

Doing the Cool and Unusual with PROC FORMAT

Marty Hultgren, SAS® Institute, Minneapolis, MN

ABSTRACT

As an instructor at SAS I am privileged to work with students of all levels, from beginner to advanced, and one of the joys of my job is sharing little golden SAS nuggets with students – cool and useful things to do that can help back at work. The nuggets I present in this paper all concern the use of PROC FORMAT. I am assuming the reader has had some experience with PROC FORMAT and the VALUE statement, but I will spend a little time reviewing basic PROC FORMAT syntax, before demonstrating some other very useful techniques.

INTRODUCTION

This paper will take SAS programmers with a little familiarity with PROC FORMAT on a trip through some of its less commonly-used features. I will cover the following topics:

- brief review
- using a control dataset
- the FMTLIB option
- using the PICTURE statement
- storing formats permanently and the FMTSEARCH option
- overlapping ranges
- nested formats
- creating hyperlinks with PROC FORMAT

BRIEF REVIEW

I'd like to start with a short review of PROC FORMAT, concentrating on the VALUE statement.

The FORMAT procedure begins with the PROC FORMAT statement. Following the PROC FORMAT statement will be one or more VALUE statements, each defining a format. By default formats created with PROC FORMAT are stored in the work library.

```
PROC FORMAT;  
    VALUE format-name      range-1 = 'label'  
                           range-2 = 'label'  
                           ...  
                           range-n = 'label';  
RUN;
```

Format names can be 32 characters long in SAS 9, and no longer than 8 characters in earlier releases of SAS. Format names cannot end in a number, and character formats must begin with a '\$', while numeric formats can begin with either a letter or an underscore. The ranges are the values in the variables you wish to format. Character ranges should be quoted. The labels are the text-strings you would like to see written in place of the stored values. Labels should also be quoted.

What follows are two VALUE statements, demonstrating the syntax used to create both a character and a numeric format:

```
value $Genres 'B'='Blues'  
              'J'='Jazz'  
              'C'='Choral'  
              'P'='Pop';  
  
value money   0 -< 25000 = 'Up to 25K'  
              25000 -< 50000 = '25K to 50K'  
              50000 -< 75000 = '50K to 75K';
```

While these simple VALUE statements can be wonderful choices, we'll now take a look at several other capabilities of PROC FORMAT which add a lot of functionality.

USING A CONTROL DATASET

As most SAS programmers know, there is nearly always more than one way to get a job done in SAS. This small rule is no different when creating formats – there are several ways code can be written to create formats. If your format will be a small one, it is often easiest and fastest to type all the values yourself. But what if you need a format to display values for hundreds of manufacturing codes? It would be much more pleasant to have SAS do the typing for you, and that, essentially, is what happens when you use the CNTLIN option in PROC FORMAT to create a control data set.

In some cases, all the information (data) needed to build a format is already in place. Or at least in *some* place, and your job is then simple – somehow point PROC FORMAT to that place, and tell it what's what. In other words, tell PROC FORMAT the format name, and where to get both the values and the labels. For example, let's assume we have data sets containing alphabetic codes referring to various bird species. And we have one master data set containing all the common names of bird species and their associated code value, along with other variables:

CommonName	Code	OtherVariables....
Western Grebe	WEGR	
Clark's Grebe	CLGR	
Red-necked Grebe	RNGR	
Horned Grebe	HOGGR	
Eared Grebe	EAGR	
Least Grebe	LEGR	
Pied-billed Grebe	PBGR	
Common Loon	COLO	
<i>other rows....</i>		

Now remember what bits of information PROC FORMAT needs – a format name (which also tells SAS whether the format is a character or numeric format), a value to be replaced, and a label to write in place of the value. Of those pieces of information, the value and the label are already provided by the data, so we need only to provide a type and a name. We need to create a control data set, keeping the following in mind:

1. Any valid data set name will work
2. The variable(s) containing the values to be formatted should be renamed to START (and END if a range)
3. A RETAIN statement is best used to create the FmtName and Type variables

