CHAPTER 19

The FORMAT Procedure

Overview

The FORMAT procedure enables you to define your own informats and formats for variables. In addition, you can print the contents of a catalog that contains informats or formats, store descriptions of informats or formats in a SAS data set, and use a SAS data set to create informats or formats.

Informats determine how raw data values are read and stored. *Formats* determine how variable values are printed. For simplicity, this section uses the terminology *the informat converts* and *the format prints*.

Informats and formats tell the SAS System the data's type (character or numeric) and form (such as how many bytes it occupies; decimal placement for numbers; how to handle leading, trailing, or embedded blanks and zeros; and so forth). The SAS System provides informats and formats for reading and writing variables. For a thorough description of informats and formats that SAS provides, see the sections on formats and informats in *SAS Language Reference: Dictionary*.

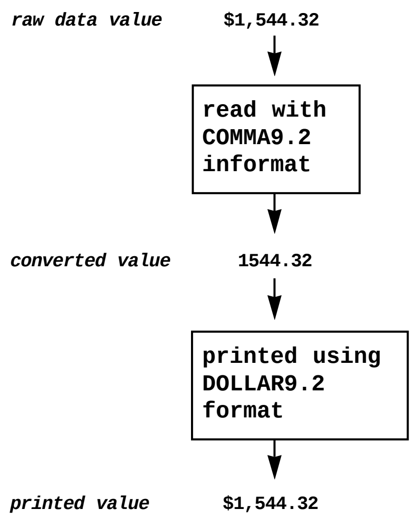
With informats, you can

- □ convert a number to a character string (for example, convert 1 to **`YES`**)
- □ convert a character string to a different character string (for example, convert **`'YES'`** to **`'OUI'`**)
- □ convert a character string to a number (for example, convert **`YES`** to 1)
- □ convert a number to another number (for example, convert 0 through 9 to 1, 10 through 100 to 2, and so forth.

With formats, you can

- □ print numeric values as character values (for example, print 1 as **`MALE`** and 2 as **`FEMALE`**)
- □ print one character string as a different character string (for example, print **`YES`** as **`OUI`**)
- □ print numeric values using a template (for example, print 9458763450 as **`945-876-3450`**).

The following figure summarizes what occurs when you associate an informat and format with a variable. The COMMA*w.d* informat and the DOLLAR*w.d* format are provided by SAS.



In the figure, SAS reads the raw data value that contains the dollar sign and comma. The COMMA9.2 informat ignores the dollar sign and comma and converts the value to 1544.32. The DOLLAR9.2 format prints the value, adding the dollar sign and comma. For more information about associating informats and formats with variables, see "Associating Informats and Formats with Variables" on page 455.

Procedure Syntax

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

Reminder: You can also use appropriate global statements with this procedure. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC FORMAT <*option(s)*>;
  EXCLUDE *entry(s)*;
  INVALUE <$>*name* <(*informat-option(s)*)>
      value-range-set(s);
  PICTURE *name* <(*format-option(s)*)>
      value-range-set-1 <(*picture-1-option(s)*)>
      <*…value-range-set-n* <(*picture-n-option(s)*)>>;
  SELECT *entry(s)*;
  VALUE <$>*name* <(*format-option(s)*)>
      value-range-set(s);

To do this	Use this statement
Exclude catalog entries from processing by the FMTLIB and CNTLOUT= options	EXCLUDE
Create an informat for reading and converting raw data values	INVALUE
Create a template for printing numbers	PICTURE
Select catalog entries from processing by the FMTLIB and CNTLOUT= options	SELECT
Create a format that specifies character strings to use to print variable values	VALUE

PROC FORMAT Statement

Reminder: You can use data set options with the CNTLIN= and CNTLOUT= data set options.See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC FORMAT <*option(s)*>;

To do this	Use this option
Specify a SAS data set from which PROC FORMAT builds an informat or format	CNTLIN=
Create a SAS data set that stores information about informats or formats	CNTLOUT=
Print information about informats or formats	FMTLIB
Specify a SAS catalog that will contain the informats or formats that you are creating in the PROC FORMAT step	LIBRARY=
Specify the number of characters of the informatted or formatted value that appear in PROC FORMAT output	MAXLABLEN=
Specify the number of characters of the start and end values that appear in the PROC FORMAT output	MAXSELEN=
Prevent a new informat or format from replacing an existing one of the same name	NOREPLACE
Print information about each format and informat on a separate page	PAGE

Options

CNTLIN=*input-control-SAS-data-set*
specifies a SAS data set from which PROC FORMAT builds informats and formats. CNTLIN= builds formats and informats without using a VALUE, PICTURE, or INVALUE statement. If you specify a one-level name, the procedure searches only the default data library (either the WORK data library or USER data library) for the data set, regardless of whether you specify the LIBRARY= option.

Tip: A common source for an input control data set is the output from the CNTLOUT= option of another PROC FORMAT step.

See also: "Input Control Data Set" on page 460

Featured in: Example 5 on page 470

CNTLOUT=*output-control-SAS-data-set*
creates a SAS data set that stores information about informats and formats that are contained in the catalog specified in the LIBRARY= option.

If you are creating an informat or format in the same step that the CNTLOUT= options appears, the informat or format that you are creating is included in the CNTLOUT= data set.

If you specify a one-level name, the procedure stores the data set in the default data library (either the WORK data library or the USER data library), regardless of whether you specify the LIBRARY= option.

Tip: You can use an output control data set as an input control data set in subsequent PROC FORMAT steps.

See also: "Output Control Data Set" on page 458

FMTLIB
prints information about all the informats and formats in the catalog that is specified in the LIBRARY= option. To get information only about specific informats or formats, subset the catalog using the SELECT or EXCLUDE statement.

Interaction: The PAGE option invokes FMTLIB.

Tip: If your output from FMTLIB is not formatted correctly, try increasing your linesize.

Tip: If you use the SELECT or EXCLUDE statement and omit the FMTLIB and CNTLOUT= options, the procedure invokes the FMTLIB option and you receive FMTLIB option output.

Featured in: Example 6 on page 472

LIBRARY=*libref*<.*catalog*>
specifies a catalog to contain informats or formats you are creating in the current PROC FORMAT step. The procedure stores these informats and formats in the catalog you specify so that you can use them in subsequent SAS sessions or jobs.

Alias: LIB=

Default: WORK.FORMATS

Tip: SAS automatically searches LIBRARY.FORMATS. You may want to use the LIBRARY libref for your format catalog.

See also: "Storing Informats and Formats" on page 456

Featured in: Example 1 on page 463

MAXLABLEN=*number-of-characters*
specifies the number of characters in the informatted or formatted value that you want to appear in the CNTLOUT= data set or in the output of the FMTLIB option. The FMTLIB option prints a maximum of 40 characters for the informatted or formatted value.

MAXSELEN=*number-of-characters*
specifies the number of characters in the start and end values that you want to appear in the CNTLOUT= data set or in the output of the FMTLIB option. The FMTLIB option prints a maximum of 16 characters for start and end values.

NOREPLACE
prevents a new informat or format that you are creating from replacing an existing informat or format of the same name. If you omit NOREPLACE, the procedure warns you that the informat or format already exists and replaces it.

PAGE
prints information about each format and informat (that is, each entry) in the catalog on a separate page.

Tip: The PAGE option activates the FMTLIB option.

EXCLUDE Statement

Excludes entries from processing by the FMTLIB and CNTLOUT= options.

Restriction: Only one EXCLUDE statement can appear in a PROC FORMAT step.

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

EXCLUDE *entry(s)*;

Required Arguments

entry(s)

specifies one or more catalog entries to exclude from processing. Catalog entry names are the same as the name of the informat or format that they store. Because informats and formats can have the same name, and because character and numeric informats or formats can have the same name, you must use certain prefixes when specifying informats and formats in the EXCLUDE statement. Follow these rules when specifying entries in the EXCLUDE statement:

- Precede names of entries that contain character formats with a dollar sign ($).
- Precede names of entries that contain numeric informats with an at sign (@).
- Precede names of entries that contain character informats with an at sign and a dollar sign (for example, @$*entry-name*).

Shortcuts to Specifying Names

You can use the colon (:) and hyphen (-) wildcard characters to exclude entries. For example, the following EXCLUDE statement excludes all formats or informats that begin with the letter **a**.

```
exclude a:;
```

In addition, the following EXCLUDE statement excludes all formats or informats that occur alphabetically between **apple** and **pear**, inclusive:

```
exclude apple-pear;
```

FMTLIB Output

If you use the EXCLUDE statement without either FMTLIB or CNTLOUT= in the PROC FORMAT statement, the procedure invokes FMTLIB.

INVALUE Statement

Creates an informat for reading and converting raw data values.

Featured in: Example 4 on page 468.

See also: The section on informats in *SAS Language Reference: Dictionary* for documentation on informats supplied by SAS.

INVALUE <$>*name* <(*informat-option(s)*)>
<*value-range-set(s)*>;

To do this	Use this option
Specify the default length of the informat	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the informat	MAX=
Specify a minimum length for the informat	MIN=
Store values or ranges in the order that you define them	NOTSORTED

To do this	Use this option
Left-justify all input strings before they are compared to ranges	JUST
Uppercase all input strings before they are compared to ranges	UPCASE

Required Arguments

name
names the informat you are creating. The name must be a SAS name up to seven characters long, not ending in a number. If you are creating a character informat, use a dollar sign ($) as the first character, with no more than six additional characters. A user-defined informat name cannot be the same as an informat that is supplied by SAS. Refer to the informat later by using the name followed by a period. However, do not put a period after the informat name in the INVALUE statement.

Tip: When SAS prints messages referring to a user-written informat, the name is prefixed by an at sign (@). When the informat is stored, the at sign is prefixed to the name you specify for the informat, which is why you are limited to only seven characters in the name. You need to use the at sign *only* when you are using the name in an EXCLUDE or SELECT statement; do not prefix the name with an at sign when you are associating the informat with a variable.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in "Informat and Format Options" on page 453:

DEFAULT=*length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following options:

JUST
left-justifies all input strings before they are compared to the ranges.

