# SAS Certification Prep Guide: Base Programming for SAS 9, Third Edition

by SAS Institute



---

# Chapter 7: Creating and Applying User-Defined Formats

## Overview

### Introduction

If you read "Creating List Reports" on page 112, you learned to associate formats with variables either temporarily or permanently.

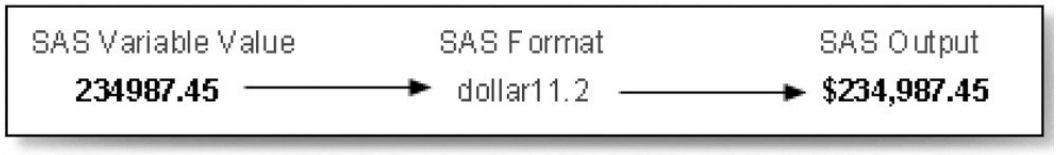


**Figure 7.1:** Variable and Format

But sometimes you might want to create custom formats for displaying variable values. For example, you can format a product number so that it is displayed as descriptive text, as shown below.


**Figure 7.2:** Variable and Custom Format

Using the FORMAT procedure, you can define your own formats for variables. You can store your formats temporarily or permanently, and you can display a list of all your formats and descriptions of their values.

### Objectives

In this chapter, you learn to

- create your own formats for displaying variable values
- permanently store the formats that you create
- associate your formats with variables.

## Introduction to PROC FORMAT

### Overview

Sometimes variable values are stored according to a code. For example, when the PRINT procedure displays the data set Perm.Employee, notice that the values for JobTitle are coded, and they are not easily interpreted.

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | 112 | 29996.63 |
| 2 | Lisa | Helms | 105 | 18567.23 |
| 3 | John | Higgins | 111 | 25309.24 |
| 4 | Amy | Larson | 113 | 32696.78 |
| 5 | Mary | Moore | 112 | 28945.89 |
| 6 | Jason | Powell | 103 | 35099.55 |
| 7 | Judy | Riley | 111 | 25309.37 |

| | | | | |
|---|---|---|---|---|
| 8 | Neal | Ryan | 112 | 28180.85 |

**Figure 7.1:** Perm.Employee Data Set

You can display more descriptive values for these variables. Here is how a report that contains formatted values for the variable JobTitle might look. The predefined SAS formats cannot help here.

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | technical writer | 23936.63 |
| 2 | Lisa | Helms | text processor | 18567.23 |
| 3 | John | Higgins | assoc. technical writer | 25303.24 |
| 4 | Amy | Larson | senior technical writer | 32636.78 |
| 5 | Mary | Moore | technical writer | 28945.89 |
| 6 | Jason | Powell | manager | 35033.55 |
| 7 | Judy | Riley | assoc. technical writer | 25303.37 |
| 8 | Neal | Ryan | technical writer | 28180.85 |

**Figure 7.2:** Perm.Employee Data Set with Formatted Values for JobTitle

However, you can use the FORMAT procedure to define your own formats for displaying values of variables.

### Invoking PROC FORMAT

To begin a PROC FORMAT step, you use a PROC FORMAT statement.

---

General form, PROC FORMAT statement:

**PROC FORMAT** <*options*>*;*

where *options* include

- **LIBRARY**=*libref* specifies the libref for a SAS data library to contain a permanent catalog of user-defined formats
- **FMTLIB** displays a list of all of the formats in your catalog, along with descriptions of their values.

---

Any time you use PROC FORMAT to create a format, the format is stored in a format catalog. If the SAS data library does not already contain a format catalog, SAS automatically creates one. If you do not specify the LIBRARY= option, the formats are stored in a default format catalog named Work.Formats.

As the libref Work implies, any format that is stored in Work.Formats is a temporary format that exists only for the current SAS session. At the end of the current session, the catalog is erased.

### Permanently Storing Your Formats

You can store your formats in a permanent format catalog named Formats when you specify the LIBRARY= option in the PROC FORMAT statement.

```
PROC FORMAT LIBRARY=libref;
```

But first, you need a LIBNAME statement that associates the libref with the permanent SAS data library in which the format catalog is to be stored.

```
libname library 'c:\sas\formats\lib';
```

When you associate a permanent format with a variable in a subsequent DATA or PROC step, you would then use the Library libref to reference the location of the format catalog.

We'll discuss the use of permanent user-defined formats later, after you learn how to create them.

Now, any format that you create in this PROC FORMAT step is stored in a permanent format catalog called Library.Formats.

```
libname library 'c:\sas\formats\lib';
proc format library=library;
   … ;
run;
```

In the program above, the catalog Library.Formats is located in the SAS data library **C:\Sas\Formats\Lib**, which is referenced by the libref Library.

