pattern used by your graphs or you can let the `PATTERN` statement generate a series of pattern definitions using either the colors list or the list of default fills.

Explicitly Specifying Patterns

To explicitly specify all the patterns in your graph, you need to do one of the following for every pattern your graph requires:

- Provide a PATTERN statement that uses the COLOR= option to specify the pattern color, for example

```
pattern1 color=red;
```

By default, the fill type SOLID.

- Provide a PATTERN statement that uses both the COLOR= option and the VALUE= option to specify the fill, for example

```
pattern1 color=blue value=r3;
```

Including COLOR= in the PATTERN statement is the simplest way to assure that you get exactly the patterns you want. When you use the COLOR= option, the PATTERN statement generates exactly one PATTERN definition for that statement. If you also use the REPEAT= option, the PATTERN definition is repeated the specified number of times.

Generating Multiple Pattern Definitions

You can also use PATTERN statements to generate multiple PATTERN definitions. To do this use the VALUE= option to specify the type of fill you want but omit the COLOR= option – for example

```
pattern1 value=r3;
```

In this case, the PATTERN statement rotates the R3 fill through all the colors in the colors list. For more information on pattern rotation, see “Understanding Pattern Sequences” on page 224.

Selecting an Appropriate Pattern

The type of fill you specify depends on the type of graph you are producing:

With...	Use...
bar and block charts (PROC GCHART), block maps (PROC GMAP)	VALUE= bar/block-pattern on page 214
contour plots (PROC GCONTOUR), map area surfaces (PROC GMAP)	VALUE=map/plot-pattern on page 215
pie and star charts (PROC GCHART)	VALUE=pie/star-pattern on page 217

Note: If you specify a fill that is inappropriate for the type of graph you are generating (for example, if you specify VALUE=L1 in a PATTERN statement for a choropleth map), SAS/GRAPH ignores the PATTERN statement and continues searching for a valid pattern. If it does not find a definition with a valid fill specification, it uses default patterns instead. \triangle

Controlling Outline Colors

Whenever you use PATTERN statements, the default outline color is the same as the fill color, for example, a blue bar has a blue outline. The effect is the same as specifying

COUTLINE=SAME. Even when the procedure runs out of user-defined patterns and generates default patterns, the outlines continue to match the interior fill color.

To change the outline color of any pattern, whether default or user-defined, use the COUTLINE= option in the action statement that generates the chart.

The Effect of the CPATTERN= Graphics Option

Although the CPATTERN= graphics option is used most often with default patterns, it does affect the PATTERN statement. With default patterns (no PATTERN statements specified) it

- specifies the color for all patterns
- causes default patterns to use hatched fills instead of the default SOLID.

In conjunction with the PATTERN statement it does the following:

- With a PATTERN statement that only specifies a fill (VALUE=), CPATTERN= determines the color of that fill. For example, these statements produce two green, hatched patterns:

```
goptions cpattern=green;
pattern1 value=x3;
pattern2 value=x1;
```

- With a PATTERN statement that only specifies a color (COLOR=), the COLOR= option overrides the CPATTERN= color, but CPATTERN= causes the fill to be hatched, not the default SOLID. For example, these statements produce one red, hatched pattern:

```
goptions cpattern=green;
pattern1 color=red;
```

See also the description of CPATTERN="CPATTERN" on page 316.

Specifying Version 6 Patterns

If you specify the V6COMP graphics option, SAS/GRAPH generates patterns by rotating the appropriate Version 6 default patterns through all the colors in the colors list. With V6COMP, all patterns are outlined in the same color as the fill.

Specifying Device-Dependent Hardware Patterns

You can specify device-dependent hardware patterns with the types of device drivers described in this section.

GDDM Drivers

GDDM drivers include several sets of hardware patterns. These patterns include both predefined and user-defined (device-dependent) fill patterns. When you use a hardware pattern with a GDDM driver, specify the name of the device-dependent pattern set you want the driver to use. This name will be stored in the GPROLOG string in the device entry for the driver. Specify the name of the pattern set in either of these ways:

- Use the GPROLOG= graphics option to assign the pattern set name to the GPROLOG string.
- Enter the pattern set name in the Gprolog window of the device entry for the GDDM device driver.

If you do not specify a pattern set name, the device uses a predefined pattern.

Values for *nnn* for predefined patterns are 1 through 16. Values for *nnn* for device-dependent patterns are 65 through 128.

Information regarding both types of fill patterns can be found in *GDDM Application Programming Guide*. For additional information on specifying hardware patterns with GDDM drivers, see also the *GDDM Base Programming Reference*.

TEK42xx Series Terminal Drivers

TEK42xx series terminal drivers support the predefined fill patterns found in the Technical Reference Guide for each terminal. These drivers can also support user-defined fill patterns. Values for *nnn* for these drivers are numbers less than 175.

HPLJxxxx Drivers

HPLJxxxx drivers for the HP LaserJet support the predefined shading levels and predefined fill patterns for rectangle fill only. These patterns are documented in the appropriate HP LaserJet technical manual. Values for *nnn* for shading levels are 001 through 008. Values for fill patterns are 009 through 014.

Metagraphics Drivers

Metagraphics drivers can use the hardware patterns supported by the device for which they are written. When you specify hardware patterns for a metagraphics driver, values of *nnn* can range from 0 through 999.

Understanding Pattern Sequences

Pattern sequences are sets of PATTERN definitions that SAS/GRAPH automatically generates when a PATTERN statement specifies a fill but not a color. In this case, the specified fill is used once with every color in the colors list. If REPEAT= 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 colors list, either default or user-specified, contains more than one color.

In this case, the PATTERN statement rotates the fill specified by VALUE= through every color in the colors list, generating one PATTERN definition for every color in the list. After every color has been used once, SAS/GRAPH goes to the next PATTERN statement. For example, suppose you specified the following colors list and PATTERN statements for bar/block patterns:

```
goptions colors=(blue red green) ctext=black;
pattern1 color=red value=x3;
pattern2 value=r3;
pattern3 color=blue value=l3;
```

Here, **PATTERN1** generates the first PATTERN definition. **PATTERN2** omits COLOR=, so the specified fill is rotated through all three colors in the colors list before the

PATTERN3 statement is used. This table shows the color and fill of the PATTERN definitions that would be generated if nine patterns were required:

Definition Number	Source	Characteristics:	
		Color	Fill
1	PATTERN1	red	x3
2	PATTERN2	blue	r3
3	PATTERN2	red	r3
4	PATTERN2	green	r3
5	PATTERN3	blue	l3
6	first default	blue	solid
7	first default	red	solid
8	first default	green	solid
9	second default	blue	x1

Notice that after all the PATTERN statements are exhausted, the procedure begins using the default bar and block patterns, beginning with SOLID. Each fill from the default list is rotated through all three colors in the colors list before the next default fill is used.

Repeating Pattern Sequences

If you use REPEAT= but not COLOR=, the sequence generated by cycling the definition through the colors list is repeated the number of times specified by REPEAT=. For example, these statements illustrate the effect of REPEAT= on PATTERN statements both with and without explicit color specifications:

```
goptions colors=(red blue green);
pattern1 color=gold repeat=2;
pattern2 value=x1 repeat=2;
```

Here, **PATTERN1** is used twice and **PATTERN2** cycles through the list of three colors and then repeats this cycle a second time:

Sequence Number	Source	Characteristics: Color	Fill
1	PATTERN1	gold	solid (first default)
2	PATTERN1	gold	solid (first default)
3	PATTERN2	red	x1
4	PATTERN2	blue	x1
5	PATTERN2	green	x1
6	PATTERN2	red	x1

Sequence Number	Source	Characteristics: Color	Fill
7	PATTERN2	blue	x1
8	PATTERN2	green	x1

SYMBOL Statement

The SYMBOL statement defines the characteristics of symbols that display the data plotted by PROC GPLOT and PROC GCONTOUR.

Used by:

GCONTOUR and GPLOT procedures

Global

Assigned by default

Description

SYMBOL statements create SYMBOL definitions, which are used by the GPLOT and GCONTOUR procedures. For the GPLOT procedure, SYMBOL definitions control

- 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 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.

Syntax

```
SYMBOL<1...99>
  <COLOR=symbol-color>
  <MODE=EXCLUDE | INCLUDE>
  <REPEAT=number-of-times>
  <STEP=distance< units>>
  <appearance-option(s)>
  <interpolation-option>;
```

appearance-options can be one or more of these:

BWIDTH=*box-width*

CI=*line-color*

CO=*color*

CV=*value-color*

FONT=*font*

HEIGHT=*symbol-height<units>*

LINE=*line-type*

POINTLABEL<=(*label-description(s)*) | NONE>

VALUE=*special-symbol* | *text-string* | NONE

WIDTH=*thickness-factor*

interpolation-option can be one of these:

- general methods
 - INTERPOL=JOIN
 - INTERPOL=*map/plot-pattern*
 - INTERPOL=NEEDLE
 - INTERPOL=NONE
 - INTERPOL=STEP<*placement*><J><S>
- high-low interpolation methods
 - INTERPOL=BOX<*option(s)*><00...25>
 - INTERPOL=HILO<C><*option(s)*>
 - INTERPOL=STD<1 | 2 | 3><*variance*><*option(s)*>
- regression interpolation methods
 - INTERPOL=R<*type*><0><CLM | CLI<50...99>>
- spline interpolation methods
 - INTERPOL=L<*degree*><P><S>
 - INTERPOL=SM<*nn*><P><S>
 - INTERPOL=SPLINE<P><S>

Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters
IN	inches
PCT	percentage of the graphics output area
PT	points.

If you omit *units*, a unit specification is searched for in this order:

- 1 the GUNIT= option in a GOPTIONS statement
- 2 the default unit, CELLS.

BWIDTH=*box-width*

specifies the width of the box generated by either the INTERPOL=BOX or INTERPOL=HILOB option. *Box-width* can be any number greater than 0. By default, the value of *box-width* is the same as the value of the WIDTH= option, whose default value is 1. Therefore, if you specify a value for WIDTH= and omit BWDITH=, the width of the box changes accordingly.

Featured in: “Example 4. Creating and Modifying Box Plots” on page 273

CI=*line-color*

specifies a color for an interpolation line (GPLOT) or a contour line (GCONTOUR). If you omit CI= but specify CV=, CI= assumes the value of CV=. In this case, CI= and CV= specify the same color, which is the same as specifying COLOR= alone.

If you omit CI=, the color specification is searched for in this order:

- 1 the CV= option
- 2 the COLOR= option
- 3 the CSYMBOL= option in a GOPTIONS statement
- 4 each color in the colors list sequentially before the next SYMBOL definition is used.

See also: “Using Color” on page 247

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266

CO=*color*

specifies a color for

- outlines of filled areas generated by the INTERPOL=*map/plot-pattern* option
- confidence limit lines generated by the INTERPOL=*Rseries* option
- staffs, boxes, and bars generated by the high-low interpolation methods: INTERPOL=HILO, INTERPOL=BOX, and INTERPOL=STD.

If you omit the CO= option, the search order for a color specification depends on the interpolation method being used.

See also: “Using Color” on page 247

Featured in: “Example 5. Filling the Area between Plot Lines” on page 276 and “Example 4. Creating and Modifying Box Plots” on page 273

COLOR=*symbol-color*

C=*symbol-color*

specifies a color for the entire definition, unless it is followed by a more explicit specification. For the GPLOT procedure, this includes plot symbols, the plot line, confidence limit lines, and outlines. For the GCONTOUR procedure, this includes contour lines and labels.

Using the COLOR= option is exactly the same as specifying the same color for both the CI= and CV= options.

If COLOR= precedes CI= or CV= in the same statement, CI= or CV= is used instead.

If you do not use COLOR= or CI=, CV=, and CO=, the color specification is searched for in this order:

- 1 the CSYMBOL= option in a GOPTIONS statement
- 2 each color in the colors list sequentially before the next SYMBOL definition is used.

See also: “Using Color” on page 247

CV=*value-color*

specifies a color for

- plot symbols in the GPLOT procedure
- the filled areas generated by the INTERPOL=*map/plot-pattern* option
- contour labels in the GCONTOUR procedure.

If you omit CV= but specify CI=, CV= assumes the value of CI=. In this case, CV= and CI= specify the same color, which is the same as specifying COLOR= alone.

If you omit CV=, the color specification is searched for in this order:

- 1 the CI= option
- 2 the COLOR= option
- 3 the CSYMBOL= option in a GOPTIONS statement
- 4 each color in the colors list sequentially before the next SYMBOL definition is used.

See also: “Using Color” on page 247

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266, “Example 5. Filling the Area between Plot Lines” on page 276, and “Example 4. Creating and Modifying Box Plots” on page 273

FONT=*font*

F=*font*

specifies the font for the plot symbol (GPLOT) or contour-label text (GCONTOUR) specified by VALUE=. The *font* specification can be

- the name of a software font. For example, FONT=MARKER specifies a software font that is stored in the catalog SASHELP.FONT.
- a hardware font specification of the form HW*xxxxnnn* or *hardware-font-name*:

HWxxxxnnn

HW identifies the font as a hardware font, *xxx* are the last two or three characters of the module name as listed in the Module field in the device entry's Detail window, and *nnn* is the Chartype number of the hardware font as listed in the device entry's Chartype window (for example, FONT=HWDMX001).

hardware-font-name

specifies the name of a hardware font as shown in the device entry's Chartype window (for example, FONT="Palatino-Italic"). The name must be enclosed in double quotation marks.

By default, no font is specified, and the symbol specified by VALUE= is taken from the special symbol table shown in Figure 8.21 on page 243. To use symbols from the special symbol table, omit FONT=.

You can use FONT= to specify a symbol font, such as Marker, that contains a symbol that you want to use in your plot. In this case, the string specified by VALUE= is the character code for the symbol. For example, this definition specifies a heart:

```
symbol font=marker value=N;
```

You can also use FONT= to specify a text font, such as Swiss. In this case, the string specified by VALUE= appears in the plot:

```
symbol font=swiss value=star;
```

Here, the word "star" is displayed in the plot.

To cancel a font specification and return to the default special symbol table, enter a null value:

```
symbol font=, value=star;
```

See also: the VALUE= on page 241 option, "Specifying Plot Symbols" on page 246, and Chapter 6, "SAS/GRAPH Fonts," on page 125

Featured in: Example 2 on page 643

HEIGHT=*symbol-height*<*units*>

H=*symbol-height*<*units*>

specifies the height in number of units of plot symbols (GPLOT) or contour labels (GCONTOUR).

Note: HEIGHT= affects only the height of the symbols and labels on the plot; it does not affect the height of any symbols that may appear in a legend. Δ

See also: the SHAPE= option on page 192 in the LEGEND statement

Featured in: "Example 4. Creating and Modifying Box Plots" on page 273, "Example 3. Rotating Plot Symbols through the Colors List" on page 271, and Example 2 on page 643

INTERPOL=BOX<*option(s)*><00...25>

I=BOX<*option(s)*><00...25>

produces box and whisker plots. The bottom and top edges of the box are located at the sample 25th and 75th percentiles. The center horizontal line is drawn at

the 50th percentile (median). By default, INTERPOL=BOX, in which case the vertical lines, or whiskers, are drawn from the box to the most extreme point within 1.5 interquartile ranges. (An interquartile range is the distance between the 25th and the 75th sample percentiles.) Any value more extreme than this is marked with a plot symbol.

Values for *option(s)* are one or more of these:

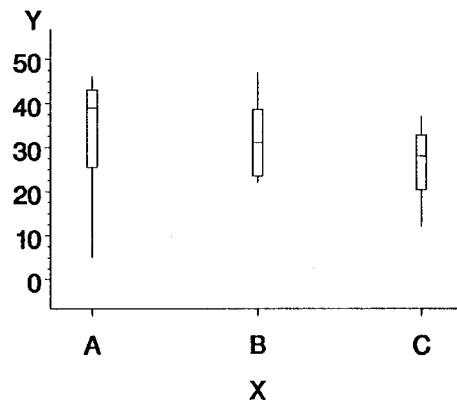
- F fills the box with the color specified by CV= and outlines the box with the color specified by CO=
- J joins the median points of the boxes with a line
- T draws tops and bottoms on the whiskers.

In addition, you can specify a percentile to control the length of the whiskers within the range 00 through 25. These are examples of percentile specifications and their effect:

- 00 high/low extremes. INTERPOL=BOX00 is *not* the same as the default, INTERPOL=BOX.
- 01 1st percentile low, 99th high
- 05 5th percentile low, 95th high
- 10 10th percentile low, 90th high
- 25 25th percentile low, 75th high; since the box extends from the 25th to the 75th percentile, no whiskers are produced.

Figure 8.15 on page 230 shows the type of plot INTERPOL=BOX produces.

Figure 8.15 Box Plot



Note: If you use HAXIS= or VAXIS= in the PLOT statement or ORDER= in an AXIS definition to restrict the range of axis values, by default any observations that fall outside the axis range are excluded from the interpolation calculation. See the MODE= option on page 238 Δ

You cannot use the GPLOT procedure PLOT statement option AREAS= with INTERPOL=BOX.

To increase the thickness of all box plot lines, including the box, whiskers, join line, and top and bottom ticks, use the WIDTH= option.

To increase the width of the box itself, use the BWIDTH= option. By default the value of BWIDTH= is the same as the value of WIDTH=. Therefore, if you specify a value for WIDTH= and omit BWIDTH=, the width of the box changes.

