

SAS V6.06 PROC FORMAT: Stored Statically or Dynamically

Bill Parman, Lincoln National Corporation

INTRODUCTION

The purpose of this paper is to illustrate how to implement SAS* 6.06 permanent formats in the MVS environment. Two general categories of formats will be presented. The first category is static formats; those whose values do not change. The second category is dynamic formats; those whose values are data dependent.

This paper makes the following assumptions:

1. The reader is reasonably skilled in the coding of SAS data steps.
2. The reader is familiar with the use of SAS libraries and catalogs.
3. The reader is experienced with the use of instream PROC FORMATS.
4. FSEDIT is not available.

STATIC FORMATS

When the content of the VALUE and INVALUE statement of formats and informats, rarely, if ever, change, they are considered static. When these types of formats and informats are used by several programs, there is benefit in making them permanent formats or informats.

To illustrate static user-defined formats and informats, suppose several programs have been written where each program possesses a hardcoded copy of the following PROC FORMAT (Yes, there are two SAS functions that will convert state postal codes to state names, but please indulge me?):

```
PROC FORMAT LIBRARY=LIBRARY;  
  VALUE $MWSFM  
    'ND' = 'North Dakota'  
    'SD' = 'South Dakota'  
    'NB' = 'Nebraska'  
    'KS' = 'Kansas'  
    'MO' = 'Missouri'  
    'IA' = 'Iowa'  
    'MN' = 'Minnesota'  
    'WI' = 'Wisconsin'  
    'IL' = 'Illinois'  
    'IN' = 'Indiana'  
    'MI' = 'Michigan'  
    'OH' = 'Ohio'  
  OTHER = 'Non-member State';
```

If another state were to be added to the above VALUE statement, each program possessing this PROC FORMAT would have to be changed; whereas, if the FORMAT were permanently stored in the FORMATS catalog of a private SAS library to which each program had access, only the permanent format would have to be updated. The programs would remain unchanged.

Below is an example of a the SAS program necessary to store the above format in the FORMATS catalog of TS01D.V6SASLIB.DATA:

Program 1

```
1. LIBNAME LIBRARY 'tsoid.V6SASLIB.DATA';
2. PROC FORMAT LIBRARY=LIBRARY;
   VALUE $MWSFM
     'ND' = 'North Dakota'
     'SD' = 'South Dakota'
     'NB' = 'Nebraska'
     'KS' = 'Kansas'
     'MO' = 'Missouri'
     'IA' = 'Iowa'
     'MN' = 'Minnesota'
     'WI' = 'Wisconsin'
     'IL' = 'Illinois'
     'IN' = 'Indiana'
     'MI' = 'Michigan'
     'OH' = 'Ohio'
     OTHER = 'Non-member State';
RUN;
```

Figure 1. Creating a static, permanent format

For an explanation of the program in Figure 1, see the list below.

- | <i>Item</i> | <i>Explanation</i> |
|-------------|---|
| 1 | Associate the private SAS library, into which the permanent format is to be stored, to the libref, LIBRARY.

Note: When creating a stored format, the libref, to which the private SAS library is associated, does not have to be LIBRARY. |
| 2 | Create the permanent format, MWSFM, in LIBRARY.FORMATS.

The combination of the LIBNAME statement in line 1 and the LIBRARY= option in line 2, cause the format, MWSFM, to be permanently stored in the FORMATS catalog of TSOID.V6SASLIB.DATA. |

Program 2

```
1. LIBNAME LIBRARY 'tsoid.V6SASLIB.DATA';
   OPTIONS NODATE;
   DATA TEMP;
     INFILE CARDS;
     INPUT @01 lname      $char10.
           @11 fname      $char06.
           @17 state      $char02.
           @19 age         2.
           @21 company     $char25.;
     OUTPUT TEMP;
   CARDS;
     Smith      John  ND37Sunny Sun Flowers
     Jones      Jane  KY23Old Crow Distillery
     Jackson    Barb  MI41General Motors
   ;
   RUN;
   TITLE1 'Conference Attendance List';
   PROC PRINT DATA=TEMP NOOBS;
2.   FORMAT STATE $MWSFM.;
   RUN;
```

Figure 2. Referencing a Stored Format

For an explanation of the program in Figure 2 on page 2, see the list below:

- | Item | Explanation |
|-------------|---|
| 1 | Associate the private SAS library, from which stored formats are to be referenced, to the libref, LIBRARY.

Note: When existing formats are to be referenced, the SAS library file possessing their FORMATS catalog MUST be associated with the libref, LIBRARY. |
| 2 | Reference the permanent format, MWSFM.

