

User Defined Formats
Michel Schexanyder
Louisiana State University
irmike@lsu.edu

Abstract

SAS user defined formats provide a tool which can save programming and execution time as well as improve the appearance of output. User defined formats enhance the appearance of reports by substituting intelligible descriptions in place of obscure codes. The proc format cntlin= option converts a code table into a user defined format. A SAS permanent library is the appropriate place to store user defined formats. With the growth of this format library comes the need to document its contents. This documentation consists of: How can the format be updated? Where did the descriptions come from? When was the format last updated? SAS automatically provides some of the needed documentation. User defined formats can subset a secondary file before sorting and merging with a primary file.

User Defined Formats

A format statement associates formats with variables. When the SAS system writes the values of variables, it uses the format associated with the variables to print the values. In addition to the many predefined SAS formats, a SAS user can create user defined formats. The following format procedure creates a user defined format in the work library.

```
proc format;  
  value $SEXF  
    '1' = 'MALE '  
    '2' = 'FEMALE';
```

You can build a format for the names of the ten divisions in your company or the descriptions of the 3,000 account numbers in your accounting system. The format statement is both a data statement and a proc statement. A format statement in a proc step associates the format with the variables for the duration of that procedure. Using a format statement in a data step permanently associates a format with a variable by changing the descriptor information of the SAS data set containing the variable.

Three ways of using the account description format are:

```
format ACCTNBR $ACCTDESC.;  
ACCTDESC = put (ACCTNBR, $ACCTDESC.);  
ACCTDESC = ACCTNBR || ' ' || put (ACCTNBR, $ACCTDESC.);
```

The second and third methods create a new variable. The third method produces a new variable containing both the code and its description. For example 115300100 Biology - Research

Prior to Version 6 of SAS user defined formats were stored in a partitioned data set (or a directory) with each member containing one or more proc formats. Through an include statement, a program gained access to the format. With Version 6, SAS standardized the concept of a format library as a place to store user defined formats. Now one libname statement with a libref for the format library gives a program access to all formats in the format library, no include statements required. All of the advantages of user formats carry forward to the format library. New programs take advantage of existing formats. Also changes and updates to formats affect all programs that reference the format library

Building a User Defined Format Library

Our objective is to build a user defined format library to store all formats that we want easy access. Each SAS program will need one libname statement to point to this format library. We will show how to add new formats to the library and how to keep track of the formats in the library. SAS automatically provides some format documentation data.

First, build a permanent SAS data set to store all SAS user defined formats. This step is dependent on your operating system. Three sources of formats to add to this format library are: code tables from production systems (example, the account descriptions of your account numbers), code tables you build on your own, and proc formats stored in partitioned data sets. The following code builds a format of account description for your account numbers. The data set ACCOUNTS must contain the account number (ACCTNBR) and the account description (ACCTDESC).

```
libname LIBRARY 'PERM.SAS.LIBRARY' DISP=OLD; run;

data format; set ACCOUNTS;
  TYPE = 'C';
  length START $9 LABEL $30;
  START = ACCTNBR;
  FMTNAME= 'ACCTDESC'; LABEL=ACCTDESC;
  keep TYPE FMTNAME START LABEL;
proc sort; by FMTNAME START;
data format; set format; by FMTNAME START;
  if first.start;
proc format cntlin=format LIBRARY=LIBRARY;
```

The keep statement in the above code shows the four variables needed to define a format. The proc format converts the four variables contained in the cntlin= data set into a format. The resulting format is stored in the library specified after the library=.

The method is extremely effective if you have access to a data set containing the three variables: code type, code, and description. Ideally your company has such a data set for each of its application systems (accounting, payroll, purchasing, facilities, etc.). In this ideal case one program for each application system can add or update all the formats for that application system.

Adding an existing proc format to a Format Library

The following code transfers a format to the permanent format library.

```
proc format; value $SEXF
    '1' = 'MALE' '2' = 'FEMALE';
proc format cntlout=CNTRL;
data CNTRL; set CNTRL(rename=FMTCNAME=FMTC);
    length FMTCNAME $8;
    FMTCNAME =FMTC;
libname LIBRARY 'PERM.SAS.LIBRARY' DISP=OLD; run;
proc format cntlin=CNTRL LIBRARY=LIBRARY;
```

The cntlout= option on the proc formats creates a format data set for all formats in the working library. The last proc format transfers the format to the format library, just as in the previous case.

Tracking the formats in a format library

Once you have added several formats to your library, you may need to keep track of your formats. One method is to build a file which contains the source of the format values, the program that accesses the format values, the format name, and a format description. The example below shows some of the formats in the Chart of Accounts (COA) system which are built by the program contained in perm.prod.prgms(coafmts).

1	COA	PERM.PROD.PRGMS (COAFMTS)
2	\$ACCTDST	Account distribution code
2	\$ACCTNAM	Account name
2	\$AGENCYF	Agency code (federal accounts)
2	\$AAF	AA AUXILIARY ACCOUNT
2	\$ACF	AC AGENCY
2	\$ATF	AT ACCOUNT TYPE
2	\$BAF	BA BILLING ADDRESS
2	\$BCF	BC BILLING CYCLE
2	\$BFF	BF BILLING FORM
2	\$BTF	BT BILLING TYPE

The file shown above is part of a control file for the format library. Before adding a new format to the format library, insure that the new format name is not already in use.