For a scatter effect with the box, use a multiple plot request, as in this example:


```

symbol1 i=none v=star color=green;
symbol2 i=box v=none color=blue;
proc gplot data=test;
  plot (y y)*x / overlay;

```

Featured in: “Example 4. Creating and Modifying Box Plots” on page 273

INTERPOL=HILO<C><*option*>

I=HILO<C><*option*>

specifies that a solid vertical line connect the minimum and maximum Y values for each X value. The data should have at least two values of Y for every value of X; otherwise, the single value is displayed without the vertical line.

By default, for each X value, the mean Y value is marked with a tick as shown in Figure 8.16 on page 232.

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.

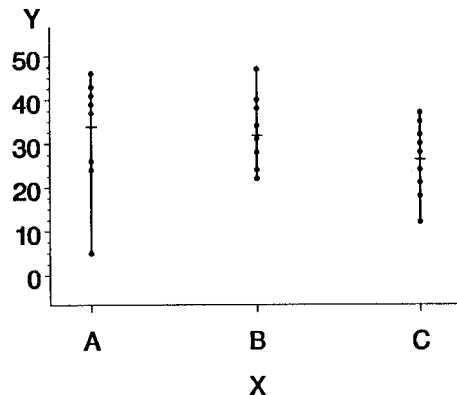
T adds tops and bottoms to each line.

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 8.16 on page 232 shows the type of plot INTERPOL=HILO produces. Plot symbols in the form of dots have been added to this figure.

Figure 8.16 High-Low Plot



To increase the thickness of all lines generated by the INTERPOL=HILO option, use the WIDTH= option.

Note: If you use HAXIS= or VAXIS= in the PLOT statement or ORDER= in an AXIS definition to restrict the range of axis values, by default any observations that fall outside the axis range are excluded from the interpolation calculation. See the MODE= option on page 238. Δ

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266

INTERPOL=JOIN

I=JOIN

connects data points with straight lines. Points are connected in the order they occur in the input data set. Therefore, the data should be sorted by the independent (horizontal axis) variable.

If the data contain missing values, the observations are omitted. However, the plot line is not broken at missing values unless the SKIPMISS option is used.

See also: the SKIPMISS on page 826 option and “Missing Values” on page 807

INTERPOL=L<degree><P><S>

I=L<degree><P><S>

specifies a Lagrange interpolation to smooth the plot line. Specify one of these values for *degree*:

1 | 3 | 5 specifies the degree of the Lagrange interpolation polynomial. By default, *degree* is 1.

In addition, you can specify one or both of these:

P specifies a parametric interpolation

S sorts a data set by the independent variable before plotting its data.

The Lagrange methods are useful chiefly when data consist of tabulated, precise values. A polynomial of the specified degree (1, 3, or 5) is fitted through the nearest 2, 4, or 6 points. In general, the first derivative is not continuous. If the values of the horizontal variable are not strictly increasing, the corresponding parametric method (L1P, L3P, or L5P) is used.

Specifying INTERPOL=L1P, INTERPOL=L3P, or INTERPOL=L5P results in a parametric Lagrange interpolation of degree 1, 3, or 5, respectively. Both the horizontal and vertical variables are processed with the Lagrange method and a parametric interpolation of degree 1, 3, or 5, using the distance between points as a parameter.

INTERPOL=*map/plot-pattern*

I=*map/plot-pattern*

specifies that a pattern fill the polygon that has been defined by the data points.

Values for *map/plot-pattern* are

MEMPTY

ME

an empty pattern. EMPTY and E are valid aliases

MSOLID

MS

a solid pattern. SOLID and S are valid aliases

Mdensity<*style*<*angle*>>

a shaded pattern.

Density specifies the density of the pattern's shading:

1...5 1 produces the lightest shading and 5 produces the heaviest.

Style specifies the direction of pattern lines:

N parallel lines (the default)

X crosshatched lines.

Angle specifies the starting angle for parallel or crosshatched lines:

0...360 the degree at which the parallel lines are drawn. By default, *angle* is 0 (lines are parallel to the horizontal axis).

The INTERPOL=*map/plot-pattern* option only works if the data are structured so that the data points and, consequently, the plot lines form an enclosed area. The plot lines should not cross each other.

See also: "PATTERN Statement" on page 211

Featured in: "Example 5. Filling the Area between Plot Lines" on page 276

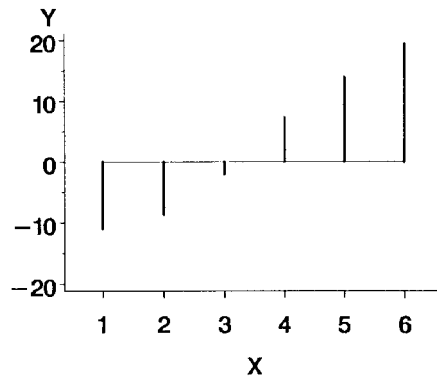
INTERPOL=NEEDLE

I=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 if it is greater than 0. The horizontal line is drawn automatically.

Figure 8.17 on page 234 shows the type of plot INTERPOL=NEEDLE produces. Plot symbols are not displayed in this figure.

Figure 8.17 Needle Plot



You cannot use the PLOT statement option AREAS= with INTERPOL=NEEDLE.

INTERPOL=NONE
I=NONE

produces plots with unconnected data points (scatter plots). If no interpolation method is specified in a SYMBOL statement and if the graphics option INTERPOL= is not used, INTERPOL=NONE is the default.

You cannot use the PLOT statement option AREAS= with INTERPOL=NONE.

INTERPOL=R<type><0><CLM | CLI<50...99>>
I=R<type><0><CLM | CLI<50...99>>

specifies that a plot is a regression analysis. By default, regression lines are not forced through plot origins and confidence limits are not displayed.

Type specifies the type of regression. Specify one of these values for *type*:

L requests linear regression representing the regression equation

$$Y = \beta_0 + \beta_1 X$$

Q requests quadratic regression representing the regression equation

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2$$

C requests cubic regression representing the regression equation

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

By default, *type* is L. The regression line is drawn in the line type specified in the LINE= option. By default, the type of the regression line is 1.

Note: Specify *type* if you use either 0, or CLI, or CLM. Δ

To force the regression line through a (0,0) origin, specify

0 eliminates the β_0 parameter, or intercept, from the regression equation. If the origin is at (0,0), also forces the regression line through the origin. For example, if you specify 0 for a linear regression, the plot line represents the equation

$$Y = \beta_1 X$$

Note: To force the regression line through the origin (0,0) when the data ranges do not place the origin at (0,0), use the GPLOT procedure options HZERO and VZERO (ignored if the

data contain negative values), or use HAXIS and VAXIS to specify axes ranges from 0 to maximum data value. If the data ranges contain negative values and HAXIS and VAXIS specify ranges starting at 0, only values within the displayed range are used in the interpolation calculations. Δ

To display confidence limits, specify one of these:

CLM displays confidence limits for mean predicted values

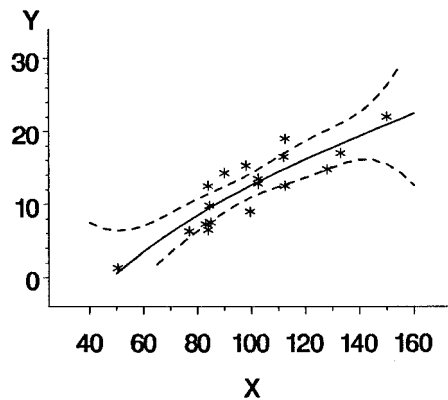
CLI displays confidence limits for individual predicted values.

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 8.18 on page 235 shows the type of plot INTERPOL=RCCLM95 produces (cubic regression analysis with 95% confidence limits).

Figure 8.18 Plot of Regression Analysis and Confidence Limits



Featured in: Example 4 on page 839

INTERPOL=SM<nn><P><S>

I=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

0...99 produces a cubic spline that minimizes a linear combination of the sum of squares of the residuals of fit and the integral of the square of the second derivative (Reinsch 1967)*. The greater the *nn* value, the smoother the fitted curve. By default, the value of *nn* is 0.

In addition, specify one or both of these:

P specifies a parametric cubic spline

S sorts data by the independent variable before plotting.

* Reinsch, C.H. (1967), "Smoothing by Spline Functions," *Numerische Mathematik*, 10, 177-183.

INTERPOL=SPLINE<P><S>

I=SPLINE<P><S>

specifies that the interpolation for the plot line use a spline routine.

INTERPOL=SPLINE produces the smoothest line and is the most efficient of the nontrivial spline interpolation methods.

Spline interpolation smoothes a plot line using a cubic spline method with continuous second derivatives (Pizer 1975)*. This method uses a piecewise third-degree polynomial for each set of two adjacent points. The polynomial passes through the plotted points and matches the first and second derivatives of neighboring segments at the points.

Specify one or both of these:

P specifies a parametric spline interpolation method. This interpolation uses a parametric spline method with continuous second derivatives. Using the method described earlier for the spline interpolation, a parametric spline is fitted to both the horizontal and vertical values. The parameter used is the distance between points

$$t = \sqrt{(x^2 + y^2)}$$

If two points are so close together that the computations overflow, the second point is not used.

S sorts a data set by the independent variable before plotting its data.

Note: When points on the graph are out of range of the axis values, the curve is clipped. If an end point is out of range, no curve is drawn. Out-of-range conditions may 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. Δ

INTERPOL=STD<1 | 2 | 3><variance><option(s)>

I=STD<1 | 2 | 3><variance><option(s)>

specifies that a solid line connect the mean Y value with $\pm 1, 2,$ or 3 standard deviations for each X.

Note: By default, 2 standard deviations are used. Δ

The sample variance is computed about each mean, and from it, the standard deviation s_y is computed. *Variance* can be one or both of these:

M computes $s_{\bar{y}}$,

P computes sample variances using a pooled estimate, as in a one-way ANOVA model.

In addition, specify one of these values for *option(s)*:

B connects the minimum and maximum Y values with bars instead of lines.

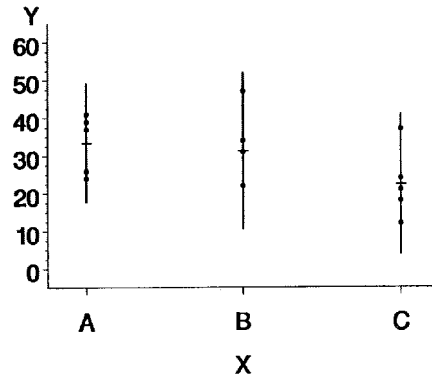
J connects the means from bar to bar with a line.

* Pizer, Stephen M. (1975), *Numerical Computing and Mathematical Analysis*, Chicago: Science Research Associates, Inc., Chapter 4.

- 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 8.19 on page 237 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 8.19 Plot of Standard Deviations



Note: By default, the vertical axis ranges from the minimum to the maximum Y value in the data. If the requested number of standard deviations from the mean covers a range of values that exceeds the maximum or is less than the minimum, the STD lines are cut off at the minimum and maximum Y values. When this cutoff occurs, rescale the axis using VAXIS= in the PLOT statement or ORDER= in an AXIS definition so that the STD lines are shown. Δ

If you restrict the range of axis values by using HAXIS= or VAXIS= in a PLOT statement or ORDER= in an AXIS definition, by default any observations that fall outside the axis range are excluded from the interpolation calculation. See the MODE= on page 238 option.

To increase the thickness of all lines generated by the INTERPOL=STD option, use the WIDTH= option.

You cannot use the PLOT statement option AREAS= with INTERPOL=STD.

INTERPOL=STEP<placement><J><S>

I=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.

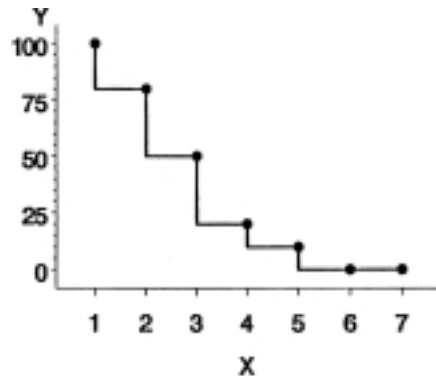
In addition, specify one or both of these:

- J produces steps joined with a vertical line.

S sorts unordered data by the independent variable before plotting.

Figure 8.20 on page 238 shows the type of plot INTERPOL=STEPJR produces. Plot symbols in the form of dots have been added to this figure.

Figure 8.20 Step Plot



LINE=*line-type*

L=*line-type*

specifies the line type of the plot line in the GPLOT procedure, or the contour line in the GCONTOUR procedure:

- 1 a solid line.
- 2...46 a dashed line.

Line types are shown in Figure 8.22 on page 249. By default, LINE=1.

MODE=EXCLUDE | INCLUDE

specifies that interpolation calculations exclude or include data values that are outside the range of plot axes. By default, MODE=EXCLUDE, which excludes values outside the axis range from any calculations.

If you control the range of values displayed on an axis by using HAXIS= and VAXIS= in the GPLOT procedure, or ORDER= in an AXIS definition, any data points that lie outside of the range of the axes are discarded before the calculations are done for interpolation lines. This 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.

See also: “Values Out of Range” on page 807

POINTLABEL<=(*label-description(s)*) | NONE>

labels plot points. *Label-description(s)* defines the appearance or the values of the labels, or both. POINTLABEL without any specified descriptions labels points with the Y value. NONE suppresses the point labels. *Label-description(s)* can be used to change the variable whose values are used to label points, and/or to change features of the label text, such as the color, font, or size of the text.

Note: If you do not specify a color on a SYMBOL statement, the symbol definition is rotated through the colors list before the next SYMBOL statement is used. Thus, if your plot contains multiple plot lines and you want to limit your POINTLABEL specification to a single line, you must specify a color on the SYMBOL statement that contains the POINTLABEL description. Δ

Label-description(s) can be one or more of these:

COLOR=*text-color*

C=*text-color*

specifies the color of the label text. The default is the first color from the colors list.

FONT=*font* | NONE

F=*font* | NONE

specifies the font for the text. See Chapter 6, "SAS/GRAPH Fonts," on page 125 for details on specifying *font*. If you omit **FONT=**, a font specification is searched for in this order:

- 1 the **FTEXT=** option in a **GOPTIONS** statement
- 2 the default hardware font, NONE.

HEIGHT=*text-height* <*units*>

H=*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.

JUSTIFY=CENTER | LEFT | RIGHT

J=C | L | R

specifies the horizontal alignment of the label text. The default is CENTER. The location of the point label is relative to the location of the corresponding data point.

POSITION=TOP | MIDDLE | BOTTOM

specifies the vertical placement of the label text. The default is TOP. The location of the point label is relative to the location of the corresponding data point.

"#var" | **"#x:#y <\$char>"** | **"#y:#x <\$char>"** | **"#x:#y"** *\$char* | **"#y:#x"** *\$char*
 specifies the variable(s) whose values will label the plot points. The variable specification must be enclosed in either single or double quotation marks. The first specified variable must be prefixed with a pound sign (#). If a second variable is specified, it must be prefixed with a colon and a pound sign (:#). Optionally, when you specify both the X and Y variables, you can specify the character to display as the delimiter between variable values in the plot label.

By default if **POINTLABEL** 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 will label plot points:

```
POINTLABEL=( "#sales" )
```

Alternatively, you can label the plot points with the values of the X and Y variables, in either order. The order that you specify X and Y in the variable specification determines the order that the values are displayed in the label. The following option specifies variables HEIGHT and WEIGHT; in the label, the value for HEIGHT will be displayed, followed by the value for WEIGHT:

```
POINTLABEL=( "#height:#weight" )
```

The variables that you specify must be the plot's X and Y variables. Specifying any other variables will cause unexpected labeling.

By default when you specify both the X and Y variables, a colon (:) displays in the label to separate the values in each label. To change the character that displays as the delimiter, use the \$ syntax to specify an alternative character. The following option specifies a vertical bar (|) as the delimiter in the label:

```
POINTLABEL=("#height:#weight $|")
```

Within the quotation marks, the \$ specification can precede or follow the variable specification, but it must be separated from the variable specification by at least one space. Optionally, the \$ specification can be in its own set of quotation marks.

Note: Specifying a delimiting character with the \$ only changes the character that displays in the label. It does not change the syntax of the variable specification, which requires a colon and pound sign (:#) to precede the second variable. Δ

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.

REPEAT=*number-of-times*

R=*number-of-times*

specifies the number of times that a SYMBOL definition is applied before the next SYMBOL definition is used. By default, REPEAT=1.

The behavior of REPEAT= depends on whether any of the SYMBOL color options (CI=, CV=, CO=, and COLOR=) or the CSYMBOL= graphics option also is used:

- If any SYMBOL color option also is used in the SYMBOL definition, that SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL color option is used but the CSYMBOL= graphics option is currently in effect, the SYMBOL definition is repeated the specified number of times in the specified color.
- If no SYMBOL statement color options are used and the CSYMBOL= graphics option is not used, the SYMBOL definition is cycled through each color in the colors list, and then the entire group generated by this cycle repeats the number of times specified by REPEAT=. Thus, the total number of iterations of the SYMBOL definition depends on the number of colors in the current colors list.

See also: "Using the SYMBOL Statement" on page 243

STEP=*distance<units>*

specifies the minimum distance between labels on contour lines. The value of *distance* must be greater than zero. By default, STEP=65PCT.

