PharmaSUG 2016 - Paper TT17

Capturing Macro Code when Debugging in the Windows Environment: The Power of MFILE and the Simplicity of Pasting

Kevin R. Viel, Ph.D., inVentiv Health Clinical, Atlanta, GA

ABSTRACT

Macros are ubiquitous in a programmer's toolkit. During development, especially of long or complex or even simple macros, deciphering warnings and errors can be problematic. Copying and editing the log is resource intensive and subject to errors. The goal of this paper is to demonstrate a macro that captures the SAS code produced by a macro using the MFILE system option and allows the user to simply paste that code in the Windows environment to an Enhanced editor for review and interactive submission or %INCLUDE it, thus making possible the tracking of errors and warnings to their exact line number in the resulting log.

INTRODUCTION

Once a programmer masters the basics of the SAS System®, or earlier for the audacious, macros become an enticing skill to acquire because they reduce work while improving accuracy and the macro facility is an exploration into another "language". Regardless of one's expertise or experience, a programming error or the failure to adequately code for contingencies will inevitably occur. **Figure 1** demonstrates two trivialized examples.

```
1    %macro pharmasug ;
2
3      data _null_ ;
4        x = "Oh, happy day!" ;
5        put x
6      run ;
7
8      data _null_ ;
9        x = . ;
10       y = x ** 2 ;
11     run ;
12
13   %mend  pharmasug ;
14
15   %pharmasug ;

NOTE: Variable run is uninitialized.
Oh, happy day! .
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.00 seconds



NOTE: Missing values were generated as a result of performing an operation on missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      1 at 1:101
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Figure 1. Macro with Errors. The log from a macro with errors demonstrating the ambiguity of the notes.

Determining the source of the offending code or datum can be practically infeasible. For too long, the author would rely on the log from the execution of a macro during which the **MPRINT system option** was active. The MPRINT system option writes the SAS code generated by the executed macro to the log, which the author would manually copy to the Enhanced Editor. He would then find and replace the resulting MPRINT() prefix(es) and delete other non-code, like notes. **Figure 2** demonstrates the process.

2A

```
15   options mprint ;
16
17   %pharmasug ;
MPRINT(PHARMASUG):   data _null_ ;
MPRINT(PHARMASUG):   x = "Oh, happy day!" ;
MPRINT(PHARMASUG):   put x run ;

NOTE: Variable run is uninitialized.
Oh, happy day! .
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(PHARMASUG):   data _null_ ;
MPRINT(PHARMASUG):   x = . ;
MPRINT(PHARMASUG):   y = x ** 2 ;
MPRINT(PHARMASUG):   run ;

NOTE: Missing values were generated as a result of performing an operation on missing values.
      Each place is given by: (Number of times) at (Line):(Column).
      1 at 1:101
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

2B

```
MPRINT(PHARMASUG):   data _null_ ;
MPRINT(PHARMASUG):   x = "Oh, happy day!" ;
MPRINT(PHARMASUG):   put x run ;

MPRINT(PHARMASUG):   data _null_ ;
MPRINT(PHARMASUG):   x = . ;
MPRINT(PHARMASUG):   y = x ** 2 ;
MPRINT(PHARMASUG):   run ;
```

2C

```
data _null_ ;
x = "Oh, happy day!" ;
put x run ;

data _null_ ;
x = . ;
y = x ** 2 ;
run ;
```

Figure 2. SAS Code from Log. The log from the macro in Figure 1 run with the MPRINT system option invoked. **2A** - The unadulterated log that will be copied and pasted to the Enhanced Editor. **2B** - The code remaining after the note and other non-code were deleted manually. **2C** - The SAS code remaining after the MPRINT() prefix was replaced with nothing.

Although in this case the process was minor, the log from even macros with just a few lines can be a real problem, if the execution of the macro calls other macros. Without such complications, one might find that he or she has not deleted all of the non-code text or that he or she was too aggressive and inadvertently deleted part of the macro code. To avoid such errors of manually manipulating the text, one might write a program to extract only the SAS code. Fortunately, the SAS System provides a much easier and accurate method.

MFILE SYSTEM OPTION

The MFILE system option sends the generated SAS code to an external file. Its use requires the concomitant use of the MPRINT system option. Although the online documentation does not explicitly mention it, MFILE can also route the generated SAS code to a FILEREF. When the device-type specified in the FILENAME statements on the Windows operating system is TEMP, then the external file is temporary and managed by SAS, including its deletion at the terminal of the SAS session or when the filename is de-assigned. A further simplifying step in the Windows operating system is the ability to pipe the results of a command or program to the clip board. Obtaining the SAS code generated, thus, reduces to calling the macro presented in the paper and pasting it to an Enhanced Editor. Of course, the option to %INCLUDE the file is included if the user does not prefer to review or update the code before submission.

