Paper 40-2010

Macro Variables Make Life Easier: Applications with SAS® to Excel

Misty Johnson, State of Wisconsin Department of Health Services, Madison, WI, USA

MWSUG 2010 Milwaukee, WI

ABSTRACT

As SAS users, we all find ourselves telling SAS some basic information over and over, like the name of the report, recipient of the report, what is contained in the report and where we will write the output. Macro variables are a useful tool for the intermediate SAS user to minimize this redundancy and as a result, minimize errors by declaring important criteria only once; saving you time and making life easier. This paper will demonstrate the use of macro variables to tell SAS which data to pull, to double-check that the correct data was pulled before writing the output report, and finally define formatting done by Dynamic Data Exchange (DDE) of the output report in Excel.

INTRODUCTION

As a Research Analyst for the State of Wisconsin Department of Health Services, I create many reports each month for Counties administering various Medicaid Waiver programs. The reports I create may contain Protected Health Information (PHI) under the Health Insurance Portability and Accountability Act of 1996 (HIPAA) and must be safeguarded. Since I pull some of the same reports for different Counties, I must always be aware of what County I am currently working on to assure that I do not send County A's information to County B. To achieve this, I use macro variables to set various parameters, such as County name and data year, at the top of my program.

All of my report output is in Microsoft Excel with similar formatting, such as font, frozen top line, bold column titles, header and footer information and password protection. Instead of formatting each output report in Excel, I use Dynamic Data Exchange (DDE) in SAS to achieve the desired formatting, and I use macro variables to set values for this formatting.

LeRoy Bessler gave an excellent presentation on the functionality of DDE at MWSUG in 2007 (see references for more information), and I would recommend his paper and supporting documents for a tutorial on DDE. I am taking this one step further by using macro variables to define some of the formatting settings within DDE so that things like the name and report recipient can be used in the output file path and header on the output in an Excel document. This paper is not a tutorial on DDE per se, but an extension of using macro variables with DDE.

SET MACRO VARIABLES ON TOP

I try to set all the macro variables I will be using in the program at the very top, in a semi-comment box. I use a comment to explain each variable and where to declare it; I then copy this paragraph into all of my programs, so that I can always reference back to see exactly what data I pulled when the program was used last. For example:

```
*
>ENTER DATA YEAR HERE>>>>>>>>>;       %LET YEAR=2010; *
>ENTER AGENCY NUMBER HERE>>>>>;       %LET C_AGENCY=201300; *
>ENTER AGENCY NAME HERE>>>>>>>;       %LET AGENAME=Dane; *
>ENTER REPORT NAME HERE>>>>>>>;       %LET RPTNAME=Dane HSD L300;  *
>ENTER PASSWORD HERE>>>>>>>>>>;       %LET PASSWORD=Dane2;  *
>ENTER RECIPIENT HERE>>>>>>>>>;       %LET RECIP=Jean French;  *
>ENTER PAPER ORIENTATION HERE>;       %LET ORIENT=/*Portrait*/ Landscape;*
>ENTER PAPER SIZE HERE>>>>>>>>;       %LET PAPER1=/*Letter*/ Legal;*
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;
```

Above I declare which year of data to pull and for what County (Agency). I am also declaring some more variables that will be used at the end of the program to name the output file, password protect it, add header information and format the output Excel file.

DECLARE THE OUTPUT PATH

I use some of the macro variables that I set at the top of the program to help declare the output path. In my sample code below, I use CALL SYMPUT to obtain and format the report write date and the date stamp of the data, the "as of date", into my preferred format. These two elements will be used in my file name so I know when the report was written and what date the data was valid through. Note that the macro variable of "year" is used to pull the correct file.

```
DATA _NULL_;
  SET CY&YEAR..asofdate;
  CALL SYMPUT('currdtlong',put(today(),DATE9.));
  CALL SYMPUT('asofdatelong',PUT(asofdate,DATE9.));
RUN;
```

Next, I use a LET statement to declare two output paths. Note that the data year and agency name that I declared above is used to name the output path and file name.

```
*>ENTER OUTPUT PATH HERE>>>>;
%LET PREPATH1="H:\TEMP\PRERpt_for_&AGENAME._rtn_&currdtlong..xls";

%LET
PATH1="H:\&YEAR.\&AGENAME.\Rpt_for_&AGENAME._&year._asof_&asofdatelong._rtn_&currdtl
ong..xls";
```

I will later use DDE to format the final output report. DDE requires two files, one containing the unformatted output and a second containing the formatted, finished product. I typically write the first file, which I call a "pre-report" to my temp directory so that I can later delete the file. The final output will be written to "Path1" declared above.

