

# ***THE SAS MACRO LANGUAGE: AN OVERVIEW***

***Thomas J. Winn, Jr.  
Office of the Comptroller of Public Accounts,  
State of Texas***

## ***Why bother learning about macros?***

Have you ever wanted to "package" blocks of SAS code into components which could be invoked with a simple command? This would be desirable, for instance, in repetitive coding situations involving the processing of numerous, similar data sets in identical ways.

Have you ever wanted to write a SAS program which modifies itself while it is running? Perhaps you wanted to write flexible SAS code with the ability to accommodate itself to certain changeable details -- for instance, to check for data requiring special processing, to dynamically generate data-dependent lines in a report, or to communicate particular information between the steps in a program.

Have you ever wanted to be able to build and to utilize a customized "toolbox" of frequently-used programming modules, without having to explicitly list them in each program in which they are used?

For these situations, and for many others, use of the SAS macro language can produce the desired results. The SAS macro language is a powerful tool for producing flexible or repetitive code by using a modularized approach. It is particularly useful in applications development.

## ***What are Macros?***

Macros are stored text which contain entire blocks of SAS code, and which are identified by a name. The stored text can include SAS statements, literals, numbers, macro variables, macro functions, macro expressions, or calls to other macros. Macro variables are used to facilitate symbolic substitution of strings of text, whereas macros can be used to manipulate SAS source statements.

Macro information can be inserted at any point in a SAS program simply by referring to the macro entity by name, preceded by a special character, which distinguishes macro statements from ordinary SAS code. The percent sign (%) and ampersand (&) are "macro triggers." If either of these triggers are followed by a non-blank character, the macro facility takes control over processing. Macro variables are identified using the ampersand, and macros are identified with the percent sign. The macro facility constructs and edits SAS source statements by substituting the currently-defined values of macro variables, and also by replacing macro names with the stored text which is associated with each of them.

## ***The SAS Macro Language is a programming language.***

The macro facility follows instructions which are written using a special language.

The SAS macro language can be thought of as a separate programming language that works in conjunction with the SAS language. It has variables, statements, functions, expressions, and syntax.

### ***Macro Variables.***

Macro variables are used to facilitate symbolic substitution of strings of text. Macro variables are named following the usual rules for SAS names. A macro variable can be referenced by placing an ampersand immediately before the macro variable name and, optionally, by placing a period after the macro variable name. A macro variable reference causes the macro facility to substitute the current value of the macro variable for the reference. There are two types of macro variables: automatic macro variables, which are defined by the SAS supervisor, and those which are defined by the user. Once it has been appropriately defined, a macro variable can be used repeatedly in the SAS job.

Automatic macro variables are created by the SAS Supervisor when the SAS System is invoked. They include information such as the time, day, and date on which the job began executing, and the name of the most recently created SAS data set. Automatic macro variables are available for the duration of a SAS job and can be referenced anywhere in any SAS program.

The %LET statement is the most common way to create user-defined macro variables and to assign values to these variables. The value assigned to a macro variable is treated as a character string by the macro facility, but it isn't necessary to enclose the string in quotes. If quotes are used, the quotes become part of the string. There is no such thing as a numeric macro variable -- in the macro language, everything is stored text. Here is the general syntax

```
%LET macrovariablename = value;
```

The name of a macro variable must conform to the customary rules for names in SAS. Here is a particular example:

```
%LET OPT = N NOOBS UNIFORM;
```

Then, the statement

```
PROC PRINT &OPT;
```

would resolve as (that is, be replaced by)

```
PROC PRINT N NOOBS UNIFORM;
```

### ***Macros.***

Macros are used to manipulate SAS source statements. A macro is defined by enclosing its text between a %MACRO statement and a %MEND statement. The

%MACRO statement can include a parameter list (either positional or keyword type) to define special macro variables that can be referenced within the macro.

Here is the general syntax which is most commonly used:

```
%MACRO macroname(parameters);  
... macrotext ...  
%MEND macroname;
```

Macro variables which are defined within a macro generally cannot be resolved outside of the macro. It is possible to invoke a macro from within another macro. When this happens, the macros are said to be "nested". In general, a macro variable can be resolved in the macro within which it is defined, and in all other macros that are nested within that macro.

