%SYSFUNC - The Brave New Macro World

Chris Yindra, C. Y. Training Associates

ABSTRACT

The new macro function %SYSFUNC (SAS® rel 6.12) allows access by the macro processor to most data step functions and several SCL functions. Solutions to many common macro tasks can now be simpler and more efficient. In addition, many of the SCL functions allow the programmer to manipulate external files directly from the macro processor without forcing step boundary conditions with data steps. %SYSFUNC also allows the macro processor to do floating point arithmetic.

Note: These features are also supported in the data step with the SYSFUNC function.

INTRODUCTION

This paper will demonstrate several uses of the new macro function %SYSFUNC including checking for the existence of data sets and external files, retrieving information about data sets, and manipulating external files. Emphasis will be placed on the SCL functions available to %SYSFUNC. This paper is intended for those with macro programming experience.

The examples in this paper were run using SAS rel 6.12 on WINDOWS® 95.

FUNCTIONS AVAILABLE TO %SYSFUNC

All data step functions are available EXCLUDING:

DIF	INPUT	PUT	DIM	LAG	RESOLVE
HBOUND		LBOUND		SYMGET	

Instead of INPUT and PUT you can use INPUTC, INPUTN and PUTC, PUTN for character or numeric formats, respectively.

Note: while values returned by macro functions are not limited to the current maximum length of data step variables of 200 characters, the values returned by SAS functions are limited to 200 characters.

The following SCL functions are available:

ATTRC	FDELETE	FRLEN	SYSTEM
ATTRN	FETCH	FSEP	VARFMT
CEXIST	FETCHOBS	FWRITE	VARINFMT
CLOSE	FEXIST	GETOPTION	VARLABEL
CUROBS	FGET	GETVARC	VARLEN
DCLOSE	FILEEXIST	GETVARN	VARNAME
DINFO	FILENAME	LIBNAME	VARNUM
DNUM	FILEREF	LIBREF	VARTYPE
DOPEN	FINFO	MOPEN	
DOPTNAME	FNOTE	NOTE	
DOPTNUM	FOPEN	OPEN	
DREAD	FOPTNAME	PATHNAME	
DROPNOTE	FOPTNUM	POINT	
DSNAME	FPOINT	REWIND	
EXIST	FPOS	SYSMSG	
FAPPEND	FPUT	SYSRC	
FCLOSE	FREAD		
FCOL	FREWIND		

Note: The SYSMSG function allows the programmer to return system messages to the LOG.

REVIEW OF MULTIPLE AMPERSANDS

Multiple ampersands can be used to allow the value of a macro variable to become another macro variable reference. The macro variable reference will be re-scanned until the macro variable is resolved.

The following demonstrates how macro variables with multiple ampersands are resolved.

Symbol Table

Macro Variable Name	Macro Variable Value
A	FREIGHT
B	PASSENGER
C	SPECIAL
CODE	A

Resolving a macro variable:

1. &CODE $\xrightarrow{\text{1st scan}}$ A

2. &&CODE $\xrightarrow{\text{1st scan}}$ &CODE $\xrightarrow{\text{2nd scan}}$ A

$$\begin{tikzcd} \text{[\&\&] CODE} \arrow[d] \\ \text{[\&][CODE]} \end{tikzcd}$$

On the first scan - && resolves to &, CODE held as a token

$$\begin{tikzcd}
\text{3. \&\&\&CODE} \arrow[r] \arrow[d, dash, shift left] \arrow[d, dash, shift right] & \text{\&A} \arrow[r] & \text{FREIGHT} \\
\text{[\&\&][\&CODE]} \arrow[d, shift left] \arrow[d, shift right] & & \\
\text{[\&][A]} & &
\end{tikzcd}$$

On the first scan -
&& resolves to &, &code to A.

CHANGING THE FORMAT OF A MACRO VARIABLE

The format of macro variables can be changed with the INPUTN, INPUTC or PUTN, PUTC functions. To change a macro variable using a numeric informat use the INPUTN function. To change a macro variable using a character format, use the PUTC function.

Syntax:
```
val = %SYSFUNC(INPUTC(char val, informat));
val = %SYSFUNC(INPUTN(num val, informat));

val = %SYSFUNC(PUTC(char val, format));
val = %SYSFUNC(PUTC(num val, format));
```

Example 1

%SYSFUNC allows us to convert a macro variable using a format without having to resort to a data step.

