

FORMAT FESTIVAL – An Introduction to SAS Formats and Informats

David Maddox, Bellsouth Telecommunications, Inc.

ABSTRACT

FORMATS provide an instruction or template that SAS uses to output data values. They can be used to control the written appearance, or, in some cases, to group data values for analysis. INFORMATS, conversely, provide an instruction or template for reading data values. Both constructs add some powerful capabilities to the software, and these capabilities are available to novice and advanced SAS programmers alike. This workshop will focus on:

- The various types of FORMATS and INFORMATS
- Uses of FORMATS and INFORMATS
- Creating your own FORMATS and INFORMATS

In addition, we will look at some 'not so obvious' uses that are sure to make life easier in certain cases. We'll also take a look at Version 9 enhancements. This workshop is primarily for those just beginning to use SAS, but intermediate programmers may find it useful as well.

INTRODUCTION

FORMATS and INFORMATS are powerful components of the SAS language. As best I can tell, these good friends have been around since the early days of SAS. In fact, much of the material discussed in this paper can be found in Version 5 or Version 6 documentation. Newer versions of SAS just expand on the basics. You can bet that for most any new release of SAS that there will be a list of new FORMATS and INFORMATS. In this paper, we will examine FORMATS and INFORMATS from a general perspective and hopefully answer some of the questions of 'What, Where, How, When, and Why'. So, let's "Begin at the Beginning" by looking first at INFORMATS.

INFORMATS

One of the first tasks of data processing is getting data into a machine format so that it can be manipulated. In SAS, INFORMATS can help us do just that. INFORMATS are an instruction or template for reading data values. They are particularly useful in handling non-standard values. When we see a number written with embedded commas, we know that it is a number with commas added to enhance readability. But, a computer would interpret that same number as a character string. We could allow the computer to read the number in and store it as a string, but then what if we wanted to later use the number in a calculation or a number of different calculations. INFORMATS can help immensely by allowing us to tell the program how to interpret the data.

INFORMAT TYPES AND CATEGORIES

There are two basic types of INFORMATS – SAS supplied and User Created. The SAS System ships with a large number of INFORMATS that are automatically available to users in both interactive and batch environments. In addition, users have the option of creating their own INFORMATS to accommodate certain non-standard data situations that they may face. INFORMATS also can be characterized as to whether they are for character, numeric or date values. The current SAS documentation lists some additional categories such as DBCS, but most users will not be concerned with these additional categories. The basic form of an INFORMAT depends on its category.

Character INFORMATS have the following form - **\$<INFORMAT Name>w.** where the \$ sign designates the INFORMATS as being for character data. The w, or width dimension, is the total width of the field. Examples of character INFORMATS are:

\$CHAR7. Reads a character field with a width of seven. Does not trim leading or trailing blanks.

\$HEX4. Converts hexadecimal data to character data.

Numeric INFORMATS omit the "\$" sign and include a decimal width. Examples of numeric INFORMATS are:

COMMA7.2. – This INFORMAT would remove embedded commas and \$ signs. It would also convert a left parentheses to a minus sign.

PD4. – Reads packed decimal – a favorite of mainframer's. Two digits are fit into each bit and using a half-byte for a sign. This one's been a lifesaver on numerous occasions. The column width is four.

Date INFORMATS are used to convert date and time information into a SAS date format. And now a word about dates in SAS. Dates are represented as numeric variables. A SAS date is a number which represents the number of days since January 1, 1960. By representing dates in this way, it is easy to do calculations with SAS dates. SAS provides a number of date INFORMATS that can be used to convert the many forms in which dates may occur into a SAS date variable.

It's important to note that all INFORMATS have a 'period' as the final character of the name . This is a distinctive feature of both INFORMATS and FORMATS. Leaving off the period will result in SAS misinterpreting an INFORMAT or FORMAT name and will most assuredly generate errors.

SPECIFYING INFORMATS

There are several ways to specify INFORMATS for use in a program, regardless of whether they are supplied by SAS or created by the user, as we will discuss in the next section:

- INFORMATS can be specified in an INPUT statement.
- They can also be used in an INPUT, INPUTC, AND INPUTN function
- They can be used in an INFORMAT statement in either a DATA Step or PROC Step.
- They can be assigned in an ATTRIB statement in a DATA Step or PROC Step.