UPCASE
converts all raw data values to uppercase before they are compared to the possible ranges. If you use UPCASE, make sure the values or ranges you specify are in uppercase.

value-range-set(s)
specifies raw data and values that the raw data will become. The *value-range-set(s)* can be one or more of the following:

value-or-range-1 <..., value-or-range-n>=informatted-value | [*existing-informat*]
The informat converts the raw data to the values of *informatted-value* on the right side of the equal sign.

informatted-value
is the value you want the raw data in *value-or-range* to become. Use one of the following forms for *informatted-value*:

'character-string'
is a character string up to 200 characters long. Typically, *character-string* becomes the value of a character variable when you use the informat to convert

raw data. Use *character-string* for *informatted-value* only when you are creating a character informat. If you omit the single quotation marks around *character-string*, the INVALUE statement assumes the quotation marks are there.

For hex literals, you can use up to 199 typed characters, or up to 98 represented characters at 2 hex characters per represented character.

number
is a number that becomes the informatted value. Typically, *number* becomes the value of a numeric variable when you use the informat to convert raw data. Use *number* for *informatted-value* when you are creating a numeric informat. The maximum for *number* depends on the host operating environment.

ERROR
treats data values in the designated range as invalid data. SAS assigns a missing value to the variable, prints the data line in the SAS log, and issues a warning message.

SAME
prevents the informat from converting the raw data as any other value. For example, the following GROUP. informat converts values 01 through 20 and assigns the numbers 1 through 20 as the result. All other values are assigned a missing value.

```
invalue group 01-20= _same_
              other= .;
```

existing-informat
is an informat that is supplied by SAS or a user-defined informat. The informat you are creating uses the existing informat to convert the raw data that match *value-or-range* on the left side of the equals sign. If you use an existing informat, enclose the informat name in square brackets, for example, [date9.] or with parentheses and vertical bars, for example, (|date9.|). *Do not enclose the name of the existing informat in single quotation marks.*

value-or-range
See "Specifying Values or Ranges" on page 454.
Consider the following examples:

- The $GENDER. character informat converts the raw data values **F** and **M** to **1** and **2**:

```
invalue $gender 'F'='1'
                'M'='2';
```

  The dollar sign prefix indicates that the informat converts character data.

- When you are creating numeric informats, you can specify character strings or numbers for *value-or-range*. For example, the TRIAL. informat converts any character string that sorts between **A** and **M** to the number 1 and any character string that sorts between **N** and **Z** to the number 2. The informat treats the unquoted range 1–3000 as a numeric range, which includes all numeric values between 1 and 3000:

```
invalue trial 'A'-'M'=1
              'N'-'Z'=2
               1-3000=3;
```

  If you use a numeric informat to convert character strings that do not correspond to any values or ranges, you receive an error message.

- The CHECK. informat uses _ERROR_ and _SAME_ to convert values of 1 through 4 and 99. All other values are invalid:

```
invalue check 1-4=_same_
               99=.
            other=_error_;
```

PICTURE Statement

Creates a template for printing numbers.

Featured in: Example 1 on page 463 and Example 9 on page 479

See also: The section on formats in *SAS Language Reference: Dictionary* for documentation on formats supplied by SAS.

PICTURE *name* <(*format-option(s)*)>
  <*value-range-set-1* <(*picture-1-option(s)*)>
  <*...value-range-set-n* <(*picture-n-option(s)*)>>>;

To do this	Use this option
Control the attributes of the format	
Specify a fuzz factor for matching values to a range	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the format	MAX=
Specify a minimum length for the format	MIN=
Specify multiple pictures for a given value or range and for overlapping ranges	MULTILABEL
Store values or ranges in the order that you define them	NOTSORTED
Round the value to the nearest integer before formatting	ROUND
Control the attributes of each picture in the format	
Specify a character that completes the formatted value	FILL=
Specify a number to multiply the variable's value by before it is formatted	MULTIPLIER=
Specify that numbers are message characters rather than digit selectors	NOEDIT
Specify a character prefix for the formatted value	PREFIX=

Required Arguments

name
names the format you are creating. The name must be a SAS name up to eight characters long, not ending in a number. A user-defined format cannot be the name of a format supplied by SAS. Refer to the format later by using the name followed by

a period. However, do not put a period after the format name in the PICTURE statement.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in "Informat and Format Options" on page 453:

DEFAULT= *length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following arguments:

DATATYPE=DATE | TIME | DATETIME
specifies that you can use *directives* in the picture as a template to format date, time, or datetime values. See the definition of directives on page 445 for a list.

DECSEP='*character*'
specifies the separator character for the fractional part of a number.

Default: . (a decimal point)

DIG3SEP='*character*'
specifies the three-digit separator character for a number.

Default: , (a comma)

FILL='*character*'
specifies a character that completes the formatted value. If the number of significant digits is less than the length of the format, the format must complete, or fill, the formatted value:

- □ The format uses *character* to fill the formatted value if you specify zeros as digit selectors.
- □ The format uses zeros to fill the formatted value if you specify nonzero digit selectors. The FILL= option has no effect.

If the picture includes other characters, such as a comma, which appear to the left of the digit selector that maps to the last significant digit placed, the characters are replaced by the fill character or leading zeros.

Default: ' '(a blank)

Interaction: If you use the FILL= and PREFIX= options in the same picture, the format places the prefix and then the fill characters.

Featured in: Example 9 on page 479

MULTILABEL
allows the assignment of multiple labels or external values to internal values. The following PICTURE statements show the two uses of the MULTILABEL option. In each case, number formats are assigned as labels. The first PICTURE statement assigns multiple labels to a single internal value. Multiple labels may also be assigned to a single range of internal values. The second PICTURE statement assigns labels to overlapping ranges of internal values. The MULTILABEL option allows the assignment of multiple labels to the overlapped internal values.

```
picture abc (multilabel)
   1000='9,999'
   1000='9999';
```

```
picture overlap (multilabel)
   /* without decimals */
   0-999='999'
   1000-9999='9,999'

   /* with decimals */
   0-9='9.999'
   10-99='99.99'
   100-999='999.9';
```

Only multilabel-enabled procedures such as PROC MEANS, PROC SUMMARY, and PROC TABULATE can use multiple labels. All other procedures recognize only the primary label. The *primary label* for a given entry is the external value that is assigned to the first internal value or range of internal values that matches or contains the entry when all internal values are ordered sequentially. For example, in the first PICTURE statement, the primary label for 1000 is 1,000 because the format 9,999 is the first external value that is assigned to 1000. The secondary label for 1000 is 1000, based on the 9999 format.

In the second PICTURE statement, the primary label for 5 is 5.000 based on the 9.999 format that is assigned to the range 0–9 because 0–9 is sequentially the first range of internal values containing 5. The secondary label for 5 is 005 because the range 0–999 occurs in sequence after the range 0–9. Consider carefully when you assign multiple labels to an internal value. Unless you use the NOTSORTED option when you assign variables, the SAS System stores the variables in sorted order. This may produce unexpected results when variables with the MULTILABEL format are processed. For example, in the second PICTURE statement, the primary label for 15 is 015, and the secondary label for 15 is 15.00 because the range 0–999 occurs in sequence before the range 10–99. If you want the primary label for 15 to use the 99.99 format you may want to change the range 10–99 to 0–99 in the PICTURE statement. The range 0–99 occurs in sequence before the range 0–999 and will produce the desired result.

MULTIPLIER=*n*

specifies a number that the variable's value is to be multiplied by before it is formatted. For example, the following PICTURE statement creates the MILLION. format, which formats the variable value 1600000 as **`$1.6M`**:

```
picture million low-high='00.0M'
       (prefix='$' mult=.00001);
```

Alias: MULT=

Default: 10^n, where *n* is the number of digits after the first decimal point in the picture. For example, suppose your data contain a value 123.456 and you want to print it using a picture of '999.999'. The format multiplies 123.456 by 10^3 to obtain a value of 123456, which results in a formatted value of **`123.456`**.

Example: Example 1 on page 463

NOEDIT

specifies that numbers are message characters rather than digit selectors; that is, the format prints the numbers as they appear in the picture. For example, the following PICTURE statement creates the MILES. format, which formats any variable value greater than 1000 as **`>1000 miles`**:

```
picture miles 1-1000='0000'
          1000<-high='>1000 miles'(noedit);
```

PREFIX='*prefix*'
specifies a character prefix to place in front of the value's first significant digit. You must use zero digit selectors or the prefix will not be used.

The picture must be wide enough to contain both the value and the prefix. If the picture is not wide enough to contain both the value and the prefix, the format truncates or omits the prefix. Typical uses for PREFIX= are printing leading dollar signs and minus signs. For example, the PAY. format prints the variable value 25500 as **$25,500.00**:

```
picture pay low-high='000,009.99'
                     (prefix='$');
```

Default: no prefix

Interaction: If you use the FILL= and PREFIX= options in the same picture, the format places the prefix and then the fill characters.

Featured in: Example 1 on page 463 and Example 9 on page 479

ROUND
rounds the value to the nearest integer before formatting. Without the ROUND option, the format multiplies the variable value by the multiplier, truncates the decimal portion (if any), and prints the result according to the template you define. With the ROUND option, the format multiplies the variable value by the multiplier, rounds that result to the nearest integer, and then formats the value according to the template.

Tip: Note that the ROUND option rounds a value of .5 to the next highest integer.

value-range-set
specifies one or more variable values and a template for printing those values. The *value-range-set* is the following:

value-or-range-1 <..., value-or-range-n>='picture'

picture
specifies a template for formatting values of numeric variables. The picture is a sequence of characters in single quotation marks. The maximum length for a picture is 40 characters. Pictures are specified with three types of characters: digit selectors, message characters, and directives. You can have a maximum of 16 digit selectors in a picture.

Digit selectors are numeric characters (0 through 9) that define positions for numeric values. A picture format with nonzero digit selectors prints any leading zeros in variable values; picture digit selectors of 0 do not print leading zeros in variable values. If the picture format contains digit selectors, a digit selector must be the first character in the picture.