This example converts a macro variable date string into a macro variable containing the SAS date representation using an existing SAS format.

```
%LET MYDATE = 971006;

%PUT ORIGINAL VALUE: &MYDATE;

%MACRO CHNGFMT(INVAR,INFMT);
  %LET &INVAR =
        %SYSFUNC(INPUTN(&&&INVAR,&INFMT));
%MEND;

%CHNGFMT(MYDATE,YYMMDD6.);

%PUT SAS DATA VALUE: &MYDATE;
```

Result:

```
ORIGINAL VALUE: 971006
SAS DATA VALUE: 13793
```

Example 2

This example converts a macro variable using a user format.

```
PROC FORMAT;
  VALUE STATEFMT 1 = 'CONNECTICUT'
                 2 = 'MARYLAND'
                 3 = 'NEW YORK';
RUN;

%LET STATEVAR = 2;

%PUT ORIGINAL VALUE: &STATEVAR;

%MACRO CHNGVAL(INVAR,FMT);
   %LET &INVAR =
          %SYSFUNC(PUTN(&&&INVAR,&FMT));
%MEND;

%CHNGVAL(STATEVAR,STATEFMT.);

%PUT NEW VALUE: &STATEVAR;
```

Result:

```
ORIGINAL VALUE: 2
NEW VALUE: MARYLAND
```

READING SAS DATA SETS

There are many functions that allow you to access and manipulate SAS data sets with %SYSFUNC. Many of these features were previously available in the macro language, however many often took a round about approach. While %SYSFUNC allows you to access data set observations, I'll focus on accessing data set descriptor information.

Data set functions discussed in this paper:

OPEN -	Opens a SAS data set.
CLOSE -	Closes a SAS data set.
EXISTS -	Checks for the existence of a SAS data set.
LIBNAME -	Assigns a libref.
LIBREF -	Verifies that a libref has been assigned.
ATTRC -	Returns the value of character attributes of a data set.
ATTRN -	Returns the value of numeric attributes of a data set.
VARFMT -	Returns a variables format.
VARNUM -	Returns a variables position.
VARTYPE -	Returns a variables type.

Some of these functions return values while others return a return code. The value of the return code is function dependent. A typical variable name to pass a return code into is RC although this is not required. The standard syntax for a function that returns a return code is:

```
rc =  %SYSFUNC(functionname(argument));
```

Note: While accessing data sets, the SYSMSG() function can be used to return error messages to the log.

Example 3

A common task for a macro programmer is to generate a report whether or not a data set exists. This is useful when a user may attempt to report off of a data set that has not been created.

You can accomplish this in a macro with the EXISTS function.

Syntax
```
 rc = %SYSFUNC(EXISTS, data set name);
```

The values of the return code are:

0 - The data set does not exist
1 - The data set does exist

```
%MACRO CHECKIT(DSN);
   %IF %SYSFUNC(EXIST(&DSN)) = 1 %THEN %DO;
      PROC CONTENTS DATA=&DSN;
   %END;
   %ELSE %DO;
       DATA _NULL_;
       FILE PRINT;
       PUT "THE DATASET &DSN DOES NOT EXIST";
   %END;
      RUN;
%MEND;

%CHECKIT(CYLIB.JUNK);
```

Result:

```
THE DATASET CYLIB.JUNK DOES NOT EXIST
```

Many attributes of a data set besides its existence can be retrieved by the %SYSFUNC. For instance, the ATTRN function returns the value of numeric attributes of a data set. Some of the attributes are:

CRDTE	- Creation date
MODTE	- Modification date
NOBS	- Number of physical observations (includes observations marked for deletion.
NLOBS	- Number of logical observations (excludes observations marked for deletion.
NVARS	- Number of variables in the data set.

The ATTRN function requires that the data set has been opened with the OPEN function; The OPEN function returns a unique numeric identifier for a data set. Many functions use this assigned identifier as an argument instead of the data set name.

Any data set that is opened with the OPEN function should be closed with the CLOSE function. The close function returns a return code. A non zero return code on the on the CLOSE function is an error condition.

Syntax
```
%LET dsid = %SYSFUNC(OPEN(data set name));
   %IF &dsid = 0 %THEN
        %PUT %SYSFUNC(SYSMSG());
%LET nvar = %SYSFUNC(ATTRN(&dsid,attribute);
%LET rc = %SYSFUNC(CLOSE(&dsid));
    %IF &rc NE 0 %THEN
        %PUT %SYSFUNC(SYSMSG());
```

In this syntax, DSID is the numeric identifier associated with the data set opened with the OPEN function. If this identifier is 0, an error occurred opening the data set. We can use SYSMSG() to write the error to the LOG.