We'll see some examples of using an INPUT statement to specify INFORMATS in the examples below.

CREATING USER DEFINED INFORMATS

Although SAS provides a large number of INFORMATS for general use, users may find it necessary to create their own for specific purposes. INFORMATS can be created with PROC FORMAT. In the following example, the INFORMAT \$ganpa. is created and used to read an NPA (area code). It translates the three digit string to an NPA name. The variable is stored as the name in place of the original string.

```
proc format;
    invalue $ganpa '229'='Albany'
                  '706','762'='Athens'
                  '404','470','678','770'='Atlanta'
                  '478'='Macon'
                  '912'='Savannah';

run;
data ex1;
length npa $8;
input npa $ganpa3. nxx $ 4-6 line $ 7-10 revenue 12.2;
cards;
404423AAAA50.0
229543XXXX75.0
912333BBBB25.0
912767CCCC31.75
678221DDDD46.0
478332EEEE34.5
770681FFFF21.7
706728GGGG45.8
229476HHHH31.2
;
run;
proc print data=ex1;
run;
```

Obs	npa	nxx	line	revenue
1	Atlanta	423	AAAA	50.00
2	Albany	543	XXXX	75.00
3	Savannah	333	BBBB	25.00
4	Savannah	767	CCCC	31.75
5	Atlanta	221	DDDD	46.00
6	Macon	332	EEEE	34.50
7	Atlanta	681	FFFF	21.70
8	Athens	728	GGGG	45.80
9	Albany	476	HHHH	31.20

You will notice that in the EX1 data set that the length of the character variable 'npa' has been increased to '8' in order to accommodate the name rather than the three-digit character string. In this particular example, the use of the INFORMAT in the INPUT statement provides only a temporary association for the life of the DATA STEP. Declare the INFORMAT in an INFORMAT or ATTRIB statement in order to achieve a permanent association. The following output from a PROC CONTENTS includes a permanently associated 'npa' variable and the \$ganpa. INFORMAT.

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Informat
3	line	Char	4	
1	npa	Char	8	\$GANPA8.
2	nxx	Char	3	
4	revenue	Num	8	

When a program containing user defined INFORMATS is executed, the INFORMATS must be made available in one of two ways. Either include the PROC FORMAT code that creates the INFORMAT in the program or create a permanent INFORMAT using PROC FORMAT. For example, to create a permanent INFORMAT \$ganpa., then assign a LIBNAME to a specific library where a FORMATS catalog is located. Then, specify the option in the PROC FORMAT step to reference the LIBRARY LIBNAME.

```
proc format library=library;
    invalue $ganpa '229'='Albany'
                  '706','762'='Athens'
                  '404','470','678','770'='Atlanta'
                  '478'='Macon'
                  '912'='Savannah';

run;

proc format library=library fmtlib;
select @$ganpa;
run;
```

The FMTLIB produces the following information about the \$ganpa. INFORMAT. This option can be used to check the user created INFORMAT for accuracy and completeness.

```
-----
|          INFORMAT NAME: @$GANPA  LENGTH:      8  NUMBER OF VALUES:      9
|  MIN LENGTH:      1  MAX LENGTH:  40  DEFAULT LENGTH  8  FUZZ:      0
|-----|-----|-----|
| START          | END          | INVALUE (VER. V7|V8  04JUL2006:18:25:37) |
|-----+-----+-----|
| 229            | 229          | Albany
| 404            | 404          | Atlanta
| 470            | 470          | Atlanta
| 478            | 478          | Macon
| 678            | 678          | Atlanta
| 706            | 706          | Athens
| 762            | 762          | Athens
| 770            | 770          | Atlanta
| 912            | 912          | Savannah
|-----|-----|-----|
```

As we can see from this very brief look at INFORMATS that they can be very useful in reading in a data in a variety of forms. Now, let move on to FORMATS.