VERIFY, THEN PRINT

The next part of my SAS program pulls the data for the Agency number declared as a macro variable and creates the report. Within the body of the program, I format this Agency number into a character variable with length of 25 called *Agency*. This would turn, for example, Agency #201300 into "Dane County". Before I write the file to Excel, I use a DATA step to double check that the data I pulled was for the Agency I named by number. Since I declare both the Agency name and number, this is my double check to avoid a potential HIPAA violation of sending one County's information to another County.

```
DATA _NULL_; SET REPORT;
    IF _N_=1 THEN CALL SYMPUT('AGENCYNAME',PUT(AGENCY,$25.)); RUN;
DATA _NULL_; set CY&YEAR..asofdate;
IF (LOWCASE(SCAN("&AGENAME.",1)))=(LOWCASE(SCAN("&AGENCYNAME.",1)))
    THEN CALL SYMPUT('ANSWER',PUT("YES",$5.));
    ELSE CALL SYMPUT('ANSWER',PUT("NO",$5.)); RUN;
%MACRO PRINT_REPORT;
%IF &ANSWER.=YES %THEN
    %DO;
          PROC DBLOAD DBMS=EXCEL DATA= REPORT; PATH=&PREPATH1.;
          PUTNAMES=YES; LIMIT=0; label; reset all; LOAD; RUN;
    %END;
RUN;
%MEND PRINT_REPORT;
    %PRINT_REPORT; RUN; /* INVOKE MACRO TO WRITE TO EXCEL IF CORRECT COUNTY */
```

The first DATA step above opens my report, that is waiting to be written to Excel, and examines the first line. It grabs the value for "Agency" and stores it as a macro variable called "agencyname". The second DATA step compares the agency name that I declared at the beginning of my program, a macro variable named "agename" with "agencyname" that's actually in the report. I use a SCAN to determine if they are equal, for example, "Dane" would equal "Dane County". I create another macro variable called "answer" with the answer to my security question.

If the "agency" contained in the report matches the "agency" that I named in the beginning of the program, I use PROC DBLOAD to write the file to Excel. I use PROC DBLOAD because I have declared labels for all my variables in a late DATA step in the body of the program (not illustrated), and PROC DBLOAD will print these variable labels in Excel nicely; PROC EXPORT does not support labels.

Although PROC DBLOAD prints the variable labels and number formats well, it doesn't exactly look "finished". It comes out in "System" font size 10. I much prefer Times New Roman font, a frozen, bold top line, headers and footers to name the report and recipient, as well as a password to protect the document. This is why I use DDE to accomplish these tasks for me.

USE MACRO VARIABLES TO DEFINE THE FINAL OUTPUT

At the top of the program, you will recall that I defined both the orientation and paper size for the final output. Before DDE can use this information, I need to code these macro variables into a format useable by DDE. This is achieved with a short macro named "Compile_DDE" and is shown below.

```
%GLOBAL ORIENTATION PAPER;
OPTIONS SYMBOLGEN;
%MACRO COMPILE_DDE(ORIENT=,PAPER1=);
  %IF &ORIENT.=Portrait %THEN %LET ORIENTATION=1;
  %ELSE %LET ORIENTATION=2;
  %IF &PAPER1.=Letter %THEN %LET PAPER=1;
  %ELSE %LET PAPER=5;
%MEND COMPILE_DDE;
%COMPILE_DDE(ORIENT=&ORIENT.,PAPER1=&PAPER1.);
%PUT ORIENTATION=&ORIENTATION.;
%PUT PAPER=&PAPER.;
```

This short macro simply converts my wordy declarations of “use landscape orientation” and “print it on legal-sized paper” from the top of my SAS program into code that will be useable by DDE.