SAS will attempt to find the format, MWSFM, in WORK.FORMATS catalog. Because the format, MWSFM, is not in WORK.FORMATS and the SAS library file, 'tsoid.V6SASLIB.DATA', has been associated with the libref, LIBRARY, SAS will continue searching for MWSFM in LIBRARY.FORMATS. |

Conference Attendance List				
LNAME	FNAME	STATE	AGE	COMPANY
Smith	John	North Dakota	37	Sunny Sun Flowers
Jones	Jane	Non-member State	23	Old Crow Distilleries
Jackson	Barb	Ohio	41	Buckeyes Unlimited

Figure 3. Output from Program 2

DYNAMIC FORMATS

Dynamic formats and informats are those whose value and invalue statement, respectively, depends on the content of a temporary or permanent SAS data set or an external file.

Permanent Dynamic Formats Used for Code Conversions

Suppose that a VSAM file exists within a system that maintains the correspondence of various codes and their descriptions. The codes maintained in this VSAM table change on a frequent basis. Numerous SAS programs are to be written that need the descriptions from this VSAM file for reporting purposes.

To solve the above problem, selected code groups could be read into a temporary SAS data set. Using the temporary SAS data set of selected groups, codes and descriptions, another temporary or permanent SAS data set can be created that can be used by PROC FORMAT as the CNTLIN= data set. In this manner each time the job is run, a new CNTLIN= data set is created that reflects any changes that may have occurred in the VSAM file since the last run. Hence, the resulting formats and informats flex with the changes that occur in the VSAM file.

Suppose that the VSAM file has the following record description. The key is made up of GROUP and CODE. (Keep in mind that a card file is being used to simulate the VSAM file.)

Field	Position/Description
GROUP	1 - 3 / Group Id
CODE	5 - 6 / Code
DESC	8 - 30 / Code Description

For the following illustration, assume that there are four sets of codes maintained in the VSAM file: car codes, region codes, dealer codes, and unit price codes.

One of the programs in this system is to print a sales report summarizing total sales by dealer within region. The input data set contains codes for region, dealer, and car. When the report is printed, the description of each code is to be printed instead of the code itself. In the case of unit price, an informat is needed to convert the car code into a numeric unit price so that total sales can be calculated.

Suppose that the input file to be used for the report has the following record description.

Field	Position/Description
REGION	1 - 2 / Region Code
DEALER	4 - 5 / Dealer Code
CAR	7 - 8 / Car Code
UNITS	10 - 12 / Units sold

See Figure 4 on page 5 and Figure 5 on page 7 for a complete solution. To simulate the VSAM file, a CARD data set will be used.

Program 3

```

1. LIBNAME LIBRARY 'tsoid.V6SASLIB.DATA';
   OPTIONS NODATE;
2. DATA CODES;
   INFILE CARDS;
   INPUT @01 GROUP      $CHAR03.;
   IF GROUP IN('005' '010' '015' '020');
   INPUT @05 CODE      $CHAR02.;
   @08 DESC      $CHAR30.;
   OUTPUT CODES;
3. CARDS;
   005 05 2-Door Sedan
   005 10 4-Door Sedan
   005 15 2-Door Hardtop
   010 10 North
   010 30 South
   015 05 Honest Harry's Auto Sales
   015 10 Tom's Trustworthy Auto Sales
   015 15 Gary's Good Deal Auto Sales
   020 05 3400.00
   020 10 3500.00
   020 15 4500.00
   ;
   RUN;
4. DATA TEMP(DROP=GROUP RENAME=(CODE=START DESC=LABEL));
5.   LENGTH FMTNAME $ 7;
6.   RETAIN FMTNAME
7.     TYPE      'C';
8.   SET CODES;
9.   BY GROUP NOTSORTED;
   IF FIRST.GROUP THEN
     SELECT (GROUP);
       WHEN ('005') FMTNAME = 'CARFMT';
       WHEN ('010') FMTNAME = 'REGFMT';
       WHEN ('015') FMTNAME = 'DEALFMT';
       WHEN ('020')
         DO;
           FMTNAME = 'UPRFMT';
           TYPE      = 'I';
         ;
     ;
10.  END;
     OUTPUT TEMP;
     RUN;
     *;
11. PROC FORMAT LIBRARY=LIBRARY CNTLIN=TEMP;
     RUN;

```

Figure 4. Dynamic stored formats and code conversions - Part I

For an explanation of the program in Figure 4, see the list below.

