Paper RF-09-2015

%SYSFUNC Is Your Friend

Kaushal Raj Chaudhary[1], Deanna Naomi Schreiber-Gregory[2]

[1]Sanford Research, Sioux Falls, SD, [2]National University, La Jolla, CA

ABSTRACT

SAS® DATA step has a number of functions but macro functions are meager in macro language. %SYSFUNC allows the use of almost all DATA step and user written functions in macro environment, thus bridging the gap between SAS DATA step and macro language. This adds a lot of flexibility for SAS programmers to write programs in SAS macro. In this paper we will introduce %SYSFUNC and present several examples of the uses of it.

INTRODUCTION

Macro functions are functions in SAS macro language. They work with text strings both in macro definitions or open code. For example, %LENGTH returns the length of a string and %SUBSTR produces substring of a character string.

```
%let Macvar=MWSUG2015;
%let len=%length(&Macvar);
%let substring=%substr(&Macvar,1,5);
%put Lenght=&len Substring=&substring;
```

Log output:
Length=8 Substring=MWSUG

There are different types of macro functions.

- Macro character functions (e.g. %SCAN,%INDEX)
- Macro evaluation functions (e.g. %SYSEVAL)
- Macro quoting functions (e.g. %STR, %BQUOTE)

But these functions are limited in number. We need something “more” that can perform wide range of data manipulation in macro language. Here comes %SYSFUNC for this purpose. %SYSFUNC is less understood but versatile SAS macro function which makes possible the use of numerous SAS DATA step function in the macro environment

Syntax:

%SYSFUNC (function (argument(s), <, format>)
%QSYSFUNC (function (argument(s), <, format>)

Function: name of the function to use
Function called can be DATA step functions, user-defined functions by PROC FCMP, but it cannot be macro function.
Argument (s): one or more arguments used by function. It can be macro variable reference or text expression that produces arguments for function.
Format: It is optional, specifies the format to be applied to the result of function. It can be SAS system format or user -defined format created by PROC FORMAT.

%QSYSFUNC does similar functions as %SYSFUNC. Additionally, it masks the special characters in its argument.

%SYSFUNC virtually supports all DATA step functions with some exceptions listed below. Input and Put are not supported in %SYSFUNC. Instead INPUTN, INPUTC, and PUTN, PUTC are used for character or numeric formats, respectively.

DIF	ALLCOMB	ALLPERM
IORCMDG	DIM	HBOUND
LBOUND	INPUT	LAG
LEXPERK	LEXCOMB	LEXCOMBI

LEXPERM	MISSING	PUT
RESOLVE	SYMGET	All Variable Information Functions

SOME EXAMPLES OF THE USES OF %SYSFUNC

We will see some examples of %SYSFUNC in this section.

CHECKING THE EXISTENCE OF A DATA SET

```
%put %sysfunc(exist(sashelp.class));
```

Log output:
1

In the program above %SYSFUNC executes the EXIST function to check the existence of a SAS data set. It returns 1 (if the data set exists) or 0 (if the data set does not exist). SASHELP.CLASS data set exists in SAS system; hence it outputs 1.

GETTING DESCRIPTOR INFORMATION OF A DATA SET

The ATTRC and ATTRN functions can be used to get descriptor information of a SAS data set. They return the value of numeric and character attributes of a data set, respectively. These functions work in conjunction with the OPEN and CLOSE functions. The OPEN function opens the data set and returns a unique numeric data set identifier. The CLOSE function closes the data set.

Some character and numeric attributes of a data set include.

Numeric attributes:

ANY - specifies whether the data set has observations or variables.
CRDTE - specifies the date that the data set was created.
INDEX - specifies whether the data set supports indexing.
MODTE - specifies the last date and time that the data set was modified.
NVARS - specifies the number of variables in the data set.
NOBS - specifies the number of observations in the data set

Character attributes:

LIB - returns the libref of the SAS library in which the data set resides
MEM - returns the SAS data set name.
LABEL - returns the label assigned to the data set.
TYPE - returns the SAS data set type.
MTYPE - returns the SAS library member type.
MODE - returns the mode in which the SAS data set was opened.

Macro “datainfo” below returns the libref of SAS library, SAS data set name, and number of observations and variables of given a data set (SASHELP.CLASS) if the data set can be opened.

```
%macro datainfo(ds);
   %local lib name nvars nobs;
   %let dsid = %sysfunc(open(&ds,i));
   %if &dsid=1 %then
      %do;
            %let lib=%sysfunc(attrc(&dsid,LIB));
            %let name=%sysfunc(attrc(&dsid,MEM));
           %let nobs =%sysfunc(attrn(&dsid,NOBS));
           %let nvars=%sysfunc(attrn(&dsid,NVARS));
   %let rc = %sysfunc(close(&dsid));
         %put &lib &name &nobs &nvars;
      %end;
   %else
      %put %sysfunc(sysmsg());
%mend datainfo;
%datainfo(sashelp.class)
```

GETTING VARIABLE INFORMATION OF A DATA SET

There are some DATA step functions which can get the variable information of a SAS data set. For example, VARNAME and VARTYPE return the name and type (numeric or character) of a variable, respectively.

Other functions are:

VARFMT - Returns a variable format.
VARINFMT- Returns a variable informat.
VARNUM - Returns a variable position.

```
%let dsid=%sysfunc(open(sashelp.class,i));
%let VariableName=%sysfunc(varname(&dsid,4));
%let VariableType=%sysfunc(vartype(&dsid,4));
%put VariableName=&VariableName VariableType=&VariableType;
```

Log output:
VariableName=Height VariableType=N

"Height" is the fourth variable of SASHELP.CLASS data set and it is numeric.