EXPLANATION OF SAS CODE

The M_MFILE macro appears in **Code 1** (the appendix). **Lines 1-7.** The macro statement includes five macro parameters, four of which have default values. TEXT_PATH is the path of the physical, permanent file to which the user may write the macro call, for instance, if he or she might like to adapt the call for future code.

INCLUDE is an indicator variable that when set to Y runs the SAS code generated by the target macro via the %INCLUDE statement with the SAS system option SOURCE2 invoked so that the log includes the SAS code. MCALL is the macro call of interest. CLIP is an indicator variable. When set to its default value of Y, the SAS code is written to the Windows clipboard and can then be copied (for instance, by typing Cntl-V) to the destination of choice, such as the Enhanced Editor. DELETE is an indicator variable that when set to Y will delete the mprint.sas file if TEXT_PATH is not missing.

Lines 9-12. If the word PUT is included before the macro name in the MCALL macro parameter, then the macro will write the macro call to the log without executing it. This is useful if the macro call includes macro variables or complex values for its macro parameter and the user wants to verify it before executing it. The match is determined using the PRXMATCH() function, one of the powerful Perl regular expression functions available in the SAS system. Since it is called as an argument to the %QSYSFUNC() macro function, the arguments should not be quoted with (double) quotation marks. The first argument is "/^put/i". The forward slashes demarcate the pattern, which is "^put". The caret (^) indicates that the three byte match has to start in the first position (byte). The match pattern is "put". The modifier "i" makes the match case insensitive. The second argument is the macro call provided to MCALL macro parameter with any special characters or mnemonic operators masked in its value resolved at macro execution. The special characters or mnemonic operators masked by %QSYSFUNC() are the same as those masked by %NRBQUOTE() macro function.

Lines 14-24. If "put" was the first three bytes of the MCALL macro parameter value and, hence, WRITE_MFILE will have the value 1, then these lines provide the logic to execute the FILENAME statement for the MPRINT fileref as a temporary file or as a permanent file mprint.sas with the path provided by the TEXT_PATH macro parameter. If the path is provided, then it overrides the creation of a temporary file. Lines 21-22 execute the OPTIONS statement invoking the system options MPRINT and MFILE.

Line 26. This is the macro call. At times, care must be taken to mask certain tokens while providing the values to the M_MFILE macro call. For instance, the values of the macro parameters of the target macro might have characters like & or % that can cause issues if not properly quoted. The section MACRO QUOTING will address this in more detail.

Lines 28-63. This section of code is only written and run if "put" was not the first three bytes of the macro parameter value MCALL. Lines 31-33 "toggle" the system options MPRINT and MFILE so that the generated SAS code from this point is no longer written to the fileref MPRINT. That remaining SAS code would be from execution of the M_MFILE macro itself and is not of interest, unless the user modified the macro and is debugging. The generated code includes the OPTION statements in Lines 31-33. This demarcates the code of interest. With no reason to run this code again, we "remove" it from the file by replacing it with blank spaces. The reason that we do not just delete the record from the raw text file is that M_MFILE edits the raw file "in place" by virtue of the SHAREBUFFERS option to the INFILE statement and that the INFILE and FILE statements reference the same fileref, MPRINT. Since no variable was explicitly provided to the INPUT statement, SAS reads the entire line into the variable _INFILE_. Lines 44-48 determine the length of _INFILE_ and create a variable SPACE that is that length but has only spaces. Recall that the REPEAT() function repeats its first argument one more time than its second argument specifies. To preserve the "leading" spaces and to write the exact size of the original line, the $VARYING256. format is used, which requires a variable immediately following it that provides the length. If the macro parameter CLIP is set to Y, then contents of the MPRINT fileref are read by the Windows DOS command TYPE and piped to the Windows clipboard. In Line 55 the path and file name of the file referenced by the MPRINT fileref are obtained by the data step function PATHNAME. Since the path or file name might contain spaces, the value provided to the operating system is quoted. More complex coding might be required if the path has what might be termed "questionable" characters. The simplicity of the data step in Lines 57-59 hints at the utility of pipes.

