

A Quick Introduction to PROC FORMAT

Steve Reading, SAS Institute Inc., Atlanta, GA

Abstract

The Format Procedure enables users to quickly and easily improve the appearance of reports and graphs, process data unique to their computing environment, and dramatically improve the programming efficiency of their application. This Beginning Tutorial will provide real-life examples and tips on putting PROC FORMAT to work.

Introduction

The power and benefits of PROC FORMAT can be easily overlooked by the beginning SAS® user. This procedure empowers the user to rapidly improve the physical appearance of text and graphical output as well as the programming efficiency of the application itself. This short paper examines both of these benefits and clarifies how PROC FORMAT can help deliver more meaningful and understandable information.

What are Formats and Informats?

Formats and informats enable users to control the appearance of data in reports and graphs as well as control how data are read by the SAS System, respectively. The SAS System supplies numerous formats and informats to read in and write out date values, binary, ASCII, EBCDIC, etc. The FORMAT procedure allows users to create their own formats and informats and integrate them with those supplied by the SAS System.

Formats allow more meaningful character or numeric representations of underlying data. As an example, your company may store information about departments as a series of characters or numbers. At SAS Institute, the Atlanta Regional Office is included in department SM6. This short abbreviation is more efficient in terms of storage requirements, but is more difficult to understand in a report. In the report itself, you may wish to use a long description such as "Atlantic Coast Region".

Informats allow the SAS System to interpret the underlying or raw data and read it into a SAS data set. Dates are a good example of this. Suppose your date information is stored in a flat file as "10/01/96". You could easily read this string as a character field using the \$8. informat. However, that would not really be a true representation of that data. Often it is necessary to subtract one date from another to calculate the number of days. This could not be done accurately if the variable was a character value. Therefore, it would be better to use the MMDDYY8. informat. This informat allows the SAS System to understand that the field is a date value and that it needs to be stored as a numeric value, even though the field itself contains "/".

User Written Formats and Informats

The FORMAT procedure enables users to create their own formats and informats that will more accurately represent data unique to their computing environment. Let's use the example above for the

various sales and marketing groups at SAS Institute.

If you know the names of these groups, it is a simple matter to type the full name as you would like it to appear and equate that name with the three character abbreviation found in the data. The program in Figure One illustrates the creation of the \$REGION character formatⁱ.

```
libname myfmts 'c:\demo611\sesug';
options fmtsearch=(myfmts);
proc format library=myfmts;
value $region 'SM1' = 'Mid-Atlantic'
              'SM2' = 'Midwest'
              'SM3' = 'Metro New York'
              'SM4' = 'Southeast'
              'SM5' = 'Independent'
              'SM6' = 'Atlantic Coast'
              'SM7' = 'Northeast'
              'SM8' = 'Ohio Valley'
              'SM9' = 'Rocky Mountain';
run;
```

Figure 1

The OPTIONS statement sets the Format Search (FMTSEARCH) option, telling the SAS System to look in MYFMTS library for user-written formats. The use of the LIBRARY option on the PROC FORMAT statement permanently stores the format for future use.

In just a few moments you have been able to create a format to transform the report depicted in Figure Two into the more visually appealing report shown in Figure Three.

The SAS System			
OBS	REGION	REVENUE	
1	SM1	3937439	
2	SM2	945633	
3	SM3	2998565	
4	SM4	1500699	
5	SM5	355699	
6	SM6	1800959	
7	SM7	1166989	
8	SM8	1699565	
9	SM9	1632555	

Figure 2

Revenue Summary		
OBS	REGION	REVENUE
1	Mid-Atlantic	\$3,937,439
2	Midwest	\$945,633
3	Metro New York	\$2,998,565
4	Southeast	\$1,500,699
5	Independent	\$355,699
6	Atlantic Coast	\$1,800,959
7	Northeast	\$1,166,989
8	Ohio Valley	\$1,699,565
9	Rocky Mountain	\$1,632,555

Figure 3

This example uses the \$REGION format along with a SAS supplied DOLLAR formatⁱⁱ. The report was created with the PROC PRINT program shown in Figure Four.

```
options ls=79 ps=56 nodate nonumber
        fmtsearch=(myfmts);
proc print data=sasug.regions;
format region $region. revenue dollar10.;
title 'Revenue Summary';
run;
```

Figure 4

Simply adding the FORMAT statement to the PRINT procedure can quickly and easily improve the appearance of the report and provide greater information to the recipient.