Note: If you specify units of PCT or CELLS, STEP= calculates the distance between the labels based on the width of the graphics output area, not the height. For example, if you specify STEP=50PCT and if the graphics output area is 9 inches wide, the distance specified is 4.5 inches. A value less than 10 percent is ignored and 10 percent is used instead. Δ

When you use STEP=, specify the minimum distance that you want between labels. The option then calculates how many labels it can fit on the contour line, taking into account the length of the labels and the minimum distance you specified. Once it has calculated how many labels it can fit while retaining the minimum distance between them, it places the labels, evenly spaced, along the

line. Consequently, the space between labels may be greater than what you specify, although it will never be less.

In general, to increase the number of labels from the default, reduce the value of *distance*.

If the procedure cannot write the label at a particular location on the contour, for example because the contour line makes a sharp turn, the label may 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 may also cause labels to be omitted, since this forces the procedure to select smoother labeling locations, which may not be available on some contours.

Featured in: Example 2 on page 643

VALUE=*special-symbol* | *text-string* | NONE

V=*special-symbol* | *text-string* | NONE specifies:

- a plot symbol for the data points (GPLOT). By default, VALUE=PLUS, which produces the + symbol for the plot symbol.
- contour-label text in a contour plot (GCONTOUR). By default with the AUTOLABEL option, GCONTOUR labels contour lines with the contour variable's value at that contour level.

VALUE=NONE suppresses plot symbols at the data points, or labels on the contour lines.

Values for *special-symbol* are the names and characters shown in Figure 8.21 on page 243. The special symbol table can be used only if the FONT= option is not used or a null value is specified:

```
font=,
```

Note: To specify a single quotation mark, you must enclose it in double quotation marks: Δ

```
value="'"
```

To specify a double quotation mark, you must enclose it in single quotation marks:

```
value='"'
```

In some operating environments, punctuation characters may require single quotes.

If you use VALUE=*text-string* to specify a plot symbol, you must also use the FONT= option to specify a symbol font or a text font. If you specify a symbol font, the characters in the string are character codes for the symbols in the font. If you specify a text font, the characters in the string are displayed. If you specify a text string containing quotes or blanks, enclose the string in single quotes.

For example, if you specify this statement, the plot symbol is the word "plus" instead of the symbol +:

```
symbol font=swiss value=plus;
```

See also: the FONT= option on page 229 and "Specifying Plot Symbols" on page 246

Featured in: "Example 3. Rotating Plot Symbols through the Colors List" on page 271, "Example 4. Creating and Modifying Box Plots" on page 273, and Example 2 on page 643

WIDTH=*thickness-factor*

W=*thickness-factor*



































specifies the thickness of interpolated lines (GPLOT) or contour lines (GCONTOUR), where *thickness-factor* is a number. The thickness of the line increases directly with *thickness-factor*. By default, WIDTH=1.

WIDTH= also affects all the lines in box plots (INTERPOL=BOX), high-low plots with bars (INTERPOL=HILOB), and standard deviation plots (INTERPOL=STD). It also affects the outlines of the area generated by the AREAS= option in the PLOT statement of the GPLOT procedure.

Note: By default, the value specified by WIDTH= is used as the default value for the BWIDTH= option. For example, specifying WIDTH=6 also sets BWIDTH= to 6 unless you explicitly assign a value to BWIDTH=. Δ

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266 and “Example 4. Creating and Modifying Box Plots” on page 273

Figure 8.21 Special Symbols for Plotting Data Points

VALUE=	Plot Symbol	VALUE=	Plot Symbol
PLUS		% (percent)	
X		& (ampersand)	
STAR		' (single quote)	
SQUARE		= (equals)	
DIAMOND		- (hyphen)	
TRIANGLE		@ (at)	
HASH		* (asterisk)	
Y		+ (plus)	
Z		> (greater than)	
PAW		. (period)	
POINT		< (less than)	
DOT		, (comma)	
CIRCLE		/ (slash)	
_ (underscore)		? (question mark)	
" (double quote)		((left parenthesis)	
# (pound sign)	) (right parenthesis)	
\$ (dollar sign)		: (colon)	

Note: The words or special characters in the VALUE= column are entered exactly as shown.

Using the SYMBOL Statement

A SYMBOL statement specifies one or more options that indicate the color and other attributes used by the GPLOT procedure or the GCONTOUR procedure. For GPLOT, 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 only be applied to contour plots when the AUTOLABEL option is specified on GCONTOUR. Δ

You can define up to 99 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

- 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 245 and “Using Generated Symbol Sequences” on page 249.

Although it is common practice, you do not have to start with SYMBOL1, and you do not have to use sequential statement numbers. When assigning SYMBOL definitions, SAS/GRAPH software starts with the lowest-numbered definition and works upward, ignoring gaps in the numbering.

Altering or Canceling SYMBOL Statements

SYMBOL statements are additive. If you define a SYMBOL statement and later submit another SYMBOL statement with the same number, the new SYMBOL statement defines or cancels only the options that are included in the new statement. Options that are not included in the new statement are not changed and remain in effect.

Assume you define SYMBOL4 as:

```
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 RESET= in a GOPTIONS statement:

```
goptions reset=symbol;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current SYMBOL definitions as well as other settings.

To display current SYMBOL definitions in the LOG window, use the GOPTIONS procedure with the SYMBOL option:

```
proc goptions symbol nolist;
run;
```

Controlling Consecutive SYMBOL Statements

If you specify consecutively numbered SYMBOL statements and you want SAS/GRAPH to use each definition only once, use color specifications to ensure each SYMBOL statement generates only one symbol definition. You can

- specify colors on each SYMBOL statement, using the COLOR=, CI=, CV=, or CO= options. This method lets you explicitly assign colors for each definition. For example, these statements generate two definitions:

```
symbol1 value=star color=green;
symbol2 value=square color=yellow;
```

- specify a default color for all SYMBOL statements using the CSYMBOL= option on 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 colors list to a single color using the COLORS= option on the GOPTIONS statement. This method makes it easy to specify the same color for each definition when you want the color to apply to other definitions also, such as PATTERN definitions.

For more information on specifying colors for symbol definitions, see “Using Color” on page 247.

If you do not use color to limit a SYMBOL statement to a single symbol definition, SAS/GRAPH generates multiple symbol definitions from that statement by rotating the current definition through the colors list (for more details, see “Using Generated Symbol Sequences” on page 249). Because SAS/GRAPH uses symbol definitions in the order they are generated, this means that the n th symbol definition applied to a graph does not necessarily correspond to the SYMBOL n statement.

For example, assuming no color is specified on the CSYMBOL= graphics option, these statements generate four definitions:

```
goptions colors=(red blue green);
symbol1 value=star;
symbol2 value=square color=yellow;
```

Because no color is specified on SYMBOL1, SAS/GRAPH rotates the symbol definition through the colors list, which has three colors. Thus, SYMBOL1 defines the first three applied symbol definitions, and SYMBOL2 defines the 4th:

Sequence Number	Source	Characteristics: Color	Symbol
1	SYMBOL1	red	star
2	SYMBOL1	blue	star
3	SYMBOL1	green	star
4	SYMBOL2	yellow	square

In this case, if a graph needs only three symbols, the SYMBOL2 definition is not used.

To make the n th applied symbol definition correspond to the SYMBOL n statement, limit each SYMBOL statement to a single color, using one of the techniques listed at the beginning of this section.

Setting Definitions for PROC GPLOT

The following topics apply only for SYMBOL statements used with PROC GPLOT:

- 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 uses to mark the data points on a plot. Plot symbols can be

- special symbols from Figure 8.21 on page 243
- characters from symbol fonts
- text strings.

By default, the plot symbol is the + symbol. To specify a special symbol, use VALUE= to specify a name or a character from Figure 8.21 on page 243:

```
symbol1 value=hash color=green;
symbol2 value=) color=blue;
```

This example uses color to ensure each SYMBOL statement generates only one definition. You can omit color specifications to let SAS/GRAPH rotate symbol definitions through the colors list. For details, see “Using Generated Symbol Sequences” on page 249.

To use plot symbols other than those in Figure 8.21 on page 243, 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 VALUE= is displayed as the plot symbol. (See VALUE= on page 241 and FONT= on page 229.)

This table illustrates some of the ways you can define a plot symbol:

Definition	Plot Symbol
symbol1 value=plus;	+
symbol2 value=+;	⊕
symbol3 font=swiss value=plus;	plus
symbol4 font=marker value=u;	■
symbol5 value="**";	⊕

Specifying a Default Interpolation Method

The INTERPOL= option in a GOPTIONS statement specifies a default interpolation method to be used with all SYMBOL definitions. This default interpolation method is in effect unless you specify a different interpolation in a SYMBOL statement. If the GOPTIONS statement does not specify an interpolation method, the default for each SYMBOL statement is NONE.

Sorting Data with Spline Interpolation

If you want the Gplot procedure to sort by the horizontal axis variable before plotting, add the letter S to the end of any of the spline interpolation methods (INTERPOL=L, INTERPOL=SM, and INTERPOL=SPLINE). For example, suppose you want to overlay three plots (Y1*X1, Y2*X2, and Y3*X3) and for each plot, you want the X variable sorted in ascending order. Use these statements:

```
symbol1 i=splines c=red;
symbol2 i=splines c=blue;
```



```

symbol3 i=splines c=green;

proc gplot;
  plot y1*x1 y2*x2 y3*x3 / overlay;
run;

```

Using Color

Generally, there are two ways to explicitly specify color for SYMBOL statements:

- specify colors on the SYMBOL statements
- specify a color on the CSYMBOL= graphics option.

You can also let SAS/GRAPH rotate symbol definitions through the colors list. For details, see “Using Generated Symbol Sequences” on page 249.

Specifying Colors with SYMBOL Statements

The SYMBOL statement has these options for specifying color:

- The CV= option specifies color for plot symbols in GPLOT, or for contour labels in GCONTOUR.
- The CO= option specifies color for confidence limit lines and area outlines in GPLOT.
- The CI= option specifies color for plot lines in GPLOT, or contour lines in GCONTOUR.
- The COLOR= option specifies color for the entire symbol. For GPLOT, this includes plot symbols, plot lines, and outlines. For GCONTOUR, this includes contour lines and labels.

CV= and CI= have the same effect as using COLOR= 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 CV= and CI= specify the same color.

In general, CI=, CV=, and CO= color specific areas of the symbol. Use these options to produce symbols and plot lines of different colors without having to overlay multiple plot pairs. For example, if you request regression analysis with confidence limits, use this statement to assign red to the plot symbol, blue to the regression lines, and green to the confidence limit lines:

```

symbol cv=red ci=blue co=green;

```

The COLOR= option colors the entire symbol or those portions of it not colored by one of the other color options. If COLOR= precedes CI= or CV=, 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 colors 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 colors list. Δ

If the SYMBOL color options and the CSYMBOL= graphics option are not used, the SYMBOL definition cycles through each color in the colors list before the next definition is used. For details, see “Using Generated Symbol Sequences” on page 249.

Specifying Color with CSYMBOL=

The CSYMBOL= option on the GOPTIONS statement specifies the default color to be used by all SYMBOL definitions:

```
goptions csymbol=green;  
symbol1 value=star;  
symbol2 value=square;
```

In this example, both SYMBOL statements use green.

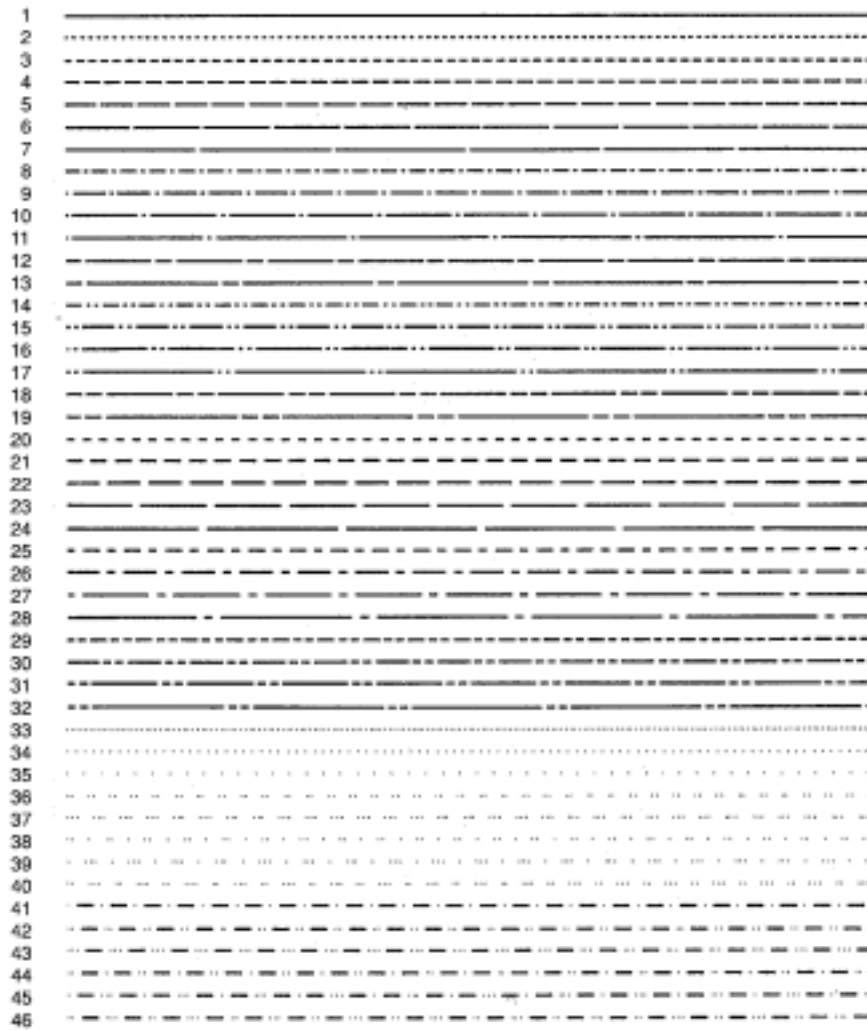
CSYMBOL= is overridden by any of the SYMBOL statement color options. See “Using Color” on page 247 for details.

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 8.22 on page 249 shows the line types represented by these numbers. By default, the line type is 1 for plot and contour lines, and 2 for confidence limit lines.

Figure 8.22 Line Types



Note: These line types also are used by other statements and procedures. Some options accept a line type of 0, which produces no line. Δ

Using Generated Symbol Sequences

Symbol sequences are sets of SYMBOL definitions that are automatically generated by SAS/GRAPH software if any of these conditions is true:

- no valid SYMBOL definition is available. In this case, default symbol sequences are generated by rotating symbol definitions through the color specified on the GOPTIONS statement's CSYMBOL= option. If a CSYMBOL= color is not in effect, the definitions are rotated through the colors list.
- a SYMBOL statement specifies color but not a plot symbol for the GPLOT procedure, or a line type for the GCONTOUR procedure (assuming GCONTOUR does not specify the needed line types). In this case, a default plot symbol or line type is used with the specified color and only one definition is generated.

- a SYMBOL statement specifies a plot symbol for GPLOT or a line type for GCONTOUR, but no color options. In this case, the specified plot symbol or line type is used once with the color specified by the CSYMBOL= graphics option. If a CSYMBOL= color is not in effect, the specified plot symbol or line type is rotated through the colors list.

If REPEAT= 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 colors list.

- Definitions used for GPLOT rotate plot symbols through the colors 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 colors list. If more than one definition is required, it uses the same plot symbol or line type with the next color in the colors list and continues until all the colors have been used once. If more definitions are needed, SAS/GRAPH selects the second default plot symbol or line type and rotates it through the colors list. It continues in this fashion, selecting default plot symbols or line types and cycling them through the colors list until all the required definitions are generated.

If a color has been specified with the CSYMBOL= option on the GOPTIONS statement, each default plot symbol or line type is used once with the specified color, and the colors in the colors list are ignored.

Symbol Sequences Generated from SYMBOL Statements

If a SYMBOL statement does not specify color, and if the CSYMBOL= graphics option is not used, the symbol definition is rotated through every color in the colors list before the next SYMBOL definition is used:

```
goptions colors=(blue red green);
symbol1 cv=red i=join;
symbol2 i=spline v=dot;
symbol3 cv=green v=star;
```

Here, the SYMBOL1 statement generates the first SYMBOL definition. The SYMBOL2 statement does not include color, so the first default plot symbol is rotated through all colors in the colors list before the SYMBOL3 statement is used. This table shows the colors and symbols that would be used if nine symbol definitions were required for PROC GPLOT:

Sequence Number	Source	Characteristics:		
		Color	Symbol	Interpolation
1	SYMBOL1	cv=red	first default	join
2	SYMBOL2	color=blue	dot	spline
3	SYMBOL2	color=red	dot	spline
4	SYMBOL2	color=green	dot	spline
5	SYMBOL3	cv=green	star	NONE

Sequence Number	Source	Characteristics:		
		Color	Symbol	Interpolation
6	first default	color=blue	first default	default
7	first default	color=red	first default	default
8	first default	color=green	first default	default
9	second default	color=blue	second default	default

Notice that after the SYMBOL statements are exhausted, the procedure begins using the default definitions (sequences 6 through 9). Each plot symbol from the default list is rotated through all colors in the colors 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 colors list.

