ABSTRACT
The macro facility is a part of base SAS® software that enables you to write dynamic programs. Programs that utilize the macro facility can
• use your data to write the program for you
• pass information between SAS steps
• repeatedly submit SAS statements.

This macro tutorial will demonstrate how to create reports and manage SAS data sets using macro variables in a maintenance-free program.

INTRODUCTION
This paper discusses how you can use macro variables to generate a SAS data set for data entry for monthly data. The data set will have the name of the next month and year based on the current month and year. Also, the program will delete the data set created two months ago that contains last month’s data. For example, if today is 11MAR96, the data set created today is named APR96 and the data set deleted today is named FEB96. The level of SAS software programming experience is beginning to intermediate.

The products discussed in this paper are
• base SAS
• SAS/FSP® software.

This paper is relevant for all operating systems utilizing Release 6.06 or later of SAS software.

OBTAINING DATE INFORMATION
Every month a new data set is created containing scheduling information for the next month. Macro variables make the program dynamic and maintenance free. Functions extract date information. For example, functions can determine

1. today’s date
2. the three-letter abbreviation of a month
3. the year portion of a date
4. the previous month’s or next month’s date.

length thismon $ 3;
1 today = today();
2 thismon = put(today, worddate3.);
3 tyear = year(today);
4 lastmon = put(intnx('month', today, -1), worddate3.);
5 nextmon = put(intnx('month', today, 1), worddate3.);

By assigning today’s date to a variable, the TODAY function will not need to execute multiple times throughout the program. That action saves CPU time.

Manipulating Year Values
The next problems to solve are to
• create a new data set for next year
• delete a data set from the previous year.
The month values are already assigned appropriately. However, you need to assign the year values.
If the NEXTMON is January, then you need to increment the year value for the new data set by one. Otherwise, the year values are the same for this month and next month. You need the year value for the current month later when you copy the current month’s data set descriptor portion to the next month’s so it has the same variables.

if upcase(nextmon) = 'JAN' then yyear = tyear + 1;
else yyear = tyear;

If LASTMON is December, then the year value is not the same as the current year. Therefore, you need to create a new year variable for last year. You will need last year’s year value later when you delete old data sets. However, if last month wasn’t December, the year is the same for this month and last month.

if upcase(lastmon) = 'DEC' then lyear = tyear - 1;
else lyear = tyear;

CREATING MACRO VARIABLES
The date information generated in the DATA step is used to create SAS data set names in subsequent PROC steps. In order to utilize that information in the PROC steps, it must be stored independent of the SAS data set. That storage location is a symbol table. Macro variable values are stored in the symbol table. One way you can assign a value to a macro variable is with the SYMPUT routine.

The first argument in the SYMPUT routine is the macro variable name. The second argument is the text assigned to the macro variable. It can be a
• literal value
• DATA step variable name
• DATA step expression.

call symput('month', nextmon);
call symput('yearmac', put(yyear, 4.4));
call symput('lyearmac', put(yyear, 4.4));
call symput('lmonth', thismon);
call symput('lmonth', lastmon);

The SYMPUT routine obtains the current value of the SAS data set variable from the program data vector (PDV) during execution time and transfers it to the symbol table.

Macro variables store character values only. Therefore, you must convert the numeric year values to character with the PUT function before assigning the values to the macro variables. If you do not convert the numeric values, they are automatically converted using the BEST12. format.

If you use DATA step variable values to assign macro variable values
• the maximum number of characters you can assign to the macro variable is 200
• all leading and trailing blanks are retained and stored as part
of the macro variable value.

**USING THE MACRO VARIABLES**

All of the statements you have seen so far are located in the
DATA step. The DATA step can create SAS data set variables
as well as macro variables. The macro variables that you create
in the DATA step with CALL SYMPUT can be used later in the program in

- DATA steps
- PROC steps
- global statements.

The macro processor resolves macro variables before the SAS
statements are compiled. The macro processor receives names
that begin with a macro trigger. An ampersand (&) is a macro
trigger.

To create a new SAS data set with the name of the next month
and year (for example, APR96), use the MONTH and YEARMAC
macro variables.

PROC FSEEDIT enables you to create a new SAS data set with the
same attributes as an existing SAS data set using the options
NEW= and DATA=.

Use the TMONTH and TYEARMAC macro variables to reference
the SAS data set with this month's name (for example, MAR96).

```
proc fseedit new=data.APR96
   like=data.MAR96;
```

DATA is an arbitrary library reference name for a permanent SAS
data library.

You can reference multiple consecutive macro variables that are
not separated by blanks. The macro processor resolves each
macro trigger independently.

After the macro processor resolves the macro variables, it sends
the code to the input stack. The input stack is an area of
memory that the SAS Supervisor reads before routing text to the
appropriate compiler.

The PROC FSEEDIT statement that is shown is the result of the
resolved macro variables in March 1996.

```
proc fseedit new=data.APR96
   like=data.MAR96;
```

You can use PROC DATASETS and the macro variables
LMONTH and LYEARMAC to delete last month's SAS data set.

```
proc datasets lib=data;
delete &lmmonth&lyearmac;
```

This is the statement that is sent back to the input stack in March
1996.

```
proc datasets lib=data;
delete FEB96;
```

**THE FINAL PROGRAM**

Here's the complete program discussed in this paper:

```
libname data '/users/mysaslib';
data _null_;
length thismon $ 3;
today = today(1);
thismon = put(today, worddate3.);
tyear = year(today);
lastmon = put(intnx('mon', today, -1), worddate3.);
nextmon = put(intnx('mon', today, 1), worddate3.);
if upcase(lastmon) = 'JAN' then year = tyear + 1;
else year = tyear;
if upcase(lastmon) = 'DEC' then lyear = tyear - 1;
else lyear = tyear;
call symput('month', th.smon);
call symput('yearmac', put(year, 4.));
call symput('tyearmac', put(year, 4.));
call symput('lyearmac', put(year, 4.));
call symput('tmmonth', t.month);
call symput('lmonth', l.month);
proc fseedit newdata.&month&yearmac
   like=data.&month&yearmac;
proc datasets lib=data;
delete &lmmonth&lyearmac;
runt;
quit;
```

**CONCLUSION**

You can create and use macro variables to make your programs
data-dependent and maintenance free. The data-dependent
program discussed in this paper will process without the
programmer providing dates. The program uses functions and
macro variables to do all the work.

For more information on macro variables and the macro
processor, please refer to either SAS® Macro Language Course
Notes (order #P58213) or SAS® Guide to Macro Processing,

SAS and SAS/FSI® are registered trademarks or trademarks of
SAS Institute Inc. in the USA and other countries. © indicates
USA registration.

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