Two more parameters that DDE will need to know are the number of variables and number of observations contained in the “preliminary” output already written to Excel. To accomplish this, I used a tip from SAS Tech Support, Usage Note 8743: How can I determine the number of variables in a data set? (http://support.sas.com/kb/8/743.html).

```
*DETERMINE THE NUMBER OF COLUMNS OF YOUR FINAL DATA SET;
*FROM SAS TECH SUPPORT:  "Usage Note 8743: How can I determine the number of
variables in a data set?";
%let nvar=%sysfunc(attrn(%sysfunc(open(&syslast.,i)),nvars));
%let nobs=%EVAL(%sysfunc(attrn(%sysfunc(open(&syslast.,i)),nobs))+1);
%put nvar=&nvar.;
%put nobs=&nobs.;
```

The last data set I was working on in this program was called “Report”, which was written to Excel through PROC DBLOAD as “pre-report”, or preliminary output. This file called “Report” is what the system variable “&SYSLAST” is referring to. Note that I add one (1) to the number of observations, or system variable “nobs”, to account for the variable titles in the first row. Now DDE knows how many observations my preliminary output has, as well as how many variables, which translates to number of rows and columns for Excel.

USE DDE TO FORMAT THE FINAL OUTPUT

Now we will invoke the power of DDE through a macro I named “formatme”, shown in it’s entirety below, before discussing in parts.

```
*****************************************************************;
****  NOW USE DDE TO FORMAT OUTPUT IN EXCEL                      *;
*****************************************************************;
%MACRO FORMATME(PREPATH1=,PATH1=,PASSWORD=,NVAR=,NOBS=,ORIENTATION=,PAPER=);
OPTIONS NOXWAIT NOXSYNC;
* OPEN A BLANK EXCEL WORKBOOK ;
X '"C:\Program Files\Microsoft Office\OFFICE11\EXCEL.EXE"';
DATA _NULL_;
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO START */
RUN;

FILENAME DDECMDS DDE "EXCEL|SYSTEM";

DATA _NULL_;
FILE DDECMDS;
PUT %unquote(%str(%'[OPEN(&PREPATH1.)]%'));
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO OPEN FILE */
PUT '[WORKBOOK.ACTIVATE("Sheet1")]'; /*ACTIVATE SHEET 1*/
************  PAGE SETUP  ***************************;
PUT %unquote(%str(%'[PAGE.SETUP("&L REPORT NAME: &RPTNAME.' '0d'x 'RUNDATE: &D
FOR: &RECIP.' '0d'x 'AUTHOR: Misty Johnson   SOURCE: &YEAR. HSRS LTS Module","&L &F
&R Page &P of
&N",.25,.25,.75,.5,FALSE,TRUE,TRUE,TRUE,&ORIENTATION.,&PAPER.,100,"AUTO",2,TRUE,600,
.25,.25)]%'));
************  FORMAT PAGE  ***************************;
PUT %unquote(%str(%'[SELECT("R1C1:R1C&NVAR.")]%')); /*SELECT TOP LINE*/
    PUT '[FONT.PROPERTIES("","BOLD")]'; /*BOLD TOP LINE*/
    PUT '[BORDER(0,0,0,0,1)]'; /*UNDERLINE TOP LINE*/
    PUT '[FREEZE.PANES(TRUE,0,1)]'; /*FREEZE TOP LINE*/
```

```
PUT %unquote(%str(%'[SELECT("R1C1:R&NOBS.C&NVAR.")]%')); /*SELECT WHOLE PAGE*/
    PUT '[FONT.PROPERTIES("Times New Roman",,10,,,,,,,1)]'; /*CHANGE FONT*/
    PUT '[COLUMN.WIDTH(,,,3)]'; /*AUTOFIT COLUMN WIDTH*/
    PUT '[SET.PRINT.TITLES("R1")]'; /*USE TOP ROW AS TITLES*/

PUT '[SELECT("R1C1")]'; /*REMOVE HIGHLIGHTING, PLACE CURSOR AT TOP*/

PUT %unquote(%str(%'[SAVE.AS(&PATH1.,,"&PASSWORD.")]%'));
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO SAVE */
PUT '[ERROR(FALSE)]';
*PUT '[QUIT()]'; /*(OPTIONAL: CLOSE EXCEL)*/
RUN;
%MEND FORMATME;
%FORMATME(PREPATH1=&PREPATH1.,PATH1=&PATH1.,PASSWORD=&PASSWORD.,NVAR=&NVAR.,NOBS=&NO
BS., ORIENTATION=&ORIENTATION.,PAPER=&PAPER.);
```