Note: This chapter uses 9's as nonzero digit selectors. △

Message characters are nonnumeric characters that print as specified in the picture. The following PICTURE statement contains both digit selectors (99) and message characters (**illegal day value**). Because the DAYS. format has nonzero digit selectors, values are printed with leading zeros. The special range OTHER prints the message characters for any values that do not fall into the specified range (1 through 31).

```
picture days 01-31='99'
       other='99-illegal day value';
```

For example, the values 02 and 67 print as

```
                                  02
                 67-illegal day value
```

Directives are special characters that you can use in the picture to format date, time, or datetime values.

Restriction: You can only use directives when you specify the DATATYPE= option in the PICTURE statement.

The permitted directives are

%a	Locale's abbreviated weekday name
%A	Locale's full weekday name
%b	Locale's abbreviated month name
%B	Locale's full month name
%d	Day of the month as a decimal number (1–31), with no leading zero
%H	Hour (24–hour clock) as a decimal number (0–23), with no leading zero
%I	Hour (12–hour clock) as a decimal number (1–12), with no leading zero
%j	Day of the year as a decimal number (1–366), with no leading zero
%m	Month as a decimal number (1–12), with no leading zero
%M	Minute as a decimal number (0–59), with no leading zero
%p	Locale's equivalent of either AM or PM
%S	Second as a decimal number (0–59), with no leading zero
%U	Week number of the year (Sunday as the first day of the week) as a decimal number (0,53), with no leading zero
%w	Weekday as a decimal number (1= Sunday, 7)
%y	Year without century as a decimal number (0–99), with no leading zero
%Y	Year with century as a decimal number
%%	%

Any directive that generates numbers can produce a leading zero, if desired, by adding a 0 before the directive. This applies to %d, %H, %I, %j, %m, %M, %S, %U, and %y. For example, if you specify %y in the picture, then 2001 would be formatted as '1', but if you specify %0y, then 2001 would be formatted as '01'.

value-or-range
See "Specifying Values or Ranges" on page 454.

Building a Picture Format: Step by Step

This section shows how to write a picture format for formatting numbers with leading zeros. In the SAMPLE data set, the default printing of the variable Amount has leading zeros on numbers between 1 and –1:

```
options nodate pageno=1 linesize=64
        pagesize=60;
data sample;
```

```
   input Amount;
   datalines;
-2.05
-.05
-.01
  0
 .09
 .54
 .55
6.6
14.63
;
```

```
             Default Printing of the Variable Amount           1

                         Obs    Amount

                          1      -2.05
                          2      -0.05
                          3      -0.01
                          4       0.00
                          5       0.09
                          6       0.54
                          7       0.55
                          8       6.60
                          9      14.63
```

The following PROC FORMAT step creates the NOZEROS. format, which eliminates leading zeros in the formatted values:

```
libname library 'SAS-data-library';


proc format library=library;
   picture nozeros
           low -  -1  =  '00.00'
              (prefix='-')
           -1 <-<  0  =     '99'
              (prefix='-.' mult=100)
            0  -< 1   =     '99'
              (prefix='.'  mult=100)
            1  - high =  '00.00';
run;
```

Table 19.1 on page 447 explains how one value from each range is formatted. Figure 19.1 on page 448 provides an illustration of each step. The circled numbers in the figure correspond to the step numbers in the table.

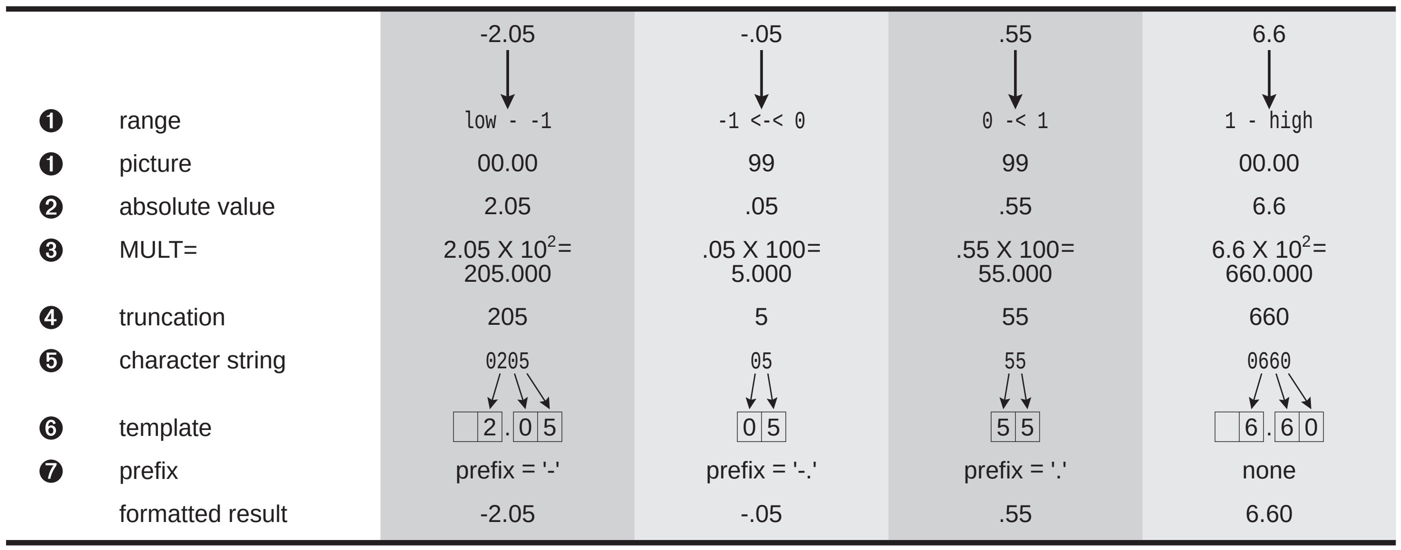
Table 19.1 Building a Picture Format

Step	Rule	In this example
1	Determine into which range the value falls and use that picture.	In the second range, the exclusion operator < appears on both sides of the hyphen and excludes –1 and 0 from the range.
2	Take the absolute value of the numeric value.	Because the absolute value is used, you need a separate range and picture for the negative numbers in order to prefix the minus sign.
3	Multiply the number by the MULT= value. If you do not specify the MULT= option, the PICTURE statement uses the default. The default is 10^n , where *n* is the number of digit selectors to the right of the decimal[1] in the picture. (Step 6 discusses digit selectors further.)	Specifying a MULT= value is necessary for numbers between 0 and 1 and numbers between 0 and –1 because no decimal appears in the pictures for those ranges. Because MULT= defaults to 1, truncation of the significant digits results without a MULT= value specified. (Truncation is explained in the next step.) For the two ranges that do not have MULT= values specified, the MULT= value defaults to 100 because the corresponding picture has two digit selectors to the right of the decimal. After the MULT= value is applied, all significant digits are moved to the left of the decimal.
4	Truncate the number after the decimal. If the ROUND option is in effect, the format rounds the number after the decimal to the next highest integer if the number after the decimal is greater than or equal to .5.	Because the example uses MULT= values that ensured that all of the significant digits were moved to the left of the decimal, no significant digits are lost. The zeros are truncated.
5	Turn the number into a character string. If the number is shorter than the picture, the length of the character string is equal to the number of digit selectors in the picture. Pad the character string with leading zeros. (The results are equivalent to using the Z*w.* format. Z*w.* is explained in the section on SAS formats in *SAS Language Reference: Dictionary*.	The numbers 205, 5, and 660 become the character strings `0205`, `05`, and `0660`, respectively. Because each picture is longer than the numbers, the format adds a leading zero to each value. The format does not add leading zeros to the number 55 because the corresponding picture only has two digit selectors.

Step	Rule	In this example
6	Apply the character string to the picture. The format only maps the rightmost *n* characters in the character string, where *n* is the number of digit selectors in the picture. Thus, it is important to make sure that the picture has enough digit selectors to accommodate the characters in the string. After the format takes the rightmost *n* characters, it then maps those characters to the picture from left to right. Choosing a zero or nonzero digit selector is important if the character string contains leading zeros. If one of the leading zeros in the character string maps to a nonzero digit selector, it and all subsequent leading zeros become part of the formatted value. If all of the leading zeros map to zero digit selectors, none of the leading zeros become part of the formatted value; the format replaces the leading zeros in the character string with blanks.[2]	The leading zero is dropped from each of the character strings **0205** and **0660** because the leading zero maps to a zero digit selector in the picture.
7	Prefix any characters that are specified in the PREFIX= option. You need the PREFIX= option because when a picture contains any digit selectors, the picture must begin with a digit selector. Thus, you cannot begin your picture with a decimal point, minus sign, or any other character that is not a digit selector.	The PREFIX= option reclaims the decimal point and the negative sign, as shown with the formatted values **-.05** and **.55**.

1 A decimal in a PREFIX= option is not part of the picture.
2 You can use the FILL= option to specify a character other than a blank to become part of the formatted value.

Figure 19.1 Formatting One Value in Each Range



		-2.05	-.05	.55	6.6
❶	range	`low - -1`	`-1 <-< 0`	`0 -< 1`	`1 - high`
❶	picture	00.00	99	99	00.00
❷	absolute value	2.05	.05	.55	6.6
❸	MULT=	$2.05 \times 10^2=$ 205.000	.05 X 100= 5.000	.55 X 100= 55.000	$6.6 \times 10^2=$ 660.000
❹	truncation	205	5	55	660
❺	character string	`0205`	`05`	`55`	`0660`
❻	template	□2.05	05	55	□6.60
❼	prefix	prefix = '-'	prefix = '-.'	prefix = '.'	none
	formatted result	-2.05	-.05	.55	6.60

The following PROC PRINT step associates the NOZEROS. format with the AMOUNT variable in SAMPLE:

```
proc print data=sample noobs;
   format amount nozeros.;
   title 'Formatting the Variable Amount';
   title2 'with the NOZEROS. Format';
run;
```

```
                  Formatting the Variable Amount                1
                     with the NOZEROS. Format

                              Amount

                              -2.05
                               -.05
                               -.01
                                .00
                                .09
                                .54
                                .55
                               6.60
                              14.63
```

CAUTION:

The picture must be wide enough for the prefix and the numbers. In this example, if the value –45.00 were formatted with NOZEROS. the result would be 45.00 because it falls into the first range, low - –1, and the picture for that range is not wide enough to accommodate the prefixed minus sign and the number. △

Specifying No Picture

This PICTURE statement creates a *picture-name* format that has no picture:

```
picture picture-name;
```

SELECT Statement

Selects entries from processing by the FMTLIB and CNTLOUT= options.