### ***Macro Programming Statements and Macro Functions.***

Most macro programming statements appear similarly, and are used in the same way, as corresponding ordinary-SAS programming statements, except that they begin with a percent sign (%). For example, the %IF-%THEN-%ELSE, %DO, %DO %WHILE, %DO %UNTIL, %END, and %GOTO statements are used in precisely the same manner as the familiar IF-THEN-ELSE, DO, DO WHILE, DO UNTIL, END, and GOTO statements. These statements facilitate conditional code generation, iterative processing, and branching within a macro.

The SAS macro language includes several functions for manipulating strings which are similar to character string functions which might be used in a DATA step in ordinary SAS. %LENGTH, %SUBSTR, %INDEX, and %SCAN work just as you might suppose.

Single quotes ( ' ') and double quotes ( " ") don't have the same effect in the SAS macro language as in ordinary SAS. If a macro variable reference is placed within single quotes, it will not resolve (that is, it will be treated as constant text, and no symbolic substitution will occur). However, macro variable references enclosed in double quotes will resolve. The SAS language uses quotes to indicate character constants. However, since quotes are considered part of the text in the SAS macro language, quoting functions take the place of quotes in macro expressions. Quoting functions are very useful for removing the customary syntactical meanings of special characters (like semicolons, unmatched parentheses, apostrophes, operators, and mnemonics) in character strings. Consult the SAS documentation for further information.

### ***A Few Examples of the Use of Macros.***

Here is an example of a macro, named REPET, which uses keyword-type parameters in an iterative loop to generate multiple DATA and PROC steps:

```
%MACRO REPET(FIRST=, LAST=);  
  %LOCAL I;  
  %DO I=&FIRST %TO &LAST;  
    DATA NET&I;  
      INFILE IN&I;  
      INPUT DEPT $ INCOME EXPENSE;
```

```

                                PROFIT = INCOME - EXPENSE;
PROC PRINT;
                                TITLE "REPORT OF INCOME & EXPENSE FOR 19&I";
                                %END;
                                RUN;
                                %MEND REPET;

```

Simply defining a macro does not cause it to execute. Here is how to invoke the previously-defined macro (Notice that the null default values of the parameters are replaced with desired values. Also notice that the statement does not end with a semicolon.):

```
%REPET (FIRST=85, LAST=91)
```

Following is an example in which macro variables are used for the start date of the time period under consideration in a batch reporting job which normally would be run each Monday morning. Before submitting the program, the appropriate date elements are inserted as arguments in the MDY function, named START, below. If the program is run on a Monday, and if the desired time period is the preceding week, then it isn't necessary to do anything to START. This program is easy to maintain and use, since the start date is only entered once at the beginning of the program, and the macro facility automatically substitutes values for two different forms of the date as often as they are required.

```

DATA _NULL_;
    START = MDY(11,05,91);
    %GLOBAL STRTDA1 STRTDA2;
    IF &SYSDAY=Monday THEN DO;
        START2 = &SYSDATE - 7;
        CALL SYMPUT("STRTDA1", LEFT(PUT(START2, WORDDATE.)));
        CALL SYMPUT("STRTDA2", LEFT(PUT(START2, DATE.)));
    END;
    ELSE DO;
        CALL SYMPUT("STRTDA1", LEFT(PUT(START, WORDDATE.)));
        CALL SYMPUT("STRTDA2", LEFT(PUT(START, DATE.)));
    END;

DATA A;
    INFILE IN1;
    INPUT @1 VAR1 PD6. @7 VAR2 $2.
           @9 VAR3 $4. @13 DATEVAR MMDDYY6.;
    IF VAR2 EQ '02';
    IF DATEVAR GE "&STRTDA2"D;
PROC PRINT;
    VAR VAR1 VAR3;
    TITLE "Records Processed Since &STRTDA1";

```

The preceding program makes use of two automatic macro variables, SYSDAY and SYSDATE, which are created when the SAS System is invoked. The SYMPUT routine is used to transfer information from the DATA step to macro variables, during the execution phase of the DATA step.