Dynamic Data Exchange (DDE) can get long and complicated; and unfortunately, this is a long chunk of code, but it's filled with goodies. Let's discuss it in blocks.

INVOKE THE MACRO, START DDE

Code snippet for discussion:

```
%MACRO FORMATME(PREPATH1=,PATH1=,PASSWORD=,NVAR=,NOBS=,ORIENTATION=,PAPER=);
OPTIONS NOXWAIT NOXSYNC;
* OPEN A BLANK EXCEL WORKBOOK ;
X '"C:\Program Files\Microsoft Office\OFFICE11\EXCEL.EXE"';
DATA _NULL_;
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO START */
RUN;

FILENAME DDECMDS DDE "EXCEL|SYSTEM";
```

DDE does not have to be in a macro. I put the DDE in a macro because I want to declare the variables that I will use in DDE as macro variables, and see their declaration clearly at the top of my program. The macro variables that I will use in the DDE, clearly shown in the MACRO statement, are:

- PREPATH1 – this is where the preliminary output was written.
- PATH1 – this is where the final, formatted output will be written (they must be different).
- PASSWORD – this will be the case-sensitive password used to open the final Excel document.
- NVAR – this is the number of variables that we determined in the prior step. This will be the number of columns in the final Excel document.
- NOBS – this is the number of observations plus one (1) that we determined in the prior step. This will be the number of rows, counting the first row with titles, in the final Excel document.
- ORIENTATION – this is the paper orientation (letter or landscape) of the final Excel document.
- PAPER – this is the paper size (letter, legal, etc) of the final Excel document.

The X statement, followed by the location of Microsoft Excel, opens a blank Excel workbook. The SLEEP statement gives the computer a bit of time to start Excel. The FILENAME statement establishes a communication link with Excel.

OPEN THE PRELIMINARY OUTPUT FILE

Code snippet for discussion:

```
DATA _NULL_;
FILE DDECMDS;
PUT %unquote(%str(%'[OPEN(&PREPATH1.)]%'));
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO OPEN FILE */
PUT '[WORKBOOK.ACTIVATE("Sheet1")]'; /*ACTIVATE SHEET 1*/
```

This bit of code opens your preliminary file, hence why I like to call the path "prepath". Note that anytime you call a macro variable in DDE, you must use %UNQUOTE(%STR(%.....)). So whenever you see this, you know that one of

the macro variables declared at the very top of the program is being called into service. I normally put a number one (1) after the macro variable “prepath” because sometimes I have more than one output file. The macro name “prepath1” indicates to me that I have only one output file. Now that the preliminary file is accessed and open, next we will format it.

FORMAT THE PRELIMINARY OUTPUT FILE – PAGE SETUP

Code snippet for discussion:

```
************  PAGE SETUP  ****************************;
PUT %unquote(%str(%'[PAGE.SETUP("&L REPORT NAME: &RPTNAME.' '0d'x 'RUNDATE: &D
FOR: &RECIP.' '0d'x 'AUTHOR: Misty Johnson   SOURCE: &YEAR. HSRS LTS Module","&L &F
&R Page &P of &N",
.25,.25,.75,.5,FALSE,TRUE,TRUE,TRUE,&ORIENTATION.,&PAPER.,100,"AUTO",2,TRUE,600,.25,
.25)]%'));
```

The above code demonstrates DDE formatting the page setup. Note again the presence of %UNQUOTE(%STR(%.....)), because we’re about to use some of our time-saving macro variables. The first PUT statement is very long, but what it is doing is no different that what you would do in Excel by clicking on File, then Page Setup.