Restriction: Only one SELECT statement can appear in a PROC FORMAT step.

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

Featured in: Example 6 on page 472.

SELECT *entry(s)*;

Required Arguments

entry(s)
specifies one or more catalog entries for processing. Catalog entry names are the same as the name of the informat or format that they store. Because informats and formats can have the same name, and because character and numeric informats or formats can have the same name, you must use certain prefixes when specifying informats and formats in the SELECT statement. Follow these rules when specifying entries in the SELECT statement:

- Precede names of entries that contain character formats with a dollar sign ($).
- Precede names of entries that contain numeric informats with an at sign (@).
- Precede names of entries that contain character informats with an at sign and a dollar sign, for example, @$*entry-name*.

Shortcuts to Specifying Names

You can use the colon (:) and hyphen (-) wildcard characters to select entries. For example, the following SELECT statement selects all formats or informats that begin with the letter **a**.

```
select a:;
```

In addition, the following SELECT statement selects all formats or informats that occur alphabetically between **apple** and **pear**, inclusive:

```
select apple-pear;
```

FMTLIB Output

If you use the SELECT statement without either FMTLIB or CNTLOUT= in the PROC FORMAT statement, the procedure invokes FMTLIB.

VALUE Statement

Creates a format that specifies character strings to use to print variable values.

Featured in: Example 2 on page 465.

See also: The chapter on formats in *SAS Language Reference: Dictionary* for documentation on formats supplied by SAS.

VALUE <$>*name* <(*format-option(s)*)>
<*value-range-set(s)*>;

To do this	Use this option
Specify the default length of the format	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the format	MAX=
Specify a minimum length for the format	MIN=

To do this	Use this option
Specify multiple values for a given range, or for overlapping ranges	MULTILABEL
Store values or ranges in the order that you define them.	NOTSORTED

Required Arguments

name
names the format you are creating. The name must be a SAS name up to eight characters long, not ending in a number. Character format names must have a dollar sign ($) as the first character and no more than seven additional characters. A user-defined format cannot be the name of a format supplied by SAS. Refer to the format later by using the name followed by a period. However, do not put a period after the format name in the VALUE statement.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in “Informat and Format Options” on page 453:

DEFAULT=*length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following options:

MULTILABEL
allows the assignment of multiple labels or external values to internal values. The following VALUE statements show the two uses of the MULTILABEL option. The first VALUE statement assigns multiple labels to a single internal value. Multiple labels may also be assigned to a single range of internal values. The second VALUE statement assigns labels to overlapping ranges of internal values. The MULTILABEL option allows the assignment of multiple labels to the overlapped internal values.

```
value one (multilabel)
   1='ONE'
   1='UNO'
   1='UN'

value agefmt (multilabel)
   15-29='below 30 years'
   30-50='between 30 and 50'
   51-high='over 50 years'
   15-19='15 to 19'
   20-25='20 to 25'
   25-39='25 to 39'
   40-55='40 to 55'
   56-high='56 and above';
```

Only multilabel-enabled procedures such as PROC MEANS, PROC SUMMARY, and PROC TABULATE can use multiple labels. All other procedures recognize only the primary label. The *primary label* for a given entry is the external value that is assigned to the first internal value or range of internal values that matches or contains the entry when all internal values are ordered sequentially. For example, in the first VALUE statement, the primary label for 1 is ONE because ONE is the first external value that is assigned to 1. The secondary labels for 1 are UNO and UN. In the second VALUE statement, the primary label for 33 is **`25 to 39`** because the range 25–39 is sequentially the first range of internal values that contains 33. The secondary label for 33 is **`between 30 and 50`** because the range 30–50 occurs in sequence after the range 25–39.

value-range-set(s)

specifies one or more variable values and a character string or an existing format. The *value-range-set(s)* can be one or more of the following:

value-or-range-1 <..., value-or-range-n>='formatted-value' | [*existing-format*]

The variable values on the left side of the equals sign print as the character string on the right side of the equals sign.

formatted-value

specifies a character string that becomes the printed value of the variable value that appears on the left side of the equals sign. Formatted values are always character strings, regardless of whether you are creating a character or numeric format.

Formatted values can be up to 200 characters. For hex literals, you can use up to 199 typed characters, or up to 98 represented characters at 2 hex characters per represented character. Some procedures, however, use only the first 8 or 16 characters of a formatted value.

If you omit the single quotation marks around *formatted-value*, the VALUE statement assumes them to be there.

If a formatted value contains a single quotation mark, write it as two separate single quotation marks:

```
value sect 1='Smith''s class'
           2='Leung''s class';
```

Tip: Formatting numeric variables does not preclude your using those variables in arithmetic operations. SAS uses stored values for arithmetic operations.

existing-format

specifies a format supplied by SAS or an existing user-defined format. The format you are creating uses the existing format to convert the raw data that match *value-or-range* on the left side of the equals sign.

If you use an existing format, enclose the format name in square brackets, for example, [date9.] or with parentheses and vertical bars, for example, (|date9.|). *Do not enclose the name of the existing format in single quotation marks.*

Using an existing format can be thought of as *nesting* formats. A nested level of one means that if you are creating the format A with the format B as a formatted value, the procedure only has to use one existing format to create A.

Tip: Avoid nesting formats more than one level. The resource requirements increase dramatically with each additional level.

value-or-range

For details on how to specify *value-or-range*, see "Specifying Values or Ranges" on page 454.

Consider the following examples:

□ The $STATE. character format prints the postal code for selected states:

```
value $state 'Delaware'='DE'
             'Florida'='FL'
```

```
                  'Ohio'='OH';
```

The variable value **`Delaware`** prints as **`DE`**, the variable value **`Florida`** prints as **`FL`**, and the variable value **`Ohio`** prints as **`OH`**. Note that the $STATE. format begins with a dollar sign.

- The ANSWER. numeric format, writes the values 1 and 2 as **`yes`** and **`no`**:

```
value answer 1='yes'
             2='no';
```

Specifying No Ranges

This VALUE statement creates a *format-name* format that has no ranges:

```
value format-name;
```

Informat and Format Options

This section discusses options that are valid in the INVALUE, PICTURE, and VALUE statements. These options appear in parentheses after the informat or format name. They affect the entire informat or format that you are creating.

DEFAULT=*length*

specifies the default length of the informat or format. The value for DEFAULT= becomes the length of the informat or format if you do not give a specific length when you associate the informat or format with a variable.

The default length of a format is the length of the longest formatted value.

The default length of an informat depends on whether the informat is character or numeric. The default length of character informats is the length of the longest informatted value. The default of a numeric informat is 12 if you have numeric data to the left of the equals sign. If you have a quoted string to the left of the equals sign, the default length is the length of the longest string.

FUZZ=*fuzz-factor*

specifies a fuzz factor for matching values to a range. If a number does not match or fall in a range exactly but comes within *fuzz-factor*, the format considers it a match. For example, the following VALUE statement creates the LEVELS. format, which uses a fuzz factor of .2:

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 means that if a variable value falls within .2 of a value on either end of the range, the format uses the corresponding formatted value to print the variable value. So the LEVELS. format formats the value 2.1 as **`B`**.

If a variable value matches one value or range without the fuzz factor, and also matches another value or range with the fuzz factor, the format assigns the variable value to the value or range that it matched without the fuzz factor.

Default: 1E–12 for numeric formats and 0 for character formats.

Tip: Specify FUZZ=0 to save storage space when you use the VALUE statement to create numeric formats.

Tip: A value that is excluded from a range using the < operator does not receive the formatted value, even if it falls into the range when you use the fuzz factor.

MAX=*length*
specifies a maximum length for the informat or format. When you associate the format with a variable, you cannot specify a width greater than the MAX= value.

Default: 40

Range: 1–40

MIN=*length*
specifies a minimum length for the informat or format.

Default: 1

Range: 1–40

NOTSORTED
stores values or ranges for informats or formats in the order that you define them. If you do not specify NOTSORTED, values or ranges are stored in sorted order by default. Use NOTSORTED if

- you know the likelihood of certain ranges occurring, and you want your informat or format to search those ranges first to save processing time.
- you want to preserve the order that you define ranges when you print a description of the informat or format using the FMTLIB option.
- you want to preserve the order that you define ranges when you use the ORDER=DATA option and the PRELOADFMT option to analyze class variables in PROC MEANS, PROC SUMMARY, or PROC TABULATE.

Tip: SAS automatically sets the NOTSORTED option when you use the CPORT and the CIMPORT procedures to transport informats or formats between host platforms with different standard collating sequences. This can occur when you transport informats or formats between ASCII and EBCDIC host platforms.

Specifying Values or Ranges

As the syntax of the INVALUE, PICTURE, and VALUE statements indicates, you must specify values as *value-range-sets*. On the left side of the equals sign you specify the values that you want to convert to other values. On the right side of the equals sign, you specify the values that you want the values on the left side to become. This section discusses the different forms you can use for *value-or-range*, which represents the values on the left side of the equals sign. For details on how to specify values for the right side of the equals sign, see "Required Arguments" for the appropriate statement.

The INVALUE, PICTURE, and VALUE statements accept numeric values on the left side of the equals sign. INVALUE and VALUE also accept character strings on the left side of the equals sign.

As the syntax shows, you can have multiple occurrences of *value-or-range* in each *value-range-set*, with commas separating the occurrences. Each occurrence of *value-or-range* is either one of the following:

value
a single value, for example, 12 or`'CA'`. Enclose character values in single quotation marks; if you omit the quotes around *value*, PROC FORMAT assumes the quotes to be there.

You can use the keyword OTHER as a single value. OTHER matches all values that do not match any other value or range.

range
a list of values, for example, 12–68 or `'A'-'Z'`. For ranges with character strings, be sure to enclose each string in single quotation marks. For example, if you want

a range that includes character strings from A to Z, specify the range as **'A'-'Z'**, with single quotes around the **A** and around the **Z**.