The following data step creates a temporary control data set, here called CTRL. The RENAME= data set option is used to rename the CODE variable to START. To specify a range we would need both a START and an END variable – in this case only START is needed. If the format being created will be a character format, you can choose to begin the name with a '\$', or not. If you do not begin your name with a \$, SAS will add it when the format is built. You must also create a TYPE variable, with either 'c' or 'n' as the value.

```
data ctrl;  
  set birds(rename=(code=start commonname=label));  
  retain fmtname 'birds' type 'c';  
run;
```

Label	Start	FmtName	Type
Western Grebe	WEGR	birds	c
Clark's Grebe	CLGR	birds	c
Red-necked Grebe	RNGR	birds	c
Horned Grebe	HOGGR	birds	c
Eared Grebe	EAGR	birds	c
Least Grebe	LEGR	birds	c
Pied-billed Grebe	PBGR	birds	c
Common Loon	COLO	birds	c

Once the control data set is built, a very simple PROC FORMAT step can run and create the \$birds format:

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

USING THE FMTLIB OPTION

Now that the \$BIRDS format has been output, is there a way to see it before using it? The FMTLIB option on the PROC FORMAT statement will print a table in the output window, which describes formats in a library. Here, the LIB option is pointing to the WORK library, and the FMTLIB option is causing PROC FORMAT to build a table describing all formats in that library. Notice that the START and END values are the same, because there was no range:

```
proc format lib=work fmtlib;
run;
```

```
„fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff†
,      FORMAT NAME: $BIRDS   LENGTH:   17   NUMBER OF VALUES:    8   ,
,   MIN LENGTH:    1  MAX LENGTH:  40  DEFAULT LENGTH 17  FUZZ:      0   ,
‡fffffffffffffffff...fffffffffffffffff...fffffffffffffffffffffffffffff%
,START              ,END              ,LABEL (VER. V7|V8   19JUL2004:07:11:36),
‡fffffffffffffffff^fffffffffffffffff^fffffffffffffffffffffffffffff%
,CLGR              ,CLGR              ,Clark's Grebe              ,
,COLO              ,COLO              ,Common Loon                ,
,EAGR              ,EAGR              ,Eared Grebe                 ,
,HOGR              ,HOGR              ,Horned Grebe                 ,
,LEGR              ,LEGR              ,Least Grebe                  ,
,PBGR              ,PBGR              ,Pied-billed Grebe            ,
,RNGR              ,RNGR              ,Red-necked Grebe             ,
,WEGR              ,WEGR              ,Western Grebe                ,
$fffffffffffffffff·fffffffffffffffff·fffffffffffffffffffffffffffffœ
```

USING THE PICTURE STATEMENT

Another wonderful capability of PROC FORMAT is provided with the PICTURE statement. The PICTURE statement allows programmers to create templates to be used when displaying numeric data. The following bulleted points describe and explain the PICTURE statement:

- Only numeric formats can be created using PICTURE.
- Templates are built using *digit selectors*.
- Digit selectors are either 0, to suppress leading zeros, or 9 (or any non-zero digit) to maintain leading zeros. For example, the digit selectors 009.99 would display 45 as 45.00, while 999.99 would display 45 as 045.00.
- Digits to the right of the decimal point are *always* displayed, regardless of the digit selector used. So both the digit selectors '009.99' and '009.00', would display 45 as 45.00.
- Leading non-numeral characters are ignored. Use the PREFIX option to include them.
- Non-numeral characters *following* a leading 0 or 9 will be displayed as typed.

To demonstrate the PICTURE statement, imagine running a PROC FREQ and writing the statistics out to a data set, in order to present them your own way. The percentages that PROC FREQ creates have already been multiplied by 100, so using the PERCENT format will greatly inflate your results. The following shows how you can present a percentage like 12.34 with a trailing percent sign:

```
proc format;
  picture MyPct low -< 0   = '009.999%' (prefix='(')
                  0 - high = '009.999%';
run;
```

Notes:

Note that in this case I'm specifying two ranges, and that for the first, I'm using the PREFIX option. Also note the use of 0s and 9s. With the PICTURE statement, integers act as placeholders. Use 9 to include leading zeros, and 0 to suppress them. The other characters seen above, the "%" and the ")" will print as they appear in the statement. Since the picture must begin with a number, the PREFIX option is used to insert some nonnumeric character at the beginning of a value, in this case the open parenthesis.