Notice that LIB= is an acceptable abbreviation for the LIBRARY= option.

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

Also notice that you can specify a catalog name in the LIBRARY= option, and you can store formats in any catalog. The catalog name must conform to SAS naming conventions.

```
proc format lib=library.catalog;
```

Now that you know how to store your own formats, let's see how to create them.

## Defining a Unique Format

### Overview

You can use the VALUE statement to define a format for displaying one or more values.

---

General form, VALUE statement:

```
VALUE format-name
        range1='label1'
        range2='label2'
        …;
```

where *format-name*

- must begin with a dollar sign ($) if the format applies to character data
- must be a valid SAS name
- cannot be the name of an existing SAS format
- cannot end in a number
- does not end in a period when specified in a VALUE statement.

**Additional Note** In SAS 8.2, the format name must be a SAS name up to eight characters in length, not ending in a number.

**Additional Note** Beginning with SAS®9, a numeric format name can be up to 32 characters in length. A character format name can be up to 31 characters in length.

---

Notice that the statement begins with the keyword VALUE and ends with a semicolon after all the labels have been defined. The following VALUE statement creates the JOBFMT format to specify descriptive labels that will later be assigned to the variable JobTitle:

```
proc format lib=library;
   value jobfmt
         103='manager'
         105='text processor'
         111='assoc. technical writer'
         112='technical writer'
         113='senior technical writer';
run;
```

The VALUE range specifies

- a single value, such as **24** or **'S'**.
- a range of numeric values, such as **0-1500**.
- a range of character values enclosed in quotation marks, such as **'A'-'M'**.
- a list of unique values separated by commas, such as **90,180,270** or **'B', 'D', 'F'.** These values can be character values or numeric values, but not a combination of character and numeric values (because formats themselves are either character or numeric).

When the specified values are character values, they must be enclosed in quotation marks and must match the case of the variable's values. The format's name must also start with a dollar sign ($). For example, the VALUE statement below defines the $GRADE format, which displays the character values as text labels.

```
proc format lib=library;
   value $grade
         'A'='Good'
         'B'-'D'='Fair'
         'F'='Poor'
         'I','U'='See Instructor';
run;
```

When the specified values are numeric values, they are not enclosed in quotation marks, and the format's name should not begin with a dollar sign ($). The VALUE statement that defines the format JOBFMT assigns labels to numeric values.

### Specifying Value Ranges

You can specify a non-inclusive range of numeric values by using the less than symbol (<) to avoid any overlapping. In this example, the range of values from **0** to less than **13** is labeled as child. The next range begins at **13**, so the label teenager would be assigned to the values **13** to **19**.

```
proc format lib=library;
   value agefmt
         0-<13='child'
         13-<20='teenager'
         20-<65='adult'
         65-100='senior citizen';
run;
```

You can also use the keywords LOW and HIGH to specify the lower and upper limits of a variable's value range. The keyword LOW does not include missing numeric values. The keyword OTHER can be used to label missing values as well as any values that are not specifically addressed in a range.

```
proc format lib=library;
   value agefmt
         low-<13='child'
         13-<20='teenager'
         20-<65='adult'
         65-high='senior citizen'
         other='unknown';
run;
```

**Additional Note** If applied to a character format, the keyword LOW includes missing character values.

When specifying a label for displaying each range, remember to

- enclose the label in quotation marks
- limit the label to 256 characters
- use two single quotation marks if you want an apostrophe to appear in the label, as in this example:

```
000='employee''s jobtitle unknown';
```

To define several formats, you can use multiple VALUE statements in a single PROC FORMAT step. In this example, each VALUE statement defines a different format.

```
proc format lib=library;
   value jobfmt
         103='manager'
         105='text processor'
         111='assoc. technical writer'
         112='technical writer'
         113='senior technical writer';
   value $respnse
         'Y'='Yes'
         'N'='No'
         'U'='Undecided'
         'NOP'='No opinion';
run;
```

The SAS log prints notes informing you that the formats have been created.

SAS Log (Partial Listing)

```
01  proc format lib=library;
02     value jobfmt
03           103='manager'
04           105='text processor'
05           111='assoc. technical writer'
06           112='technical writer'
07           113='senior technical writer';
    NOTE: Format JOBFMT has been written to LIBRARY.FORMATS.
```

**Figure 7.3:** SAS Log

Because you have defined the JOBFMT format for displaying the values of JobTitle, the format can be used with PROC PRINT for more legible output.