The code found in the appendix produces the following report using the data shown above and data stored in the format library. Observation 64 describes the account name format (acctnam) which converts a 9 character account number into a 25 character account description. The format was last updated on 9/3/97 and there are 19,027 account numbers and descriptions.

```
----- SOURCE=COA SRCDESC=PERM.PROD.PRGM(S COAFMTS) -----
```

OBS	NAME	TYPE	LENGTH	DATE	COUNT	DESC
63	ACCTDST	FORMATC	9,9,3	09/03/97	19027	Account distribution code
64	ACCTNAM	FORMATC	9,9,25	09/03/97	19027	Account name
65	AGENCYF	FORMATC	9,9,2	09/03/97	5933	Agency code (federal accounts)
66	AAF	FORMATC	3,3,25	09/03/97	32	AA AUXILIARY ACCOUNT
67	ACF	FORMATC	2,2,25	09/03/97	51	AC AGENCY
68	ATF	FORMATC	1,1,17	09/03/97	7	AT ACCOUNT TYPE
69	BGTOBJ	FORMATC	6,6,4	02/26/98	1190	Budget object
70	BGTTYPE	FORMATC	6,6,2	02/26/98	1190	Budget type
71	BAF	FORMATC	3,3,25	09/03/97	1189	BA BILLING ADDRESS
72	BCF	FORMATC	1,1,25	09/03/97	20	BC BILLING CYCLE
73	BFF	FORMATC	2,2,25	09/03/97	19	BF BILLING FORM
74	BTF	FORMATC	2,2,17	09/03/97	2	BT BILLING TYPE

SAS automatically maintains the data contained in the type, length, and date columns of this report.

The following code produces a report of federal agency codes and descriptions.

```
%let fmtname=AGENCYF;
libname LIBRARY 'PERM.SAS.LIBRARY' DISP=SHR; run;
TITLE1 'SAS Format Library';
TITLE2 "Source: PERM.SAS.LIBRARY";
proc format library=library cntlout=format;
data format; set format;
    where fmtname="&fmtname";
proc print; by fmtname;
    var start label;
```

The following code generates an alphabetical list of the formats in the library.

```
libname LIBRARY 'PERM.SAS.LIBRARY' DISP=SHR; run;
TITLE1 'SAS Format Library';
TITLE2 "Source: PERM.SAS.LIBRARY";
proc CATALOG cat=LIBRARY.FORMATS;
    contents out=cat;
proc sort data=cat; by NAME;
data cat; set cat (rename=name=fmtname);
    length = SUBSTR(DESC,15,8); DROP DESC;
proc print; by libname memname;
```

Additional uses of user defined formats

So far we have we have used formats to retrieve a description in place of a code (state name in place of the state abbreviation). Formats can be used for table lookup. This includes cross walk tables. You may or may not want to include these formats in your main format library.. You are not limited to only one format library besides the work library. Option Fmtsearch (SAS Technical Report P-222, p90) overcomes the limitation of only two sources of user defined formats as is stated on page 296 of the SAS Procedures Guide Version 6, Third Edition.

Use of a user defined format in a where clause can subset a secondary file prior to sorting and merging with a primary file. In the following example the data set track.fresh contains a set of student records. The following code builds a format for each student. The format value for each student is COHORT. The library=track stores the format with the original data track.fresh.

```
OPTION FMTSEARCH=(TRACK SSNCHNG);
data format; set TRACK.FRESH;          * primary file;
    TYPE = 'C';
    length START $9 LABEL $5;
    START = SSN;
    FMTNAME= 'COHORT'; LABEL='COHORT';
    keep TYPE FMTNAME START LABEL;
proc sort; by FMTNAME START;

data format; set format; by FMTNAME START;
    if FIRST.START;                  * eliminate duplicates;
proc format cntlin=format LIBRARY=TRACK;

data ADDRESS; set DIR.ADDRESS;        * secondary file;
    WHERE put(SSN,$COHORT.)= 'COHORT';
proc sort; by SSN;

data FRESH; merge TRACK.FRESH(IN=F) ADDRESS(IN=A); by SSN; if F;
```

References

SAS Procedure Guide Version 6, Third Edition (56080).

SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07 (59139).

SAS Professional Programming Secrets, Aster and Seidman, Windcrest/ McGraw Hill
Highly recommended.

Appendix A

The following code generates a SAS user defined format library report.

```
OPTIONS LS=80 ;
TITLE1 'SAS Format Library';
data DESC; INFILE DESC;
TITLE2 'Source: PERM.PROD.SOURCE(LIBRARY)';
INPUT @1 RCD $1.
      @9 F1 $8.
      @21 F2 $30.;
RETAIN SOURCE SRCDESC;
length name $8;
if RCD GT '0';
if RCD = '1' THEN DO; SOURCE=F1; SRCDESC=F2; END;
if RCD = '2' THEN DO; NAME =F1; DESC =F2; END;
if RCD = '2' ;
keep SOURCE SRCDESC NAME DESC;
proc sort; by NAME;

libname LIBRARY 'PERM.SAS.LIBRARY' DISP=SHR; run;
proc CATALOG cat=LIBRARY.FORMATS;
  contents out=cat;
proc sort data=cat; by NAME;
data cat; set cat;
  length = SUBSTR(DESC,15,8); DROP DESC;

proc format library=library cntlout=format;
proc summary nway data=format(rename=fmtname=name);
  class NAME;
  output out=freq;

data cat; merge cat(IN=F) freq(IN=D); by NAME;
  rename _freq_=count;

data cat; merge cat(IN=F) DESC(IN=D); by NAME;
proc sort; by SOURCE DESC;
proc print uniform ; by SOURCE SRCDESC;
  var NAME TYPE LENGTH DATE COUNT DESC;
proc printto print=disk;
proc print uniform ; by SOURCE SRCDESC;
  var NAME TYPE LENGTH DATE COUNT DESC;
```

The proc printto directs the output of the second proc print to disk. This enables a programmer to have access to the report in a window on his terminal.