SAS® Shorts: Valuable Tips for Everyday Programming

Jeff McCartney and Raymond Hu,
Social and Scientific Systems, Inc., Bethesda, MD

ABSTRACT & INTRODUCTION

The tips presented in this paper should prove valuable to SAS programmers of all levels. Examples selected were developed over time by our company staff as practical solutions and have become part of our standard pool of resources. Examples are short and utilitarian in nature and include references to supplementary documentation such as SAS OnlineDoc®. Included are examples showing how to use PROC PRINTTO to direct SAS logs and output to user-defined files; using the ?? modifier of the INPUT function to determine if a SAS character string contains only numeric data; using the RUN CANCEL statement to conditionally run a DATA step or PROC step; user written macros to display start time, end time, and duration of SAS jobs and how to store that macro in a SAS autocall library; adding icons to the program editor, log, and output windows; how to print two small output jobs on one output page; a "trick" to have PROC PRINT display variables in alphabetical order; sending email within SAS; and using the explorer window to display selected variables in the order you want. Most examples are based on the Microsoft Windows® platform but some may apply across platforms.

PROC PRINTTO TO DIRECT SAS LOGS AND OUTPUT TO SPECIFIED FILES

Sometimes you may need to combine multiple SAS programs into one program, say, to streamline code for an overnight batch run. However, you may prefer to maintain the SAS log and SAS output files created by each piece separately rather than having just one log and output file created for the combined job. The following example program SASTEST01.SAS allows you to do this.

In this example, programs SASTEST01_PART1.SAS and SASTEST01_PART2.SAS are to be submitted, with the logs being directed to "c:\SAS Programs\SAS Logs\&sasprgnm..log" and the SAS output being directed to "c:\SAS Programs\SAS Lists\&sasprgnm..out".

```
/*********************************/
/* PROGRAM NAME: SASTEST01.SAS    */
/*********************************/

%let sasprgnm=sastest01_part1;
options source2; /* puts code in log */
proc printto
   log = "c:\SAS Programs\SAS Logs\
&sasprgnm..log"
   print="c:\SAS Programs\SAS Lists\
&sasprgnm..out" new;
%include "c:\SAS Programs\
&sasprgnm..sas";
proc printto;
run;

%let sasprgnm=sastest01_part2;
options pageno=1;  /* reset page # */
proc printto
   log  = "c:\SAS Programs\SAS Logs\
&sasprgnm..log"
   print="c:\SAS Programs\SAS Lists\
&sasprgnm..out" new;
%include "c:\SAS Programs\
&sasprgnm..sas";
proc printto;
run;

/* The End! */
```

DETERMINE IF A CHARACTER STRING CONTAINS ONLY NUMBERS USING THE INPUT FUNCTION AND THE SPECIAL "??" FORMAT MODIFIER

The following excerpt is from SAS OnlineDoc documentation:

> **? or ??**
>
> The optional question mark (?) and double question mark (??) format modifiers suppress the printing of both the error messages and the input lines when invalid data values are read. The ? modifier suppresses the invalid data message. The ?? modifier also suppresses the invalid data message and, in addition, prevents the automatic variable _ERROR_ from being set to 1 when invalid data are read.

Below is an example of using ?? to determine whether a variable contains non-numeric values or not:

```
data _null_;

x = “12345678”;

if (input(x, ?? 8.) eq .) then
put ‘non-numeric’;
else put ‘numeric’;

run;
```

Running SAS would return “Numeric” in the above example. If we used X=”123a5678”, SAS would return “Non-Numeric”. Note that the input format in the above example is “8.” So only the first 8 bytes of the character string are checked. Thus, X=123456789a would return “Numeric” as it would only be checking the first 8 bytes of the string.

RUN CANCEL STATEMENT

RUN CANCEL terminates the current step without executing it. SAS prints a message that indicates that the step was not executed. Following is an example of when this may be useful:

```
/*****************************/
/* PROGRAM NAME:             */
/* RUN_VALID_GENDER_ONLY.SAS  */
/*****************************/

%LET CANCEL = ;

DATA TEMP1;

LENGTH IDNO   $ 6
       GENDER $ 1
       HIRED  $ 3;

INFILE CARDS DELIMITER = ',';
INPUT IDNO GENDER HIRED;
CARDS;
001234,F,NO
001459,M,YES
002340,F,YES
003401,F,YES
004201,M,YES
004209,M,NO
005621,0,NO
;

RUN;

DATA TEMP2;
SET TEMP1;

/* Assign a value of CANCEL to the  */
/* macro variable if GENDER is not  */
/* M or F.                          */

IF (GENDER NOT IN ('M','F'))
    THEN CALL SYMPUT('CANCEL','CANCEL');

RUN;

PROC FREQ DATA=TEMP2;
TABLES GENDER*HIRED /LIST MISSING;
RUN &CANCEL;

/* The End! */
```