If you specify **'A-Z'**, the procedure interprets it as a three-character string with **A** as the first character, a hyphen (-) as the second character, and a **Z** as the third character.

If you omit the quotes, the procedure assumes quotes around each string. For example, if you specify the range **abc-zzz**, the procedure interprets it as **'abc'-'zzz'**.

You can use LOW or HIGH as one value in a range, and you can use the range LOW-HIGH to encompass all values. For example, these are valid ranges:

```
low-'ZZ'
35-high
low-high
```

You can use the less than (<) symbol to exclude values from ranges. If you are excluding the first value in a range, put the < after the value. If you are excluding the last value in a range, put the < before the value. For example, the following range does not include 0:

```
0<-100
```

Likewise, the following range does not include 100:

```
0-<100
```

The following ranges show how to avoid overlapping ranges using noninclusive notation:

```
'AA'-<'AJ'=1 'AJ'-'AZ'=2
```

AJ is part of the second range, not the first.

If you overlap values in ranges, PROC FORMAT assigns the value to the first range. For example, in the following ranges, the value **AJ** is part of the first range:

```
'AA'-'AJ'=1 'AJ'-'AZ'=2
```

Each *value-or-range* can be up to 200 characters. If *value-or-range* has more than 200 characters, the procedure truncates the value after it processes the first 200 characters.

Note: You do not have to account for every value on the left side of the equals sign. Those values are converted using the default informat or format. For example, the following VALUE statement creates the TEMP. format, which prints all occurrences of 98.6 as **NORMAL**:

```
value temp 98.6='NORMAL';
```

If the value were 96.9, the printed result would be **96.9**. △

Concepts

Associating Informats and Formats with Variables

Table 19.2 on page 456 summarizes the different methods for associating informats and formats with variables.

Table 19.2 Associating Informats and Formats with Variables

Step	Informats	Formats
In a DATA step	Use the ATTRIB or INFORMAT statement to permanently associate an informat with a variable. Use the INPUT function or INPUT statement to associate the informat with the variable only for the duration of the DATA step.	Use the ATTRIB or FORMAT statement to permanently associate a format with a variable. Use the PUT function or PUT statement to associate the format with the variable only for the duration of the DATA step.
In a PROC step	The ATTRIB and INFORMAT statements are valid in base SAS procedures. However, in base SAS software, typically you do not assign informats in PROC steps because the data have already been read into SAS variables.	Use the ATTRIB statement or the FORMAT statement to associate formats with variables. If you use either statement in a procedure that produces an output data set, the format is permanently associated with the variable in the output data set. If you use either statement in a procedure that does not produce an output data set, the statement associates the format with the variable only for the duration of the PROC step.

Tips

□ Do not confuse the FORMAT statement with the FORMAT procedure. The FORMAT and INFORMAT statements associate an existing format or informat (either standard SAS or user-defined) with one or more variables. PROC FORMAT creates user-defined formats or informats.

□ It is often useful to assign informats in the FSEDIT procedure in SAS/FSP software and in the BUILD procedure in SAS/AF software.

See Also

□ For complete documentation on the ATTRIB, INFORMAT, and FORMAT statements, see the section on statements in *SAS Language Reference: Dictionary*.

□ For complete documentation on the INPUT and PUT functions, see the section on functions in *SAS Language Reference: Dictionary*.

□ See “Formatted Values” on page 59 for more information and examples of using formats in base SAS procedures.

Storing Informats and Formats

PROC FORMAT stores user-written informats and formats as entries in SAS catalogs.* You use the LIBRARY= option in the PROC FORMAT statement to indicate the catalog. The name of the catalog entry is the name of the format or informat. The entry types are

□ FORMAT for numeric formats

* Catalogs are a type of SAS file and reside in a SAS data library. If you are unfamiliar with the types of SAS files or the SAS data library structure, see the section on SAS files in *SAS Language Reference: Dictionary*.

- □ FORMATC for character formats
- □ INFMT for numeric informats
- □ INFMTC for character informats.

Temporary Informats and Formats

Informats and formats are temporary when you do not specify the LIBRARY= option in the PROC FORMAT statement. If you omit the LIBRARY= option, PROC FORMAT stores the informats and formats in the temporary catalog WORK.FORMATS. You can retrieve temporary informats and formats only in the same SAS session or job in which they are created. To retrieve a temporary format or informat, simply include the name of the format or informat in the appropriate SAS statement. The SAS System automatically looks for the format or informat in the WORK.FORMATS catalog.

Permanent Informats and Formats

If you want to use a format or informat that is created in one SAS job or session in a subsequent job or session, you must permanently store the format or informat in a SAS catalog.

You can create permanent informats and formats by using the LIBRARY= option in the PROC FORMAT statement. See the discussion of the LIBRARY= option in "PROC FORMAT Statement" on page 435.

Accessing Permanent Informats and Formats

After you have permanently stored an informat or format, you can use it in later SAS sessions or jobs. If you associate permanent informats or formats with variables in a later SAS session or job, SAS must be able to access the informats and formats. Thus, you must use a LIBNAME statement to assign a libref to the library that stores the catalog that stores the informats or formats.

SAS always searches the WORK.FORMATS and the LIBRARY.FORMATS catalogs for any user-defined informats or formats that you associate with variables. If you want to specify a search order for catalogs, or if you want to specify additional catalogs for SAS to search, use the SAS system option FMTSEARCH=. For further information on FMTSEARCH=, see the section on SAS system options in *SAS Language Reference: Dictionary*. For an example that uses the LIBRARY= and FMTSEARCH= options together, see Example 8 on page 476.

CAUTION:

Serious complications arise if you do not save informats and formats that are permanently associated with variables in a data set. △

If you reference an informat or format that the SAS System cannot find, you receive an error message and processing stops unless the SAS system option NOFMTERR is in effect. When NOFMTERR is in effect, the SAS System uses the *w.* or $*w.* default format to print values for variables with formats it cannot find. For example, to use NOFMTERR, use this OPTIONS statement:

```
options nofmterr;
```

Refer to the section on SAS system options in *SAS Language Reference: Dictionary* for more information on NOFMTERR.

Results

Output Control Data Set

The output control data set contains information that describes informats or formats. You can use output control data sets, or a set of observations from an output control data set, as an input control data set in a subsequent PROC FORMAT step. You create an output control data set with the CNTLOUT= option in the PROC FORMAT statement.

Output control data sets contain an observation for every value or range in each of the informats or formats in the LIBRARY= catalog. The data set consists of variables that give either global information about each format and informat created in the PROC FORMAT step or specific information about each range and value.

The variables in the output control data set are

DEFAULT
: numeric variable that indicates the default length for format or informat

END
: character variable that gives the range's ending value

EEXCL
: character variable that indicates whether the range's ending value is excluded. Values are

Y	the range's ending value is excluded
N	the range's ending value is not excluded

FILL
: numeric variable whose value is the value of the FILL= option

FMTNAME
: character variable whose value is the format or informat name

FUZZ
: numeric variable whose value is the value of the FUZZ= option

HLO
: character variable that contains range information about the format or informat in the form of eight different letters that can appear in any combination. Values are

F	standard SAS format or informat used for formatted value or informatted value
H	range's ending value is HIGH
I	numeric informat range (informat defined with unquoted numeric range)
L	range's starting value is LOW
N	format or informat has no ranges, including no OTHER= range
O	range is OTHER
R	ROUND option is in effect
S	NOTSORTED option is in effect

LABEL
: character variable whose value is the informatted or formatted value or the name of a standard SAS informat or format

LENGTH
: numeric variable whose value is the value of the LENGTH= option

MAX
: numeric variable whose value is the value of the MAX= option

MIN
: numeric variable whose value is the value of the MIN= option

MULT
: numeric variable whose value is the value of the MULT= option

NOEDIT
: character variable whose value indicates whether the NOEDIT option is in effect. Values are

`1`	NOEDIT option is in effect
`0`	NOEDIT option is not in effect

PREFIX
: character variable whose value is the value of the PREFIX= option

SEXCL
: character variable that indicates whether the range's starting value is excluded. Values are

`Y`	the range's starting value is excluded
`N`	the range's starting value is not excluded

START
: character variable that gives the range's starting value

TYPE
: character variable that indicates the type of format. Possible values are

`C`	character format
`I`	numeric informat
`J`	character informat
`N`	numeric format (excluding pictures)
`P`	picture format

Output 19.1 on page 459 shows an output control data set that contains information on all the informats and formats created in "Examples" on page 463.

Output 19.1 Output Control Data Set for PROC FORMAT Examples

```
                                             An Output Control Data Set                                               1

                                                                                                                D L
    F                                                                    D                                    D A A
    M                                                                    E  L        P           N          D I T N
    T              S                                   L                 F  E        R           O   S E    E G A G
    N              T                                   A                 A  N    F   E       M F E T E E    C 3 T U
  O A              A                E                  B           M  M  U  G    U   F       U I D Y X X H  S S Y A
  b M              R                N                  E           I  A  L  T    Z   I       L L I P C C L  E E P G
  s E              T                D                  L           N  X  T  H    Z   X       T L T E L L O  P P E E

  1 BENEFIT LOW                          7304 WORDDATE20.          1 40 20 20  1E-12      0.00   0 N N N LF
  2 BENEFIT             7305 HIGH               ** Not Eligible ** 1 40 20 20  1E-12      0.00   0 N N N H
  3 NOZEROS LOW                            -1 00.00                1 40  5  5  1E-12 -  100.00   0 P N N L  . ,
  4 NOZEROS               -1                0 99                   1 40  5  5  1E-12 -. 100.00   0 P Y Y    . ,
  5 NOZEROS                0                1 99                   1 40  5  5  1E-12 .  100.00   0 P N Y    . ,
  6 NOZEROS                1 HIGH             00.00                1 40  5  5  1E-12    100.00   0 P N N H  . ,
  7 PTSFRMT                0                3 0%                   1 40  3  3  1E-12      0.00   0 N N N
  8 PTSFRMT                4                6 3%                   1 40  3  3  1E-12      0.00   0 N N N
  9 PTSFRMT                7                8 6%                   1 40  3  3  1E-12      0.00   0 N N N
 10 PTSFRMT                9               10 8%                   1 40  3  3  1E-12      0.00   0 N N N
 11 PTSFRMT               11 HIGH             10%                  1 40  3  3  1E-12      0.00   0 N N N H
 12 USCURR  LOW              HIGH             000,000              1 40  7  7  1E-12 $    1.61   0 P N N LH . ,
 13 CITY    BR1              BR1              Birmingham UK        1 40 14 14      0      0.00   0 C N N
 14 CITY    BR2              BR2              Plymouth UK          1 40 14 14      0      0.00   0 C N N
 15 CITY    BR3              BR3              York UK              1 40 14 14      0      0.00   0 C N N
 16 CITY    US1              US1              Denver USA           1 40 14 14      0      0.00   0 C N N
 17 CITY    US2              US2              Miami USA            1 40 14 14      0      0.00   0 C N N
 18 CITY    **OTHER**        **OTHER**        INCORRECT CODE       1 40 14 14      0      0.00   0 C N N O
 19 EVAL    C                C                                   1 1 40  1  1      0      0.00   0 I N N
 20 EVAL    E                E                                   2 1 40  1  1      0      0.00   0 I N N
 21 EVAL    N                N                                   0 1 40  1  1      0      0.00   0 I N N
 22 EVAL    O                O                                   4 1 40  1  1      0      0.00   0 I N N
 23 EVAL    S                S                                   3 1 40  1  1      0      0.00   0 I N N
```