CREATING A DATE FORMAT WITH THE PICTURE STATEMENT

Another nice use of the PICTURE statement is for the creation of date formats. Date formats are built using *directives*, which are special characters that you can use to specify how to format date, time, or datetime values. In order to use directives, you must specify the DATATYPE= option in the PICTURE statement.

Here's a short list of some directives (look under The FORMAT Procedure in the SAS Procedures Guide for a complete list):

- %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
- %m Month as a decimal number (1–12), with no leading zero
- %y Year without century as a decimal number (0–99), with no leading zero
- %Y Year with century as a decimal number

Before looking at an example, there are a few important things to mention:

- text can be used along with directives
- directives and text should be placed in single quotes to prevent attempts by the macro processor to resolve directives as macro calls
- directives which produce numbers do not return leading zeros by default. Adding a 0 before the directive will generate a leading zero. For example, %y will format 2004 as '4', where %0y will format 2004 as '04'
- the DATATYPE option should be used with any range that contains a number, whether or not directives are used. The 3rd range below has no directives, but the year value, 2004, would cause problems if DATATYPE=DATE were not specified.

```
proc format;
  picture MyDate
    '01jan2004'd - '30dec2004'd = '%0d-%b-%Y' (datatype=date)
    '31dec2004'd = '%0d-%b-%Y New Years!' (datatype=date)
    other='Not in 2004' (datatype=date);
run;

data _null_;
  date1='27sep2004'd;
  date2='31dec2004'd;
  put date1 = MyDate11. /
      date2 = MyDate22.;
run;
```

```
date1=27-SEP-2004
date2=31-DEC-2004 New Years!
```

STORING FORMATS PERMANENTLY

When you create a format, SAS will store it in the Formats catalog in your WORK library by default. The Formats catalog will be created if it does not already exist. Then, when SAS encounters a format in your code, it uses the following search pattern to look for that format:

1. SAS formats

2. WORK formats
3. LIBRARY libref formats
4. Other libraries specified using FMTSEARCH

SAS will automatically search for formats in the first 3 locations. So if you store your formats permanently and assign a libref to that location called LIBRARY, that location will be searched. But what if you don't want to use the name LIBRARY as your libref? Or what if you have format catalogs in 2 locations referenced by 2 librefs? The FMTSEARCH option on the global OPTIONS statement will add new locations to the search path.

In the following example a format is being created and stored in the TEST library on my Windows machine. Because no catalog name is being specified on the PROC FORMAT statement, PROC FORMAT will create a catalog called FORMATS.

```
libname test 'c:\temp';
proc format lib=test;
  value $grade 'A'='Excellent'
               'B'='Very Good'
               'C'='Fair'
               'D'='Poor'
               'F'='Fail';
run;
```

If I try to use \$GRADE format in this or a later SAS session without first telling SAS where to find it, I will get an error message like the following:

ERROR: The format \$GRADE was not found or could not be loaded.

Telling SAS where to find the \$GRADE format is done via the FMTSEARCH option, specified on an OPTIONS statement:

OPTIONS FMTSEARCH=(*libraryname*)

Multiple library names can be specified within the parentheses, separated by a space. If no catalog name is specified with the library name, SAS assumes the catalog to search is called FORMATS. The following OPTIONS statement shows FMTSEARCH referencing the TEST library (and by default, the FORMATS catalog). The log excerpt shows that the PROC PRINT step ran successfully:

```
options fmtsearch=(test);

proc print data=grades;
  format grade $grade.;
run;
```

NOTE: There were 9 observations read from the data set WORK.GRADES.

