

Paper 88-26

Use Call Symput and %SYSFUNC for More Informative Titles and Footnotes

Lynn Palmer, State of California, Berkeley, California

ABSTRACT

Information available when your program runs can be placed into titles, footnotes or output using Call Symput and starting in Version 7, %SYSFUNC. These features can be very helpful when running reports routinely. Instead of changing dates, counts, or file names in your code, the program can change them for you. This paper will discuss the use of Call Symput and %SYSFUNC for gaining access to generated information and will show examples of using this information.

INTRODUCTION

The title of your report is what your users will see first. For maximum impact, you want to give as much useful information as you can in your titles. Dates and counts can make titles more informative.

SYMPUT is a routine that you CALL inside a data step to produce a global symbolic variable. The variable can not be referenced inside the creating data step. The data step must end with a RUN; statement before you reference the variable in your code. The value of the macro variable is assigned during the execution of the data step. The syntax is:

```
DATA dataset(can be _NULL_);
  CALL SYMPUT(name,value);
RUN;
```

In Version 7 of SAS, %SYSFUNC and %QSYSFUNC became available. What makes these two functions so useful is they can be assigned outside of a data set. A global variable is created that can be placed directly in to the title or footnote line. %QSYSFUNC is required when a symbol is embedded in the data. The syntax is:

```
%SYSFUNC(function(argument(s))<,format>);
```

DATES

Often, the date that SAS® software places in the upper right-hand corner of the pages of your output or your log is enough. In the situation where a different date is needed, you can suppress the default date by placing, NODATE in an OPTIONS statement (OPTIONS NODATE;) before the procedure that prints your report or at the top of your code. You can use CALL SYMPUT to generate a variable with the date that you need.

Using CALL SYMPUT, you can generate an earlier date, the current date, or a later date and format it in any SAS software format you would like. To use CALL SYMPUT to generate a date variable, you can use a null data step or your current data step.

A null data step can be placed anywhere in your program. At the top or just before the title lines will prevent searching through code to find the variable when you want to make changes in your titles.

The date you assign can be a fixed date coded into your program, a date retrieved from your data, a date generated by SAS software and your system, or a future or past date calculated using another date. By using the function, PUT, you format the date. For example,

A date coded :

```
DATA _NULL_;
  CALL SYMPUT('date1','07/01/1998');
RUN;
```

Your date from your data (the last observation in this data step):

```
DATA BB; SET AA;
  CALL SYMPUT('date2', PUT( yourdate,
  MMDDYY6.));
RUN;
```

A generated date:

```
DATA _NULL_;
  CALL SYMPUT('date3',
  PUT(TODAY(),WORDDATE.));
RUN;
```

A calculated date from a generated date:

```
DATA _NULL_;
  CALL SYMPUT('date4',
  PUT(TODAY()+14,WORDDATE.));
RUN;
```

Information presented in titles is the first thing seen on a report, so spacing and formatting are important. With CALL SYMPUT, you can format the date with any of SAS software date formats. You can, also, remove any extra spaces with SAS software functions. The syntax is as follows:

```
CALL SYMPUT(name,function1( function2
(function3(PUT(date), format.))););
```

PUT is need for the date to be formatted. An unformatted SAS date is a number representing days before(-) or since 01/01/1960. An example with spaces removed and the date formatted:

```
CALL SYMPUT('date1',
  TRIM(LEFT(PUT(TODAY()),
  WORDDATE.)));
```

Some of the character functions that you might consider using are LEFT, RIGHT, TRIM, and UPCASE. The some of the more useful date functions are DATE, TODAY, and WEEKDAY. A

few of the formats you can use are: DATEw., DAYw., MMYyww., MONNAMEw., MONYyww., QTRRw., WEEKDATEw., WEEKDATEw., WORDDATEw., YEARw.. Chapter 14 of the SAS Language Manual describes these formats and their results.