Input Control Data Set

You specify an input control data set with the CNTLIN= option in the PROC FORMAT statement. The FORMAT procedure uses the data in the input control data set to construct informats and formats. Thus, you can create informats and formats without writing INVALUE, PICTURE, or VALUE statements.

The input control data set must have these characteristics:

- □ For both numeric and character formats, the data set must contain the variables FMTNAME, START, and LABEL, which are described in "Output Control Data Set" on page 458. The remaining variables are not required.
- □ If you are creating a character format, a character informat, or a PICTURE statement format, you must specify a TYPE variable with the value that indicates the type of informat or format you are creating.
- □ If range values are to be noninclusive, the variables SEXCL and EEXCL must each have a value of **Y**. Inclusion is the default.

You can create more than one format from an input control data set if the observations for each format are grouped together.

You can use a VALUE, INVALUE, or PICTURE statement in the same PROC FORMAT step with the CNTLIN= option. If the VALUE, INVALUE, or PICTURE statement is creating the same informat or format that the CNTLIN= option is creating, the VALUE, INVALUE, or PICTURE statement creates the informat or format and the CNTLIN= data set is not used. You can, however, create an informat or format with VALUE, INVALUE, or PICTURE and create a different informat or format with CNTLIN= in the same PROC FORMAT step.

For an example featuring an input control data set, see Example 5 on page 470.

Procedure Output

The FORMAT procedure prints output only when you specify the FMTLIB option or the PAGE option in the PROC FORMAT statement. The printed output is a table for each format or informat entry in the catalog specified in the LIBRARY= option. The output also contains global information and the specifics of each value or range defined for the format or informat.

The FMTLIB output shown in Output 19.2 on page 461 contains a description of the NOZEROS. format, which is created in "Building a Picture Format: Step by Step" on page 445, and the EVAL. informat, which is created in Example 4 on page 468.

Output 19.2 Output from PROC FORMAT with the FMTLIB Option

```
                FMTLIB Output for the NOZEROS. Format and the              1
                                EVAL. Informat

   ----------------------------------------------------------------------------
   |       FORMAT NAME: NOZEROS  LENGTH:    5   NUMBER OF VALUES:    4        |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   5  FUZZ: STD        |
   |--------------------------------------------------------------------------|
   |START           |END             |LABEL  (VER. 7.00    29MAY98:10:00:24) |
   |----------------+----------------+----------------------------------------|
   |LOW             |              -1|00.00                P-  F  M100        |
   |             -1<               0<99                    P-. F  M100        |
   |               0|               1<99                   P.  F  M100        |
   |               1|HIGH            |00.00                P   F  M100        |
   ----------------------------------------------------------------------------


   ----------------------------------------------------------------------------
   |     INFORMAT NAME: @EVAL    LENGTH:    1   NUMBER OF VALUES:    5        |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   1  FUZZ:        0   |
   |--------------------------------------------------------------------------|
   |START           |END             |INVALUE(VER. 7.00    29MAY98:10:00:25) |
   |----------------+----------------+----------------------------------------|
   |C               |C               |                                       1|
   |E               |E               |                                       2|
   |N               |N               |                                       0|
   |O               |O               |                                       4|
   |S               |S               |                                       3|
   ----------------------------------------------------------------------------
```

The fields are described below in the order they appear in the output, from left to right:

INFORMAT NAME
FORMAT NAME
the name of the informat or format. Informat names begin with an at-sign (@).

LENGTH

the length of the informat or format. PROC FORMAT determines the length in the following ways:

- □ For character informats, the value for LENGTH is the length of the longest raw data value on the left side of the equals sign.
- □ For numeric informats
  - □ LENGTH is 12 if all values on the left side of the equals sign are numeric.
  - □ LENGTH is the same as the longest raw data value on the left side of the equal sign.
- □ For formats, the value for LENGTH is the length of the longest value on the right side of the equal sign.

In the output for @EVAL., the length is 1 because 1 is the length of the longest raw data value on the left side of the equals sign.

In the output for NOZEROS., the LENGTH is 5 because the longest picture is 5 characters.

NUMBER OF VALUES

the number of values or ranges associated with the informat or format. NOZEROS. has 4 ranges, EVAL. has 5.

MIN LENGTH

the minimum length of the informat or format. The value for MIN LENGTH is 1 unless you specify a different minimum length with the MIN= option.

MAX LENGTH

the maximum length of the informat or format. The value for MAX LENGTH is 40 unless you specify a different maximum length with the MAX= option.

DEFAULT LENGTH

the length of the longest value in the INVALUE or LABEL field, or the value of the DEFAULT= option.

FUZZ

the fuzz factor. For informats, FUZZ always is 0. For formats, the value for this field is STD if you do not use the FUZZ= option. STD signifies the default fuzz value.

START

the beginning value of a range. FMTLIB prints only the first 16 characters of a value in the START and END columns.

END

the ending value of a range. The exclusion sign (<) appears after the values in START and END, if the value is excluded from the range.

INVALUE
LABEL

INVALUE appears only for informats and contains the informatted values. LABEL appears only for formats and contains either the formatted value or picture. The release of the SAS System and the date on which the format or informat was created are in parentheses after INVALUE or LABEL.

For picture formats, such as NOZEROS., the LABEL section contains the PREFIX=, FILL=, and MULT= values. To note these values, FMTLIB prints the letters **P**, **F**, and **M** to represent each option, followed by the value. For example, in the LABEL section, **P-.** indicates that the prefix value is a dash followed by a period.

FMTLIB prints only 40 characters in the LABEL column.

Examples

Several examples in this section use the PROCLIB.STAFF data set. In addition, many of the informats and formats that are created in these examples are stored in LIBRARY.FORMATS. The output data set shown in "Output Control Data Set" on page 458 contains a description of these informats and the formats.

```
libname proclib 'SAS-data-library';
```

PROCLIB.STAFF contains information about a small subset of employees who work for a corporation that has sites in the U.S. and Britain. The data contain the name, identification number, salary (in British pounds), location, and date of hire for each employee. The FORMAT statement in the DATA step assigns the standard SAS format DATE7. to the variable HireDate.

```
data proclib.staff;
   input Name & $16. IdNumber $ Salary
         Site $ HireDate date7.;
   format hiredate date7.;
   datalines;
Capalleti, Jimmy  2355 21163 BR1 30JAN79
Chen, Len         5889 20976 BR1 18JUN76
Davis, Brad       3878 19571 BR2 20MAR84
Leung, Brenda     4409 34321 BR2 18SEP74
Martinez, Maria   3985 49056 US2 10JAN93
Orfali, Philip    0740 50092 US2 16FEB83
Patel, Mary       2398 35182 BR3 02FEB90
Smith, Robert     5162 40100 BR5 15APR86
Sorrell, Joseph   4421 38760 US1 19JUN93
Zook, Carla       7385 22988 BR3 18DEC91
;
```

Example 1: Creating a Picture Format

Procedure features:

PROC FORMAT statement options:
- LIBRARY=

PICTURE statement options:
- MULT=
- PREFIX=

LIBRARY libref

LOW and HIGH keywords

Data set:

PROCLIB.STAFF on page 463.

This example uses a PICTURE statement to create a format that prints the values for SALARY in the data set PROCLIB.STAFF in U.S. dollars.

Program

The LIBNAME statement specifies a SAS data library to store permanent informats and formats. The libref LIBRARY is useful because SAS automatically searches for informats and formats in any library referenced with the LIBRARY libref.

```
libname proclib 'SAS-data-library-1 ';
libname library 'SAS-data-library-2';


options nodate pageno=1 linesize=80 pagesize=40;
```

LIBRARY=LIBRARY stores the USCURR. format in the catalog LIBRARY.FORMATS.

```
   proc format library=library;
```

The format USCURR. uses the MULT= value of 1.61 and a prefix of $. Any number you format with the USCURR. format will be multiplied by 1.61 and then applied to the picture. The picture contains six digit selectors: five for the salary and one for the dollar sign prefix. LOW-HIGH ensures that all values are formatted.

```
      picture uscurr low-high='000,000' (mult=1.61 prefix='$');
   run;
```

PROC PRINT prints PROCLIB.STAFF. The FORMAT statement associates the USCURR. format with Salary for the duration of this procedure step only. The LABEL statement associates the label with Salary for the duration of this step only.

```
proc print data=proclib.staff noobs label;
   label salary='Salary in U.S. Dollars';
   format salary uscurr.;
   title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

Output

```
               PROCLIB.STAFF with a Format for the Variable Salary              1

                                         Salary in
                            Id             U.S.                         Hire
            Name          Number         Dollars      Site              Date

            Capalleti, Jimmy    2355      $34,072     BR1     30JAN79
            Chen, Len           5889      $33,771     BR1     18JUN76
            Davis, Brad         3878      $31,509     BR2     20MAR84
            Leung, Brenda       4409      $55,256     BR2     18SEP74
            Martinez, Maria     3985      $78,980     US2     10JAN93
            Orfali, Philip      0740      $80,648     US2     16FEB83
            Patel, Mary         2398      $56,643     BR3     02FEB90
            Smith, Robert       5162      $64,561     BR5     15APR86
            Sorrell, Joseph     4421      $62,403     US1     19JUN93
            Zook, Carla         7385      $37,010     BR3     18DEC91
```

Example 2: Creating a Format for Character Values

Procedure features:
VALUE statement
OTHER keyword

Data set:
PROCLIB.STAFF on page 463.

Format: USCURR on page 464.

This example uses a VALUE statement to create a character format that prints a value of a character variable as a different character string.

Program

The LIBNAME statement specifies a SAS data library to store permanent informats and formats. The libref LIBRARY is useful because SAS automatically searches for informats and formats in any library referenced with the LIBRARY libref.

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

LIBRARY=LIBRARY stores the $CITY. format in the catalog LIBRARY.FORMATS.