The previous code produces the following message in the log:

```
391
392  PROC FREQ DATA=TEMP2;
393  TABLES GENDER*HIRED /LIST MISSING;
394  RUN &CANCEL;

NOTE: The procedure was not executed at
the user's request.
NOTE: PROCEDURE FREQ used:
         real time         0.04 seconds
```

MACROS TO DISPLAY STARTING AND ENDING TIME OF A SAS JOB

Add the user written SAS macros %JOBSTART and %JOBEND to SAS programs to display the starting and ending time of your SAS jobs.

```
%JOBSTART:

%macro jobstart;
   options nonotes nosource nosource2;
   /* Macro to generate job ID, date */
   /* and start time for PC SAS jobs */
   %global s_t_a_rt d_a_t_e;
   data _null_;
      startyme = time();
      length jobid $4;
      startymc = put(startyme, 5.);
      jobid = startymc;
      jobdate = date();
      stardate = put(jobdate, 10.);
      contour = '~~~~~~~~~~~~~~~~~~~~~~~~';
      note = '%JOBSTART: ';
      put note '|' contour @35 '|';
      put note '| JOB ID    : ' jobid 4.
          '      |';
      put note '| DATE      : '
         jobdate mmddyy. ' |';
      put note '| START TIME: '
         startyme time. ' |';
      put note '|' contour @35 '|';
      call symput ('s_t_a_rt', startymc);
      call symput ('d_a_t_e', stardate);
   run;
   options notes source source2;
%mend jobstart;

%JOBEND:

%macro jobend (sendto=);
   options nonotes nosource nosource2;
   /* Macro to generate finish time */
   /* and duration for PC SAS jobs, */
   /* %jobstart must be run before. */
    data _null_;
     fintyme = time();
     durtyme = fintyme - &s_t_a_rt;

     /* if job spans days, add  */
     /* 86400 seconds per day   */
     /* correct the duration    */
     findate = date();
     numdays = findate - &d_a_t_e;
     if durtyme < 0 then
       durtyme = durtyme +
                 numdays * 86400;

    contour = '~~~~~~~~~~~~~~~~~~~~~~~~';
    note = '%JOBEND: ';

    put note '|' contour @35 '|';
```

```
    put note '| FINISH TIME : '
       fintyme time. ' |';
    put note '| JOB DURATION: '
       durtyme time. ' |';
    put note '|' contour @35 '|';

    %if %length(&sendto) > 0 %then %do;
       %let foo =
         x
g:\compsvcs\macrosas.lib\jobinfo\send to
&sendto %str(" Remote job finished ";);
    %end;

   run;

    %if %length(&sendto) > 0 %then %do;
        &foo
    %end;
options notes source source2;
%mend jobend;
```

Store commonly used SAS macros such as these into a network directory accessible by multiple staff, for example on "G:\PROGRAMMERS\MACROS\JOBINFO". Then modify the SASV8.CFG file to include this path in the SASAUTOS statement as follows:

```
-SET SASAUTOS  (
                "!sasroot\core\sasmacro"
                "!sasext0\graph\sasmacro"
                "!sasext0\stat\sasmacro"

G:\PROGRAMMERS\MACROS\JOBINFO
                 )
```

Finally, add the calls to %JOBSTART and %JOBEND to your SAS program as in the following example.

```
/* TEST_JOB.SAS */

%jobstart;

data temp1;
x = 'A';
y = 7;
run;

proc print n uniform data=temp1;
title2 'Example Showing Job Starting and
Ending Times';

%jobend;

/* The End! */
```

The above code produces the following SAS log:

```
1    /* TEST_JOB.SAS */
2
3    %jobstart;
%JOBSTART: |~~~~~~~~~~~~~~~~~~~~~~|
%JOBSTART: | JOB ID    : 5699     |
%JOBSTART: | DATE      : 06/11/01 |
%JOBSTART: | START TIME: 15:49:56 |
%JOBSTART: |~~~~~~~~~~~~~~~~~~~~~~|
4
5    data temp1;
6    x = 'A';
7    y = 7;
8    run;

NOTE: The data set WORK.TEMP1 has 1
observations and 2 variables.
NOTE: DATA statement used:
      real time           0.16 seconds


9
10   proc print n uniform data=temp1;
11   title2 'Example Showing Job Starting
and Ending Times';
12
13   %jobend;
%JOBEND: |~~~~~~~~~~~~~~~~~~~~~~|
%JOBEND: | FINISH TIME : 15:49:57 |
%JOBEND: | JOB DURATION:  0:00:01 |
%JOBEND: |~~~~~~~~~~~~~~~~~~~~~~|
14
15   /* The End! */
```