Example 4

Determine if a data set has been modified by FSEDIT or an AF application since its creation (this will not work on data sets that have been recreated).

```
PROC FSEDIT DATA=CYLIB.EMPLOY;
RUN;

%MACRO DSMOD(DSNAME);
  %LET DSID = %SYSFUNC(OPEN(&DSNAME));
  %IF (&DSID = 0) %THEN
                   %PUT MSG = %SYSFUNC(SYSMSG());
  %ELSE %DO;
    %LET CDATE =
                   %SYSFUNC(ATTRN(&DSID,CRDTE));
    %LET MDATE =
                  %SYSFUNC(ATTRN(&DSID,MODTE));
    %LET RC = %SYSFUNC(CLOSE(&DSID));
    %IF &CDATE NE &MDATE %THEN
             %PUT The data set has been modified;
  %END;
%MEND;

%DSMOD(CYLIB.EMPLOY);
```

Result:
The data set has been modified.

Example 5

Another common task is to generate a report whether or not a SAS data set exists or has 0 observations. This I useful when a user defined selection criteria does not select any observations to be written out to the resulting data set and the programmer wants a report returned stating this.

To accomplish this we will start with the macro created in example 3. We also need to check for 0 observations in a data set. The NLOBS argument to the ATRN function will return the number of logical observations in a data set (not counting those marked for deletion).

```
DATA TEMP;
 SET CYLIB.EMPLOY;
 WHERE DEPT = 'ZZ';   * NO SUCH DEPARTMENT;
RUN;

%MACRO GENRPT(DSN);
   %IF %SYSFUNC(EXIST(&DSN)) = 1 %THEN %DO;
     %LET DSID = %SYSFUNC(OPEN(&DSN));
     %LET NUMOBS =
            %SYSFUNC(ATTRN(&DSID,NLOBS));
     %IF &NUMOBS GT 0 %THEN %DO;
       PROC PRINT DATA=&DSN;
     %END;
     %ELSE %DO;
       DATA _NULL_;
        FILE PRINT;
        PUT "NO OBSERVATIONS IN &DSN";
     %END;
     %LET RC = %SYSFUNC(CLOSE(&DSID));
   %END;
   %ELSE %DO;
     DATA _NULL_;
      FILE PRINT;
      PUT "THE DATASET &DSN DOES NOT EXIST";
   %END;
     RUN;
%MEND;

%GENRPT(TEMP);
```

Result:
NO OBSERVATIONS IN TEMP.

The ATTRC function can be used to return a variety of character attributes of a data set. Some of the attributes are:

LABEL	- The label assigned to the data set
SORTEDBY	- The names of the BY variables in order (empty if the data set is not sorted).

As with the ATTRN function, the ATTRC function requires that the data set has been opened with the OPEN function. Data sets that are opened should be closed with the CLOSE function.

Syntax
```
%LET dsid = %SYSFUNC(OPEN(data set name));
%LET cvar = %SYSFUNC(ATTRC(&dsid,attribute);
%LET rc = %SYSFUNC(CLOSE(&dsid));
```

Example 6

Sorting a data set that has previously been sorted according to the same criteria requires redundant overhead. We can use ATTRC to check and see if a data set has already been sorted according to the specified criteria. To accomplish this we provide the BY statement sort string and compare this to the SORTEDBY attribute. Blanks are COMPRESSed out to avoid syntax confusion.

```
PROC SORT DATA=CYLIB.EMPLOY;
 BY DEPT SALARY;
RUN;

%LET SRTSTRNG = DEPT SALARY;

%MACRO ISSORT(DSNAME);
  %LET DSID = %SYSFUNC(OPEN(&DSNAME));
  %IF (&DSID = 0) %THEN
              %PUT MSG = %SYSFUNC(SYSMSG());
  %ELSE %DO;
     %LET SEQ =
          %SYSFUNC(ATTRC(&DSID,SORTEDBY));
     %LET RC = %SYSFUNC(CLOSE(&DSID));
     %IF %SYSFUNC(COMPRESS(&SRTSTRNG)) NE
        %SYSFUNC(COMPRESS(&SEQ)) %THEN %DO;
           PROC SORT DATA=&DSNAME;
                BY &SRTSTRNG;
            RUN;
     %END;
    %ELSE %PUT A sort was not required;
  %END;
%MEND;

%ISSORT(CYLIB.EMPLOY);
```

Result:
A sort was not required.