OVERLAPPING RANGES

There are times when one value can be grouped in different ways. For example, I might decide to categorize the number of days in each season in Minnesota which are below or above zero, and I might decide to further categorize them by 'comfort rating' (mine). So 40F might be in both the 'Above Zero' and the 'Quite Chilly' categories. To do this in the same table I can create a format using the MULTILABEL option. Note that this type of format can only be used with 3 procedures: TABULATE, MEANS and SUMMARY.

```
proc format;
    value jazz      01 = 'Artie Shaw'
                   02 = 'Duke Ellington'
                   03 = 'Bill Evans'
                   04 = 'Miles Davis';

    value pop       100 = 'The Grateful Dead'
                   101 = 'Sparklehorse'
                   102 = 'The Cure'
                   103 = 'Tom Waits'
                   104 = 'Lauryn Hill';

    value choral    201 = 'Tallis Scholars'
                   202 = 'The Dale Warland Singers';

    value music     1-99  = [jazz.]
                   100-199 = [pop.]
                   200-299 = [choral.];
```

```
run;

data _null_;
  input music @@;
  put music= music.;
cards;
1 2 102 104 201 202
;
```

```
music=Artie Shaw
music=Duke Ellington
music=The Cure
music=Lauryn Hill
music=Tallis Scholars
music=The Dale Warland Singers
```

**CRE
ATING
HYPERLINKS**

WITH PROC FORMAT

Finally, here is a simple example of how proc format can be used to create hyperlinks in an html table produced by the Output Delivery System, allowing details to be presented for each cell of a table produced by PROC TABULATE. I am assuming the reader has some familiarity with html, and can see that html tags are being assigned as the labels for different musical styles. When the html file is created, each music genre will hyperlink to another file containing the titles of compact discs within that genre.

```
proc format;
  value $MusicLib
    'African' = '<A HREF="African.html"> African </A>'
    'Asian'   = '<A HREF="Asian.html">   Asian   </A>'
    'Big Band' = '<A HREF="BigBand.html"> Big Band </A>'
    'Blues'    = '<A HREF="Blues.html">   Blues   </A>'
    'Classic'  = '<A HREF="Classic.html"> Classical </A>'
    'Folk'     = '<A HREF="Folk.html">    Folk    </A>'
    'Jazz'     = '<A HREF="ModJazz.html"> Jazz    </A>'
    'Other'    = '<A HREF="Other.html">   Other   </A>'
    'Reggae'   = '<A HREF="Reggae.html">  Reggae   </A>'
    'Rock'     = '<A HREF="Rock.html">    Rock    </A>';
run;
```

After creating the \$MusicLib format, it can be referenced in another step, in this case PROC TABULATE. The row titles produced by PROC TABULATE will hyperlink to other html files.

```
ods html file='c:\temp\new\MusicLib.html' style=festival;

proc tabulate data=musiclib f=3.;
  class category;
  var count;
  table category, count='Number of CDs'*sum=' '
    / box='Music Category' indent=0;
  format category $musiclib.;
run;

ods html close;
```

Output produced by PROC TABULATE:

Music Category	Number of CDs
African	9
Asian	7
Blues	24
Classical	24
Folk	16
Jazz	44
Rock	45

HTML file hyperlinked to from the 'African' row title value in the above PROC TABULATE report:

African CDs

Artist	Album Title
Fela Kuti	Open and Close
Various	African Odyssey
Habib Koite	Ma Ya
Seydu	Diamond Tears
Oliver Mtukudzi	Tuku Music
S.E. Rogie	Palm Wine Guitar Music

CONCLUSION

There are many ways to make use of PROC FORMAT besides the VALUE statement. I hope this paper will make use of some of these other techniques.

REFERENCES

SAS Institute Inc., *SAS Language Reference: Dictionary*, Cary, NC: SAS Institute Inc., 2000.

SAS Institute Inc., *SAS OnlineDoc®*, Version 8, Cary, NC: SAS Institute Inc., 2000.

Lund, Pete (2002), "More Than Just Value: A Look Into the Depths of PROC FORMAT", *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference CD*, Paper 4.

Staum, Roger (2000), "SAS Software Formats: Going Beneath the Surface", *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference CD*, Paper 2.

CONTACT INFORMATION

Marty Hultgren
SAS Institute, Inc.
800 Nicollet Mall, Suite 2800
Minneapolis, MN 55402
Email: Marty.Hultgren@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute, Inc. in the USA and other countries.