If you use REPEAT= but no color, the sequence generated by cycling the definition through the colors list is repeated the number of times specified by REPEAT=. For example, these statements define a colors list and illustrate the effect of REPEAT= on SYMBOL statements both with and without explicit color specifications:

```
goptions colors=(blue red green);
symbol1 color=gold repeat=2;
symbol2 value=star color=cyan;
symbol3 value=square repeat=2;
```

Here, SYMBOL1 is used twice, SYMBOL2 is used once, and SYMBOL3 rotates through the list of three colors and then repeats this cycle a second time:

Sequence Number	Source	Characteristics:		
		Color	Symbol	Interpolation
1	SYMBOL1	gold	first default	default
2	SYMBOL1	gold	first default	default
3	SYMBOL2	cyan	star	default
4	SYMBOL3	blue	square	default
5	SYMBOL3	red	square	default
6	SYMBOL3	green	square	default
7	SYMBOL3	blue	square	default
8	SYMBOL3	red	square	default
9	SYMBOL3	green	square	default

TITLE, FOOTNOTE, and NOTE Statements

The TITLE, FOOTNOTE, and NOTE statements control the content, appearance, and placement of text.

Used by:

GCHART, GCONTOUR, GFONT, GMAP, GPLOT, GPRINT, GSLIDE, and G3D procedures

Global: TITLE and FOOTNOTE

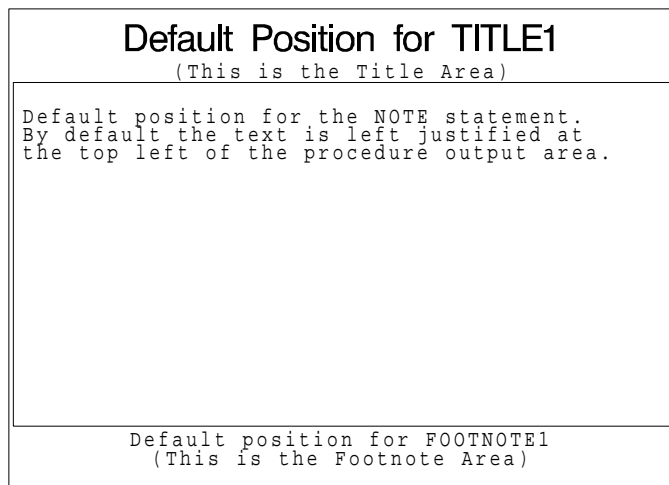
Description

TITLE, FOOTNOTE, and NOTE statements add text to maps, plots, charts, and text slides. With these statements you can

- control the content, appearance, and placement of the text, including color, size, font, and alignment
- underline or draw boxes around the text
- draw straight lines on the output.

Figure 8.23 on page 252 shows the default appearance and placement of titles, footnotes, and notes on the graphics output area.

Figure 8.23 Default Placement of Titles, Footnotes, and Notes



Titles are centered at the top of the graphics output in the *title area*. They are positioned in numeric order with the lowest-numbered TITLE at the top of the title area and the highest-numbered TITLE at the bottom of the title area.

TITLE statements have these default characteristics:

- TITLE1 is twice the height of all other titles and uses the SWISS font.
- All other TITLE statements are one unit high and use the default hardware font.

Footnotes are positioned similarly in the *footnote area* at the bottom of the graphics output area, with the lowest numbered FOOTNOTE at the top of the footnote area. Unless otherwise specified, they use the default hardware font and are one unit high.

Space for the title area and the footnote area is taken from the procedure output area. The more titles and footnotes you specify and the bigger they are, the smaller the procedure output area will be.

Notes are positioned at the top of the procedure output area and are left justified. The statements appear one below another in the order they appear in the program. Unless otherwise specified, they use the default hardware font and are one unit high.

For more information on titles, footnotes, and notes in the graphics output area, see “Placement of Graphic Elements in the Graphics Output Area” on page 34.

Syntax

TITLE<1...10>< *text-argument(s)*>;

FOOTNOTE<1...10>< *text-argument(s)*>;

NOTE< *text-arguments(s)*>;

text-argument(s) can be one or more of these in any order:

'text-string'

text-options

text-options can be one or more options from any or all of the following categories:

- appearance options
 - COLOR=*color*
 - FONT=*font*
 - HEIGHT=*text-height*<*units*>
- placement and spacing options
 - JUSTIFY=LEFT | CENTER | RIGHT
 - LSPACE=*line-space*<*units*>
 - MOVE=(*x,y*)<*units*>
- baseline angling and character rotation options
 - ANGLE=*degrees*
 - LANGLE=*degrees*
 - ROTATE=*degrees*
- boxing, underlining, and line drawing options
 - BCOLOR=*background-color*
 - BLANK=YES
 - BOX=1...4
 - BSPACE=*box-space*<*units*>
 - DRAW=(*x,y...*,*x-n,y-n*)<*units*>
 - UNDERLIN=0...3

Options

When the syntax of an option includes *units*, use one of these:

CELLS	character cells
CM	centimeters
IN	inches
PT	points
PCT	percentage of the graphics output area

If you omit *units*, a unit specification is searched for in this order:

- 1 the GUNIT= option in a GOPTIONS statement
- 2 the default unit, CELLS.

ANGLE=*degrees*

A=*degrees*

specifies the angle of the *baseline* of the text string(s) following the option 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).

Angled titles or footnotes may require more vertical space and, consequently, may 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.

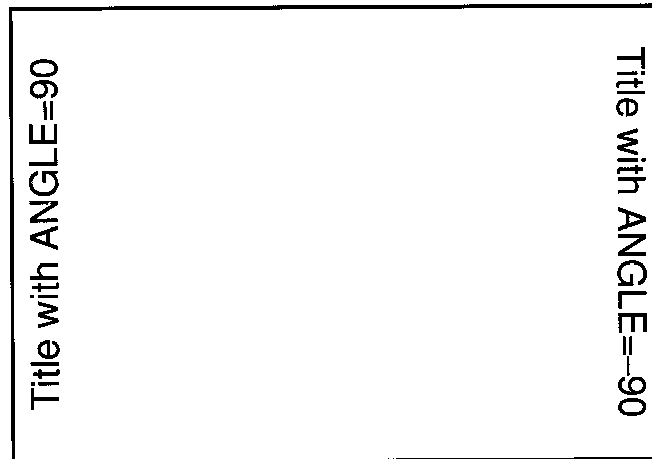
Using the ANGLE= option after one text string and before another can reset some options to their default values. See “Using Options That Can Reset Other Options” on page 265.

ANGLE= 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:*

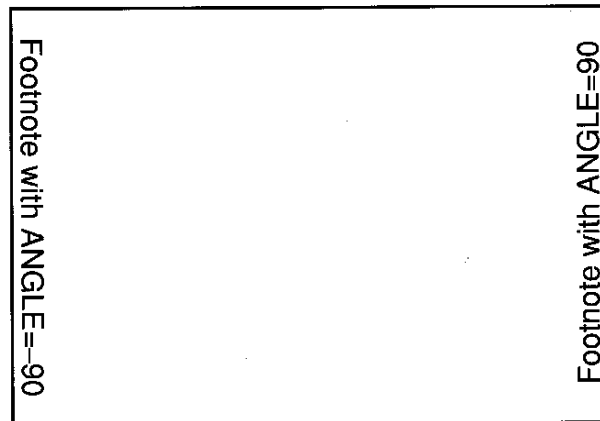
Figure 8.24 on page 254 shows how ANGLE=90 degrees or ANGLE=-90 degrees positions and rotates title text.

Figure 8.24 Positioning Titles with the ANGLE= Option



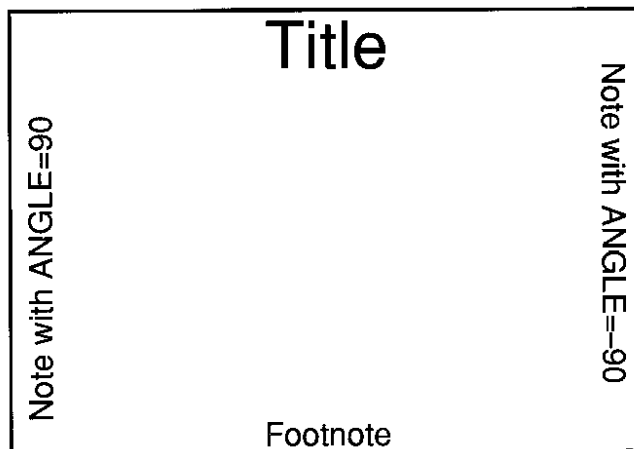
- *With the FOOTNOTE statement:*

Figure 8.25 on page 255 shows how ANGLE=90 degrees or ANGLE=-90 degrees positions and rotates footnote text.

Figure 8.25 Positioning Footnotes with the ANGLE=Option

- *With the NOTE statement:*

Figure 8.26 on page 255 shows how ANGLE= 90 degrees or -90 degrees positions and rotates note text.

Figure 8.26 Positioning Notes with the ANGLE= Option

See also: the LANGLE= option on page 260 and the ROTATE= option on page 262

Featured in: “Example 6. Enhancing Titles” on page 278

BCOLOR=*background-color*

BC=*background-color*

specifies the background color of a box produced by the BOX= option. If you omit BOX=, BCOLOR= is ignored. By default, the background color of the box is the same as the background color for the entire graph. The color of the frame of the box is determined by the color specification used in BOX=.

Note: BCOLOR= may be reset by ANGLE= or JUSTIFY=, or by MOVE= with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

See also: the BOX= on page 256 option

Featured in: “Example 6. Enhancing Titles” on page 278

BLANK=YES

BL=YES

protects the box and its contents from being overwritten by any subsequent graphics elements by blanking out the area where the box is displayed. BLANK= enables you to overlay graphics elements with boxed text. It is ignored if you omit BOX=. Because titles and footnotes are written from the highest numbered to the lowest numbered, the BLANK= option only blanks out titles and footnotes of a lower number.

Note: BLANK= may be reset by ANGLE= or JUSTIFY=, or by MOVE= with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

See also: the BOX= on page 256 option

Featured in: “Example 6. Enhancing Titles” on page 278

BOX=1...4

BO=1...4

draws a box around one line of text. A value of 1 produces the thinnest box lines; 4 produces the thickest. Boxing angled text does not produce an angled box; the box is sized to include the angled text.

The color of the box is either:

- the color specified by the COLOR= option in the statement
- the default text color.

COLOR= affects only the frame of the box. To color the background of the box, use BCOLOR=.

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 JUSTIFY= to create multiple lines of text within a box.

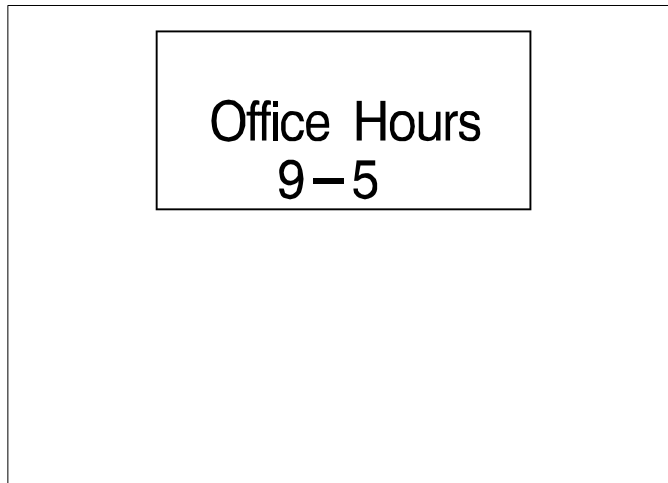
To draw a box around multiple lines of text, you can either

- Use MOVE= with relative coordinates to position the lines of text where you want them and enclose them with BOX=. For example, this statement produces the boxed note shown in Figure 8.27 on page 257:

```
note font=swiss justify=center box=3
      'Office Hours'          move=(40pct,-12pct) '9-5';
```

- Use the DRAW= option to draw the box and do not use BOX=.

Figure 8.27 Using the BOX= Option and the MOVE= Option to Box Multiple Lines of Text



Note: BOX= may be reset by ANGLE= or JUSTIFY=, or by MOVE= with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

See also: the BCOLOR= on page 255 option and the BLANK= on page 256 option and the BSPACE= on page 257 option

Featured in: “Example 6. Enhancing Titles” on page 278

BSPACE=*box-space*<*units*>

BS=*box-space*<*units*>

specifies the amount of space between the boxed text and the box. The space above the text is measured from the font maximum, and the space below the text is measured from the font minimum. By default, BSPACE=1. If BOX= is not used, BSPACE= is ignored.

The spacing is uniform around the box. For example, BSPACE=.5IN leaves one-half inch of space between the text and the top, bottom, and sides of the box.

Note: BSPACE= may be reset by ANGLE= or JUSTIFY=, or by MOVE= with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

See also: the BOX= on page 256 option

COLOR=*color*

C=*color*

specifies the color for the following text, box, or line. COLOR= affects all text, lines, and boxes that follow it and stays in effect until another COLOR= specification is encountered.

Change colors as often as you like. For example, this statement produces a title with red text in a box with a blue frame and a cream background:

```
title color=red 'Total Sales' color=blue
      box=3 bcolor=cream;
```

Although BCOLOR= controls the background color of the box, the frame color is controlled with the COLOR= that precedes BOX=.

If you omit COLOR=, 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 colors list.

See also: the BCOLOR= on page 255 option

DRAW=(*x,y...*,*x-n,y-n*)<*units*>

D=(*x,y...*,*x-n, y-n*)<*units*>

draws lines anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Absolute Coordinates	Relative Coordinates
<i>x</i> < <i>units</i> >	\pm <i>x</i> < <i>units</i> >
<i>y</i> < <i>units</i> >	\pm <i>y</i> < <i>units</i> >

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. DRAW= does not affect the positioning of text.

The starting point for lines specified with relative coordinates begins at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued and the relative draw is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates. For example, DRAW=(+0,+0,+0,1IN) draws a vertical line from the end of the text to one inch from the bottom of the graphics output area.

FONT=*font*

F=*font*

specifies the font for the subsequent text. See Chapter 6, “SAS/GRAPH Fonts,” on page 125 for details on specifying *font*. 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.

Featured in: “Example 6. Enhancing Titles” on page 278

HEIGHT=*text-height*<*units*>

H=*text-height*<*units*>

specifies the height of text characters in number of units. By default, HEIGHT=1. Height is measured from the font minimum to the capline. Ascenders may extend above the capline, depending on the font.

If your text line is too long to be displayed in the height specified in HEIGHT=, 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 HEIGHT=, a text height specification is searched for in this order:

□ for a TITLE1 statement:

- 1 the HTITLE= option in a GOPTIONS statement
- 2 the HTEXT= option in a GOPTIONS statement
- 3 the default value, 2.

By default, a TITLE1 title is twice the height of all other titles.

□ for all other TITLE statements and the FOOTNOTE and NOTE statements:

- 1 the HTEXT= option in a GOPTIONS statement
- 2 the default value, 1.

Featured in: “Example 1. Ordering Axis Tick Marks with SAS Datetime Values” on page 266 and “Example 6. Enhancing Titles” on page 278

JUSTIFY=LEFT | CENTER | RIGHT

J=L | C | R

specifies the alignment of the text string. The default depends on the statement with which you use JUSTIFY=:

- for a FOOTNOTE statement the default is CENTER
- for a NOTE statement the default is LEFT
- for a TITLE statement the default is CENTER.

All the text strings following JUSTIFY= are treated as a single string and are displayed as one line that is left-, right-, or center-aligned.

You can change the justification within a single line of text. For example, this NOTE statement displays a date on the left side of the output and the page number on the same line on the right:

```
note 'June 28, 1997' justify=right 'Page 3';
```

In addition, you can use JUSTIFY= to produce multiple lines of text by repeating JUSTIFY= with the same value before the text string for each line. Multiple lines of text with the same justification are blocked together. For example, this TITLE statement produces a three-line title with each line right-justified:

```
title justify=right 'First Line'
      justify=right 'Second Line'
      justify=right 'Third Line';
```

You can get the same effect with three TITLE statements, each specifying JUSTIFY=RIGHT. If you produce a block of text by specifying the same justification for multiple text strings, and then change the justification for an additional text string, that text is placed on the same line as the first string specified in the statement.

Note: Using JUSTIFY= 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 265. △

Featured in: “Example 3. Rotating Plot Symbols through the Colors List” on page 271

LANGLE=*degrees*

LA=*degrees*

specifies the angle of the *baseline* of the following 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 may require more vertical space and consequently may increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using BOX= with angled text does not produce an angled box; the box is sized to accommodate the angled note.

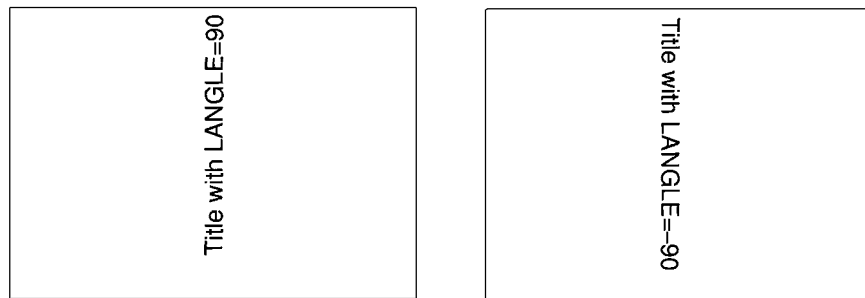
Unlike ANGLE=, LANGLE= does not reset any other options. Therefore, LANGLE= is easier to use because you do not need to repeat options after a text break.

LANGLE= has the same effect on the text as ANGLE=, except when you specify an angle of 90 degrees or -90 degrees. With these specifications, the result depends on the statement in which you use the option:

- *With the TITLE statement:*

Figure 8.28 on page 260 shows how LANGLE=90 degrees and LANGLE=-90 degrees positions and rotates titles.