The PAGE.SETUP command string is filled with individual settings, each separated by a comma, which represent the little check-boxes / value-boxes in the Page Setup in Excel. Instead of clicking on things like “landscape” and “legal-sized paper”, we’re asking DDE and SAS to do this for us. You can download the file explaining macro functions in Excel called “Macrofun.exe” from http://support.microsoft.com/kb/128185 for more information on all the settings I’m about to describe for PAGE.SETUP. Here is the PAGE.SETUP command followed by a detailed explanation:

PAGE.SETUP(header, footer, left margin, right margin, top margin, bottom margin, row and column headings, cell gridlines, center horizontally, center vertically, orientation, paper size, scale, page number of first page, page order, black & white cells, quality, header margin, footer margin, print cell notes, draft quality)

- header – this is the header information.
- footer – this is the footer information.
- left margin – the left margin in inches.
- right margin – the right margin in inches.
- top margin – the top margin in inches.
- bottom margin – the bottom margin in inches.
- row and column headings – the row and column headings checkbox (TRUE means the box is “checked”).
- cell gridlines – the gridlines checkbox (TRUE means the box is “checked”).
- center horizontally – center on page horizontally checkbox (TRUE means the box is “checked”).
- center vertically – center on page vertically checkbox (TRUE means the box is “checked”).
- orientation – page orientation (1=portrait or 2=landscape).
- paper size – a number from 1 to 26 specifying paper size (1=letter, 5= legal).
- scale – a number representing the percentage of reduction / enlargement or TRUE = size to 1page.
- page number of first page – specify number on first page or AUTO for 1.
- page order – pagination order (1=down, then over or 2=over, then down).
- black & white cells – print as black & white (TRUE) or FALSE allows for color.
- quality – print quality in dots per inch (600 or 1200; omitting it will yield default of 600).
- header margin – header margin in inches from page edge.
- footer margin – footer margin in inches from page edge.
- print cell notes –print cell notes (I’ve omitted this setting which is default for FALSE).
- draft quality – draft quality checkbox (I’ve omitted this setting which is default for FALSE).

Note that the settings for the header and footer are quite long. Here is the code printed in an easier format for discussion:

```
Header: "&L REPORT NAME: &RPTNAME.' '0d'x '
          RUNDATE: &D   FOR: &RECIP.' '0d'x '
          AUTHOR: Misty Johnson   SOURCE: &YEAR. HSRS LTS Module"

Footer: "&L &F &R Page &P of &N"
```

I have used formatting codes to control the placement of my header and footer information. My header typically has three lines in the top left corner; therefore I've used &L to left justify, and ' '0d'x ' for a carriage return. I have also used some of the macro variables in the header information, namely, Report Name, Recipient and Data Year. The formatting code &D will print the current date for "run date".

Similarly, the footer is left justified (&L), the name of the workbook, (&F), then right justify (&R) the word "Page", a space and the current page number (&P), the word "of" another space and the total number of pages in the document (&N).

FORMAT THE PRELIMINARY OUTPUT FILE – FORMAT THE PAGE

Code snippet for discussion:

```
************  FORMAT PAGE  ****************************;
PUT %unquote(%str(%'[SELECT("R1C1:R1C&NVAR.")]%')); /*SELECT TOP LINE*/
    PUT '[FONT.PROPERTIES("","BOLD")]'; /*BOLD TOP LINE*/
    PUT '[BORDER(0,0,0,0,1)]'; /*UNDERLINE TOP LINE*/
    PUT '[FREEZE.PANES(TRUE,0,1)]'; /*FREEZE TOP LINE*/
PUT %unquote(%str(%'[SELECT("R1C1:R&NOBS.C&NVAR.")]%')); /*SELECT WHOLE PAGE*/
    PUT '[FONT.PROPERTIES("Times New Roman",,10,,,,,,1)]'; /*CHANGE FONT*/
    PUT '[COLUMN.WIDTH(,,,3)]'; /*AUTOFIT COLUMN WIDTH*/
    PUT '[SET.PRINT.TITLES("R1")]'; /*USE TOP ROW AS TITLES*/

PUT '[SELECT("R1C1")]'; /*REMOVE HIGHLIGHTING, PLACE CURSOR AT TOP*/
```

