Paper 80-25

A Macro Utility for Generating Formatted Log Comments

Pete Lund, WA State Caseload Forecast Council, Olympia, WA

Abstract

If you've ever %included code or used macros with embedded SAS code, it is often frustrating to miss inline comments in the log or be unable to clearly track what the program has done. You can always turn on the SOURCE2 or MPRINT/MLOGIC options, but that sometimes gets to be too much. Likewise, you can add your own comments with %PUT, but you have to go to a lot of work to make them look nice.

This paper discusses a macro which formats comments which are written to the SAS log. The comment is bordered with asterisks and the user can control indentation, justification and line breaks. Macro references in the comment are resolved so comments can be made dynamic. The border automatically adjusts to the size of the longest line and there is no limit to the length or number of lines in the comment. You get consistent, formatted comments with fewer keystrokes than keying all those %PUT statements yourself.

The Call

To create your comment, simply pass it as the parameter of the %COMMENT macro. By default, separate comment lines are delimited by commas. When processed, the comment will be displayed in the log, bounded by asterisks, with each line left justified. For example, the following invocation:

```
%COMMENT(This is a test,of the comment macro);
```

would produce the following in the log:

```
************************
* This is a test       *
* of the comment macro *
************************
```

There is no limit to the number of lines or the length of each line, though lines will wrap at the LINESIZE limit. As noted, a comma is the default line separator. Often a comma is needed in the comment. The delimiter can be changed with the DLM= option. Simply put DLM=x, where x is any character, as the end of the comment and the specified character will be used as the line delimiter. For example, modifying the earlier example:

```
%COMMENT(This is a test, a new one,~
      of the comment macro~works well! DLM=~);
```

would produce the following in the log:

```
*****************************
* This is a test, a new one *
* of the comment macro      *
* works well!               *
*****************************
```

Notice that actual line breaks in the call to the macro are ignored. Only those specified by a line delimiter are used.

Adding Some Formatting

There are a number of options which can enhance the look of your comments. Each option is entered on a separate comment line.

Option	Action
/C	Centers following lines
/L or /c	Left justifies following lines
/R	Right justifies following lines
/I*n*	Indents following lines *n* spaces
/D	Inserts a line of dashes
/H	Inserts a line of asterisks
DLM=*x*	Changes default line break delimiter to *x*

The following example demonstrates the use of many of the options.

```
%comment(/C|NOTE!|/D|/L|
      This demonstrates the use of the|
      COMMENT macro!||/H||It is|/I3|
      - easy|- convenient|- versatile||/L|
      Get one for all your friends!!|/D|/R|
      Today is %sysfunc(date(),worddate.)
      DLM=|);
```

produces the following:

```
************************************
*             NOTE!                *
*----------------------------------*
* This demonstrates the use of the *
* COMMENT macro!                   *
*                                  *
************************************
*                                  *
* It is                            *
*    - easy                        *
*    - convenient                  *
*    - versatile                   *
*                                  *
* Get one for all your friends!!   *
*----------------------------------*
*         Today is April 11, 2000  *
************************************
```

Notice the format changes and the fact that macro references and %SYSFUNC calls are resolved.

How Does It Do That?!

An annotated version of the code for the macro is shown in Appendix A. In-line comments have been removed from the "production" version for the sake of space. Let's

Acknowledgments

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

References

Lund, Peter, "Always There When I Need You: Macros to Create Directories", *Proceedings of the Sixteenth Annual Pacific Northwest SAS Users Group Regional Conference*, 1997.

Polzin, Jeff and Susan O' Connor, "Macro Facility Enhancements for Release 6.09E and Release 6.11", *Proceedings of the Twenty-First Annual SAS Users Group International Conference*, Cary NC: SAS Institute Inc., 1996.

SAS Institute Inc., *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary NC: SAS Institute Inc., 1995.

Author Contact Information

Copies of the macro, including comments and syntax instructions, are available via e-mail. Please designate whether you prefer the macro in the body of the e-mail or as a text attachment.