Calculated dates consist of a date, an operation, and a number or a variable containing a number. Adding or subtracting a number of days from your date will give you a future date or a past date. If you want your report returned in two weeks with the missing data completed, you add 14 to your system date. The code looks like this:

```
DATA _NULL_;
CALL SYMPUT('date4', PUT(TODAY()+14,
WORDDATE.));
RUN;
```

The variable which CALL SYMPUT creates is referred to by an & and the name. In the example above, the variable name is: date4. You would use &date4 in your title or footnote. When you use this variable in your title or footnote, you need to use double quotes around the text. Double quotes are only require for the title line or footnote in which the macro variable is contained. For example:

```
TITLE1 'USING CALL SYMPUT';
TITLE2 'FOR TITLE DATES';
TITLE3 "REPORT RUN &date1";
FOOTNOTE "DATA DUE &date4";
```

The following report is produced:

```

          USING CALL SYMPUT
          FOR TITLE DATES
REPORT RUN July 1, 1998

OBS    DATE        REPORT
  1     01/28/1998    I
  2     01/29/1998    M

          DATA DUE July 15, 1998
    
```

%SYSFUNC and %QSYSFUNC

There are a couple of ways dates can be handled using %SYSFUNC. You can assign the date to a symbolic variable and use that variable in your title or you can use the function in your title. For example:

```
%LET date1 = %SYSFUNC(TODAY(),WORDDATE.);
TITLE "REPORT RUN ON &date1";
```

Or in the title statement:

```
TITLE "REPORT RUN ON %SYSFUNC (TODAY() ,
WORDDATE.);";
```

Although the above code will give you the date in your title, SAS will leave the space for the maximum length date. To have your dates nicely adjusted to the left, you need to use another %function. To use the symbol, "%", you need to use %QSYSFUNC. An example which left adjusts the date and trims away excess spaces is:

```
%LET date2 =
%QSYSFUNC(LEFT(%QSYSFUNC(TRIM(%QSYSFUNC
(TODAY(),WORDDATE.))));
TITLE "REPORT RUN ON &date2"; or
TITLE "REPORT RUN ON %QSYSFUNC(LEFT
(%QSYSFUNC(TRIM(%QSYSFUNC(TODAY()),
WORDDATE.))));";
```

%SYSFUNC works with alpha/numeric data.

%QSYSFUNC works with data containing special characters and mnemonic operators, such as: & % ' " () + - = ~ ^ ~ ; , blank AND OR NOT EQ NE LE LT GE and GT.

COUNTS

SAS software is great for producing many desired counts in reports. One or more of those counts in your title can make a better presentation of the data in your report. CALL SYMPUT can be used to put a total count of your data into the title. There are two ways to get a total count of records in your data set. The first example is longer, but shows a method that can be used to get other statistics besides totals:

```
DATA AA; INFILE ddname;
INPUT @1 yourdata;
each_rec = 1; *dummy variable;
RUN;
PROC FREQ;
TABLE each_rec/OUT = tcount NOPRINT;
DATA _NULL_; SET tcount; RETAIN
count;
CALL SYMPUT('base',TRIM( LEFT (
count)));
RUN;
TITLE "COUNT: &base RECORDS";
```

The simpler code is:

```
DATA AA; SET BB END=EOF;
IF EOF THEN
CALL SYMPUT('countAA',_N _);
RUN;
TITLE1 "THE RECORDS IN AA =
&countAA";
```

The first example does the same thing as the second. Sometimes the count or statistic that you need is more complex that a total of the records in your data set. The output from the data sets generated from SAS procedures, usually OUT= , gives a variety of information beyond totals to included in titles. Using PROC FREQ to generate a data set, you have access to frequency as well as total. Using a DATA _NULL_ step, you can get other calculated values. CALL SYMPUT will always take the last observation's value unless you set up your code to point to a particular value or record. For example:

```
DATA _NULL_ ; SET tcount;
IF _N_ = 5 THEN CALL SYMPUT( 'case5',
percent);
RUN;
```

This code, using the tcount output data set from your PROC FREQ, will give you the percent of the fifth category from your PROC FREQ table.

Again, the value can be formatted, and leading blanks can be removed using SAS software functions and formats. When using numbers, TRIM and LEFT are recommended. You can also format for dollars amounts, percents, commas, and decimals. Remember to use PUT with formats.

FILES NAMES

Knowing from where the data came for a particular report when you produce many reports can often be challenging, as well always typing the file names into footnotes can be time consuming. With CALL SYMPUT, you can have SAS software put your file names into your footnotes for you. JFCB in an INFILE statement for a sequential file and PROC CONTENT with a DATA _NULL_ step for a SAS data set will allow you to create a macro variable in which the string containing the name of your file resides. JFCB is the SAS system variable that holds

the name of your file.