The above code demonstrates DDE formatting the spreadsheet. I use a SELECT statement to select the top line of the preliminary output, which contains the variable labels, to bold the text, underline it, and freeze the top line so the labels remain stationary while you scroll through the final report. Notice the macro variable &NVAR in the first SELECT statement; knowing how many variables are in your output allows DDE to only format the number of columns it needs to

Similarly, when we want to change the font by hand in Excel, we would "select all". DDE accomplishes this in the second SELECT statement above by selecting the cells starting at Row1, Column1 through the last row and the last column. What number is the last row and the last column? It was already set by SAS through the macro variables &NOBS and &NVAR. Note that we added one (1) to the macro variable &NOBS to "count" the top line which contains our variable (column) titles. If we were to use the system variable &NOBS and not add one, the last observation (row) would not be formatted with the desired font.

After "selecting all", DDE style, we use DDE to change the font, auto fit the column width to our data, and allow the first row to print at the top on all hard copy sheets at the column titles. More information can be found on all these DDE commands in the "Macrofun.exe" help file. Again, these are all parameters you can set by hand in Excel though "File", then "Page Setup"; but why set them by hand when SAS can do it for you? After formatting our report, select the first cell, which places the cursor at "home" and removes any "selected" cells for a nice, neat appearance.

SAVE THE FINAL OUTPUT

Code snippet for discussion:

```
PUT %unquote(%str(%'[SAVE.AS(&PATH1.,,"&PASSWORD.")]%'));  /* SAVE THE FINAL FILE */
Z=SLEEP(3);  /* WAIT 3 SECONDS FOR EXCEL TO SAVE */
PUT '[ERROR(FALSE)]';
*PUT '[QUIT()]'; /*(OPTIONAL: CLOSE EXCEL)*/
RUN;
```

Now that we have formatted the final output in Excel, we save it to the final path destination that I call &PATH1. I also secure the file with a password, one that I designate at the beginning of the program with a macro variable called &PASSWORD. I wait 3 seconds for Excel to save the file, deal with any errors and I could optionally close Excel when I'm finished. I usually comment this out, because I want to leave Excel open after the file is finished, because I usually then just hit the email "button" in Excel to send the file to its recipient, all secure with its password protection.

CONCLUSION

In conclusion, I've demonstrated how macro variables can make things easier for you. All of us have written some really neat programs that pull different things, declare a year here and a program there, but when you go to run it for another year or another program, you have to be SURE that you declared what you needed in all the right places. However, usage of macro variables ensures that you state your criteria only ONCE, in a highly visible place, and that it is honored throughout the program.

DDE holds real power to format Excel reports into a polished document, saving much time. This process is improved upon when the formatting criteria are reliant upon macro variables set at the beginning of the program.

REFERENCES

Bessler, LeRoy. 2007. "SAS®-with-Excel Application Development Tools and Techniques, Expanded Edition". MWSUG 2007 Proceedings.

Delwiche, Lora D. and Slaughter, Susan J. 1998. *The Little SAS® Book, Second Edition*, Cary, NC: SAS Institute Inc.

Microsoft Support. 2005. "Macrofun.exe" from http://support.microsoft.com/kb/128185

SAS Tech Support, Usage Note 8743: "How can I determine the number of variables in a data set?" http://support.sas.com/kb/8/743.html

CONTACT INFORMATION

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

Name: Misty Ann Johnson
Enterprise: State of Wisconsin Department of Health Services
Address: 1 W Wilson St, Rm 418
City, State ZIP: Madison, WI 53707
Work Phone: 608-267-9583
E-mail: misty.johnson@wi.gov