WORKING WITH EXTERNAL FILES AND DIRECTORY

%SYSFUNC can be used with DATA step functions to work with external files and directory outside of DATA step. The FILEEXIST function checks the existence of an external file and returns 1 or 0 depending on the existence of external file. It returns 1 if the file exists or 0 if the file does not exist.

```
%put %sysfunc(fileexist(C:\Users\chaudhak\Desktop\SYSFUNC.docx));
```

Log output:
1

Some other functions include:

Fopen - Opens an external file and returns a file identifier value.
Dopen - Opens a directory and returns a directory identifier value.
Libname - Assigns or deassigns a libref for a SAS library.
Pathname - Returns the physical name of an external file or a SAS library.

Please see the SAS documentation for more lists of functions.

GETTING SAS SYSTEM OPTIONS

%SYSFUNC executes the GETOPTION function to retrieve the value of a SAS system option.

```
%put %sysfunc(getoption(ORIENTATION));
```

Log output:
Portrait

SAS uses portrait orientation of paper when printing to a printer.

USER DEFINED FUNCTION AND FORMAT

SAS has capability of creating user-defined function using FCMP procedure. %SYSFUNC can execute those functions. Here user-defined function F2C converts temperature from Fahrenheit to Celsius.

```
proc fcmp outlib=work.Myfun.demo;
  function f2c(f);
    c=(f-32)/1.8;
       return(c);
  endsub;
run;
options cmplib=work.Myfun;
```

```
%let ctemp=%sysfunc(f2c(77));
%put &ctemp;
```

Log output:
25

We can also use user-defined format created from PROC FORMAT in addition to SAS supplied formats in %SYSFUNC. In example below we are creating a numeric format “BMI” to format body mass index (BMI) values. %SYSFUNC uses the PUTN function to return the formatted value. A person with BMI value of 35 would be obese.

```
proc format;
  value BMI
       low-18.5 ="Underweight"
       18.5-24.9="Normal weight"
       25-29.9 ="Overweight"
       30-high ="Obese";
 run;
%let BMI=35;
%let rc=%sysfunc(putn(&BMI,BMI.));
%put &rc;
```

Log output:
Obese

USING MACRO VARIABLE IN DATA STEP

%SYSFUNC enables the use of macro variables in a DATA step to create variables. In example below new variable “var3” has been created by performing function on macro variables &var1 and &var2.

```
%let var1=1;
%let var2=2;
data maxvar;
   var3=%sysfunc(max(&var1,&var2));
run;
```

FORMATTING MACRO VARIABLE

%SYSFUNC eliminates the need of an additional DATA step to format a macro variable. A DATA step can be used to produce formatted value of macro variable as below.

```
data _null_;
  today = put(date(),worddate18.);
  call symput('Date_',today);
run;
title "Assignment 1 &Date_";
```

It is just one step process in %SYSFUNC.

```
title "Assignment 1 %sysfunc(date(),worddate18.)";
```

FLOATING POINT NUMBER GENERATION

Mathematical operations in macro facility return only integer values. %SYSFUNC returns floating point number if the function referenced by %SYSFUNC supports floating point numbers.

```
%put %sysfunc(sqrt(27),3.1);
```

Log output:
5.2

The output value has been formatted to one decimal point.

QUOTING %SYSFUNC

%QSYSFUNC is macro quoting counterpart of %SYSFUNC. It substitutes %SYSFUNC when there is a need to quote the return value from a function. In example below referencing the TODAY function by %SYSFUNC causes an error. Switching to %QSYSFUNC removes the meaning of special characters in formatted date, thus avoiding the error.

```
%let date=%sysfunc(left(%qsysfunc(today(),worddate.)));
%put &date;
```

Log output:
September 16, 2015

NESTING %SYSFUNC CALLS

There are situations where nesting of functions becomes necessary for data manipulation. Functions in %SYSFUNC, however, cannot be nested, but %SYSFUNC call can be nested. For example, two %SYSFUNC has been called with their respective function arguments compress and round to clean the value of macro variable “macvar”.

```
%let macvar= 8 23.12;
%let max=%sysfunc(round(%sysfunc(compress(&macvar))));
%put &max;
```

Log output:
823

CONCLUSION

This paper introduced %SYSFUNC and presented few examples of uses of it. It has really widened the horizon of data manipulation capability in SAS macro environment, thus making the life of SAS programmers easier.

REFERENCES

Yindra, C. (1998, March). *%SYSFUNC – The Brave New Macro World.* Paper presented at SAS® SUGI23, Nashville, TN. in the Advanced Tutorials Section.

Moriak, C. (2002, September). *%SYSFUNC: A Macro Variable Can't Function Without It.* Paper presented at SAS® NESUG Conference (15th), Buffalo, NY. in the Posters Section.

Shannon, D. (2009, October). *Get Funky with %SYSFUNC.* Presented at PHUSE 2009, Switzerland.

Field, A., & Miles, J. (2012). *Discovering Statistics Using SAS*® , Thousand Oaks, CA: Sage Publications.

SAS® Institute Inc. 2014. SAS /STAT® 9.4 *User's Guide*. Cary, NC: SAS® Institute Inc.

SAS® Institute Inc. 2014. *%SYSFUNC and QSYSFUNC Functions.*. SAS® 9.2 Macro Language: Reference. Cary, NC: SAS® Institute Inc.

CONTACT INFORMATION

Your comments, questions, and suggestions are valued and encouraged. Contact the authors at:

Kaushal Raj Chaudhary
Sanford Research
Sioux Falls, SD
Email: kaushal.chaudhary@SanfordHealth.org

Deanna Naomi Schreiber-Gregory
National University
La Jolla, CA / Moorhead, MN
E-mail: d.n.schreibergregory@gmail.com