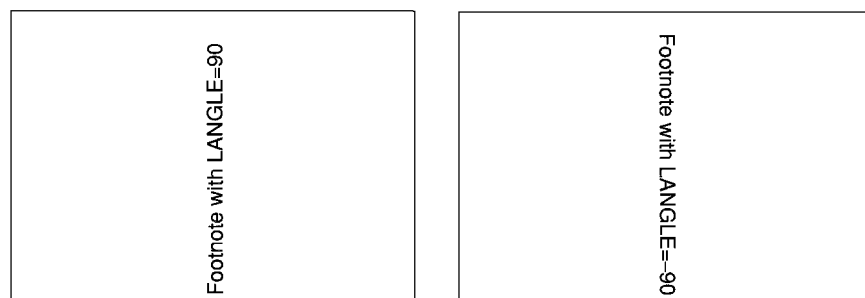
Figure 8.28 Positioning Titles with the LANGLE= Option



- *With the FOOTNOTE statement:*

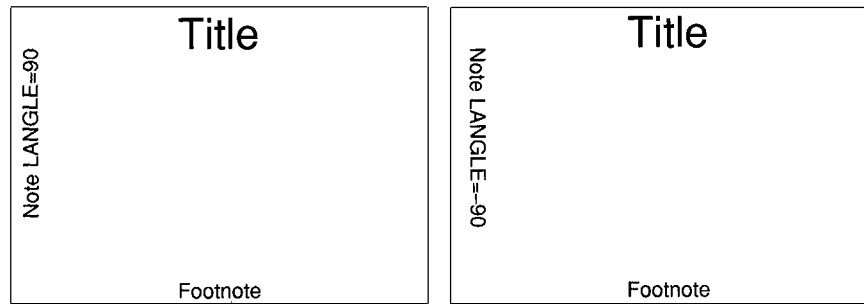
Figure 8.29 on page 260 shows how LANGLE=90 degrees and LANGLE=-90 degrees positions and rotates footnotes.

Figure 8.29 Positioning Footnotes with the LANGLE= Option



- *With the NOTE statement:*

Figure 8.30 on page 261 shows how LANGLE=90 degrees and LANGLE=-90 degrees positions and rotates notes.

Figure 8.30 Positioning Notes with the LANGLE= Option

See also: the ANGLE= on page 253 option

LSPACE=*line-space* <units>

LS=*line-space* <units>

specifies the amount of spacing *above* lines of note and title text and the amount of spacing *below* lines of footnote text. For notes and titles, the spacing is measured from the capline of the current line to the font minimum of the line above. For footnotes, the spacing is measured from the font minimum of the current line to the capline of the line below. By default, LSPACE=1.

Note: LSPACE= may be reset by ANGLE= or JUSTIFY=, or by MOVE= with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

MOVE=(*x,y*) <units>

M=(*x,y*) <units>

positions subsequent text or lines anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Absolute Coordinates	Relative Coordinates
<i>x</i> <units>	\pm <i>x</i> <units>
<i>y</i> <units>	\pm <i>y</i> <units>

Commas between coordinates are optional; you can use blanks instead.

The starting point for lines specified with relative coordinates begins with the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued and the relative move is measured from where a zero-length text string would have ended, given the normal placement for the statement. You can mix relative and absolute coordinates.

MOVE= overrides a JUSTIFY= specified for the same text string.

If a NOTE, FOOTNOTE, or TITLE statement uses MOVE= to position the text so that the statement does not use its default position, the text of the next NOTE, FOOTNOTE, or TITLE statement occupies the unused position and no blank lines are displayed.

Note: If you specify MOVE= with at least one absolute coordinate and if the option follows one text string and precedes another, some options can be reset to their default values. See “Using Options That Can Reset Other Options” on page 265. Δ

Featured in: “Example 2. Specifying Logarithmic Axes” on page 269 and “Example 6. Enhancing Titles” on page 278

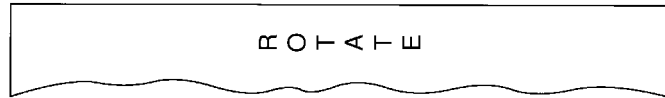
ROTATE=*degrees*

R=*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 ANGLE= or LANGLE=. 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 8.31 on page 262 shows how characters are positioned when ROTATE=90 is used with the default (horizontal) baseline.

Figure 8.31 Tilting Characters with the ROTATE= Option



See also: the ANGLE= on page 253 option

Featured in: “Example 6. Enhancing Titles” on page 278

text-string(s)

is one or more strings up to 200 characters. You must enclose text strings in single or double quotation marks. The text appears exactly as you type it in the statement, including uppercase and lowercase characters and blanks.

To use single quotation marks or apostrophes within the title, you can either

- use a pair of single quotation marks together:

```
footnote 'All''s Well That Ends Well';
```

- enclose the text in double quotation marks:

```
footnote "All's Well That Ends Well";
```

Because FOOTNOTE, NOTE, and TITLE statements concatenate all text strings, the strings must contain the correct spacing. With a series of strings, add blanks at the beginning of a text string rather than at the end, as in this example:

```
note color=red 'Sales:' color=blue ' 2000';
```

With some fonts, you produce certain characters by specifying a hexadecimal value. A trailing **x** identifies a string as a hexadecimal value. For example, this statement* produces the title **Profits Increase £ 3,000**:

```
title font=swiss 'Profits Increase ' '18'x '3,000';
```

For more information see “Specifying Special Characters” on page 130.

In addition, you can embed one or more of the following in the string:

#BYLINE

substitutes the entire BY line without leading or trailing blanks for **#BYLINE** in the text string, and displays the BY line in the footnote, note, or title produced by the statement.

* This statement assumes you are using a U.S. key map.

#BYVAL n | #BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value in the footnote, note, or title produced by the statement. Specify the variable with one of these:

n specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Featured in: “Example 7. Using BY-group Processing to Generate a Series of Charts” on page 280 and “Example 9. Combining Graphs and Reports in a Web Page” on page 287

#BYVAR n | #BYVAR(*BY-variable-name*)

substitutes the name of the BY-variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label in the footnote, note, or title produced by the statement. Specify the variable with one of these:

n specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

A BY variable name displayed in a title, note, or footnote is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

For more information, see “Substituting BY Line Values in a Text String” on page 266.

UNDERLIN=0...3**U=0...3**

underlines subsequent text. Values of 1, 2 and 3 underline with an increasingly thicker line. UNDERLIN=0 halts underlining for subsequent text.

Underlines follow the text baseline. If you use an LANGLE= or ANGLE= option for the line of text, the underline is drawn at the same angle as the text.

Underlines do not break up to follow rotated characters. See the ROTATE= on page 262 option.

To make the text and the underline the same color, specify a COLOR= *before* the UNDERLIN= that precedes the text string. To make the text a different color, specify COLOR= *after* the UNDERLIN=.

Note: UNDERLIN= may be reset by ANGLE= or JUSTIFY=, option, or by the MOVE= option with absolute coordinates. See “Using Options That Can Reset Other Options” on page 265 for details. Δ

Featured in: “Example 6. Enhancing Titles” on page 278

Using TITLE and FOOTNOTE Statements

You can define TITLE and FOOTNOTE statements anywhere in your SAS program. They are global and remain in effect until you cancel them or until you end your SAS

session. All currently defined FOOTNOTE and TITLE statements are automatically displayed.

You can define up to ten TITLE statements and ten FOOTNOTE statements in your SAS session. A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement. You do not have to start with TITLE1 and you do not have to use sequential statement numbers. Skipping a number in the sequence leaves a blank line.

You can use as many text strings and options as you want, but place the options before the text strings they modify. See “Using Multiple Options” on page 264.

The most recently specified TITLE or FOOTNOTE statement of any number completely replaces any other TITLE or FOOTNOTE statement of that number. In addition, it cancels all TITLE or FOOTNOTE statements of a higher number. For example, if you define TITLE1, TITLE2, and TITLE3, resubmitting the TITLE2 statement cancels TITLE3.

To cancel individual TITLE or FOOTNOTE statements, define a TITLE or FOOTNOTE statement of the same number without options (a null statement):

```
title4;
```

But remember that this will cancel all other existing statements of a higher number.

To cancel all current TITLE or FOOTNOTE statements, use the RESET= graphics option in a GOPTIONS statement:

```
goptions reset=footnote;
```

Specifying RESET=GLOBAL or RESET=ALL also cancels all current TITLE and FOOTNOTE statements as well as other settings.

Using the NOTE Statement

NOTE statements are local, not global, and they must be defined within a procedure or RUN-group with which they are used. They remain in effect for the duration of the procedure that includes NOTE statements in any of its RUN-groups or until you end your SAS session. All notes defined in the current RUN group, as well as those defined in previous RUN-groups, are displayed in the output as long as the procedure remains active.

You can use as many text strings and options as you want, but place the options before the text strings they modify. See “Using Multiple Options” on page 264.

Using Multiple Options

In each statement you can use as many text strings and options as you want, but you must place the options before the text strings they modify. Most options affect all text strings that follow them in the same statement, unless the option is explicitly reset to another value. In general, TITLE, FOOTNOTE, and NOTE statement options stay in effect until one of these events occurs:

- the end of the statement is reached
- a new specification is made for that option.

For example, this statement specifies that one part of the note be red and another part blue, but the height for all the text is 4:

```
note height=4 color=red 'Red Tide'
      color=blue ' Effects on Coastal Fishing';
```

Setting Defaults

You can set default characteristics for titles (including TITLE1 definitions), footnotes, and notes by using the following graphics options in a GOPTIONS statement:

CTITLE=*color*

sets the default color for all titles, footnotes, and notes; overridden by the COLOR= option in a TITLE, FOOTNOTE, or NOTE statement.

CTEXT=*text-color*

sets the default color for all text; overridden by CTITLE= for titles, footnotes, and notes.

FTITLE=*title-font*

sets the default font for TITLE1 definitions; overridden by FONT= in the TITLE1 statement.

FTEXT=*text-font*

sets the default font for all text, including the TITLE1 statement if FTITLE= 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 HTITLE= is not used; overridden by the HEIGHT= option a TITLE, FOOTNOTE, or NOTE statement.

See Chapter 9, “Graphics Options and Device Parameters Dictionary,” on page 301 for a complete description of each option.

Using Options That Can Reset Other Options

The ANGLE=, MOVE=, and JUSTIFY= options affect the position of the text and cause *text breaks*. (To cause a text break, MOVE= 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. Δ

If in a TITLE, FOOTNOTE, or NOTE statement, before the first text string, you use an option that can be reset (such as UNDERLIN=) and before the second string you use an option that resets it (such as JUSTIFY=), 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 UNDERLIN= *before* the second text string and *after* JUSTIFY=:

```
title underlin=2 'Line 1' justify=left
      underlin=2 'Line 2';
```

Substituting BY Line Values in a Text String

To use the #BYVAR and #BYVAL options, insert the option in the text string at the position you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character, either a space or other nonalphanumeric character, such as the quote that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string. To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text. If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

If you use a #BYVAR or #BYVAL specification for a variable that is not named in the BY statement (such as #BYVAL2 when there is only one BY-variable or #BYVAL(ABC) when ABC is not a BY-variable or does not exist), or if there is no BY statement at all, the substitution for #BYVAR or #BYVAL does not occur. No error or warning message is issued and the option specification is displayed with the rest of the string. The graph will continue to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see “BY Statement” on page 177.

Note: This feature is not available in the Data Step Graphics Interface or in the Annotate facility since BY lines are not created in a DATA step. Δ

Example 1. Ordering Axis Tick Marks with SAS Datetime Values

Features:

AXIS statement options:

```
COLOR=
LABEL=
MAJOR=
MINOR=
OFFSET=
ORDER=
```

FOOTNOTE statement option:

```
HEIGHT=
```

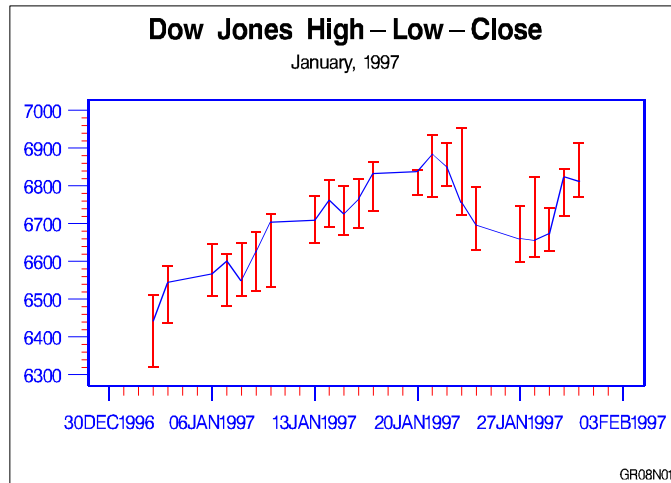
SYMBOL statement options:

```
CI=
CV=
INTERPOL=
WIDTH=
```

GOPTIONS statement options:

FTITLE=
 GUNIT=
 HTEXT=
 HTITLE=

Sample library member: GR08N01



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. The default unit specification for heights in the graph are percent of the graphics output area as specified by `GUNIT=` in the `GOPTIONS` statement. The `GOPTIONS` statement also specifies the default fonts for `TITLE1` and for other text.

Set the graphics environment. `GUNIT=` specifies the units in percent of the graphics output area. `HTITLE=` specifies the height for `TITLE1` text. `HTEXT=` specifies the height for all other text. `FTITLE=` specifies `SWISSB` as the font for `TITLE1`.

```
goptions reset=global gunit=pct border
         cback=white
         colors=(black blue green red)
         ftitle=swissb ftext=swiss htitle=6
         htext=4;
```

Create the data set. `DOWHLC` contains the high, low, and close values of the Dow Jones Industrial index for each business day for a month.

```
data dowhlc;
  input date date9. high low close;
  format date date9.;
  datalines;
02JAN1997    6511.38    6318.96    6442.49
03JAN1997    6586.42    6437.10    6544.09
...more data lines...
30JAN1997    6621.82    6481.75    6600.66
31JAN1997    6621.82    6481.75    6600.66
;
```

Prepare the data for a high-low plot. DOWHLC2 generates three records for each date, storing each date's high, low, and close values in variable DOW.

```
data dowhlc2;
  set dowhlc;
  drop high low close;
  dow=high; output;
  dow=low; output;
  dow=close; output;
```

Define titles and footnote. HEIGHT=3 in the FOOTNOTE statement overrides the height specified by HTEXT= in the GOPTIONS statement.

```
title1 'Dow Jones High-Low-Close';
title2 'January, 1997';
footnote height=3 justify=right 'GR08N01 ' ;
```

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. CV= colors the vertical lines, and CI= colors the line that joins the close values. WIDTH= controls the thickness of the line that joins the close points.

```
symbol interpol=hiloctj
  cv=blue
  ci=red
  width=2;
```

Define characteristics of the horizontal axis. ORDER= uses a SAS date value to set the major tick marks. OFFSET= moves the first and last tick marks to make room for the tick mark value. COLOR= makes all axis elements red. MAJOR= and MINOR= modify the size and color of the major and minor tick marks.

```
axis1 order=('30DEC96'd to '03FEB97'd by week)
  offset=(3,3)
  color=blue
  label=none
  major=(height=3 width=2)
  minor=(number=6 color=red height=2 width=1)
  width=3;
```

Define characteristics of the vertical axis. LABEL=NONE suppresses the AXIS label. The COLOR= suboption in MINOR= overrides the COLOR= option.

```
axis2 color=blue
  label=none
  major=(height=3)
  minor=(number=4 color=red height=1)
  offset=(2,2);
```

Generate the plot and assign AXIS definitions. HAXIS= assigns AXIS1 to the horizontal axis, and VAXIS= assigns AXIS2 to the vertical axis.

```
proc gplot data=dowhlc2;
  plot dow*date / haxis=axis1
  vaxis=axis2;
run;
quit;
```

Example 2. Specifying Logarithmic Axes

Features:

AXIS statement options:

LABEL=
 LENGTH=
 LOGBASE=
 LOGSTYLE=
 MAJOR=
 MINOR=
 VALUE=

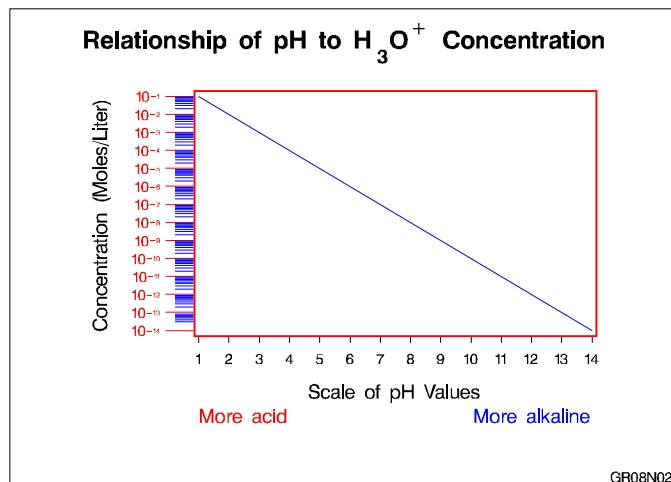
TITLE statement option:

MOVE=

GOPTIONS statement options:

GUNIT=
 VPOS=

Sample library member: GR08N02



This example illustrates the AXIS statement options LOGBASE= and LOGSTYLE=. The horizontal axis represents pH level. The vertical axis, which represents the concentration of the hydroxide ion expressed as moles per liter, is scaled logarithmically. In addition, this example shows how the TICK= parameter of the VALUE= option modifies individual tick marks.