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | 112 | 29996.63 |
| 2 | Lisa | Helms | 105 | 18567.23 |
| 3 | John | Higgins | 111 | 25309.24 |
| 4 | Amy | Larson | 113 | 32696.78 |
| 5 | Mary | Moore | 112 | 28945.89 |
| 6 | Jason | Powell | 103 | 35099.55 |
| 7 | Judy | Riley | 111 | 25309.37 |
| 8 | Neal | Ryan | 112 | 28180.85 |

**Figure 7.3:** Data Set with Unformatted Values for JobTitle

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | technical writer | 29996.63 |
| 2 | Lisa | Helms | text processor | 18567.23 |
| 3 | John | Higgins | assoc. technical writer | 25309.24 |
| 4 | Amy | Larson | senior technical writer | 32696.78 |
| 5 | Mary | Moore | technical writer | 28945.89 |
| 6 | Jason | Powell | manager | 35099.55 |
| 7 | Judy | Riley | assoc. technical writer | 25309.37 |
| 8 | Neal | Ryan | technical writer | 28180.85 |

**Figure 7.4:** Data Set with Formatted Values for JobTitle

The next section will show how to apply your formats to variables.

## Associating User-Defined Formats with Variables

### Referencing Your Formats

Remember that permanent, user-defined formats are stored in a format catalog. For example, the program below stores the format JOBFMT in a catalog named Library.Formats, which is located in the directory **`C:\Sas\Formats\Lib`** in the Windows environment.

```
libname library 'c:\sas\formats\lib';
proc format lib=library;
   value jobfmt
         103='manager'
         105='text processor'
         111='assoc. technical writer'
         112='technical writer'
         113='senior technical writer';
run;
```

To use the JOBFMT format in a subsequent SAS session, you must assign the libref Library again.

```
libname library 'c:\sas\formats\lib';
```

SAS searches for the format JOBFMT in two libraries, in this order:

- the temporary library referenced by the libref Work
- a permanent library referenced by the libref Library.

SAS uses the first instance of a specified format that it finds.

**Additional Note** You can delete formats using PROC CATALOG or the SAS Explorer window.

### Assigning Your Formats to Variables

Just as with SAS formats, you associate a user-defined format with a variable in a FORMAT statement.

```
data perm.employee;
   infile empdata;
   input @9 FirstName $5. @1 LastName $7. +7 JobTitle 3.
         @19 Salary comma9.;
   format salary comma9.2 jobtitle jobfmt.;
run;
```

Don't worry about @ pointer controls in programs in this chapter, as in @9 FirstName— they don't affect the behavior of formats. To learn more about using @ pointer controls in SAS programs, see "Reading Raw Data in Fixed Fields" on page 514.

Remember, you can place the FORMAT statement in either a DATA step or a PROC step. By placing the FORMAT statement in a DATA step, you can permanently associate a format with a variable. Note that you do not have to specify a width value when using a user-defined format.

When you submit the PRINT procedure, the output for Perm.Employee now shows commas in the values for Salary, and it shows descriptive labels in place of the values for JobTitle.

```
proc print data=perm.employee;
run;
```

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | technical writer | 29,996.63 |
| 2 | Lisa | Helms | text processor | 18,567.23 |
| 3 | John | Higgins | assoc. technical writer | 25,309.24 |

| | | | | |
|---|---|---|---|---|
| 4 | Amy | Larson | senior technical writer | 32,696.78 |
| 5 | Mary | Moore | technical writer | 28,945.89 |
| 6 | Jason | Powell | manager | 35,099.55 |
| 7 | Judy | Riley | assoc. technical writer | 25,309.37 |
| 8 | Neal | Ryan | technical writer | 28,180.85 |

**Figure 7.5:** Data Set with Formatted Values for Salary and JobTitle

When associating a format with a variable, remember to

- use the same format name in the FORMAT statement that you specified in the VALUE statement
- place a period at the end of the format name when it is used in the FORMAT statement.

*Note:* The following example shows you how to create a format and then use the format in a DATA step. If you have already created the JOBFMT format, you do not need to use PROC FORMAT to recreate it here.

```
libname library 'c:\sas\formats\lib';
proc format lib=library;
   value jobfmt
         103='manager'
         105='text processor'
         111='assoc. technical writer'
         112='technical writer'
         113='senior technical writers';
run;
libname perm 'c:\data\perm';
filename empdata 'c:\data\temp\newhires.txt';
data perm.employee;
   infile empdata;
   input @9 FirstName $5. @1 LastName $7. +7 JobTitle 3.
         @19 Salary comma9.;
   format salary comma9.2 jobtitle jobfmt.;
run;
```

Notice that a period is *not* required at the end of the SAS format COMMA9.2 in the FORMAT statement. The period in this format occurs between the width specification and the decimal place specification. All formats contain periods, but only user-defined formats invariably require periods at the end of the name.