```
   proc format library=library;
```

The $CITY. format converts each of the codes BR1, BR2, and so on, to the name of the corresponding city. The keyword OTHER formats values in the data set that do not match any values on the left side of the equals sign as **INCORRECT CODE**.

```
      value  $city 'BR1'='Birmingham UK'
                   'BR2'='Plymouth UK'
                   'BR3'='York UK'
                   'US1'='Denver USA'
                   'US2'='Miami USA'
                   other='INCORRECT CODE';
   run;
```

PROC PRINT prints PROCLIB.STAFF. The LABEL statement associates the label with Salary. The FORMAT statement associates the USCURR. format (created in Example 1 on page 463) with Salary and the $CITY. format with Site. The labels and formats are not permanently assigned.

```
proc print data=proclib.staff noobs label;
   label salary='Salary in U.S. Dollars';
   format salary uscurr. site $city.;
   title 'PROCLIB.STAFF with a Format for the Variables';
   title2 'Salary and Site';
run;
```

Output

```
                  PROCLIB.STAFF with a Format for the Variables                 1
                                 Salary and Site

                                     Salary in
                         Id            U.S.                                Hire
      Name              Number       Dollars     Site                      Date

      Capalleti, Jimmy   2355        $34,072     Birmingham UK      30JAN79
      Chen, Len          5889        $33,771     Birmingham UK      18JUN76
      Davis, Brad        3878        $31,509     Plymouth UK        20MAR84
      Leung, Brenda      4409        $55,256     Plymouth UK        18SEP74
      Martinez, Maria    3985        $78,980     Miami USA          10JAN93
      Orfali, Philip     0740        $80,648     Miami USA          16FEB83
      Patel, Mary        2398        $56,643     York UK            02FEB90
      Smith, Robert      5162        $64,561     INCORRECT CODE     15APR86
      Sorrell, Joseph    4421        $62,403     Denver USA         19JUN93
      Zook, Carla        7385        $37,010     York UK            18DEC91
```

Example 3: Writing a Format for Dates Using a Standard SAS Format

Procedure features:

VALUE statement:

HIGH keyword

Data set:

PROCLIB.STAFF on page 463.

Formats:
USCURR. on page 464 and $CITY. on page 466.

This example uses an existing format that is supplied by SAS as a formatted value. Tasks include

- creating a numeric format
- nesting formats
- writing a format using a standard SAS format
- formatting dates.

Program

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';


options nodate pageno=1 linesize=80 pagesize=40;
```

LIBRARY=LIBRARY stores the BENEFIT. format in the catalog LIBRARY.FORMATS.

```
   proc format library=library;
```

The BENEFIT. format differentiates between the employees who were hired on or before 31DEC79 and those who were hired after that date. Employees hired on or before 31DEC79 are eligible for a benefits package, those hired after are not. The first range in BENEFIT. uses the LOW keyword and the SAS date constant '31DEC79'D to include all variable values up to and including December 31, 1979. For values that fall into this range, the format uses the WORDDATE*w*. format. WORDDATE*w*. is a format supplied by SAS.*

```
      value benefit  low-'31DEC79'd=[worddate20.]
```

The second range in BENEFIT. uses the SAS date constant '01JAN80'D and the keyword HIGH to include all variable values from January 1, 1980, to the most recent date. Values that fall into this range receive `** Not Eligible **` as a formatted value.

```
                     '01JAN80'd-high='  ** Not Eligible **';
   run;
```

PROC PRINT prints PROCLIB.STAFF. The FORMAT statement associates the USCURR. format (created in Example 1 on page 463) with Salary, the $CITY. format (created in Example 2 on page 465) with Site, and the BENEFIT. format with HireDate.

* For more information on SAS date constants, see the section on dates, times, and intervals in *SAS Language Reference: Concepts*. For complete documentation on WORDDATE*w*., see the section on formats in *SAS Language Reference: Dictionary*.

```
proc print data=proclib.staff noobs label;
   label salary='Salary in U.S. Dollars';
   format salary uscurr. site $city. hiredate benefit.;
   title 'PROCLIB.STAFF with a Format for the Variables';
   title2 'Salary, Site, and HireDate';
run;
```

Output

```
                  PROCLIB.STAFF with a Format for the Variables                1
                            Salary, Site, and HireDate

                                Salary in
                     Id           U.S.
Name               Number       Dollars     Site                   HireDate

Capalleti, Jimmy     2355       $34,072     Birmingham UK       January 30, 1979
Chen, Len            5889       $33,771     Birmingham UK          June 18, 1976
Davis, Brad          3878       $31,509     Plymouth UK       ** Not Eligible **
Leung, Brenda        4409       $55,256     Plymouth UK       September 18, 1974
Martinez, Maria      3985       $78,980     Miami USA         ** Not Eligible **
Orfali, Philip       0740       $80,648     Miami USA         ** Not Eligible **
Patel, Mary          2398       $56,643     York UK           ** Not Eligible **
Smith, Robert        5162       $64,561     INCORRECT CODE    ** Not Eligible **
Sorrell, Joseph      4421       $62,403     Denver USA        ** Not Eligible **
Zook, Carla          7385       $37,010     York UK           ** Not Eligible **
```

Example 4: Converting Raw Character Data to Numeric Values

Procedure feature:
INVALUE statement

This example uses an INVALUE statement to create a numeric informat that converts numeric and character raw data to numeric data.

Program

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';
```

```
options nodate pageno=1 linesize=64 pagesize=40;
```

LIBRARY=LIBRARY stores the EVAL. informat in the catalog LIBRARY.FORMATS.

```
proc format library=library;
```

The INVALUE statement creates the numeric informat EVAL., which converts the letters to their numeric equivalents. The letters **O** (Outstanding), **S** (Superior), **E** (Excellent), **C** (Commendable), and **N** (None) correspond to the numbers 4, 3, 2, 1, and 0, respectively.

```
   invalue eval 'O'=4
                'S'=3
                'E'=2
                'C'=1
                'N'=0;
run;
```

The PROCLIB.POINTS data set includes a unique four-character identification number (EmployeeId) and bonus evaluations for each employee for each quarter of the year (Q1–Q4). In the raw data, performance ratings are noted with numbers and letters. The EVAL. informat converts the value **O** to 4, the value **S** to 3, and so on. The raw data values 0 through 4 are read as themselves because they are not referenced in the definition of the informat. Converting the letter values to numbers makes it possible to calculate the total number of bonus points for each employee for the year. TotalPoints is the total number of bonus points.

```
data proclib.points;
   input EmployeeId $ (Q1-Q4) (eval.,+1);
   TotalPoints=sum(of q1-q4);
   datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

PROC PRINT prints PROCLIB.POINTS.

```
proc print data=proclib.points noobs;
   title 'The PROCLIB.POINTS Data Set';
run;
```

Output

```
                   The PROCLIB.POINTS Data Set                1

            Employee                                Total
               Id       Q1     Q2     Q3     Q4     Points

              2355       3      4      4      3       14
              5889       2      2      2      2        8
              3878       1      2      2      2        7
              4409       0      1      1      1        3
              3985       3      3      3      2       11
              0740       3      2      2      3       10
              2398       2      2      1      1        6
              5162       1      1      1      2        5
              4421       3      2      2      2        9
              7385       1      1      1      0        3
```

Example 5: Creating a Format from a Data Set

Procedure features:

PROC FORMAT statement option:

CNTLIN=

Input Control Data Set

Data sets:

PROCLIB.POINTS on page 469.

This example shows how to create a format from a SAS data set.

Tasks include

- □ creating a format from an input control data set
- □ creating an input control data set from an existing SAS data set.

Program

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';


options nodate pageno=1 linesize=80 pagesize=60;
```

Each observation in the PROCLIB.SCALE data set contains a range for the format. AMOUNT contains a percentage that will be used as the formatted value in the format.

```
data proclib.scale;
   input begin $ 1-2 end $ 5-8 amount $ 10-12;
   datalines;
0   3    0%
4   6    3%
```

```
7   8    6%
9   10   8%
11  16   10%
;
```

The DATA step creates the CTRL data set from PROCLIB.SCALE. RENAME= renames BEGIN and AMOUNT as START and LABEL, respectively. The END= option specifies a variable whose value is an end-of-file flag.

```
data ctrl(rename=(begin=start amount=label));
   set proclib.scale end=last;
```

The RETAIN statement creates the variables FMTNAME and TYPE with fixed values. The RETAIN statement is more efficient than an assignment statement in this case. RETAIN retains the value of FMTNAME and TYPE in the program data vector and eliminates the need for the value to be written on every iteration of the DATA step. FMTNAME specifies the name of the format that the input control data set creates. The TYPE variable specifies that the input control data set will create a numeric format.

```
   retain fmtname 'ptsfrmt' type 'n';
```

The IF and ELSE statements create the HLO variable. The IF statement executes only if the DATA step is writing the last observation. HLO receives a value of **h** for the last observation in the data set. The value **h** indicates that the ending value of the range is HIGH. HLO has missing values for all other observations.