ADD YOUR OWN ICONS TO THE APPLICATION TOOLBAR ON THE MAIN SAS WINDOW

Certain repetitive commands or sequences of commands you typically perform in the main SAS window may be made easier by creating a toolbar entry and associating an icon with it. For example, if you are running SAS interactively to test and debug code, you likely run your job, examine the log and output and identify program changes. You may want to clear out the log and output windows and make the program editor window active with the last code you submitted. You can do this manually by highlighting the output window and pressing the clear button, repeating this for the log window and program editor windows. But one button is all it takes if you set up your commands cleverly.

First, a note about the SAS Version 8 editors. You have two editing options, the program editor and the enhanced editor. The enhanced editor has many advantages over the program editor such as color coding (note that limited color coding is available in the program editor starting with SAS Version 8.2) and automatic indentation. However, the enhanced editor does not have a nametag associated with it which means you cannot reference it in a toolbar command. Thus, any tool defined while the enhanced editor is the active screen can only be implemented when the enhanced editor is active. Tools defined in the output window, log window, or program editor window work the same across all three windows, displaying the same icon, but this icon will not appear in the application toolbar when the enhanced editor is the active window. For reference, tools associated with the output window, the log window, and the program editor window are saved to SASUSER.PROFILE.TOOLBOX whereas tools associated with the enhanced editor are saved to SASUSER.PROFILE.SASEDIT_MAIN. Note that you may select to use the program editor instead of the enhanced editor as the default from the main SAS window by clicking on "Tools",

"Options", "Preferences", selecting the "Edit" tab, and un-checking the "Use Enhanced Editor" box.

Lets illustrate by example where the default editor is the program editor rather than the enhanced editor. Invoke SAS 8. Highlight the program editor window (as stated above, you could highlight the log or output windows and get the same results). Click on "Tools" from the toolbar and then select "Customize". Navigate to the "Customize" tab and click the "Add tool" button:

Enter command(s), e.g.:

Command: clear log; clear out; program; recall
Help text: Clear log/output; recall last submit
Tip text: Clear log/output; recall last submit

Click the "Change icon" button to select an icon to associate this command.

Click OK.

You will be prompted to save this tool in your SAS profile.

If you instead choose to use the enhanced editor, you could set up the above tool to clear the log and output windows and then manually re-activate the enhanced editor. Note that the last statements submitted in the enhanced editor are automatically recalled when you re-activate the enhanced editor (whereas, when you re-activate the program editor, the default is for it to be empty).

DISPLAY SAS DATE AND TIME AS OF TIME PROCEDURE RAN, NOT WHEN SAS SESSION STARTED

You start up an interactive SAS session at 10:45 a.m., trying to debug your program. You run it once and print the output. The output title lines display 10:45 a.m. You notice a bug, modify your program, and rerun. This output also displays a time stamp of 10:45 a.m. You do this two more times and you have four printouts on your desk. You get called into an hour-long meeting and return to your desk confused over which version of your output is the latest. Sound familiar? Insert the following code into your title lines to display the time the procedure runs, not the time the interactive SAS session started:

```
proc print;
title1 "* Current time:
%sysfunc(putn(%sysfunc(time()),time.))*";
title2 "*** Current Date: &SYSDATE***";
```

FORCING SHORT OUTPUT FROM MULTIPLE PROCEDURES ONTO ONE PHYSICAL PAGE

Sometimes you may generate SAS procedure output that is only a few lines long. For example, you may be directed to display both the largest 10 counties by population and the smallest 10 counties by population. PROC PRINT is one of the most simply SAS procedures to use and perhaps that is all that is needed. But it will print this information on two separate pages, one for each of two PROC PRINTs. A simple way to combine the output from two (or more) SAS procedures is to use the formdlim option as shown below:

```
/*********************************/
/* Program Name: TWO_PRINTS.SAS  */
/*********************************/

options formdlim=' ';    * one space
between single quotes ;

libname in1 'd:\sas shorts\data';

proc sort data=in1.county
          (keep=name pop99)
          out=temp1;
by descending pop99;
run;

proc print n uniform data=temp1 (obs=10);
format pop99 comma9.;
title1 "Ten Counties With Largest 1999
Population";
run;

proc sort data=temp1;
by pop99;
run;

proc print n uniform data=temp1 (obs=10);
format pop99 comma9.;
title1 "Ten Counties With Smallest 1999
Population";
run;

options formdlim='';   * no spaces--ie,
reset to null value ;

/* The End! */
```