If you do not format all of a variable's values, then those that are not listed in the VALUE statement are printed as they appear in the SAS data set, as shown in the following example:

```
proc format lib=library;
   value jobfmt
         103='manager'
         105='text processor';
run;
proc print data=perm.employee;
run;
```

| | FirstName | LastName | JobTitle | Salary |
|---|---|---|---|---|
| 1 | Donny | Evans | 112 | 29996.63 |
| 2 | Lisa | Helms | text processor | 18567.23 |
| 3 | John | Higgins | 111 | 25309.24 |
| 4 | Amy | Larson | 113 | 32696.78 |
| 5 | Mary | Moore | 112 | 28945.89 |
| 6 | Jason | Powell | manager | 35099.55 |
| 7 | Judy | Riley | 111 | 25309.37 |
| 8 | Neal | Ryan | 112 | 28180.85 |

**Figure 7.6:** Perm.Employee Data Set

When you build a large catalog of permanent formats, it can be easy to forget the exact spelling of a specific format name or its range of values. Adding the keyword FMTLIB to the PROC FORMAT statement displays a list of all the formats in your catalog, along with descriptions of their values.

```
libname library 'c:\sas\formats\lib';
proc format library=library fmtlib;
run;
```

When you submit this PROC step, a description of each format in your permanent catalog is displayed as output.

FORMAT NAME: JOBFMT LENGTH: 23 NUMBER OF VALUES: 5
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 23 FUZZ: STD

| START | END | LABEL (VER. V7\|V8 13MAY2011:10:02:13) |
|---|---|---|
| 103 | 103 | manager |
| 105 | 105 | text processor |
| 111 | 111 | assoc. technical writer |
| 112 | 112 | technical writer |
| 113 | 113 | senior technical writer |

**Figure 7.7:** SAS Output: Format Description

In addition to the name, range, and label, the format description includes the

- length of the longest label
- number of values defined by this format
- version of SAS that was used to create the format
- date and time of creation.

## Chapter Summary

## Text Summary

***Invoking PROC FORMAT***

The FORMAT procedure enables you to substitute descriptive text for the values of variables. The LIBRARY= option stores the new formats in a specified format catalog; otherwise, they are stored in a default catalog named Work.Formats. The keyword FMTLIB displays the formats and values that are currently stored in the catalog. The VALUE statement defines a new format for the values of a variable.

***Defining a Unique Format***

Formats can be specified for a single value, a range of values, or a list of unique values. Unique values must be separated by commas. When character values are specified, the values must be enclosed in quotation marks, and the format name must begin with a dollar sign ($). You can specify non-inclusive numeric ranges by using the less than sign (<). The keywords HIGH, LOW, and OTHER can be used to label values that are not specifically addressed in a range.

***Associating User-Defined Formats with Variables***

To access the permanent, user-defined formats in a format catalog, you'll need to use a LIBNAME statement to reference the catalog library. To associate user-defined formats with variables in the FORMAT statement, use the same format names in both the FORMAT and VALUE statements, but place a period at the end of the format name when it is used in the FORMAT statement.

## Syntax

```
LIBNAME libref 'SAS-data-library';
PROC FORMAT LIBRARY=libref FMTLIB;
       VALUE format-name
         range1 = 'label1'
```

```
          range2 'label2'
          …;
RUN;
DATA SAS-data-set;
     INFILE data-file;
     INPUT pointer variable-name informat.;
     FORMAT variable(s) format-name.;
RUN;
```

### Sample Programs

```
libname library 'c:\sas\formats\lib';
proc format library=library fmtlib;
   value jobfmt
         103='manager'
         105='text processor';
run;

data perm.empinfo;
   infile empdata;
   input @9 FirstName $5. @1 LastName $7.
         +7 JobTitle 3. @19 Salary comma9.;
   format salary comma9.2 jobtitle jobfmt.;
run;
```

### Points to Remember

- Formats—even permanently associated ones—do not affect the values of variables in a SAS data set. Only the appearance of the values is altered.
- A user-defined format name must begin with a dollar sign ($) when it is assigned to character variables. A format name cannot end with a number.
- Use two single quotation marks when you want an apostrophe to appear in a label.
- Place a period at the end of the format name when you use the format name in the FORMAT statement.