For a single sequential file:

```
DATA AA; LENGTH myjfcB $200; INFILE
ddname JFCB=myjfcB;
INPUT @1 yourdata;
CALL SYMPUT('myfile',TRIM(
LEFT(SUBSTR(myjfcB,1,40))));
RUN;
TITLE "DATA FROM: &myfile";
FOOTNOTE "DATA FROM: &myfile ON
THE MAINFRAME";
```

This code used with a partitioned data sets file or a SAS library will only give the name of the file, not the name of the member that you are using. If you are reading several files into your program with the same ddname in your JCL, you will only get the first file. The following code will allow you to use two files with one ddname.

```
DATA AA;
LENGTH myjfcB $200
myjfcB1 $200
myjfcB2 $200;
INFILE ddname JFCB=MYJFCB EOV=var1;
INPUT @1 yourdata;
IF var1=0 THEN DO; myjfcB1=myjfcB;
CALL SYMPUT('myfile1', TRIM(
LEFT(SUBSTR(myjfcB1,1,40)))); END;
IF var1=1 THEN DO; myjfcB2=myjfcB;
CALL SYMPUT('myjfcB2', TRIM(
LEFT(SUBSTR(myjfcB2,1,40)))); END;
RUN;
TITLE "DATA FROM: &myfile1 AND
&myfile2";
FOOTNOTE1 "DATA FROM: &myfile1 ON
MAINFRAME";
FOOTNOTE2 "      &myfile2 ON THE
MAINFRAME";
```

You can not use this code for three or more files unless you code a way to reset the EOV variable back to 0 each time you read a file and a way to keep track of the file name variables that you are assigning.

SAS data sets have to be handled differently. If you run a PROC CONTENTS with an OUT= statement, you can get the name of your data set and other information as well. The following is an example of using CALL SYMPUT to place the data set name, date of creation, and date last modified into a title.

```
PROC CONTENTS DATA=ddname._ALL_
OUT=filename NOPRINT;
DATA _NULL_; SET filename;
CALL SYMPUT('mydata', TRIM(LEFT(
LIBNAME)))||'|'||TRIM(LEFT(
MEMNAME));
CALL SYMPUT('mycrdt',TRIM(LEFT( PUT(
CRDATE, DATETIME18.))));
CALL SYMPUT('mymodt',TRIM(LEFT(PUT
(MODATE, DATETIME18.))));
RUN;
TITLE "DATA FROM SAS DATA SET:
&mydata";
FOOTNOTE "&mydata CREATED:
&mycrdt";
FOOTNOTE2 " LAST MODIFIED:
&mymodt";
```

There is other information about your SAS data set generated from a PROC CONTENTS that is available to

you. LIBNAME, MEMNAME, CRDATE, and MODATE are only four of the variables that SAS software places in the OUT= data set.

CONCLUSION

Informative titles can make your reports more noticed and more valued. Remember a few tips when using CALL SYMPUT. Plan to use the macro variable *after* the RUN; statement of the data step in which it is created. Use an & in front of your assigned variable name in the title or footnote text. Use double quotes around the text of the title or footnote lines in which you place your variable.

Use %SYSFUNC and %QSYSFUNC to make information available as global variables without the need to create a data step. Use these variables in titles and footnotes to make your reports more attractive. SAS software, and specifically, CALL SYMPUT and %SYSFUNC can help you have those informative titles without the work of changing code every time you run a report.

REFERENCES

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

SAS® Guide to Macro Processing, Version 6, Second Edition, SAS Institute Inc., Cary, NC, USA, 1990

SAS®OnlineDoc® Version 8, SAS Institute Inc., Cary, NC, USA, 1999.

Bryher, Monique and Landers, K. Larry, 'Taking the Mystery Out of SAS® Macro When Using Call Symput', Proceedings of the Fifth Annual Western Users of SAS® Software Regional Users Group Conference, 1997, Pages 29-34

Carpenter, Arthur L., "Using Macro Functions", Proceedings of 8th Annual Western Users of SAS Software, Inc. Regional Users Group Conference, 2000, Pages 521

TRADEMARKS

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ®indicates USA registration.

IBM is a registered trademark or trademark of International Business Machines Corporation in the USA and other countries. ®indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

E-mail: (work) lpalmer@dhs.ca.gov and
(home) lynnpalmer@home.com

Work phone: (510) 540-3296.