FORMATS

Much of what we have discussed about INFORMATS can be applied to FORMATS as well. A key point to remember is that while INFORMATS are used to convert data, FORMATS are primarily used to display data. A FORMAT is an instruction or template that SAS uses to write data values. An obvious advantage of using FORMATS is that it can greatly reduce the space that might be required to store long character strings. For example, we may elect to store a categorical variable as '1' or '2' both of which represent a much longer character string. But, when we display the data values in a report, we would probably prefer to print the longer, more descriptive strings. FORMATS are particularly important when it comes to the display of dates. As you recall, SAS date values are actually numeric variables. Date FORMATS are generally required to translate the numeric information into the date syntax with which we are familiar – in all of its various forms.

FORMAT TYPES AND CATEGORIES

As with INFORMATS, FORMATS can be categorized as SAS supplied or User Created. FORMATS also fall into the basic categories of character, numeric, or date FORMATS. The form of a FORMAT is also the same as it is for INFORMATS with the all-important requirement that the last character of the name is always a 'period'. While it may be difficult to decide whether \$ASCII7. is a FORMAT or an INFORMAT, it's the usage, or context, that will help us decide.

Specifying FORMATS

As with INFORMATS, there are several ways to specify FORMATS for use in a program, regardless of whether they are supplied by SAS or created by the user. There are also a few new methods as well.

- FORMATS can be specified in an PUT statement.
- They can also be used in an PUT function.
- They may be used in a %SYSFUNC macro function.
- They can be used in an FORMAT statement in either a DATA Step or PROC Step.
- They can be assigned in an ATTRIB statement in a DATA Step or PROC Step.

We'll see some examples of using an PUT statement to specify FORMATS in the examples below:

CREATING USER DEFINED FORMATS

Users can create their own FORMATS with PROC FORMAT. In the following example, a FORMAT is created and used to display the three digit NPA as the NPA name. The difference from the previous example is that we have constructed the data set as following:

```
data ex2;
input npa $ 1-3 nxx $ 4-7 line $ 7-10 revenue 12.2;
cards;
<data>
run;
```

When printed without any formats, the result would be as follows:

Obs	npa	nxx	line	revenue
1	404	423	AAAA	50.00
2	229	543	XXXX	75.00
3	912	333	BBBB	25.00
4	912	767	CCCC	31.75
5	678	221	DDDD	46.00
6	478	332	EEEE	34.50
7	770	681	FFFF	21.70
8	706	728	GGGG	45.80
9	229	476	HHHH	31.20

The three digit NPA is stored in the data set. But, what if we really want to display the NPA name instead. We could construct a FORMAT, which we will call \$ganpaf. and use associate it with the variable npa in a PROC PRINT by using a FORMAT statement.

```
proc format;
  value $ganpaf '229'='Albany'
               '706','762'='Athens'
               '404','470','678','770'='Atlanta'
               '478'='Macon'
               '912'='Savannah';

run;
proc print data=ex2;
  format npa $ganpaf.;
run;
```

Once again, we get a print in which the NPA name is displayed in place of the three digit NPA.

Obs	npa	nxx	line	revenue
1	Atlanta	423	AAAA	50.00
2	Albany	543	XXXX	75.00
3	Savannah	333	BBBB	25.00
4	Savannah	767	CCCC	31.75
5	Atlanta	221	DDDD	46.00
6	Macon	332	EEEE	34.50
7	Atlanta	681	FFFF	21.70
8	Athens	728	GGGG	45.80
9	Albany	476	HHHH	31.20

Whether associations are temporary or permanent follow the same rules as with INFORMATS except that using a PUT statement with a FORMAT provides a temporary association whereas using a FORMAT in a FORMAT statement provides a permanent association. Also, a user defined FORMAT may be made available in a program either by including the PROC FORMAT step that created the FORMAT or by storing it as a permanent FORMAT using the same method as with INFORMATS. FORMATS and INFORMATS reside in the same FORMATS catalog.