```
   if last then hlo='h';
   else hlo=' ';
run;
```

PROC PRINT prints the control data set, CTRL.

```
   proc print data=ctrl noobs;
      title 'The CTRL Data Set';
   run;
```

LIBRARY=LIBRARY stores the format that is created in the PROC FORMAT step in the catalog LIBRARY.FORMATS. CNTLIN= creates the format PTSFRMT. from the input control data set, CTRL.

```
   proc format library=library cntlin=ctrl;
   run;
```

PROC REPORT prints PROCLIB.POINTS and associates the PTSFRMT. format with the TotalPoints variable. The column that contains the formatted values of TotalPoints is using the alias Pctage. Using an alias enables you to print a variable twice, once with a format and once with the default format. See Chapter 32, “The REPORT Procedure,” on page 859 for more information on PROC REPORT.

```
proc report data=proclib.points nowd colwidth=12;
   column employeeid totalpoints totalpoints=Pctage;
   define pctage / format=ptsfrmt10. 'Percentage';
   title 'The Percentage of Salary For Calculating Bonus';
run;
```

Output

PROC PRINT output

```
                              The CTRL Data Set                               1

               start    end    label    fmtname    type    hlo

                0       3       0%      ptsfrmt     n
                4       6       3%      ptsfrmt     n
                7       8       6%      ptsfrmt     n
                9       10      8%      ptsfrmt     n
                11      16      10%     ptsfrmt     n       h
```

PROC REPORT output

```
                The Percentage of Salary For Calculating Bonus                1

                      Employee
                      Id          TotalPoints  Percentage
                      2355                 14  10%
                      5889                  8  6%
                      3878                  7  6%
                      4409                  3  0%
                      3985                 11  10%
                      0740                 10  8%
                      2398                  6  3%
                      5162                  5  3%
                      4421                  9  8%
                      7385                  3  0%
```

Example 6: Printing the Description of Informats and Formats

Procedure features:

PROC FORMAT statement option:

FMTLIB

SELECT statement

Format:

NOZEROS on page 446.

Informat:

EVAL. on page 469.

This example illustrates how to print a description of an informat and a format. The description shows the values that are input and output.

Program

```
libname library 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=60;
```

FMTLIB prints a description of EVAL. and NOZEROS. LIBRARY=LIBRARY points to the LIBRARY.FORMATS catalog, which contains EVAL. and NOZEROS.

```
proc format library=library fmtlib;
```

The SELECT statement selects the EVAL. informat and the NOZEROS. format, which are created in previous examples. The at-sign (@) in front of EVAL indicates that EVAL. is an informat.

```
   select @eval nozeros;
   title 'FMTLIB Output for the NOZEROS. Format and the';
   title2 'EVAL. Informat';
run;
```

Output

The output is described in “Procedure Output” on page 461.

```
                  FMTLIB Output for the NOZEROS. Format and the                 1
                                 EVAL. Informat

   ----------------------------------------------------------------------------
   |       FORMAT NAME: NOZEROS  LENGTH:    5   NUMBER OF VALUES:    4         |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   5  FUZZ: STD         |
   |--------------------------------------------------------------------------|
   |START           |END             |LABEL  (VER. 7.01    30JUN97:14:16:24)  |
   |----------------+----------------+----------------------------------------|
   |LOW             |              -1|00.00               P-  F  M100         |
   |             -1<               0<99                   P-. F  M100         |
   |              0|               1<99                   P.  F  M100         |
   |              1|HIGH            |00.00               P   F  M100         |
   ----------------------------------------------------------------------------


   ----------------------------------------------------------------------------
   |     INFORMAT NAME: @EVAL    LENGTH:    1   NUMBER OF VALUES:    5         |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   1  FUZZ:        0    |
   |--------------------------------------------------------------------------|
   |START           |END             |INVALUE(VER. 7.01    30JUN97:14:16:25)  |
   |----------------+----------------+----------------------------------------|
   |C               |C               |                                       1|
   |E               |E               |                                       2|
   |N               |N               |                                       0|
   |O               |O               |                                       4|
   |S               |S               |                                       3|
   ----------------------------------------------------------------------------
```

Example 7: Retrieving a Permanent Format

Procedure features:
PROC FORMAT statement options:
LIBRARY=

Other features:
FMTSEARCH= system option

Data sets:
SAMPLE on page 445.

This example uses the LIBRARY= option and the FMTSEARCH= system option to store and retrieve a format stored in a catalog other than WORK.FORMATS or LIBRARY.FORMATS.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=64 pagesize=60;
```

LIBRARY= stores the NOZEROS. format in the PROCLIB.FORMATS catalog.

```
proc format library=proclib;
```

The PICTURE statement creates the NOZEROS. format. NOZEROS. is explained in “Building a Picture Format: Step by Step” on page 445.

```
   picture nozeros
              low   -     -1  =  '00.00' (prefix='-'          )
               -1  <-<     0  =     '99' (prefix='-.' mult=100)
                0   -<     1  =     '99' (prefix='.'  mult=100)
                1   -   high  =  '00.00';
run;
```

The FMTSEARCH= system option adds the PROCLIB.FORMATS catalog to the search path that the SAS System uses to find user-defined formats. The FMTSEARCH= system option requires only a libref. FMTSEARCH= assumes the catalog name FORMATS if no catalog name appears. Without the FMTSEARCH= option, SAS would not find the NOZEROS. format.*

```
   options  fmtsearch=(proclib);
```

PROC PRINT prints the SAMPLE data set. The FORMAT statement associates the NOZEROS. format with the Amount variable.

```
proc print data=sample;
   format amount nozeros.;
   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

Output

```
       Retrieving the NOZEROS. Format from PROCLIB.FORMATS      1
                      The SAMPLE Data Set

                          Obs    Amount

                           1     -2.05
                           2      -.05
                           3      -.01
                           4       .00
                           5       .09
                           6       .54
                           7       .55
                           8      6.60
                           9     14.63
```

Example 8: Writing Ranges for Character Strings

Data sets:
PROCLIB.STAFF on page 463.

This example creates a format and shows how to use ranges with character strings.

Program

* For complete documentation on the FMTSEARCH= system option, see the section on SAS system options in *SAS Language Reference: Dictionary*.

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

The DATA step creates the TRAIN data set from the PROCLIB.STAFF data set, which was created earlier in "Examples" on page 463.

```
data train;
   set proclib.staff(keep=name idnumber);
run;
```

PROC PRINT prints TRAIN without a format.

```
proc print data=train noobs;
   title 'The TRAIN Data Set without a Format';
run;
```

Because the LIBRARY= option does not appear, the format is stored in WORK.FORMATS and is available only for the current SAS session.

```
proc format;
```

The VALUE statement creates the $SKILL. format, which prints each employee's identification number and the skills test they have been assigned. Employees must take either TEST A, TEST B, or TEST C, depending on their last name. The exclusion operator (<) excludes the last value in the range. Thus, the first range includes employees whose last name begins with any letter between A and D, and the second range includes employees whose last name begins with any letter between E and M. The tilde (~) in the last range is necessary to include an entire string that begins with the letter Z.

```
   value $skill  'a'-<'e','A'-<'E'='Test A'
                 'e'-<'m','E'-<'M'='Test B'
                 'm'-'z~','M'-'Z~'='Test C';
run;
```

PROC REPORT prints TRAIN. The FORMAT= option in the DEFINE statement associates $SKILL. with the Name variable. The column that contains the formatted values of Name is using the alias Test. Using an alias enables you to print a variable twice, once with a format and once with the default format. See Chapter 32, "The REPORT Procedure," on page 859for more information on PROC REPORT.

```
proc report data=train nowd;
   column name name=test idnumber;
   define test / display format=$skill. 'Test';
   title 'Test Assignment for Each Employee';
run;
```

Output

PROC PRINT output

```
                        The TRAIN Data Set without a Format                      1

                         Zook, Carla       Test C  Id
                             Name               Number

                             Capalleti, Jimmy     2355
                             Chen, Len            5889
                             Davis, Brad          3878
                             Leung, Brenda        4409
                             Martinez, Maria      3985
                             Orfali, Philip       0740
                             Patel, Mary          2398
                             Smith, Robert        5162
                             Sorrell, Joseph      4421
                             Zook, Carla          7385
```

PROC REPORT output

```
                         Test Assignment for Each Employee                       1

                         Name                Test    IdNumber
                         Capalleti, Jimmy  Test A  2355
                         Chen, Len           Test A  5889
                         Davis, Brad         Test A  3878
                         Leung, Brenda       Test B  4409
                         Martinez, Maria     Test C  3985
                         Orfali, Philip      Test C  0740
                         Patel, Mary         Test C  2398
                         Smith, Robert       Test C  5162
                         Sorrell, Joseph     Test C  4421
                         Zook, Carla         Test C  7385
```

Example 9: Filling a Picture Format

Procedure features:

PICTURE statement options:

FILL=
PREFIX=

This example

- prefixes the formatted value with a specified character
- fills the leading blanks with a specified character
- shows the interaction between the FILL= and PREFIX= options.

Program

```
options nodate pageno=1 linesize=64 pagesize=40;
```

The PAY data set contains the monthly salary for each employee.

```
data pay;
   input Name $ MonthlySalary;
   datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;
```

When FILL= and PREFIX= appear in the same picture, the format places the prefix and then the fill characters. The SALARY. format fills the picture with the fill character because the picture has zeros as digit selectors. The leftmost comma in the picture is replaced by the fill character.

```
proc format;
   picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;
```

PROC PRINT prints the PAY data set. The FORMAT statement temporarily associates the SALARY. format with the variable MonthlySalary.

```
proc print data=pay noobs;
   format monthlysalary salary.;
   title 'Printing Salaries for a Check';
run;
```

Output

```
                  Printing Salaries for a Check                 1

                      Name     MonthlySalary

                      Liu      ****$1,259.45
                      Lars     ****$1,289.33
                      Kim      ****$1,439.02
                      Wendy    ****$1,675.21
                      Alex     ****$1,623.73
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS ® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

SAS® Procedures Guide, Version 8

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

ISBN 1–58025–482–9

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

1st printing, October 1999