Formats can be permanently associated with the data set itself by using a FORMAT statement when the data set is created, through the DATASETS procedure, or interactively through the VAR window shown in Figure Five.

Variable Length Format			
	Inform	Key Label	
REGION	\$3	\$REGION.	N Region
REVENUE	8	DOLLAR10.	N Revenue

Figure 5

Building Formats from Look-up Tables

The report in Figure Six contains item numbers that are rather difficult to interpret. This report would be much easier to understand if it contained product descriptions.

Sales Report		
Number	Item	Sales
0000002		228.07
4184171		1800.00
4184181		470.40
4184623		1478.40
4184681		50.61
4184692		177.12
4184700		1320.00
4184723		8136.00

Figure 6

Most organizations have look-up tables or other data sources that are used to store this type of repetitive information. While the SAS System can utilize these data sources as look-up tables, creating formats using the Control Data Set (CNTLIN) option is a more efficient solution.

In the sample printout in Figure Seven, the look-up table lists the item number and the corresponding item description. You can use this data set to create a new format called \$PROD to be used in reports.

The SAS System			
OBS	ITEM	ITMDESC	
1	0000001	MISCELLANEOUS SALE 1	
2	0000002	MISCELLANEOUS CGP SALE	
3	0000003	MISCELLANEOUS SALE 2	
4	1000001	W-SHELF, BOTTOM	
5	1000002	WP-SHELF BOTTOM, 5/6000 EXP	
6	1000514	DUMMY, SLIPSHEET, 8000/8000, 1992	
7	1000543	DUMMY, PACKAGING COST F/ CB 900	
8	1110001	KIT - ABC	
9	1110003	KIT-COOKING ZONE POP CLIPS	
10	1112803	CARD - PRECISION FLAME	

Figure 7

First, create a small program to rename ITEM to START and ITMDESC to LABEL and create a new variable called FMTNAME. The sample program in Figure Eight will create the CNTLIN data set with the new variable names.

```
libname sesug 'c:\demo611\sesug';
libname library 'c:\demo611\sesug';

/* SORT THE DATA BY ITEM NUMBER */
proc sort data=sesug.product out=product nodupkey;
  by item;
/* DATA STEP RENAMES item AND itmdesc AND CREATES fmtname */
data prodform (rename = (item=start itmdesc=label)); /* CNTLIN IS
PRODFORM */
  set product;
  by item;
  if first.item;
  fmtname=$prod';
run;
```

Figure 8

Running the program in Figure Eight produces the data set in Figure Nine. You have all the necessary variables required of the control data set (i.e. START, LABEL, and FMTNAME).

The SAS System				
OBS	START	LABEL		FMTNAME
1	0000001	MISCELLANEOUS SALE 1		\$prod
2	0000002	MISCELLANEOUS CGP SALE		\$prod
3	0000003	MISCELLANEOUS SALE 2		\$prod
4	1000001	W-SHELF, BOTTOM		\$prod
5	1000002	WP-SHELF BOTTOM, 5/6000 EXP		\$prod
6	1000514	DUMMY, SLIPSHEET, 8000/8000, 1992		\$prod
7	1000543	DUMMY, PACKAGING COST F/ CB 900		\$prod
8	1110001	KIT - ABC		\$prod
9	1110003	KIT-COOKING ZONE POP CLIPS		\$prod

Figure 9

Now you can execute the PROC FORMAT statements in Figure Ten to create the \$PROD format. Notice that you simply use the CNTLIN option to specify the input control data set to feed information to the FORMAT procedure.

```
/* FORMAT PROCEDURE USES cntlin OPTION OR A CONTROL DATA SET TO
FEED INFORMATION */
proc format;
  cntlin=prodform;
  library=library;
run;
```

Figure 10

By using the now familiar FORMAT statement in Figure Eleven, you can quickly improve the report, making it more appealing and understandable. The reformatted report is shown in Figure Twelve.

```

options fmtsearch=(sesug) nodate nonumber ls=79 ps=56;
title 'Sales Report';

proc print data=sasug.newsales split='*' noobs;
var item sales;
format item $prod. sales dollar10.2;
label item = 'Item*Description'
      sales = 'Sales'
;
run;