Lines 65-76. If macro parameter INCLUDE has the value Y, then the SAS code generated by the target macro is immediate run in the SAS session. This option does not preclude also writing it to the clipboard and vice versa. To see the SAS code in the log, the system option SOURCE2 is invoked. The state of the SOURCE2 is recorded in Line 69 before the OPTION statement in Line 71 is executed. The path and file name are again obtained using the PATHNAME() function and the file whether permanent or temporary is run by the %INCLUDE in Line 73. The original state of the SOURCE2 options is again set by Line 75.

Lines 78-83. The macro will "clean up" including de-assigning the MPRINT fileref and optionally deleting the permanent file. Line 85 is the %MEND statement.

MACRO QUOTING

The first thing to say about macro is to read Ian Whitlock's excellent SUGI 28 paper "A Serious Look Macro Quoting" (http://www2.sas.com/proceedings/sugi28/011-28.pdf). The second thing to recommend is scouring the archives of SAS-L for post on this issue. The third thing is to disclaim that the author is NOT an expert and struggles at time. For many macros, the programmer might escape the need to quote, or to unquote, the components.

Initially, a programmer might expect the macro in **Figure 3** to cause an issue: does the comma indicate another macro parameter in the macro PHARMASUG_TITLE or M_MFILE and which macro does the first right parenthesis close? This macro call runs without issue; SAS tokenizes it correctly.

```
%macro pharmasug_title
   ( t1 =
   , t2 = %str() ;
   ) ;

   title1 "&t1." ;
   %if %length(&t2.) > 0 %then title2 %str(%')&t2.%str(%') %str(;) ;

%mend  pharmasug_title ;

%m_mfile
  ( mcall      = pharmasug_title
                    ( t1 = AZ
                    , t2 = Test
                    )
  , include    = Y
  )
```

Figure 3. Two Macro Parameters. An example of a macro call with two macro parameters.

What happens, however, if the both titles were supposed to be "A&Z, Inc"? The observant or expectant reader might have noticed that the conditional clause was not the only difference between the TITLE1 and TITLE2 statements. They were coded that way to illustrate an issue. While it might be tempting to make the entire value of the MCALL macro parameter the argument to the %NRBQUOTE() function, applying the function to the minimal text has advantages beyond pedagogy, since for the first title the use of one %NRSTR() function is insufficient. Namely, ampersand in T1 needs to be quoted from the compilation of M_MFILE and then the compilation of the TITLE1 statement generated by the PHARMASUG_TITLE macro. **Figure 4** presents one approach to arrive at the correct titles. Note that the use of single quotation marks hides the ampersand in T2 from further resolution by the macro facility, whereas that is accomplished for the ampersand in T1 by the %NRSTR().

```
%m_mfile
  ( mcall      = pharmasug_title
                    ( t1 = A%nrstr(%nrstr(&))Z%nrstr(,)Inc
                    , t2 = A%nrstr(&)Z%nrstr(,)Inc
                    )
  , include    = Y
  )
```

Figure 4. Two Macro Parameters with Special Characters. An example of a macro call with two macro parameters with values that require quoting.