READING SAS CATALOGS

The previous examples focused on SAS data sets. %SYSFUNC can also read various attributes of SAS catalogs. The CEXISTS function is used to check for the existence of a SAS catalog or catalog entry.

Syntax

```
 rc = %SYSFUNC(CEXIST(catalog entry))
```

Example 7

If a SAS data set is to be moved to a different platform or server, it may be useful to know if that data set uses any user written formats that may need to be copied as well. To accomplish this we can check every variable for any required formats. We then compare these formats with any from the LIBRARY (or other format library) library.

```
LIBNAME LIBRARY 'C:\CYDATA\';

PROC FORMAT LIBRARY = LIBRARY;
  VALUE $DPTFMT 'GIO' = 'GROUP INSURANCE'
                'GPO' = 'GROUP PENSION'
                'IIO' = 'INDIVIDUAL INS'
                'RIO' = 'REINSURANCE';

DATA EMPLOY;
 SET CYLIB.EMPLOY;
 FORMAT DEPT $DPTFMT.;

RUN;

%MACRO CHECKFMT(DSNAME);
  %LET DSID = %SYSFUNC(OPEN(&DSNAME));
  %LET NUMVARS =
%SYSFUNC(ATTRN(&DSID,NVARS));
  %DO I = 1 %TO &NUMVARS;
     %LET FMT = %SYSFUNC(VARFMT(&DSID,&I));
     %IF &FMT NE %THEN %DO;
     %LET TYPE = %SYSFUNC(VARTYPE(&DSID,&I));
     %LET FMT = %SYSFUNC(COMPRESS(&FMT,'$'));
     %LET CATENTRY=
          LIBRARY.FORMATS.&FMT.FORMAT&TYPE;
     %IF %SYSFUNC(CEXIST(&CATENTRY)) %THEN
       %PUT &FMT IS A USER WRITTEN FORMAT
            USED BY &DSNAME;
   %END;
  %END;
  %LET RC = %SYSFUNC(CLOSE(&DSID));
%MEND;

%CHECKFMT(EMPLOY);
```

Result:

```
DPTFMT. IS A USER WRITTEN FORMAT USED BY
EMPLOY.
```

MANIPULATING EXTERNAL FILES

%SYSFUNC also allows you to access and manipulate external files without the use of a data step. Files can be read and modified with the macro processor. In addition we can also access directories.

Values read from external sources are placed in a file buffer. Values in that buffer can be modified and the result can be written back to the file or to a different file. Files manipulated in this fashion are referenced by a unique id assigned with the FOPEN function, not by their fileref or name.

File and directory functions discussed in this paper:

DOPEN	- Opens a directory.
DCLOSE	- Closes a directory.
DNUM	- Number of members in a directory.
DREAD	- Returns the name of a directory member.
FOPEN	- Opens a file.
FCLOSE	- Closes a file.
FILEEXIST	- Checks for the existence of a file.
FILENAME	- Assigns a fileref.
FPUT	- Puts a value to the file buffer
FWRITE	- Writes a value from the file buffer to the file.

Like SAS data sets, all files and directories opened with the DOPEN or FOPEN function should be closed with the DCLOSE or FCLOSE function.

Example 8

Previously we tested for the existence of a SAS data set. %SYSFUNC also allows us to test for the existence of an external file. To accomplish this we use the FILEEXIST function.

Syntax

```
   rc = %SYSFUNC(FILEEXIST(file name));
```

The values of the return code are:

0 - The file does not exist
1 - The file does exist

```
%MACRO FEXISTS(FILENM);
  %LET FILEFLAG =
             %SYSFUNC(FILEEXIST(&FILENM));
  %IF (&FILEFLAG = 1) %THEN
              %PUT THE FILE &FILENM EXISTS;
  %ELSE %PUT THE FILE &FILENM DOES NOT EXIST;
%MEND;

%FEXISTS(C:\JUNK\XXX.TXT);
```

Result

```
THE FILE C:\JUNK\XXX.TXT DOES NOT EXIST
```

Here the return code is returned to the macro variable FILEFLAG. We could have also used RC, as was used in the previous examples.