```

Figure 11

Sales Report		
Item	Description	Sales
MISCELLANEOUS CGP SALE		\$228.07
EUROSTYLE TOOL SET		\$1,800.00
DELUXE CHROME TOOL SET		\$470.40
4" ADAPTER		\$1,478.40
IGNITOR		\$50.61
DECK PROTECTION PAN		\$177.12
COOKBOOK		\$1,320.00
88" COVER		\$1,344.00
FULL LENGTH COVER		\$1,122.00
ELECTRIC FAN		\$8,196.00

Figure 12

The benefits of user-created formats are now becoming more apparent. Some advantages not visible in the report are improved computer performance and ease of applying the format to the report. In more traditional look-up table scenarios, you might have been required to write code to physically reference the look-up table and perform a join or merge to extract the long descriptive information. The \$PROD format enables you to simply specify the FORMAT statement in the procedure and the SAS System does the rest.

The \$PROD format is also easy to maintain. If the look-up table is updated on a regular basis, simply rerun the CNTLIN example program above to capture the new descriptions.

Creating Templates for Formats

Often numeric data are more accurately represented by embedding spaces or other special characters within the data values. The PICTURE format enables you to

create a template for printing numeric data.

The next example contains sales figures for various cities in dollars. Although it is important to store the data to the penny, most individuals who need summary information would prefer '\$1.2 million' to '\$1,200,000'. Chart One shows the sales figures using a DOLLAR format. While the information is accurate, large numbers can sometimes be difficult to comprehend quickly.

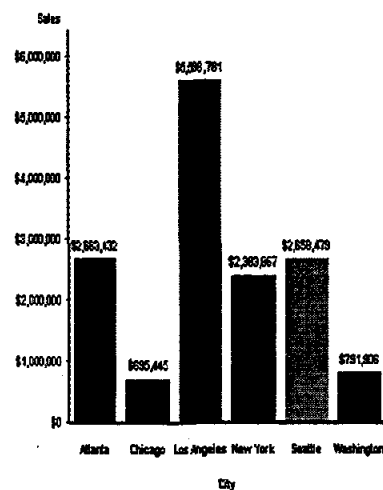


Chart 1

The program shown in Figure Thirteen creates a PICTURE format called MILLION to transform large numbers and make them easier to read.

```

proc format library=sasug;
picture million low-high='09.9M'
      ( prefix='$' mult=.00001);
run;

```

Figure 13

The "low" and "high" keywords enable you to define the format for the entire range of data values. The PICTURE specification indicates that the values should be displayed using the template of '09.9M'. Any non-zero digit is used to retain leading zeros. The "9.9" indicates

that only one leading zero is to be printed (in case we have values less than \$1 million)ⁱⁱⁱ. Any other characters print as specified in the PICTURE. The "prefix" specifies a leading character or set of characters and "mult" is a multiplication value to move the decimal point or otherwise scale the data to fit the picture template.

As in previous examples, the FORMAT statement, shown in the program in Figure Fourteen, applies the newly created MILLION format to the data. This transforms Chart One (using a normal DOLLAR format) into the new chart shown as Chart Two.

```
proc gchart data=sasug.city;
  vbar city /
    sum maxis=axis1 raxis=axis2
    width=6 patternid=midpoint
    type=SUM sumvar=salesum;
  format salesum million.;
  label salesum = 'Sales'
        city = 'City';
run;
```

Figure 14

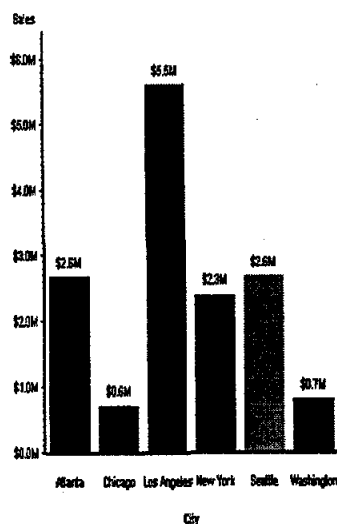


Chart 2

Chart Two can be quickly interpreted without having to count the number of commas!

Information About Your Formats

After having created all these great formats, you may need to print a listing or create a file of the format descriptions. Perhaps you have forgotten the specific ranges used or maybe documentation is needed of the formats to provide a record of a project or to inform other users. In all these instances, the format library (FMTLIB) or page (PAGE) options enable you to extract information from the format catalog and create printed output.