### Chapter Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix

**1.** If you don't specify the LIBRARY= option, your formats are stored in Work.Formats, and they exist … ?

a. only for the current procedure.

b. only for the current DATA step.

c. only for the current SAS session.

d. permanently.

**2.** Which of the following statements will store your formats in a permanent catalog? ?

a.
```
libname library 'c:\sas\formats\lib';
proc format lib=library
   …;
```

b.
```
libname library 'c:\sas\formats\lib';
format lib=library
   …;
```

c.
```
library='c:\sas\formats\lib';
proc format library
   …;
```

d.
```
library='c:\sas\formats\lib';
```

```
proc library
   …;
```

**3.** When creating a format with the VALUE statement, the new format's name ?

- cannot end with a number
- cannot end with a period
- cannot be the name of a SAS format, and…

a. cannot be the name of a data set variable.

b. must be at least two characters long.

c. must be at least eight characters long.

d. must begin with a dollar sign ($) if used with a character variable.

**4.** Which of the following FORMAT procedures is written correctly? ?

a.
```
proc format lib=library
   value colorfmt;
         1='Red'
         2='Green'
         3='Blue'
run;
```

b.
```
proc format lib=library;
     value colorfmt
           1='Red'
           2='Green'
           3='Blue';
run;
```

c.
```
proc format lib=library;
   value colorfmt;
         1='Red'
         2='Green'
         3='Blue'
run;
```

d.
```
proc format lib=library;
   value colorfmt
         1='Red';
         2='Green';
         3='Blue';
run;
```

**5.** Which of these is *false*? Ranges in the VALUE statement can specify… ?

a. a single value, such as **`24`** or **`'S'`**.

b. a range of numeric values, such as **`0-1500`**.

c. a range of character values, such as **`'A'-'M'`**.

d. a list of numeric and character values separated by commas, such as **`90,'B', 180,'D',270.`**

**6.** How many characters can be used in a label? ?

a. 40

b. 96

c. 200

d. 256

**7.** Which keyword can be used to label missing numeric values as well as any values that are not specified in a range? ?

a. LOW

b. MISS

c. MISSING

d. OTHER

**8.** You can place the FORMAT statement in either a DATA step or a PROC step. What happens when you place it in a DATA step? ?

a. You temporarily associate the formats with variables.

b. You permanently associate the formats with variables.

c. You replace the original data with the format labels.

d. You make the formats available to other data sets.

**9.** The format JOBFMT was created in a FORMAT procedure. Which FORMAT statement will apply it to the variable JobTitle in the program output? ?

a. `format jobtitle jobfmt;`

b. `format jobtitle jobfmt.;`

c. `format jobtitle=jobfmt;`

d. `format jobtitle='jobfmt';`

**10.** Which keyword, when added to the PROC FORMAT statement, will display all the formats in your catalog? ?

a. CATALOG

b. LISTFMT

c. FMTCAT

d. FMTLIB

**Answers**

**1.** Correct answer: c

If you do not specify the LIBRARY= option, formats are stored in a default format catalog named Work.Formats. As the libref Work implies, any format that is stored in Work.Formats is a temporary format that exists only for the current SAS session.

**2.** Correct answer: a

To store formats in a permanent catalog, you first write a LIBNAME statement to associate the libref with the SAS data library in which the catalog will be stored. Then add the LIB= (or LIBRARY=) option to the PROC FORMAT statement, specifying the name of the catalog.

**3.** Correct answer: d

The name of a format that is created with a VALUE statement must begin with a dollar sign ($) if it applies to a character variable.

**4.** Correct answer: b

A semicolon is needed after the PROC FORMAT statement. The VALUE statement begins with the keyword VALUE and ends with a semicolon after all the labels have been defined.

**5.** Correct answer: d

You can list values separated by commas, but the list must contain either all numeric values or all character values. Data set variables are either numeric or character.

**6.** Correct answer: d

When specifying a label, enclose it in quotation marks and limit the label to 256 characters.

**7.** Correct answer: d

MISS and MISSING are invalid keywords, and LOW does not include missing numeric values. The keyword OTHER can be used in the VALUE statement to label missing values as well as any values that are not specifically included in a range.

**8.** Correct answer: b

By placing the FORMAT statement in a DATA step, you permanently associate the defined format with variables.

**9.** Correct answer: b

To associate a user-defined format with a variable, place a period at the end of the format name when it is used in the FORMAT statement.

**10.** Correct answer: d

Adding the keyword FMTLIB to the PROC FORMAT statement displays a list of all the formats in your catalog, along with descriptions of their values.