The 'SKIP' Statement

Paul Grant, Private Healthcare Systems, Inc.

The Problem

Sooner or later *every* SAS® programmer faces the irritating problem of running only a portion of an existing SAS program.

If you're a craftsman with respect to your code, you can't use bracket-style comments (/* .... */) to block segments of code from execution because you've already used them to document your program and bracket-style comments can't be nested. For example, we all know what would happen if we tried to use bracket-style comments to prevent the following data step from being executed.

```
      /*
DATA SALARY ;
      /* ASSIGN COMMISSION RATES */
      more SAS statements
RUN ;
    */
```

The '*/' after the word 'RATES' would end the comment which began on the first line so the statements after 'RATES */' would generate an error. Clearly, not the intended result.

Comment statements (*….. ;) will do the job but, except in small programs, they're too impractical to use. It's simply too tedious to put an asterisk in front of a lot of SAS statements and remove them when you're done.

What we really need is a SKIP statement. Like a comment statement, a SKIP statement would block lines of SAS code from being compiled and executed by the SAS Supervisor. But unlike statement-style comments, the SKIP statement would hide multiple SAS statements from the SAS Supervisor. And unlike bracket-style comments, SKIP statements could be nested.

The Solution

You'll be pleased to know that this capability already exists in the SAS language - not as an undocumented feature, just an unimagined one. And as the following example shows, you can easily code one yourself.

```
      SAS statements you want to execute.....
%macro SKIP ;
      SAS statements you want to skip.....
%mend SKIP ;
      SAS statements you want to execute.....
```

How It Works

This tip works by asking the SAS macro facility to store the portion of code that you don't want to execute as a SAS macro. If the 'SKIP' macro is never called in the program, it is never passed to the SAS Supervisor to be compiled and executed.

This technique prevents *any* SAS statements from being compiled and executed: lines within a DATA or PROC step, entire DATA and/or PROC steps, and definitions of other macros.

The Details

This technique is extremely flexible and easy to use. First,
'SKIP' statements can be repeated:

```
      SAS statements you want to execute.....
%macro SKIP ;
      SAS statements you want to skip.....
%mend SKIP ;
      SAS statements you want to execute.....
%macro SKIP ;
      More SAS statements you want to skip.....
%mend SKIP ;
      SAS statements you want to execute.....
```

In this example, the second 'SKIP' macro definition simply replaces the first definition of the 'SKIP' macro.

'SKIP' statements can also be nested:

```
        SAS statements you want to execute.....

%macro SKIP ;

        SAS statements you want to skip.....

        %macro SKIP ;

            More SAS statements you want to skip.....

        %mend SKIP ;

%mend SKIP ;

        SAS statements you want to execute.....
```

In this example, the second 'SKIP' macro is defined in the local referencing environment of the first 'SKIP' macro. The first 'SKIP' macro definition is defined in the global referencing environment.

The practical point to notice in both of these examples is that you don't have to assign different names (e.g., 'SKIP1,' 'SKIP2,' etc.) to each 'SKIP' macro.

Also, you can give the 'SKIP' macro any name you want. I like 'SKIP' because it's descriptive and looks a lot better than 'JUNK' or 'OOPS'.

If the code segment you want to skip is long, you may want to use the NOSOURCE option to prevent the skipped code from being printed to your SAS log.

```
Options Nosource;

%macro SKIP ;

        SAS statements you want to skip.....

%mend SKIP ;

Options Source;

        SAS statements you want to execute.....
```

A Final Note

Anyone who has inadvertently omitted a %MEND statement at the conclusion of one of their macro definitions knows how effective this trick is. In fact, that's how I "discovered" it.

In addition to learning how to code a 'SKIP' statement, I hope you have also seen the importance of learning from your mistakes (even the "little" ones like forgetting a %MEND statement) and the role of imagination in putting what you've learned to good use.

Publication Note and References

This paper has been extensively revised since it was first published in the NESUG '92 Conference Proceedings. This earlier version also appeared in the NESUG '93, SUGI 19, WUSS '94 and NESUG '97 Conference Proceedings.

Janet Stuelpner discusses this technique in an excellent and more general paper, "Skipping, The Easy Way," published in the NESUG '96, SUGI 22, and NESUG '97 Conference Proceedings.

This technique is also described in SAS Today! A Year of Terrific Tips, by Helen and Ginger Carey and in In the Know…SAS Tips and Techniques From Around the Globe, by Phil Mason. Both of these books are published by SAS Institute.

Author

The author welcomes your comments and suggestions at:

Paul Grant
Private Healthcare Systems, Inc.
1100 Winter Street
Waltham, MA 02154
Tel: 781-895-7614
e-mail: 0005733569@mcimail.com

```
   /*
DATA SALARY ;

   /* ASSIGN COMMISSION RATES */

   .....more SAS statements .....

RUN ;
   */
```

statements you want to execute.....

%macro SKIP ;

SAS statements you want to skip.....

%mend SKIP ;

statements you want to execute.....

statements you want to execute.....

%macro SKIP ;

SAS statements you want to skip.....

%macro SKIP ;

more statements you want to skip.....

%mend SKIP ;

%mend SKIP ;

statements you want to execute.....

statements you want to execute.....

%macro SKIP ;

SAS statements you want to skip.....

%mend SKIP ;

statements you want to execute.....

%macro SKIP ;

more statements you want to skip.....

%mend SKIP ;

statements you want to execute.....

statements at the beginning of a job that you want to execute.....

%macro SKIP ;

remaining code that you want to skip...