Figure Fifteen shows the use of the FMTLIB option to create the output in Figure Sixteen. Figure Sixteen shows all the formats stored in the SESUG.FORMATS catalog (the \$PROD format was not saved due to its size). The information consists of the format name, number of values, other statistics about the format, and, finally, a listing of the actual range of values of the formats themselves.

```
proc format library=sasug fmtlib;
run;
```

Figure 15

FORMAT NAME: MILLION LENGTH: 5 NUMBER OF VALUES: 1			
NEW LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 5 PUFF: STD			
START	END	LABEL (VER. 6.11 25JAN86:10:52:13)	
LOW	HIGH	00.0M	P0 P 00.00001

FORMAT NAME: SHOREON LENGTH: 14 NUMBER OF VALUES: 9			
NEW LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 14 PUFF: 0			
START	END	LABEL (VER. 6.11 25JAN86:10:52:13)	
SHO	SHO	Mid-Atlantic	
SHO	SHO	Midwest	
SHO	SHO	Metro New York	
SHO	SHO	Southeast	
SHO	SHO	Independent	
SHO	SHO	Atlantic Coast	
SHO	SHO	Northeast	
SHO	SHO	Ohio Valley	
SHO	SHO	Rocky Mountain	

Figure 16

Sometimes you might prefer a SAS data set containing the information about your formats. The output control data set (CNTLOUT) allows a SAS data set to be created to store all the relevant

information. This is also a good way to feed information into a data dictionary for use in a data warehouse environment.

The program shown in Figure Seventeen uses the CNTLOUT option to create a data set called SESUG.FORMLST to store information about the SESUG.FORMATS catalog. The printed report in Figure Eighteen shows the contents of this newly created data set.

```
proc format library=sasug cntlout=sasug.form1st;
proc print;
run;
```

Figure 17

	F	D	P	M	S
T	L	E	R	F	E
S	A	N	G	U	I
A	E	I	Z	L	T
T	D	X	N	Z	L
1 MILLION LOW HIGH	CW-ME	1 40 5	YK-28	.00001	O P N R M H
2 REGION SAS SAS Mid-Atlantic	1 40 14 0	0 .00000			C C N R
3 REGION SAS SAS Midwest	1 40 14 0	0 .00000			C C N R
4 REGION SAS SAS Metro New York	1 40 14 0	0 .00000			C C N R
5 REGION SAS SAS Southeast	1 40 14 0	0 .00000			C C N R
6 REGION SAS SAS Indianapolis	1 40 14 0	0 .00000			C C N R
7 REGION SAS SAS Atlantic Coast	1 40 14 0	0 .00000			C C N R
8 REGION SAS SAS Northeast	1 40 14 0	0 .00000			C C N R
9 REGION SAS SAS Ohio Valley	1 40 14 0	0 .00000			C C N R
10 REGION SAS SAS Rocky Mountain	1 40 14 0	0 .00000			C C N R

Figure 18

The information stored in SESUG.FORMLST is the same as that produced by the FMTLIB option. Now, however, a portable electronic record exists that can be used to facilitate application development or otherwise augment a report.

Summary and Conclusion

In this short discussion we have covered the main points of the FORMAT procedure. First, we defined both formats and informats and how these powerful tools can be used alter the physical appearance of our reports or to more accurately read in and process raw data, respectively. Next, we continued with several examples designed to give an introduction to the capabilities of this procedure. Finally, we examined

generating reports about the formats and informats themselves.

To many SAS users, the format procedure helps expedite the process of information delivery. With these examples, you should be able to apply the concepts discussed here to your own computing environment.

References

SAS Institute Inc., *SAS® Language: Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1990.

SAS Institute Inc., SAS® Procedures Guide, Version 6, Third Edition, Cary, NC: SAS Institute Inc., 1990.

ⁱ Formats and informats with the "\$" prefix are used only for character data. The total length of the name, including the "\$", must not exceed eight characters.

ⁱⁱ As indicated above, formats and informats used with numeric data do not use a "\$" prefix.

iii Digit selectors are numeric characters (0 through 9) that define positions for numeric values. A picture format with non-zero digit selectors prints any leading zeros, while digit selectors of 0 do not print leading zeros. SAS Institute Inc., *SAS® Procedures Guide, Version 6, Third Edition*. Page 288.