The example uses the MOVE= option in a TITLE statement to position the title's subscript and superscript text.

Assign the libref and set the graphics environment. GUNIT= specifies units of percent of the graphics output area. VPOS= specifies a resolution for the vertical axis.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
          colors=(black blue green red) vpos=250
          ftitle=swissb ftext=swiss htitle=5
          htext=3;
```

Create the data set. REFLIB.CONCENTR contains the pH values and the concentration amount.

```
data reflib.concentr;
    input ph conc;
    datalines;
1  1E-1
2  1E-2
...more data lines...
13 1E-13
14 1E-14
;
```

Define title and footnote. MOVE= positions subscript 3 and superscript +. Each new position is relative to the last position specified by MOVE=.

```
title1 'Relationship of pH to H'
      move=(-0,-3) h=4 '3'
      move=(+0,+3) h=5 'O'
      move=(+0,+3) h=4 '+'
      move=(-0,-3) h=5 ' Concentration';
footnote j=r 'GR09N02 ';
```

Define symbol characteristics.

```
symbol interpol=join color=blue;
```

Define characteristics for horizontal axis. LABEL= uses the JUSTIFY= suboption to create a descriptive two-line label that replaces the variable name PH. MINOR=NONE removes all minor tick marks. LENGTH= uses the units specified by the GUNIT= graphics option to control the length of the horizontal axis.

```
axis1 label=(h=4 'Scale of pH Values'
            justify=left color=red
            h=3 'More acid'
            justify=right color=blue
            'More alkaline')
      minor=none
      length=60
      width=3;
```

Define characteristics for vertical axis. LOGBASE=10 scales the vertical axis logarithmically, using a base of 10. Each major tick mark represents a power of 10. LOGSTYLE=EXPAND displays minor tick marks in logarithmic progression. LABEL= uses the ANGLE= suboption to place the label parallel to the vertical axis. VALUE= displays the major tick mark values as 10 plus an exponent. The HEIGHT= suboption for each TICK= specification affects only the text following it. Units of CM override the default PCT specified by GUNIT=.

```
axis2 logbase=10
      logstyle=expand
      color=red
      label=(angle=90 h=4 color=black
            'Concentration (Moles/Liter)' )
      value=(tick=1 '10' height=1.5 '-14'
            tick=2 '10' height=1.5 '-13'
            tick=3 '10' height=1.5 '-12'
            tick=4 '10' height=1.5 '-11'
            tick=5 '10' height=1.5 '-10')
```



```

        tick=6 '10' height=1.5 '-9'
        tick=7 '10' height=1.5 '-8'
        tick=8 '10' height=1.5 '-7'
        tick=9 '10' height=1.5 '-6'
        tick=10 '10' height=1.5 '-5'
        tick=11 '10' height=1.5 '-4'
        tick=12 '10' height=1.5 '-3'
        tick=13 '10' height=1.5 '-2'
        tick=14 '10' height=1.5 '-1')
    major=(height=.75 cm)
    minor=(color=blue height=.5 cm);

```

Generate the plot and assign AXIS definitions. AXIS1 modifies the horizontal axis and AXIS2 modifies the vertical axis.

```

proc gplot data=reflib.concentr;
    plot conc*ph / haxis=axis1
                    vaxis=axis2;
run;
quit;

```

Example 3. Rotating Plot Symbols through the Colors List

Features:

GOPTIONS statement options:

COLORS=

HSIZE=

VSIZE=

LEGEND statement options:

LABEL=

SHAPE=

SYMBOL statement options:

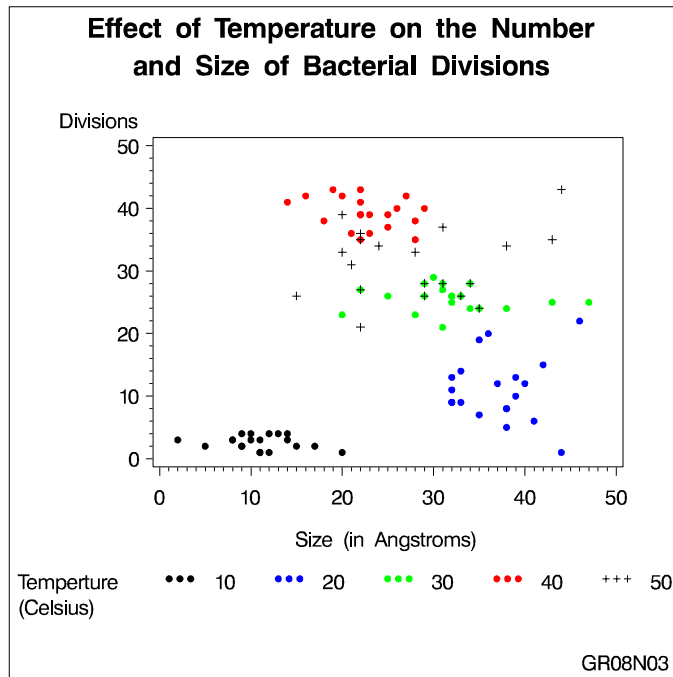
HEIGHT=

VALUE=

TITLE statement option:

JUSTIFY=

Sample library member: GR08N03



This example specifies a plot symbol on a SYMBOL statement and rotates the symbol through the colors 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.

The example uses the GOPTIONS statement to specify the colors for the color rotation. It also uses a LEGEND statement to specify a two-line legend label, and to align the label with the legend values.

Assign the libref and set the graphics environment. COLORS= specifies the colors list. This list is used by the SYMBOL statement. HSIZE= and VSIZE= specify the external dimensions of the graph. Units of IN override the default PCT specified by GUNIT=.

```
libname reflib 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swissb htitle=4 htext=3
         hsize=5in vsize=5in;
```

Create the data set. REFLIB.BACTERIA contains information about the number and size of bacterial divisions at various temperatures.

```
data reflib.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
...more data lines...
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=reflib.bacteria;
```

```

    by temp;
run;

```

Define title and footnote. JUSTIFY= breaks the title into two lines.

```

title1 'Effect of Temperature on the Number'
      justify=center 'and Size of Bacterial
                    Divisions';
footnote1 h=3 j=r 'GR08N03 ';

```

Define symbol characteristics. HEIGHT= specifies a height for the plot symbols. VALUE= specifies a dot for the plot symbol. Because no color is specified, the symbol is rotated through the colors list. Because the plot needs a fifth symbol, the default plus sign is rotated into the colors list to provide that symbol.

```

symbol1 height=2
        value=dot;

```

Define axis characteristics.

```

axis1 label=('Size (in Angstroms)') length=70;
axis2 label=('Divisions');

```

Define legend characteristics. LABEL= specifies text for the legend label. J=L specifies a new line and left-justifies the second string under the first. POSITION= aligns the top label line with the first (and in this case only) value row. SHAPE= specifies a width and height for legend values.

```

legend1 label=(position=(top left)
                'Temperature' j=1 '(Celsius)')
        shape=symbol(4,2);

```

Generate the plot.

```

proc gplot data=reflib.bacteria;
  plot div*mass=temp / frame
                                haxis=axis1
                                vaxis=axis2
                                legend=legend1;
run;
quit;

```

Example 4. Creating and Modifying Box Plots

Features:

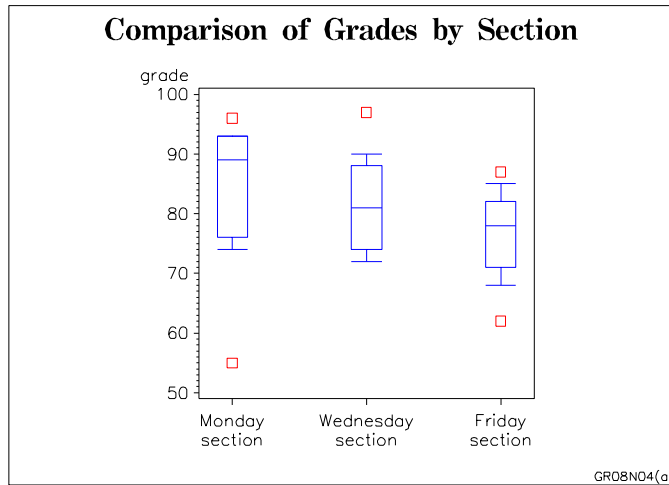
SYMBOL statement options:

```

    BWIDTH=
    CO=
    CV=
    HEIGHT=
    INTERPOL=
    VALUE=

```

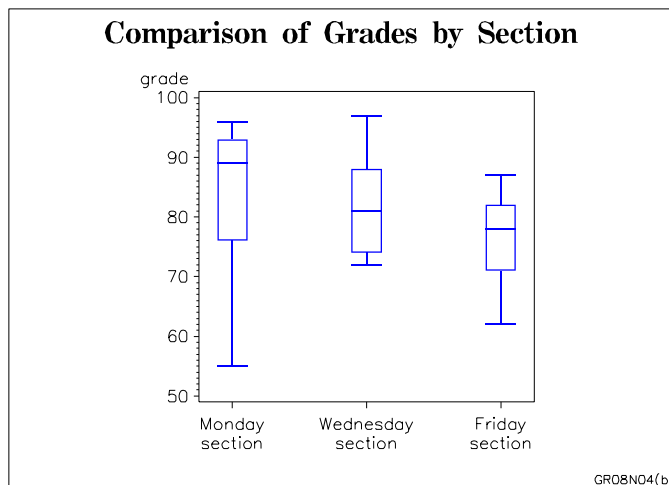
Sample library member: GR08N04



This example shows how to create box plots and how to specify SYMBOL definitions so data outside the box-plot range can be represented with data points. It also shows how to change a box plot's percentile range to see if the new range encompasses the data.

The first plot in the example uses a SYMBOL definition with INTERPOL=BOXT20 to specify a box plot with whisker tops at the 80th percentile and whisker bottoms at the 20th percentile. Data points that are outside this percentile range are represented with squares.

As illustrated in the following output, the example then changes the SYMBOL definition to INTERPOL=BOXT10, which expands the whisker range to the 90th percentile for tops and the 10th percentile for bottoms. There are no data points outside the new percentile range.



Assign the libref and set the graphics environment.

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


Example 5. Filling the Area between Plot Lines

Features:

AXIS statement option:

ORDER=

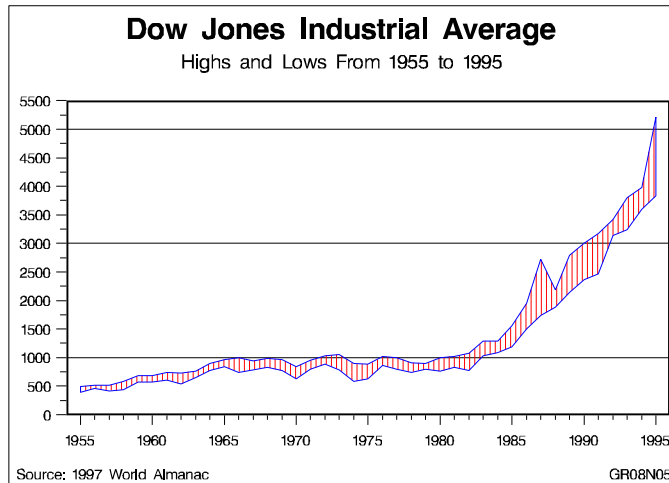
SYMBOL statement options:

CO=

CV=

INTERPOL=

Sample library member: GR08N05



This example shows how to fill the area between two plot lines by concatenating two data sets into one to form a polygon with the data points. It uses a SYMBOL statement to specify a pattern to fill the polygon and to determine the color of the area fill and the outline around the area.

The example plots yearly highs and lows for the Dow Jones Industrial Average. It separates the dependent variables HIGH and LOW to produce an upper plot line and a lower plot line. The dependent variable is named VALUE and the independent variable is named YEAR. When concatenated into one data set, AREA, the data sets form the polygon.

Set the graphics environment.

```
goptions reset=global gunit=pct border cback=white
        colors=(black blue green red)
        ftext=swissb htitle=6 htext=3;
```

Create the data set. STOCKS contains yearly highs and lows for the Dow Jones Industrial Average, and the dates of the high and low values each year.

```
data stocks;
  input year @7 hdate date9. @17 high
        @26 ldate date9. @36 low;
  format hdate ldate date9.;
  datalines;
1955 30DEC1955 488.40 17JAN1955 388.20
1956 06APR1956 521.05 23JAN1956 462.35
```

```

...more data lines...
1994 31JAN1994 3978.36 04APR1994 3593.35
1995 13DEC1995 5216.47 30JAN1995 3832.08
;

```

Restructure the data so that it defines a closed area. Create the temporary data sets HIGH and LOW.

```

data high(keep=year value)
  low(keep=year value);
  set stocks;
  value=high; output high;
  value=low; output low;
run;

```

Reverse order of the observations in LOW.

```

proc sort data=low;
  by descending year;

```

Concatenate HIGH and LOW to create data set AREA.

```

data area;
  set high low;

```

Define titles and footnote.

```

title1 'Dow Jones Industrial Average';
title2 h=4 'Highs and Lows From 1955 to 1995';
footnote j=1 ' Source: 1997 World Almanac'
         j=r 'GR08N05  ';

```

Define symbol characteristics. INTERPOL= specifies a map/plot pattern to fill the polygon formed by the data points. The pattern consists of medium-density parallel lines at 90 degrees. CV= colors the pattern fill. CO= colors the outline of the area. (If CO= were not used, the outline would be the color of the area.)

```

symbol interpol=m3n90
  cv=red
  co=blue;

```

Define axis characteristics. ORDER= places the major tick marks at 5-year intervals.

```

axis1 order=(1955 to 1995 by 5)
  label=none
  major=(height=2)
  minor=(number=4 height=1)
  offset=(2,2)
  width=3;
axis2 order=(0 to 5500 by 500)
  label=none
  major=(height=1.5) offset=(0,0)
  minor=(number=1 height=1);

```

Generate the plot using data set AREA.

```

proc gplot data=area;
  plot value*year / haxis=axis1
                   vaxis=axis2
                   vref=(1000 3000 5000);
run;

```

```
quit;
```

Example 6. Enhancing Titles

Features:

GOPTIONS statement options:

FTITLE=

FTEXT=

GUNIT=

HTITLE=

HTEXT=

TITLE statement options:

ANGLE=

BCOLOR=

BLANK=

BOX=

COLOR=

FONT=

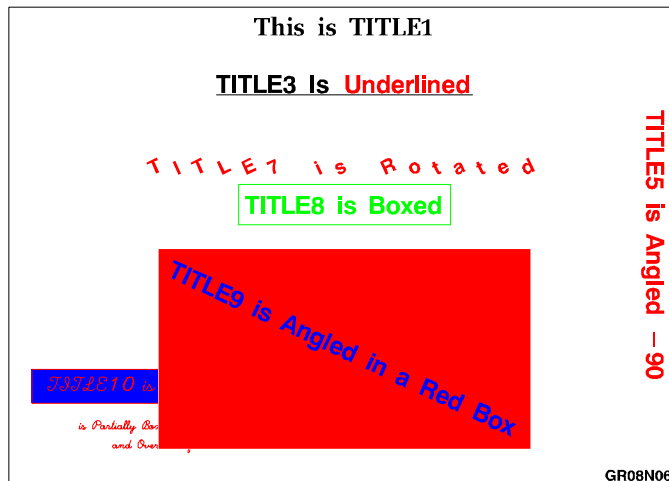
HEIGHT=

MOVE=

ROTATE=

UNDERLIN=

Sample library member: GR08N06



This example illustrates some the ways you can format title text. The same options can be used to format footnotes. The GOPTIONS statement in the example determines the font and heights used for the first title line and all remaining text in the display. GOPTIONS also determines that percentages of the graphics output area are used as the unit of measure for heights in the graph.

Assign the libref and set the graphics environment. FTITLE= assigns the font that is used by the TITLE1 statement. FTEXT= assigns the font for all other text. HTITLE=

makes the height of TITLE1 7 percent of the graphics output area, the units defined by the GUNIT= option. HTEXT= makes the height of all other text 5 percent of the graphics output area.

```
goptions reset=global gunit=pct border cback=white
        colors=(black blue green red)
        ftitle=zapfb ftext=swissb htext=5;
```

Define title1. TITLE1 uses the default font and height defined in the GOPTIONS statement.

```
title1 'This is TITLE1';
```

Define TITLE3. Because TITLE2 is not assigned, the output displays a blank line. UNDERLIN= underlines both text strings.

```
title3 underlin=1
        'TITLE3 Is'
        color=red
        ' Underlined';
```

Define TITLE5. ANGLE= tilts the line of text clockwise 90 degrees and places it at the right edge of the output.

```
title5 color=red
        angle=-90
        'TITLE5 is Angled -90';
```

Define TITLE7. ROTATE= rotates each character in the text string at the specified angle. HEIGHT= overrides HTEXT= in the GOPTIONS statement.

```
title7 height=4
        color=red
        rotate=25
        'TITLE7 is Rotated';
```

Define TITLE8. BOX= draws a green box around the text.

```
title8 color=green
        box=1
        'TITLE8 is Boxed';
```

Define TITLE9. BLANK= prevents the boxed title from being overwritten by TITLE10. The first COLOR= specifies the color of the box border, and BCOLOR= specifies the color of the box background. The second COLOR= specifies the text color.

```
title9 color=red
        box=3
        blank=yes
        bcolor=red
        color=blue
        angle=-25
        'TITLE9 is Angled in a Red Box';
```

Define TITLE10. In this statement, BOX= draws a box around the first text string. BOX= is turned off by the MOVE= that uses absolute coordinates and causes a text break.