Item	Description
1	Associate the SAS library file containing the needed FORMATS catalog to the libref, LIBRARY.
2	Create a temporary SAS data set, CODES, of selected groups, codes, and descriptions. This data set will be used as input to create the CNTLIN= data set, TEMP, to be used by PROC FORMAT in creating permanent formats and informats.
3	Card data set used to simulate the VSAM file of codes and descriptions.
4	For each observation read from CODES, the variables CODE and DESC are to be written to TEMP, but with the variable names, START and LABEL, respectively. Rather than assigning the variables CODE and DESC to START and LABEL on each iteration of the data step, they have been renamed in the DATA statement.

Since the variable, GROUP, is not needed by the CNTLIN = data set, it is dropped.

- 5 The longest format name is seven characters; therefore, the variable, FMTNAME, has its length set to seven.
- 6 This data step will create four subsets of observations. Each subset will be used to create one stored format or informat. Since the variable, FMTNAME, is part of each output observation, it needs to retain the same value for all observations common to a given subset. If it were not RETAINED, only the first observation in each subset would have a valid format name. All other observations would have a value of missing for the variable, FMTNAME.

Since the first three subsets of observations are to be of type, "C", the variable, TYPE, is retained and initialized to "C".

The last subset of observations is to be of type, "I"; therefore, the variable, TYPE, is assigned the value, "I", at the appropriate point of processing and retained for all observations common to group "020".

- 7 The SET statement reads each observation from the temporary SAS data set, CODES. Since the input data set is to be processed by groups, a BY statement is needed to establish by what variable(s) the data set is grouped. Since VSAM KSDS files are always in sequence by primary key, the data set, CODES, is in the desired sorted order. The NOTSORTED option allows FIRST-DOT (FIRST.) logic to take place without having to execute a PROC SORT beforehand.
- 8 Determine each change in the variable, GROUP. Each change in the variable, GROUP, indicates the beginning of a new format or informat.
- 9 For each change in GROUP, the SELECT statement determines what format name to assign to the variable, FMTNAME.
- 10 Because all observations common to group "020" are to create an informat, set the variable, TYPE, to a value of "I".
- 11 The PROC FORMAT will use the CNTLIN = data set, TEMP, as input and will create as many formats and informats as there are different values stored in the variable, FMTNAME. In our example that will be three formats: \$CARFMT, \$REGFMT, and \$DEALFMT; and one informat, @UPRFMT.

The LIBRARY = option instructs PROC FORMAT to place the created formats or informats as entries in the LIBRARY.FORMATS catalog. If no FORMATS catalog exists in the SAS library file, LIBRARY, one will be created. On the other hand, if a FORMATS catalog does exist, any format or informat entries that do not exist will be created; otherwise, they will be overwritten. Confirmation of this is found in the SAS log after the program is executed.

The combination of the data step starting in line 4 and the PROC FORMAT in line 11 is the equivalent of including the following instream PROC FORMAT in the program.

```

PROC FORMAT;
  VALUE $CARFMT
    '05' = '2-Door Sedan'
    '10' = '4-Door Sedan'
    '15' = '2-Door Hardtop';
  VALUE $DEALFMT
    '05' = 'Honest Harry's Auto Sales'
    '10' = 'Tom's Trustworthy Auto Sales'
    '15' = 'Gary's Good Deal Auto Sales';
  VALUE $REGFMT
    '10' = 'North'
    '20' = 'Central'
    '30' = 'South';
  INVALUE UPRFMT
    '05' = 3400
    '10' = 3500
    '15' = 4500;

```

```

                                PROGRAM 3 (Continued)

12. DATA SALES(DROP=UNITS UPRICE);
    INFILE CARDS;
    INPUT @01 REGION      $CHAR02.
          @04 DEALER      $CHAR02.
13.      @07 CAR          $CHAR02.
          @07 UPRICE      UPRFMT.
          @10 UNITS       3.;
    SALES = UNITS * UPRICE;
    OUTPUT SALES;
    CARDS;
    10 05 05 20
    10 05 10 15
    10 05 15 5
    30 15 05 10
    30 15 10 5
    30 15 15 35
    ;
    RUN;
14. PROC PRINT DATA=SALES;
    TITLE1 'ACME CORPORATION';
    TITLE2 'REGIONAL SALES REPORT';
    ID REGION DEALER;
    BY REGION DEALER;
    SUMBY DEALER;
    SUM SALES;
15.   FORMAT REGION $REGFMT.
        DEALER $DEALFMT.
        CAR $CARFMT.
        SALES DOLLAR8.;

    RUN;

```

Figure 5. Dynamic stored formats and code conversions - Part II

For an explanation of the program in Figure 5, see the list below.