The programmer must not quote indiscriminately. For instance, if the value of the parameter should be resolved during compilation of the M_MFILE macro the curious event in **Figure 5** could result. This example actually skips a step and begins with an attempt to %UNQUOTE the value, following Ian Whitlocks advice: "If the mprint looks good and the SAS compiler does not understand it, then try %UNQUOTE." **Figure 6** presents the result of an attempt to replicate the effects of the %UNQUOTE() function that Ian Whitlock described as being able to "glue together symbols to make macro objects when the macro compiler does not see them as a macro object." In this case, it appears that the &n in Line 5 resolves to &n. at compilation, but the value of the 2 for the global macro variable N is not resolved when this code is executed. However, it is properly resolved when the code of the %INCLUDE statement is compiled. Unless the programmer is starting from fresh sessions, dangers may lurk after runs with such code. Thankfully, the author is unaware of a situation in which one might want to not have the value resolve at compilation of the M_MFILE macro.

```
1    %macro pharmasug2
2       ( n = ) ;
3
4      data _null_ ;
5        do x = 1 to %unquote(&n.) ;
6          y = x ** 2 ;
7        end ;
8        put x= y= ;
9      run ;
10
11   %mend  pharmasug2 ;
12
13
```

```
14   options nomprint
15           nomfile
16           ;
17
18   %let n = 2 ;
19
20   %m_mfile
21     ( mcall      = pharmasug2
22                       ( n = %nrstr(&)n.)
23     , include    = Y
24     )
MPRINT(M_MFILE):   ;
MPRINT(PHARMASUG2):   data _null_ ;
NOTE: The macro generated output from MPRINT will also be written to external file
C:\Users\Histonis\AppData\Local\Temp\SAS Temporary Files\_TD7732_DOMDVE_\#LN00014 while OPTIONS
MPRINT and MFILE are set.

     -
     386
     200
MPRINT(PHARMASUG2):   do x = 1 to &n. ;
MPRINT(PHARMASUG2):   y = x ** 2 ;
MPRINT(PHARMASUG2):   end ;
MPRINT(PHARMASUG2):   put x= y= ;
MPRINT(PHARMASUG2):   run ;

ERROR 386-185: Expecting an arithmetic expression.

ERROR 200-322: The symbol is not recognized and will be ignored.

NOTE: The SAS System stopped processing this step because of errors.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(M_MFILE):   options nomprint nomfile

NOTE: The file/infile MPRINT is:
      Filename=C:\Users\Histonis\AppData\Local\Temp\SAS Temporary Files\_TD7732_DOMDVE_\#LN00014,
      RECFM=V,LRECL=256,File Size (bytes)=101,
      Last Modified=20Mar2016:10:05:00,
      Create Time=20Mar2016:10:05:00

NOTE: 7 records were read from the infile MPRINT.
      The minimum record length was 5.
      The maximum record length was 24.
NOTE: 7 records were written to the file MPRINT.
      The minimum record length was 5.
      The maximum record length was 24.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds



NOTE: The file TYPE is:
      Unnamed Pipe Access Device,
      PROCESS=type "C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
Files\_TD7732_DOMDVE_\#LN00014" | clip,
      RECFM=V,LRECL=256

NOTE: 0 records were written to the file TYPE.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.00 seconds


NOTE: Fileref TYPE has been deassigned.
NOTE: %INCLUDE (level 1) file C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
Files\_TD7732_DOMDVE_\#LN00014 is file C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
```

```
Files\_TD7732_DOMDVE_\#LN00014.
25  +data _null_ ;
26  +do x = 1 to &n. ;
27  +y = x ** 2 ;
28  +end ;
29  +put x= y= ;
30  +run ;

x=3 y=4
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


31  +
NOTE: %INCLUDE (level 1) ending.
NOTE: Fileref MPRINT has been deassigned.
```

Figure 5. Aggressive Macro Quoting. A perhaps inappropriately macro quoted character (&) in a macro parameter value that produced an apparently adequate MPRINT log with recovery attempted by macro unquoting.

```
1    %macro pharmasug3
2       ( n = ) ;
3
4      data _null_ ;
5        do x = 1 to %unquote(%nrstr(&)n.) ;
6          y = x ** 2 ;
7        end ;
8        put x= y= ;
9      run ;
10
11   %mend  pharmasug3 ;
12
13   options nomprint
14           nomfile
15           ;
16
17   %let n = 3 ;
18
19   %m_mfile
20     ( mcall      = pharmasug3
21                       ( n = %nrstr(&)n.)
22     , include    = Y
23     )
MPRINT(M_MFILE):   ;
MPRINT(PHARMASUG3):   data _null_ ;
NOTE: The macro generated output from MPRINT will also be written to external file
C:\Users\Histonis\AppData\Local\Temp\SAS Temporary Files\_TD8120_DOMDVE_\#LN00014 while OPTIONS
MPRINT and MFILE are set.

     -
     386
     200
MPRINT(PHARMASUG3):   do x = 1 to &n. ;
MPRINT(PHARMASUG3):   y = x ** 2 ;
MPRINT(PHARMASUG3):   end ;
MPRINT(PHARMASUG3):   put x= y= ;
MPRINT(PHARMASUG3):   run ;

ERROR 386-185: Expecting an arithmetic expression.

ERROR 200-322: The symbol is not recognized and will be ignored.

NOTE: The SAS System stopped processing this step because of errors.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(M_MFILE):   options nomprint nomfile

NOTE: The file/infile MPRINT is:
```

```
      Filename=C:\Users\Histonis\AppData\Local\Temp\SAS Temporary Files\_TD8120_DOMDVE_\#LN00014,
      RECFM=V,LRECL=256,File Size (bytes)=101,
      Last Modified=20Mar2016:10:14:21,
      Create Time=20Mar2016:10:14:21

NOTE: 7 records were read from the infile MPRINT.
      The minimum record length was 5.
      The maximum record length was 24.
NOTE: 7 records were written to the file MPRINT.
      The minimum record length was 5.
      The maximum record length was 24.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds



NOTE: The file TYPE is:
      Unnamed Pipe Access Device,
      PROCESS=type "C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
Files\_TD8120_DOMDVE_\#LN00014" | clip,
      RECFM=V,LRECL=256

NOTE: 0 records were written to the file TYPE.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.01 seconds


NOTE: Fileref TYPE has been deassigned.
NOTE: %INCLUDE (level 1) file C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
Files\_TD8120_DOMDVE_\#LN00014 is file C:\Users\Histonis\AppData\Local\Temp\SAS Temporary
Files\_TD8120_DOMDVE_\#LN00014.
24  +data _null_ ;
25  +do x = 1 to &n. ;
26  +y = x ** 2 ;
27  +end ;
28  +put x= y= ;
29  +run ;

x=4 y=9
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


30  +
NOTE: %INCLUDE (level 1) ending.
NOTE: Fileref MPRINT has been deassigned.
```