```
title10 color=red
        box=1
        bcolor=blue
        move=(5,20)
```

```

font=script
'TITLE10 is in Script and '
move=(10,12)
height=3
'is Partially Boxed, Positioned with
Explicit Moves,'
move=(15,8)
'and Overlaid by TITLE9';

```

Define footnote.

```
footnote h=3 justify=right 'GR08N06 ';
```

Display titles and footnote. All existing titles and footnotes are automatically displayed by the procedure.

```

proc gslide;
run;
quit;

```

Example 7. Using BY-group Processing to Generate a Series of Charts

Features:

AXIS statement options:

```

LABEL=
MAJOR=
MINOR=
NOPLANE
ORDER=
STYLE=
VALUE=

```

BY statement

GOPTIONS statement options:

```

HSIZE=
VSIZE=

```

OPTIONS statement option:

```
NOBYLINE
```

PATTERN statement option:

```
COLOR=
```

TITLE statement:

```
#BYVAL
```

Sample library member: GR08N07

This example uses a BY statement with the GCHART procedure to produce a separate 3D vertical bar chart for each value of the BY variable TYPE. The three charts, which are shown in Figure 8.32 on page 283, Figure 8.33 on page 283, and Figure 8.34 on page 284 following the code, show leading grain producers for 1995 and 1996.

The program suppresses the default BY lines and instead uses #BYVAL in the TITLE statement text string to include the BY variable value in the title for each chart.

The AXIS1 statement that is assigned to the vertical (response) axis is automatically applied to all three graphs generated by the BY statement. This AXIS statement

removes all the elements of the response axis except the label. The same **AXIS** statement also includes an **ORDER=** option. Because this option is applied to all the graphs, it ensures that they all use the same scale of response values.

Because no subgroups are specified and the **PATTERNID=** option is omitted, the color specified in the single **PATTERN** statement is used by all the bars.

Assign the libref and set the graphics environment. **HSIZE=** and **VSIZE=** set the horizontal and vertical size of the graphics output area.

```
libname reflib 'SAS-data-library';
options reset=global gunit=pct border cback=white
        colors=(black blue green red)
        ftitle=swissb ftext=swiss htitle=5
        htext=4 hsize=5in vsize=5in;
```

Create the data set REFLIB.GRAINLDR. REFLIB.GRAINLDR contains data about grain production in five countries for 1995 and 1996. The quantities in **AMOUNT** are in thousands of metric tons. **MEGTONS** converts these quantities to millions of metric tons.

```
data reflib.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
...more data lines...
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' = 'Indonesia'
                'USA' = 'United States';
run;
```

Suppress the default BY-line and define a title that includes the BY-value. **#BYVAL** inserts the value of the BY variable **COUNTRY** into the title of each report.

```
options nobyline;
title1 'Leading #byval(type) Producers'
      j=c '1995 and 1996';
footnotel j=r h=3 'GR08N07 ';
```

Specify a color for the bars.

```
pattern1 color=green;
```

Define the axis characteristics for the response axes. **ORDER=** 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. **MAJOR=**, **MINOR=**, and **VALUE=**

remove the tick marks and values. STYLE=0 removes the line. NOPLANE removes the 3D plane.

```
axis1 order=(0 to 550 by 100)
      label=(angle=90 'Millions of Metric Tons')
      major=none
      minor=none
      value=none
      style=0
      noplane;
```

Define midpoint axis characteristics. SPLIT= defines the character that causes an automatic line break in the axis values.

```
axis2 label=none
      split=' ';
```

Sort data according to values of BY variable. The data must be sorted before running PROC GCHART with the BY statement.

```
proc sort data=reflib.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 RAXIS= causes all three charts to have the same response axis.

```
proc gchart data=temp (where=(megtons gt 31))
      gout=reflib.excat;
      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='gr08n07';

run;
quit;
```

Figure 8.32 Output for BY Value Corn

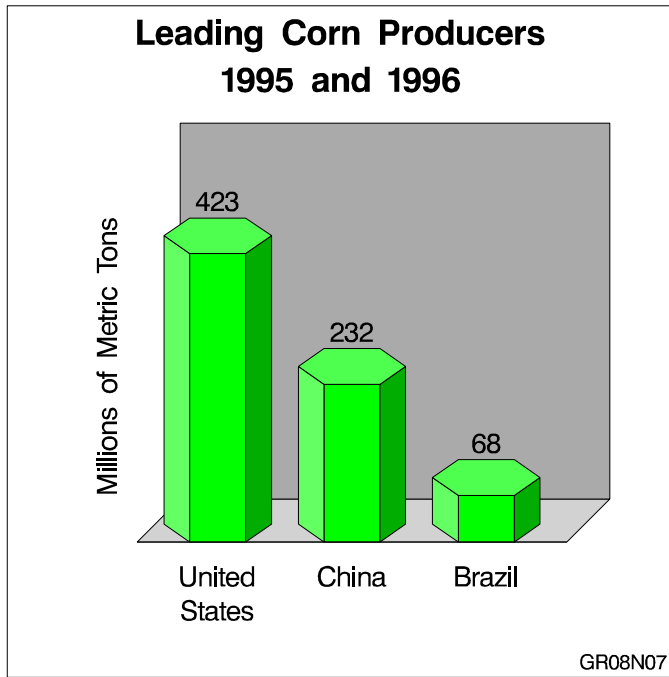


Figure 8.33 Output for BY Value Rice

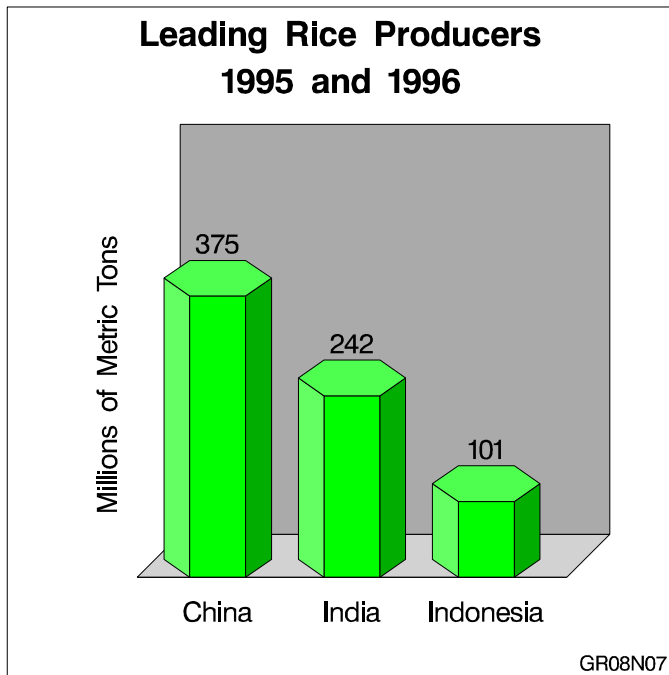
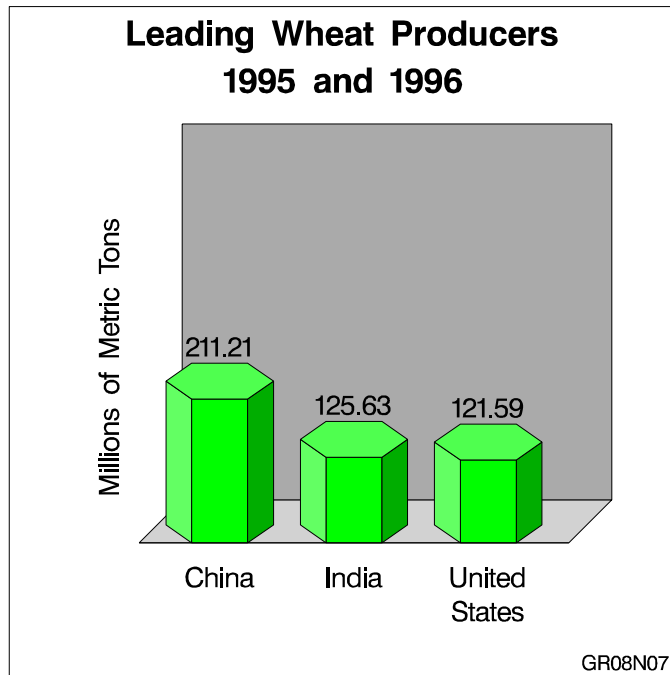


Figure 8.34 Output for BY Value Wheat



Example 8. Creating a Simple Web Page with the ODS HTML Statement

Features:

ODS HTML statement options:

BODY=

PATH=

CLOSE

GOPTIONS statement options:

COLORS=

DEVICE=

TRANSPARENCY

NOBORDER (default)

LEGEND statement options:

ACROSS=

LABEL=

MODE=

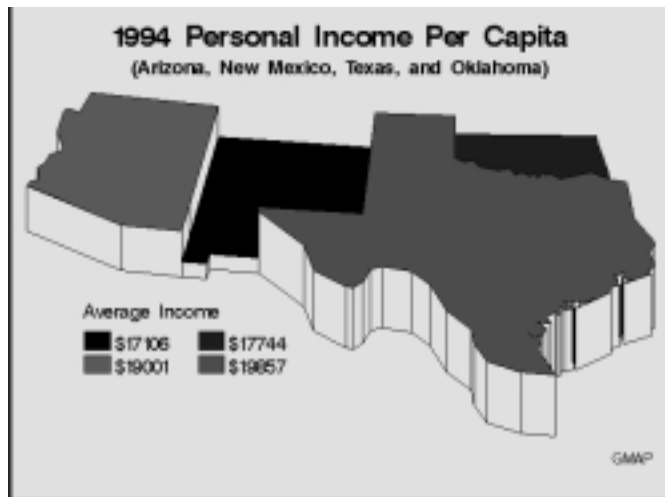
ORIGIN=

SHAPE=

WHERE statement

Sample library member: GR08N08

Figure 8.35 Displaying a Map in a Web Page



This example illustrates the simplest way to use the ODS HTML statement to create an HTML file and a GIF file that you can display in a Web browser. It generates one body file that displays one piece of SAS/GRAPH output – a map of average per capita income for four states.

This example also illustrates default pattern behavior with maps and explicit placement of the legend on the graph. It shows how the default solid map pattern rotates through every color in a colors list defined in the GOPTIONS statement. By default, the outline color is the first color in the list, in this case, BLACK.

And it shows how to use a LEGEND statement to arrange and position a legend so it fits well with the graph's layout.

Assign the libref and the Web-server path. FILENAME assigns the fileref ODSOUT, which is a destination for the HTML and GIF files produced by the example program. To assign that location as the HTML destination for program output, ODSOUT is specified later in the program on the ODS HTML statement's PATH= option. ODSOUT must point to a Web-server location if procedure output is to be viewed on the Web.

```
libname maps 'SAS-MAPS-library';
filename odsout 'path-to-Web-server-space';
```

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. Thus, the graphics output is not displayed in the GRAPH window, although it is written to the graphics catalog and to the GIF files. COLORS= on the GOPTIONS statement defines a list of four colors for the graph.

```
ods listing close;
goptions reset=global gunit=pct cback=white
         colors=(black blue green red)
         ftext=swiss ftitle=swissb htitle=6 htext=4;
```

Create the data set INCOME. INCOME contains state codes for four states and the average income of each state.

```
data income;
  input state income;
  datalines;
04      19001
```

```

35    17106
40    17744
48    19857
;

```

Assign graphics options for producing the ODS HTML output. DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. TRANSPARENCY causes the graphics output to use the Web-page background as the background of the graph. Because the default setting NOBORDER is used, the edge of the graph is not visible on the Web page.

```
goptions device=gif transparency noborder;
```

Open the ODS HTML destination. BODY= names the file for storing HTML output. PATH= specifies the ODSOUT fileref as the destination for all the HTML and GIF files.

```
ods html body='income_body.html'
      path=odsout;
```

Define titles and a footnote for the map. By default, any defined titles and footnotes are included in the graphics output (GIF file).

```
title '1994 Personal Income Per Capita';
title2 f=swissb '(Arizona, New Mexico, Texas, and Oklahoma)';
footnotel h=3 j=r 'GMAP ';
```

Define legend characteristics. ACROSS= defines the number of columns in the legend. ORIGIN= specifies an exact location for the legend. MODE= allows the legend to share the output area. LABEL= specifies a legend label and left-justifies it above the legend values. SHAPE= specifies a size and shape for the legend values.

```
legend across=2
      origin=(10,20)
      mode=share
      label=(position=top
            justify=left
            'Average Income')
      shape=bar(4,4);
```

Generate the prism map. Because the NAME= option is omitted, SAS/GRAPH assigns the default name GMAP to the GRSEG entry in the graphics catalog. This is the name that is assigned to the GIF file created by the ODS HTML statement.

```
proc gmap map=maps.us data=income;
  format income dollar6.0;
  id state;
  prism income / discrete
            legend=legend;
run;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination. You must close the HTML destination before you can view the output with a browser. ODS LISTING opens the Listing destination so that the destination is again available for displaying output during this SAS session.

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

Example 9. Combining Graphs and Reports in a Web Page

Features:

AXIS statement options:

LENGTH=

VALUE=

BY statement

GOPTIONS statement options:

BORDER

DEVICE=

TRANSPARENCY

ODS HTML statement options:

BODY=

CONTENTS=

FRAME=

PATH=

NOGTITLE

OPTIONS statement option:

NOBYLINE

TITLE statement option:

#BYVAL

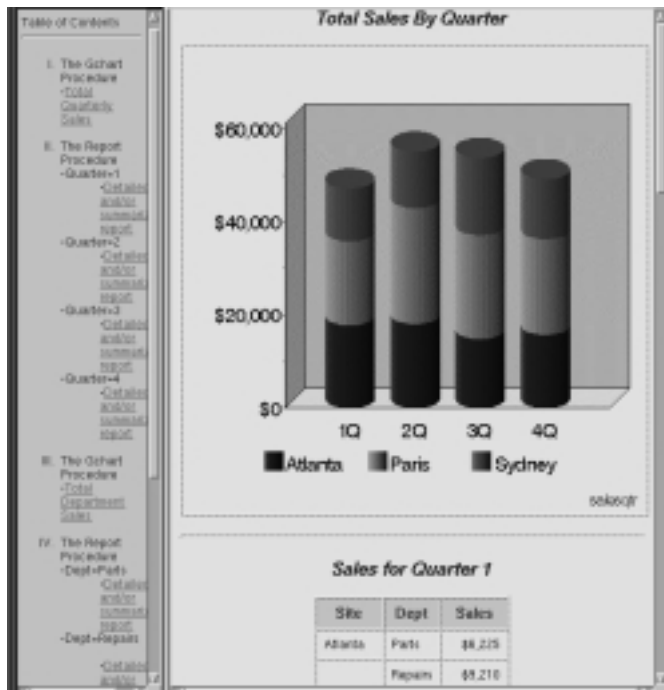
Sample library member: GR08N09

This example generates several graphs of sales data that can be accessed from a single Web page. The graphs are two bar charts of summary sales data and three pie charts that break the data down by site. Each bar chart and an accompanying report is stored in a separate body file.

The three pie charts are generated with BY-group processing and are stored in one body file. The program suppresses the default BY lines and instead includes the BY variable value in the title for each chart. The SAS/GRAPH titles are displayed in the HTML output instead of in the graphics output.

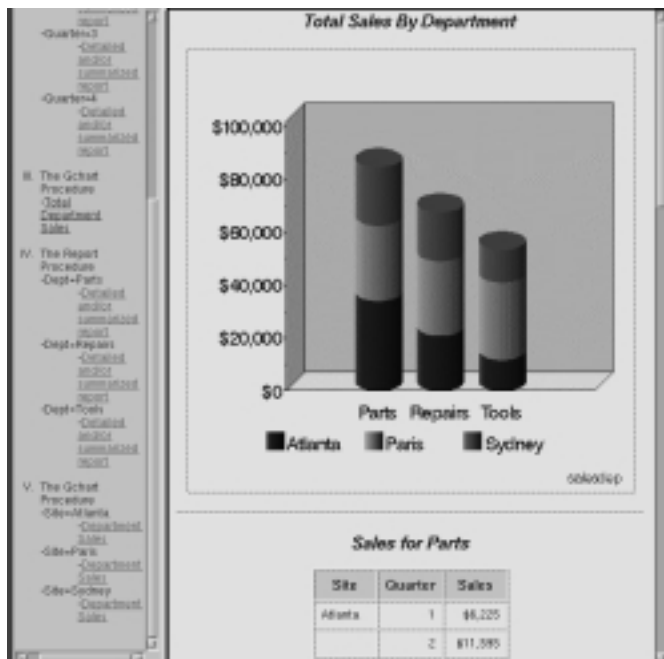
The Web page contains two frames, one that displays a Table of Contents for all the graphs, and one that serves as the display area. Links to each piece of output appear in the table of contents, which is displayed in the left frame. Initially the frame file displays the first body file, which contains a bar chart and a report, as shown in Figure 8.36 on page 288.

Figure 8.36 Browser View of Bar Chart and Quarterly Sales Report



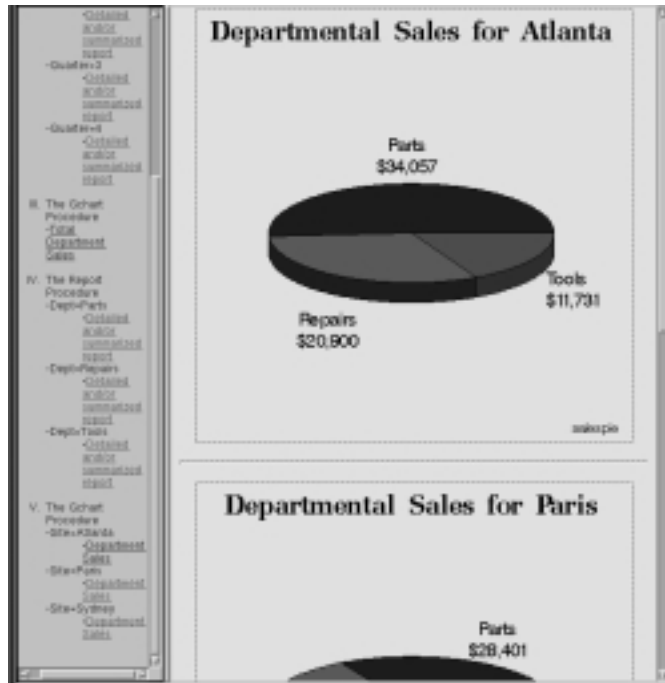
Notice that the chart title is displayed outside the graph as part of the HTML file. Select the link to *Total Department Sales* to display the second bar chart, as shown in Figure 8.37 on page 288.

Figure 8.37 Browser View of Bar Chart and Department Sales Report



Selecting any link for *Department Sales* displays the corresponding pie chart as shown in Figure 8.38 on page 289.

Figure 8.38 Browser View of Pie Charts of Site Sales



Because the pie charts are stored in one file, you can easily see all three by scrolling through the file.

Additional features include AXIS statements that specify the same length for both midpoint axes, so that the bar charts are the same width even though they have a different number of bars.

Assign the Web-server path. FILENAME assigns the fileref ODSOUT, which specifies a destination for the HTML and GIF files produced by the example program. To assign that location as the HTML destination for program output, ODSOUT is specified later in the program on the ODS HTML statement's PATH= option. ODSOUT must point to a Web-server location if procedure output is to be viewed on the Web.