Information about a file including the record length and record type can be accessed with the FINFO function. FINFO requires that a file first be opened with the FOPEN function.

Syntax

```
   file id = %SYSFUNC(FOPEN(file ref));
   value = %SYSFUNC(FINFO(file id, info item));
   rc = %SYSFUNC(FCLOSE(file id));
```

Example 9

We may find it useful to know the logical record length of an existing file. To accomplish this we use the FINFO function with the LRECL attribute.

```
FILENAME XFILE 'C:\JUNK\MYFILE.TXT';

%MACRO FILELEN(FILEREF);
  %LET FID = %SYSFUNC(FOPEN(&FILEREF));
  %LET VALUE = %SYSFUNC(FINFO(&FID,LRECL));
  %PUT LRECL = &VALUE;
  %LET FID = %SYSFUNC(FCLOSE(&FID));
%MEND;

%FILELEN(XFILE);
```

Returns

```
LRECL = 256
```

Example 10

Create a file that contains the names of all files in a sub-directory.

This requires that we first open a directory and retrieve the number of files in the directory (DOPEN and DNUM). At that point we can read the names of the files (DREAD). We then need to open a file and put the name of each file to the file buffer (FOPEN, FPUT). Once the name of a file is n the file buffer, we can write the name to the file (FWRITE). Finally, we close the directory and the file (DCLOSE, FCLOSE)

Syntax

```
          dir id = %SYSFUNC(DOPEN(file dir ref));
          file id = %SYSFUNC(FOPEN(file ref));
          value = %SYSFUNC(DREAD(file id, item));
          rc = %SYSFUNC(FPUT(file id, value));
          rc = %SYSFUNC(FWRITE(file id));

          rc = %SYSFUNC(FCLOSE(file ref));
          rc = %SYSFUNC(DCLOSE(file dir ref));
```

```
%MACRO GETNAMES(FILEDIR,OUTFILE);

 %LET FREF=X;

 %LET RC=%SYSFUNC(FILENAME(FREF,&FILEDIR));

   %IF &RC NE 0 %THEN
            %PUT %SYSFUNC(SYSMSG());

 %LET DIRID=%SYSFUNC(DOPEN(&FREF));

 %LET NEWFILE = NEWFILE;

 %LET RC =
       %SYSFUNC(FILENAME(NEWFILE,&OUTFILE));
       %IF &RC NE 0 %THEN
              %PUT %SYSFUNC(SYSMSG());
 %LET FILEID = %SYSFUNC(FOPEN(&NEWFILE,A));
          %IF &FILEID EQ 0 %THEN
              %PUT %SYSFUNC(SYSMSG());

 %LET NUMFILES = %SYSFUNC(DNUM(&DIRID));

 %DO I = 1 %TO &NUMFILES;

   %LET FILENM = %SYSFUNC(DREAD(&DIRID,&I));
   %LET RC = %SYSFUNC(FPUT(&FILEID,&FILENM));
   %LET RC = %SYSFUNC(FWRITE(&FILEID));
   %IF &RC NE 0 %THEN %PUT
            %SYSFUNC(SYSMSG());
 %END;

 %LET RC=%SYSFUNC(DCLOSE(&DIRID));
 %LET RC=%SYSFUNC(FCLOSE(&FILEID));
 %LET RC=%SYSFUNC(FILENAME(FREF));
%MEND;

%GETNAMES(C:\SAMPDATA\,C:\JUNK\ALLFILES.TXT);
```

Results in a file containing all item names in the directory.

CONCLUSION

The introduction of %SYSFUNC into the macro language gives the macro programmer a wide range of new tools to access and manipulate SAS libraries and external files.

The macro language's ability to create flexible code based on static, user, and system supplied parameters has been greatly enhanced. This paper has presented a small subset of the capabilities of %SYSFUNC and I encourage the reader to explore additional capabilities. Online documentation for %SYSFUNC on the WINDOWS® operating system can be found by selecting

HELP
ONLINE DOCUMENTATION
WHATS NEW IN RELEASE 6.12
WHATS NEW IN BASE SAS
FUNCTIONS

Questions, comments, and suggestions are welcome at:

Chris Yindra
C. Y. Training Associates, Inc
80 West Mountain Road
Canton Center, CT 06020
800-563-9484
chris@cyassociates.com
http://www.cyassociates.com

REFERENCES

SAS Institute, Inc., SAS release 6.12 Online Documentation.