Although the term MACRO does sound a bit forbidding, it's bark is certainly worse than its bite. Using MACROS eliminates the need for repetitious coding and generally makes maintaining code much easier. This paper will describe what a MACRO is, how to identify when using a MACRO would be advantageous, how to define a MACRO, how to call a MACRO, and define and describe MACRO variables both SAS-defined and User-defined.

**MACRO Defined**

A $AS Macro is a collection of SAS statements stored either temporarily or permanently under a MACRO name that can be repeatedly invoked during the execution of a SAS program.

**When MACROS should be used**

Anytime you find yourself coding virtually the same syntax with minor variations in the same program, chances are that the code could be greatly reduced by defining it as a MACRO. In addition, by reducing the amount of code, maintaining the program is simplified. Consider the following example:

**Example A:**

```
DATA BASE.Y92;
    INFILE IN92;
    INPUT@1 CNUM 5.
        @6 CNAME $CHAR20.
        @26 PDC1 15.
        @41 PDC2 15.
        @56 PDC3 15.;
DATA BASE.Y91;
    INFILE IN91;
    INPUT@1 CNUM 5.
        @6 CNAME $CHAR20.
        @26 PDC1 15.
        @41 PDC2 15.
        @56 PDC3 15.;
DATA BASE.Y90;
    INFILE IN90;
    INPUT@1 CNUM 5.
        @6 CNAME $CHAR20.
        @26 PDC1 15.
        @41 PDC2 15.
        @56 PDC3 15.;
```

The program in example A consists of three data steps. Each data step reads in observations from a flat file and creates a permanent SAS data set with five variables. The permanent $AS data sets are written to the same ddname and have the data year as part of its member name. The ddname on the infile statement also uses the data year as part of its name. After creating the permanent SAS data sets, we want to print out each observation in a separate SAS program. The program follows:

**Example B:**

```
PROC PRINT DATA=BASE.Y92 LABEL SPLIT=*
    VAR CNUM CNAME PDC1-PDC3;
    LABEL CNUM='COMPANY*NUMBER';
    LABEL CNAME='COMPANY*NAME';
    LABEL PDC1='DIRECT*PREMIUMS*WRITTEN';
    LABEL PDC2='DIRECT*PREMIUMS*EARNEO';
    LABEL PDC3='DIRECT*LOSSES*INCURRED';
    FORMAT CNUM Z5.
    FORMAT PDC1-PDC3 COMMA19.;
    TITLE1 'DATA YEAR 1992';
PROC PRINT DATA=BASE.Y91 LABEL SPLIT=*
    VAR CNUM CNAME PDC1-PDC3;
    LABEL CNUM='COMPANY*NUMBER';
    LABEL CNAME='COMPANY*NAME';
    LABEL PDC1='DIRECT*PREMIUMS*WRITTEN';
    LABEL PDC2='DIRECT*PREMIUMS*EARNEO';
    LABEL PDC3='DIRECT*LOSSES*INCURRED';
    FORMAT CNUM Z5.
    FORMAT PDC1-PDC3 COMMA19.;
    TITLE1 'DATA YEAR 1991';
PROC PRINT DATA=BASE.Y90 LABEL SPLIT=*
    VAR CNUM CNAME PDC1-PDC3;
    LABEL CNUM='COMPANY*NUMBER';
    LABEL CNAME='COMPANY*NAME';
    LABEL PDC1='DIRECT*PREMIUMS*WRITTEN';
    LABEL PDC2='DIRECT*PREMIUMS*EARNEO';
    LABEL PDC3='DIRECT*LOSSES*INCURRED';
    FORMAT CNUM Z5.
    FORMAT PDC1-PDC3 COMMA19.;
    TITLE1 'DATA YEAR 1990';
```

The example above shows three PROC PRINTs using the same LABEL, FORMAT, and VAR statements. The only differences are the name of the data set being printed and the data year indicated in the TITLE statement. The code in examples A and B can be significantly reduced by creating two MACROS.

**How to define a MACRO**

In order for one to code a MACRO, one must code the following:

```
%MACRO name (parameters);
macrotext
%MEND name;
```

The %MACRO statement begins the definition of a MACRO, assigns a name to the MACRO, and optionally includes a parameter list. The name of the MACRO must be a valid SAS name (CMS users are limited to seven characters or fewer). MACRO parameters are MACRO variables with values that are supplied by the MACRO call. Macrotext is usually a combination of SAS statements, parts of SAS statements, and MACRO Programming statements. The %MEND statement is used to end the MACRO definition.

**How to call a MACRO**

Once a MACRO has been defined, it can be called by placing a percent(%) before the name of the MACRO.

```
%macroname(ordinal values);
```

A MACRO call assigns values to the parameters when the MACRO is called. Although a semicolon is not required to end a MACRO call, it is recommended. Lastly, a MACRO call can be made anywhere in a SAS program.

The following example will define examples A and B as SAS MACROS named INP and PRT respectively:

**Example C:**

```
%MACRO INP(yR);
DATA BASE.Y&YR;
```

The program in example A consists of three data steps. Each data step reads in observations from a flat file and creates a permanent SAS data set with five variables. The permanent SAS data sets are written to the same ddname and have the data year as part of its member name. The ddname on the infile statement also uses the data year as part of its name. After creating the permanent SAS data sets, we want to print out each observation in a separate SAS program. The program follows:

**Example B:**

```
PROC PRINT DATA=BASE.Y92 LABEL SPLIT=*
    VAR CNUM CNAME PDC1-PDC3;
    LABEL CNUM='COMPANY*NUMBER';
    LABEL CNAME='COMPANY*NAME';
    LABEL PDC1='DIRECT*PREMIUMS*WRITTEN';
    LABEL PDC2='DIRECT*PREMIUMS*EARNEO';
    LABEL PDC3='DIRECT*LOSSES*INCURRED';
    FORMAT-CNUM Z5;
In MACRO `INP` the macro variable `YR` is being created. A macro variable is referenced by placing an ampersand(&) immediately before the macro variable name.

**Define and describe MACRO Variables**

A MACRO variable consists of a name and a value. There are two classes of MACRO variables:

- **automatic** - defined by the SAS supervisor
- **user-defined** - defined by the user

A list of some commonly used Automatic Macro Variables follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSDATE</td>
<td>01JAN94</td>
</tr>
<tr>
<td>SYSTIME</td>
<td>11:37</td>
</tr>
<tr>
<td>SYSDAY</td>
<td>FRIDAY</td>
</tr>
<tr>
<td>SYSVER</td>
<td>6.08</td>
</tr>
</tbody>
</table>

SYSDATE - date the SAS program began execution  
SYSTIME - time the SAS program began execution  
SYSDAY - day of the week the SAS program began execution  
SYSVER - version of the SAS system being used  

User-defined MACRO Variables can be defined with a `%LET` statement or as a parameter on a `%MACRO` statement.

The following code will demonstrate how the MACRO variable `YR` is defined in via a `%LET` statement:

```sas
%LET YR=92;
```

**CONCLUSION**

Although much of SAS MACROs and SAS MACRO Language has not been explored in this paper, you should have a basic understanding of MACROs and be able to code simple MACROs.

SAS® is the registered trademark of SAS Institute Inc., Cary, NC, USA.