```
filename odsout 'path-to-Web-server-space';
```

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. On the GOPTIONS statement, HSIZE= and VSIZE= set the horizontal and vertical size of the graphics output area. DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. TRANSPARENCY causes the graphics output to use the Web-page background as the background of the graph. BORDER is used so that the border around the graphics output area will be compatible with the borders that are created for nongraphics output.

```
ods listing close;
goptions reset=global gunit=pct border
         colors=(blue green red) ctext=black
         hsize=5in vsize=5in ftitle=zapfb
         ftext=swiss htitle=6 htext=4
```

```
device=gif transparency;
```

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;
Parts   Sydney  1  4043.97
Parts   Atlanta 1  6225.26
Parts   Paris   1  3543.97
Repairs Sydney  1  5592.82
Repairs Atlanta 1  9210.21
Repairs Paris   1  8591.98
Tools   Sydney  1  1775.74
Tools   Atlanta 1  2424.19
Tools   Paris   1  5914.25
Parts   Sydney  2  3723.44
Parts   Atlanta 2 11595.07
Parts   Paris   2  9558.29
Repairs Sydney  2  5505.31
Repairs Atlanta 2  4589.59
Repairs Paris   2  7538.56
Tools   Sydney  2  2945.17
Tools   Atlanta 2  1903.99
Tools   Paris   2  7868.34
Parts   Sydney  3  8437.96
Parts   Atlanta 3  6847.91
Parts   Paris   3  6789.85
Repairs Sydney  3  4426.46
Repairs Atlanta 3  5011.66
Repairs Paris   3  6510.38
Tools   Sydney  3  3767.10
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 Sydney  4  3012.99
Repairs Atlanta 4  2088.30
Repairs Paris   4  5530.37
Tools   Sydney  4  3817.36
Tools   Atlanta 4  4354.18
Tools   Paris   4  6511.70
;
```

Open the ODS HTML destination. 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. BODY= names the file for storing the HTML output. The contents file links to each of the body files written to the HTML destination. PATH= specifies the ODSOUT fileref as the HTML destination for all the HTML and GIF files. NOGTITLE suppresses the graphics titles from the SAS/GRAPH output and displays them through the HTML page.

```
ods html frame='sales_frame.html'
  contents='sales_contents.html'
  body='sales_body1.html'
  path=odsout
  nogtitle;
```

Define title and footnote. TITLE1 uses the font and height specified by FTITLE= and HTITLE= in the GOPTIONS statement.

```
title1 'Total Sales By Quarter';
footnote j=r h=3 'salesqtr ';
```

Define axis characteristics for the first bar chart. In AXIS2, LENGTH= 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. NAME= specifies the name of the catalog entry. Because the PATH= destination is a file storage location and not a specific file name, the name SALESQTR.GIF is assigned to the GIF file, matching the named assigned to the GRSEG on NAME=. DES= specifies the description that is stored in the graphics catalog and used in the Table of Contents.

```
proc gchart data=totals;
  format sales dollar8.;
  vbar3d quarter / discrete
    sumvar=sales
    shape=cylinder
    subgroup=site
    cframe=grayaa
    caxis=black
    width=12
    space=4
    legend=legend1
    maxis=axis2
    raxis=axis1
    des='Total Quarterly Sales'
    name='salesqtr';

run;
quit;
```

Sort the data set for the report of quarterly sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=totals out=qtrsrt;
  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.

```
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 Figure 8.36 on page 288.

```
title1 'Sales for Quarter #byval(quarter)';
proc report data=qtrsrt nowindows;
  by quarter;
  column quarter site dept sales;
  define quarter / noprint group;
  define site / display group;
  define dept / display group;
  define sales / display sum format=dollar8.;
  compute after quarter;
    site='Total';
  endcomp;
  break after site / summarize style=rowheader;
  break after quarter / summarize style=rowheader;
run;
```

Open a new body file for the second bar chart and report. Assigning a new body file closes SALES_BODY1.HTML. The contents and frame files, which remain open, will contain links to all body files.

```
ods html body='sales_body2.html' path=odsout;
```

Define title and footnote for second bar chart.

```
title1 'Total Sales By Department';
footnotel j=r h=3 'salesdep ';
```

Define axis characteristics. These AXIS statements replace the ones defined earlier. As before, LENGTH= defines the length of the midpoint axis.

```
axis1 label=none
  minor=(number=1);
  order=(0 to 100000 by 20000)
axis2 label=none length=70pct;
```

Generate the vertical bar chart of departmental sales.

```
proc gchart data=totals;
  format sales dollar8.;
  vbar3d dept / shape=cylinder
    subgroup=site
    cframe=grayaa
    width=12
    space=4
    sumvar=sales
    legend=legend1
    maxis=axis2
    raxis=axis1
    caxis=black
    des='Total Department Sales'
    name='salesdep';

run;
quit;
```

Sort the data set for the report of department sales. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=totals out=deptsort;
  by dept site;
run;
```

Reset the footnote, define a report title, and generate the report of department sales. #BYVAL inserts the value of the BY variable into the title of each report. The chart and report are shown in Figure 8.37 on page 288.

```
footnote1;
title1 'Sales for #byval(dept)';
proc report data=deptsort nowindows;
  by dept;
  column dept site quarter sales;
  define dept / noprint group;
  define site / display group;
  define quarter / display group;
  define sales / display sum format=dollar8.;
  compute after dept;
    site='Total';
  endcomp;
  break after site / summarize style=rowheader;
  break after dept / summarize style=rowheader;
run;
```

Open a new body file for the pie charts. Assigning a new file as the body file closes SALES_BODY2.HTML. The contents and frame files remain open. GTITLE displays the titles in the graph.

```
ods html body='sales_body3.html' gtitle path=odsout;
```

Sort data set in order of the BY variable before running the GCHART procedure with BY-group processing.

```
proc sort data=totals out=sitesort;
  by site;
run;
```

Define title and footnote. #BYVAL inserts the value of the BY variable SITE into the title for each output.

```
title 'Departmental Sales for #byval(site)';
footnote j=r h=3 '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 NAME= is incremented to provide a unique name for each piece of output.

```
proc gchart data=sitesort;
  format sales dollar8.;
  by site;
  pie3d dept / noheading
    outline=black
    sumvar=sales
    des='Department Sales'
    name='salespie';
run;
```

```
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination.

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

Example 10. Creating a Bar Chart with Drill-down for the Web

Features:

GOPTIONS statement option:

```
RESET=
```

ODS HTML statement options:

```
BODY=
```

```
NOGTITLE
```

```
PATH=
```

Sample library member: GR08N10

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. Figure 8.39 on page 294 shows the bar chart of sales.

Figure 8.39 Vertical Bar Chart of Total Sales

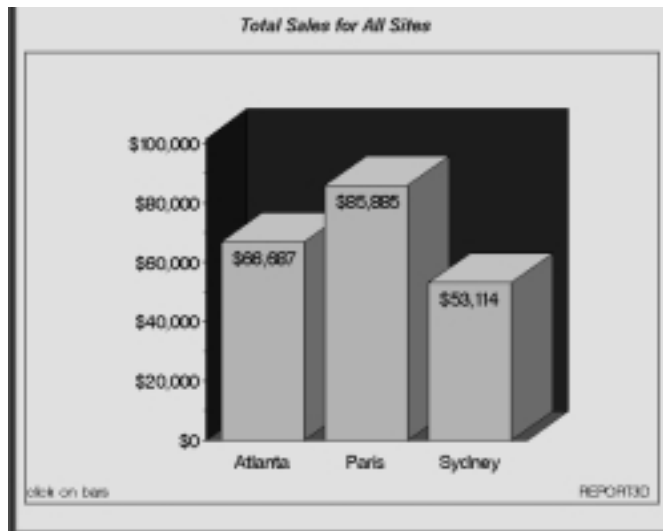


Figure 8.40 on page 295 shows the PROC REPORT output that appears when you click on the bar for Atlanta.

Figure 8.40 PROC REPORT Output Displayed in a Web Browser

Sales Report for Atlanta		
Dept	Quarter	Sales
Paris	1	\$8,025
	2	\$11,555
	3	\$8,049
	4	\$8,389
Repairs	1	\$8,210
	2	\$4,590
	3	\$5,912
	4	\$2,889
Tools	1	\$2,454
	2	\$1,364
	3	\$3,949
	4	\$4,254
Total		\$66,907

For additional information about this program, see “Details” on page 297.

Assign the Web-server path. FILENAME assigns the fileref ODSOUT, which specifies a destination for the HTML and GIF files produced by the example program. To assign that location as the HTML destination for program output, ODSOUT is specified later in the program on the ODS HTML statement’s PATH= option. ODSOUT must point to a Web-server location if procedure output is to be viewed on the Web.

```
filename odsout 'path-to-Web-server-space';
```

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING CLOSE closes the Listing destination for procedure output. On the GOPTIONS statement, DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. TRANSPARENCY causes the graphics output to use the Web-page background as the background of the graph.

```
ods listing close;
goptions reset=global gunit=pct
         colors=(black blue green red)
         hsize=7 in vsize=5 in ftitle=zapfb
         ftext=swiss htitle=6 htext=4
         device=gif transparency noborder;
```

Add the HTML variable to TOTALS and create the NEWTOTAL data set. The HTML variable SITEDRILL contains the targets for the values of the variable SITE. Each HREF value specifies the HTML body file and the name of the anchor within the body file that identifies the target graph.

```
data newtotal;
  set totals;
  length sitedrill $40;
  if site='Atlanta' then
    sitedrill='HREF="report_deptsales.html#IDX1"';

  else if site='Paris' then
    sitedrill='HREF="report_deptsales.html#IDX2"';

  else if site='Sydney' then
```

```
sitedrill='HREF="report_deptsales.html#IDX3"';
run;
```

Open the ODS HTML destination. BODY= names the file for storing HTML output. PATH= specifies the ODSOUT fileref as the HTML destination for all the HTML and GIF files. NOGTITLE suppresses the graph titles from the SAS/GRAPH output and displays them in the HTML.

```
ods html path=odsout
      body='report_body.html'
      nogtitle;
```

Define title and footnote.

```
title1 'Total Sales for All Sites';
footnote1 h=3 j=1 'click on bars' j=r 'REPORT3D ';
```

Assign a pattern color for the bars. Each bar in the graph uses the same PATTERN definition.

```
pattern color=cyan;
```

Define axis characteristics. The VBAR3D statement assigns AXIS1 to the response axis and AXIS2 to the midpoint axis.

```
axis1 order=(0 to 100000 by 20000)
      minor=(number=1)
      label=none;
axis2 label=none offset=(9,9);
```

Generate the vertical bar chart of total sales for each site. HTML= specifies SITEDRILL as the variable that contains the name of the target. Specifying HTML= causes SAS/GRAPH to add an image map to the HTML body file. NAME= specifies the name of the catalog entry.

```
proc gchart data=newtotal;
  format sales dollar8.;
  vbar3d site / discrete
        width=15
        sumvar=sales
        inside=sum
        html=sitedrill
        coutline=black
        cframe=blue
        maxis=axis2
        raxis=axis1
        name='report3d ';

run;
quit;
```

Open the file for the PROC REPORT output. Assigning a new body file closes REPORT_BODY.HTML.

```
ods html body='report_deptsales.html' path=odsout;
```

Sort the data set NEWTOTAL. The data must be sorted in order of the BY variable before running PROC REPORT with BY-group processing.

```
proc sort data=newtotal;
  by site dept quarter;
run;
```

```
quit;
```

Clear the footnote.

```
goptions reset=footnote1;
```

Suppress the default BY-line and define a title that includes the BY-value. #BYVAL inserts the value of the BY variable SITE into the title of each report.

```
options nobyline;
```

```
title1 'Sales Report for #byval(site)';
```

Print a report of departmental sales for each site.

```
proc report data=newtotal nowindows;
  by site;
  column site dept quarter sales;
  define site      / noprint group;
  define dept      / display group;
  define quarter  / display group;
  define sales     / display sum format=dollar8.;
  compute after site;
      dept='Total';
  endcomp;
  break after site / summarize style=rowheader page;
run;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination.

```
ods html close;
```

```
ods listing;
```

Details

This section provides additional information about the pieces of this program and how they work together to generate SAS/GRAPH output with drill-down functionality. It describes

- how an HREF value is built
- how the HTML= option creates an image map in the HTML file
- how the HTML file references the SAS/GRAPH output.

For a description of the relationship between the parts of the program and the HTML and GIF files, see "About SAS/GRAPH Output and the Web."

Building an HREF value

In the DATA step, the variable SITEDRILL is assigned a string that defines the link target for a data value. For example,

```
if site='Atlanta' then
  sitedrill='HREF="report_deptsales.html#IDX1"';
```

The link target is specified by the HTML HREF attribute. The HREF value tells the Web page where to link to when a user selects the region associated with the value **Atlanta**.

For example, clicking on the first bar in the chart links to the target defined by **report_deptsales.html#IDX1**. This target is composed of a filename and an anchor.

The file, **report_deptsales.html**, is generated by the PROC REPORT step. **IDX1** is the anchor that identifies the section of the file that contains the report for the first BY group, **Atlanta**.

Because anchor names increment, in order to assign them accurately you must know how many pieces of output your program generates and in what order. For example, this table lists in order the pieces of output generated by this example and their default anchor names:

Procedure	Output	Anchor name
GCHART	report3d.gif	IDX
REPORT	Atlanta report	IDX1
REPORT	Paris report	IDX2
REPORT	Sydney report	IDX3

Creating an image map

The HTML= option in the GCHART procedure is assigned the variable with the target information – in this case, SITEDRILL.

```
html=sitedrill
```

This option causes SAS/GRAPH to generate in the HTML body file the MAP and AREA elements that compose the image map. It loads the HREF attribute value from SITEDRILL into the AREA element. This image map, which is named **gqcke00k_map**, is stored in **report_body.html** (ODS generates unique map names each time you run the program, so the next time this program runs, the map name will be different):

```
<MAP NAME="gqcke00k_map">
  <AREA SHAPE="POLY"
    HREF="report_deptsales.html#IDX3"
    COORDS="423,409,423,242,510,242,510,409" >
  <AREA SHAPE="POLY"
    HREF="report_deptsales.html#IDX2"
    COORDS="314,409,314,139,401,139,401,409" >
  <AREA SHAPE="POLY"
    HREF="report_deptsales.html#IDX1"
    COORDS="205,409,205,199,292,199,292,409" >
</MAP>
```

The AREA element defines the regions within the graph that you can select to link to other locations. It includes attributes that define the shape of the region (SHAPE=) and position of the region (COORDS=) as well as the link target (HREF=).

The value assigned to the HREF= attribute is contained in the variable assigned to HTML=, in this case SITEDRILL.

Referencing SAS/GRAPH output

In the GOPTIONS statement, DEVICE=GIF causes SAS/GRAPH to create GIF files from the SAS/GRAPH output. It also adds to the open body file an IMG element that points to the GIF file. In this case, SAS/GRAPH adds the following IMG element to **report_body.html**:

```
<IMG SRC="report3d.gif" USEMAP="#gqcke00k_map">
```

The IMG element tells the Web page to get the image from the file `report3d.gif`. It also tells the Web page to use the image map `#report3d_map` to define the hot spots of the bar chart.

See Also

- For more information on the BY, LABEL, OPTIONS, and WHERE statements in base SAS software, see *SAS Language Reference: Dictionary*.

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

SAS/GRAPH® Software: Reference, Version 8

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

ISBN 1-58025-525-6

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

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

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

1st printing, October 1999

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.

OS/2®, OS/390®, and IBM® are registered trademarks or trademarks of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.