Quite often in my work, I use SAS to perform the same kind of statistical analysis and/or reporting on datasets which share the same general structure, but which differ according to industry grouping or economic region. Macros are ideal for this situation! Following is an outline of a typical case.

```

%MACRO ANALYS (REGN);
  DATA &REGN.2;
    SET &REGN;
    BY INDGRP;
    .... (SAS programming statements)
  PROC ....;
    .... (SAS programming statements)
    TITLE "Analysis for &REGN Region";
  DATA ....;
    .... (SAS programming statements)
  PROC ....;
  RUN;
%MEND ANALYS;

```

Notice the use of the parameter REGN. Macro parameters are special macro variables which are defined with the macro, and which are assigned a value when the macro is invoked. Also notice the use of a period as a macro variable delimiter in the first DATA step.

Here is how the preceding macro would be called (Notice that each invocation includes a value for the parameter REGN):

```

%ANALYS (PLAINS)
%ANALYS (METROPL)
%ANALYS (EASTEX)
%ANALYS (GULFC)
%ANALYS (CENTRAL)
%ANALYS (BORDER)
%ANALYS (OUTSTA)

```

### ***Macros can be stored in, and accessed from, macro libraries.***

The autocall facility is a way of making one or more libraries of commonly-used SAS macros available, without having to explicitly include them in each SAS program in which they are used. This allows programmers to create macros for customized functions, and to have access to them whenever they are needed. There could be personal macro libraries, or departmental macro libraries for macros to be shared by many persons. (On MVS systems, autocall libraries are partitioned data sets. On a CMS system, an autocall library is a maclib. On a PC, an autocall library is a directory consisting of files with a 'sas' extension.) The macros are stored as members in the library, where the name of the macro is the same as the name of the member. The libraries are concatenated, using the fileref SASAUTOS. Then, the autocall facility will automatically define and execute a stored macro whenever it is called in the SAS session, provided that the MAUTOSOURCE option is in effect.

### ***Where can I find more information about using macros?***

The standard macro references are: Chapter 20 of SAS Language: Reference, Version 6, First Edition, which includes a basic description; and SAS Guide to Macro Processing, Version 6, Second Edition, which contains a more thorough explanation and examples (both are 1990 SAS Institute publications). An alternative presentation of some of this material may be found in Chapters 7 and 12 of Professional SAS Programming Secrets, by Rick Aster and Rhena Seidman (Windcrest Books, 1991).

Several very excellent introductory papers about the SAS macro language have been presented at recent SUGI Conferences, especially:

- \* "Common Misunderstandings About the SAS Macro Facility," by Earl Westerlund (Proceedings of SUGI-13, 1988; pp. 1288-1292),
- \* "The SAS Macro Facility," by Maria Nicholson and Bob Pulgino (Proceedings of SUGI-13, 1988; pp. 1293-1299),
- \* "Teflon and Tang: Forging Generic Macro Tools," by Peter Kretzman (Proceedings of SUGI-13, 1988; pp. 1300-1304),
- \* "Building Macro-based Systems," by Howard Levine (Proceedings of SUGI-14, 1989; pp. 96-102),
- \* "The SAS Macro: An Aid to the User," by Robert E. Johnson (Proceedings of SUGI-15, 1990; pp. 244-250),
- \* "Macro Tricks to Astound the Folks on Thursday Morning: Ten Immediately Useful Macro Techniques," by Bruce M. Tindall and Susan M. O'Connor (Proceedings of SUGI-16, 1991; pp. 189-195),
- \* "SAS Macro Language Features for Applications Development," by Earl R. Westerlund (Proceedings of SUGI-16, 1991; pp. 245-248) and
- \* "An Introduction to the SAS Macro Language," by Gary L. Katsanis (Proceedings of SUGI-16, 1991; pp. 401-407).

Recent SUGI Conference proceedings also contain numerous papers which exemplify various uses of SAS macros.

Also, SAS Institute's training course on the SAS Macro Language is highly recommended. Attendance at this 20 minute tutorial should not be regarded as a substitute for participation in two to three days of genuine training.

### ***Summary.***

The SAS Macro Language is a powerful tool for simplifying repetitive coding, for communicating information between program steps, for generating data-dependent SAS statements, for permitting conditional execution of SAS code, and for dynamically importing certain information from the SAS Supervisor. In macro programming, a modularized approach is used. In this paper we have seen, or referred to, a few of the basic uses. And we also have listed several references for further study.

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