Figure 6. Aggressive Macro Quoting. A perhaps inappropriately macro quoted character (&) in a macro parameter value that produced an apparently adequate MPRINT log with recovery attempted by macro quoting and unquoting: "glueing" objects for the macro compiler.

CONCLUSION

Macros are a great tool in a programmer's arsenal. Inevitably, an error will surface that could be hard to trace from the log, even with MPRINT invoked. The advent of the MFILE system option to capture the SAS code generated in by a macro was a helpful advancement. This macro presented a method to %INCLUDE that code and a method to capture that code on the Windows clipboard.

Obviously future directions will include adopting the clipboard method to Unix/Linux, revising the code, if necessary, to ensure that it functions correctly regardless of operating system, and adopting it to the SAS Grid. Overall, this macro is minimal and should be understandable even to programmers unfamiliar with interaction with the Windows operating system using the SAS system. Presumably, the user has a basic understanding of the macro facility if he or she is using or reading about a macro that captures the SAS code generated by the macro call specified in its macro parameter. If not, this macro might be a tool to help study the macro facility. Hopefully, readers will find this macro helpful in the development of their macros and in clarifying the issue of macro quoting.

REFERENCES

Whitlock I. A Serious Look Macro Quoting. Proceedings of the 28th SAS Users' Group, International. (http://www2.sas.com/proceedings/sugi28/011-28.pdf).

CONTACT INFORMATION

Your comments, questions, and corrections are valued and encouraged. Contact the author at:

Name: Kevin R. Viel, Ph.D.
Enterprise: inVentiv Health Clinical
Histonis, Incorporated
E-mail: kevin.viel@inventivhealth.com
kviel@histonis.org
Web: http://www.inventivhealthclinical.com/

Code 1. The M_MFILE Macro.

```
%macro m_mfile
  ( text_path  = %str()
  , include    = N
  , mcall      =
  , clip       = Y
  , delete     = N
  ) ;

  %let write_mfile = %qsysfunc(prxmatch( /^put/i
                                       , &mcall.
                                       )
                               ) ;

  %if &write_mfile. = 0
  %then
     %do ;
         %if %length(&text_path.) > 0 %then filename mprint "&text_path.\mprint.sas" %str(;) ;
          %else filename mprint temp %str(;) ;

         options mprint
                 mfile
                 ;
       %end ;

  %&mcall.

  %if &write_mfile. = 0
  %then
     %do ;
         options nomprint
                 nomfile
                 ;

         data _null_ ;
           infile mprint
                  sharebuffers
                  ;
           file mprint ;
           input ;
           if _infile_ ~ =: "options nomprint nomfile" then put _infile_ ;
            else
              do ;
                 len = length( _infile_ ) ;
                 space = repeat( " "
                               , length( _infile_ ) - 1
                               ) ;
                 put space $varying256. len ;
             end ;
         run ;

         %if &clip. = Y
         %then
            %do ;
                filename type pipe "type ""%sysfunc(pathname(mprint))"" | clip" ;

                data _null_ ;
                  file type ;
                run ;

                filename type    clear ;
             %end ;
     %end ;

  %if     &include.     = Y
      and &write_mfile. = 0
  %then
     %do ;
         %let source2_orig  = %sysfunc(getoption(source2))  ;

```

```
 70           options source2 ;
 71
 72           %include "%sysfunc(pathname(mprint))" ;
 73
 74           options &source2_orig. ;
 75       %end ;
 76
 77    %if &write_mfile. = 0 %then filename mprint clear %str(;) ;
 78
 79    %if     %length(&text_path.) > 0
 80        and &delete.      = Y
 81        and &write_mfile. = 0
 82    %then x "del ""&text_path.\mprint.sas"" " %str(;) ;
 83
 84  %mend m_mfile ;
```