- | Item | Description |
|------|---|
| 12 | Create a temporary SAS data set, SALES, to be used in the PROC PRINT in line 14. Because the variables, UNITS and UPRICE, are not used in the final report, drop them from SALES. |
| 13 | Read the car code in as a 2-byte character string. This code will be used in the PROC PRINT in line 15 to print the description of the car. |

Reread the 2-byte car code, but use the stored informat, UPRFMT, so that the code is converted to a numeric unit price upon input. With this technique, what would, otherwise, have to be done through the use of the PUT and INPUT functions can be done all at once by using the stored informat, UPRFMT.

- 14 PROC PRINT used to generate the report in Figure 6.
- 15 FORMAT statement that utilizes the stored formats created in line 11 to print the corresponding descriptions for the codes stored in the variables, REGION, DEALER, and CAR.

ACME CORPORATION REGIONAL SALES REPORT			
REGION	DEALER	CAR	SALES
North	Honest Harry's Auto Sales	2-Door Sedan	\$68,000
		4-Door Sedan	\$52,500
		2-Door Hardtop	\$22,500
-----	-----		-----
North	Honest Harry's Auto Sales		\$143,000
North			\$143,000
South	Gary's Good Deal Auto Sales	2-Door Sedan	\$34,000
		4-Door Sedan	\$17,500
		2-Door Hardtop	\$157,500
-----	-----		-----
South	Gary's Good Deal Auto Sales		\$209,000
South			\$209,000
			=====
			\$352,000

Figure 6. Output from program 4

Figure 6. Output from program 4

Printing the Formatted Content of Stored Formats and Informats

Program 4	
LIBNAME LIBRARY 'tsoid.V6SASLIB.DATA';	
PROC FORMAT LIBRARY=LIBRARY FMTLIB;	
SELECT \$REGFMT \$DEALFMT \$CARFMT @UPRFMT;	
RUN;	

Figure 7. Printing selected FORMAT and INFORMAT catalog entries

In programs 1 and 3 (See Figure 1 on page 2, Figure 4 on page 5, and Figure 5 on page 7), the formats \$MWSFMT, \$CARFMT, \$DEALFMT, and \$REGFMT, and the informat @UPRFMT were stored in the FORMATS catalog of TSOID.V6SASLIB.DATA. To print a formatted table of selected formats and informats from this FORMATS catalog, see Figure 7 as an example.

The output from the program in Figure 7 is found in Figure 8 on page 9.

Note: It is the FMTLIB option that indicates to PROC FORMAT that the formats and informats listed in the SELECT statement are to be printed. If the SELECT statement is omitted, ALL formats and informats will be printed.

Note: Since only one of the possible formats and informats is excluded from the output, the following EXCLUDE statement could have been used instead of the SELECT statement with the same result:

EXCLUDE \$MWSFMT;

FORMAT NAME: \$CARFMT LENGTH: 14 NUMBER OF VALUES: 3 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 14 FUZZ: STD		
START	END	LABEL (VER. 6.06 27MAR91:10:03:16)
05	05	2-Door Sedan
10	10	4-Door Sedan
15	15	2-Door Hardtop

FORMAT NAME: \$DEALFMT LENGTH: 28 NUMBER OF VALUES: 3 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 28 FUZZ: STD		
START	END	LABEL (VER. 6.06 27MAR91:10:03:16)
05	05	Honest Harry's Auto Sales
10	10	Tom's Trustworthy Auto Sales
15	15	Gary's Good Deal Auto Sales

FORMAT NAME: \$REGFMT LENGTH: 7 NUMBER OF VALUES: 3 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 7 FUZZ: STD		
START	END	LABEL (VER. 6.06 27MAR91:10:03:16)
10	10	North
20	20	Central
30	30	South

INFORMAT NAME: \$UPRFMT LENGTH: 2 NUMBER OF VALUES: 3 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 2 FUZZ: STD		
START	END	LABEL (VER. 6.06 27MAR91:10:03:16)
05	05	3400
10	10	3500
15	15	4500

Figure 8. Output from the PROC FORMAT in Program 4

SUMMARY

In summary, all formats are stored. The question is where. If no specific action is taken on the part of the programmer, formats are stored in the FORMATS catalog of SAS's WORK library. On the other hand if a specific SAS library file has been associated with the libref, LIBRARY, and the LIBRARY=LIBRARY option has been specified in the PROC FORMAT statement, the programmer has control over where formats are stored.

For the use of formats whose value statement content is subject to change, the CNTLIN= option provides for the dynamic construction of PROC FORMATS.

Together, both of the above points provide advantages of speed, flexibility, maintainability, and efficiency.

Trademark Notice

*SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.