SPECIAL USES OF FORMATS

Although FORMATS are primarily designed to display values, they may also be used for some additional chores as well. The most common method of classifying values in many situations is to use a series of conditional statements. However, in SAS, FORMATS can be used to assign character classifications based on numeric values. The following example uses our data set from before. This time we will create a numeric FORMAT called revtype. and will classify the revenue as to whether it would be considered 'LOW', 'AVERAGE', or 'HIGH'. The FORMAT will be associated with the REVENUE variable.

```
proc format;
  value revtype 0 -< 35 = 'LOW'
               36 -< 55 = 'AVERAGE'
               56 -< 80 = 'HIGH';

run;
```

```

data ex3;
set ex2;
revenue1=revenue;
run;

proc print data=ex3;
format revenue revtype.;
var npa nxx line revenue1 revenue;
run;

```

This code will produce the following output:

Obs	npa	nxx	line	revenue1	revenue
1	404	423	AAAA	50.00	AVERAGE
2	229	543	XXXX	75.00	HIGH
3	912	333	BBBB	25.00	LOW
4	912	767	CCCC	31.75	LOW
5	678	221	DDDD	46.00	AVERAGE
6	478	332	EEEE	34.50	LOW
7	770	681	FFFF	21.70	LOW
8	706	728	GGGG	45.80	AVERAGE
9	229	476	HHHH	31.20	LOW

The extra DATA Step and the creation of the REVENUE1 variable allowed us to print both the revenue value and its classification based on the value of REVENUE. This methodology did not require any hard coding of conditional statements in order to assign a classification.

Another special use of FORMATS is to use it as a look-up facility. A simple example would be to check a data set against a list and, in effect, select observations from the data set based on the list. In this case, the "list" is a FORMAT called \$metro.

```

Proc format;
    value $metro '404','470','678','770'='me';
run;

data metro;
set ex2;
if put(npa,$metro.)='me';
run;

proc print data=metro;
title 'Metro Atlanta';
run;

```

< The format is used as the argument of a PUT function to select the customers on the list that had metro numbers.>

The output would be as follows:

Metro Atlanta

Obs	npa	nxx	line	revenue
1	404	423	AAAA	50.0
2	678	221	DDDD	46.0
3	770	681	FFFF	21.7

An alternative to using a FORMAT as a look-up device would be match-merging or PROC SQL joins with the lookup table as a separate table. While hand-coding a PROC FORMAT value statement could be a laborious task, there are ways to load an entire data set to create a FORMAT that includes many entries. The PROC FORMAT documentation includes information on how to use the CNTLIN option.

These are just two special ways to use the FORMAT facility. There are many other uses as well and many of them are documented in SAS Institute and in various user group papers.

VERSION 9 ENHANCEMENTS

Most version releases include new SAS supplied INFORMATS and INFORMATS, and Version 9 is no exception. In addition, the maximum length for character FORMAT names is now increased to 31, and the maximum length for numeric FORMAT names is increased to 32. As for INFORMATS, the maximum length for character INFORMAT names is now 30, and the maximum length for numeric INFORMAT names is now 31. These new limits make it possible for a user to assign more descriptive names.

CONCLUSION

This paper provides a very brief look at FORMATS and INFORMATS. Users of any language love to write elegant, concise code that is easy to manipulate and change. FORMATS and INFORMATS help SAS programmers to write some very elegant, concise, and dense code that does a lot with just a few statements. Once you've become accustomed to these tools, they will become a permanent part of your SAS repertoire.

REFERENCES

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

SAS Institute Inc., 1990.

SAS Institute Inc., SAS® Language and Procedures: Usage 2, Version 6, First Edition, Cary, NC: SAS Institute, Inc. 1991.

Delwiche, Lora D., and Susan J. Slaughter. 2003. The Little SAS® Book: A Primer, Third Edition. Cary, NC: SAS Institute, Inc.

SAS OnlineDoc® Version Eight. Copyright © 1999. SAS Institute, Inc., Cary, NC, USA.

SAS System Help for Version 9.1.3. Copyright SAS Institute Inc., Cary, NC, USA.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Maddox
Bellsouth Telecommunications, Inc.
South E3E, 3535 Colonnade Parkway
Birmingham, AL 35243
Work Phone: 205-977-0415
Fax: 205-977-8719
Email: David.Maddox@Bellsouth.com

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