Pete Lund
WA State Caseload Forecast Council
515 15th Ave SE
PO Box 40962
Olympia, WA 98504-0962
voice: (360) 902-0086
FAX: (360) 902-0084
e-mail: peter.lund@cfc.wa.gov

Appendix A

```
%macro repeat(char,times);
  %let char = %quote(&char);
  %if &char eq %then %let char = %str( );
  %sysfunc(repeat(&char,&times-1))
%mend;

%macro TrimLeft(mvar);
  %qsysfunc(trim(%qsysfunc(left(&mvar))))
%mend;

%macro Comment(Args) / pbuff;

  %let DLMpos = %index(%upcase(&syspbuff),DLM=);
  %if &DLMpos ne 0 %then %let DLM = %qsubstr(&syspbuff,&DLMpos+4,1);
  %else %let DLM = %str(,);


  %do _i = 1 %to %length(&syspbuff) - 1;
    %if %qsubstr(&syspbuff,&_i,2) eq %str(%str(&DLM)%str(&DLM)) %then
       %let syspbuff =
%qsubstr(&syspbuff,1,&_i)%str(.)%qsubstr(&syspbuff,&_i+1);
  %end;

  %if &DLMpos gt 0 %then
    %let _incom_ = %substr(&syspbuff,2,%eval(%length(&syspbuff)-8));
  %else %let _incom_ = %substr(&syspbuff,2,%eval(%length(&syspbuff)-
2));

  %let _stop_ = %length(&_incom_);
  %let _HdrLen_ = 0;
  %let _indent_ = 0;

  %do _i_ = 1 %to &_stop_ ;

    %let _ins_&_i_ =
%TrimLeft(%qscan(%quote(&_incom_),&_i_,%str(&DLM)));

    %if &&_ins_&_i_ eq %then
      %do;
         %let _Lines_ = %eval(&_i_ - 1);
         %let _i_ = &_stop_;
      %end;
    %else %if &&_ins_&_i_ ne . %then
      %do;
        %if %index(&&_ins_&_i_,/I) eq 1 %then
           %let _indent_ = %substr(&&_ins_&_i_,3);
        %else %if %qsubstr(&&_ins_&_i_,1,1) eq %quote(/) %then
          %let _indent_ = 0;
        %if %eval(%length(&&_ins_&_i_) + &_indent_) gt &_HdrLen_
%then
          %let _HdrLen_ = %eval(%length(&&_ins_&_i_) + &_indent_);
      %end;
  %end;

  %let _Border_ = %repeat(*,&_HdrLen_ + 4);

  %put ;
  %put &_Border_;

  %let _SetInd_ = 0;
  %do _i_ = 1 %to &_Lines_;
    %if &&_ins_&_i_ eq . %then
      %put * %repeat(,&_HdrLen_+1)*;
    %else %if &&_ins_&_i_ eq %quote(/D) %then
      %put *%repeat(-,%eval(&_HdrLen_ + 2))*;
    %else %if &&_ins_&_i_ eq %quote(/H) %then
      %put &_Border_;
    %else %if &&_ins_&_i_ eq %quote(/C) %then
      %let _SetInd_ = %nrstr(%eval((&_HdrLen_ - %length(&&_ins_&_i_)) / 2));
    %else %if &&_ins_&_i_ eq %quote(/c) or &&_ins_&_i_ eq %quote(/L) %then
      %let _SetInd_ = 0;
    %else %if &&_ins_&_i_ eq %quote(/R) %then
      %let _SetInd_ = %nrstr(%quote(%eval((&_HdrLen_ - %length(&&_ins_&_i_)))));
```

1 - supporting macros:
Repeat - generates repeating characters
TrimLeft - trims and left justifies a macro variable

2 - check to see if a non-default delimiter was passed. If not, set the delimiter character to a comma.

3 - put a period between two consecutive delimiters. This will be used to add a blank line.

4 - if a DLM=x was specified, strip it off &SYSPBUFF. Also, &SYSPBUFF contains the parentheses of the macro call - strip them off as well.

5 - break the incoming comment into separate macro variables, splitting on the delimiter..

6 -when the end is reached, save the number of lines which were passed.

7 -keep track of the longest line passed. If an indent toggle (/I) is active, add the number of spaces to indent to the length.

8 -start off with a line of asterisks as long as the longest line (plus room for the edges).

9 -loop through all the lines which were passed and write them to the log. If we're on a toggle line (e.g., /D, /C) do the action appropriate for the toggle. Each printed line begins and ends with an asterisk (*)

```
    %else %if %index(&&_ins_&_i_,/I) eq 1 %then
      %let _SetInd_ = %substr(&&_ins_&_i_,3);
    %else
      %do;
        %let _indent_ = %eval(%unquote(&_SetInd_) + 1);
        %put *%repeat(,&_indent_)&&_ins_&_i_%repeat(,(((&_HdrLen_-%length(&&_ins_&_i_))+1)-(&_indent_-
1)))*;
      %end;
  %end;
  %put &_Border_;
  %put ;

  %quit:
%mend;
```

10 -end with a line of asterisks as long as the longest line (plus room for the edges).