The previous example is a simple one which may solve many problems. Fancier solutions exist in SAS Version 8 using the Output Delivery System. User-defined templates may be set up to print four tables in two columns on one page if desired which is beyond the scope of this example.

PRINT VARIABLES IN A SAS DATA SET IN ALPHABETICAL ORDER USING PROC PRINT

The order of variables displayed in a PROC PRINT is the order the variables are stored on the SAS file. The following code reorders the variables alphabetically.

```
/*********************************/
/* PROGRAM NAME:                 */
/* PROC_PRINT_ALPHA_ORDER.SAS    */
/*********************************/

data temp1;
ssn = '999882222';
age = 62;
sex = 'M';
education = 'MS';
income_group = '01';
score4 = 85;
score2 = 92;
score3 = 88;
score1 = 91;

run;

proc print data=temp1;
title2 'TEMP1: Standard Proc Print';
run;

proc contents data=temp1 out=varlist
noprint;
run;

proc sql noprint;

select name into : memlist
  separated by ' ' from varlist;
run;

proc print data=temp1 (obs=1);
var &memlist;
title2 'TEMP1: Proc Print With Variables
Sorted Alphabetically';
run;

/* The End! */
```

SENDING EMAIL FROM WITHIN SAS

The SAS System Online Doc contains several pages of information and examples in the SAS Companion for the Microsoft Windows Environment section entitled “Sending Electronic Mail from within the SAS System”. It notes that you may send mail in interactive mode (e.g., run a job, highlight the output window, and mail it to a client) or in batch mode. In batch mode, you simply establish a FILENAME statement and then divert output to that filename with a FILE statement within a DATA step. An example follows:

```
/*********************************/
/* PROGRAM NAME: TEST_EMAIL.SAS  */
/*********************************/

libname in1 'd:\data';

filename out1 EMAIL
  TO=("jmccartney@s-3.com" "rhu@s-3.com")
  SUBJECT="Your SAS Job Completed!";

{DATA and/or PROC steps

/* Add a DATA step to divert to EMAIL */

data _null_;
file out1;
  put '##### TIME_STAMP.SAS Program
Completed';
  put '>>>>> Go back to office to check
results!';
run;

%jobend;

/* The End! */
```

USING THE EXPLORER WINDOW TO VIEW SELECTED VARIABLES IN THE ORDER YOU WANT

Do you take advantage of the power of the explorer window from the main SAS window to visually inspect your SAS data sets? It can be a quick and easy way to not only help you code your program by enabling you to easily see variable names (or variable labels) and the values in a spreadsheet format that is easy to navigate. You can even run queries to filter your data. Sure, you could do the same thing with a combination of PROC CONTENTS, PROC PRINT, and perhaps some simple DATA step programming. If you are not familiar with explorer, take some time to learn it. You should get a good start simply by trial and error. The explorer window gives you access to catalogs, tables, libraries, and host files. You can use it to create new libraries, file shortcuts, library members, and catalog entries. You can even use it to edit SAS files.

In any case, there is one “hidden” feature here that might come in handy, especially for those who already use explorer to examine their data. This particular tip applies for those times when you are working with SAS files with lots of variables. For example, we quite often work with medical claims data which contain several hundred variables. During our analytic programming, we may be asked to see which medical procedures are associated with given primary diagnoses, with the hypothesis that this may differ geographically. Again, when it comes time for the actual programming, we will perform these tasks using DATA step and PROC programming. However, for our first pass, inspecting the data manually has its benefits. If we point to the library containing the SAS file of medical claims and bring it up as a viewtable, we will see a spreadsheet containing hundreds of variables. The order of these variables will be the order they were created during their initial loading into SAS which generally will not be of much value. The current task is to view state, county, diagnosis code, and procedure codes for each claims record. However, these variables likely are spread across the spreadsheet such that viewing them on one screen is not possible.

There are a number of ways to get what you want but an easy way is to do the following. Assuming the viewtable window is highlighted and containing your SAS file, click “Data” on the menu bar, followed by “Hide/Unhide”. The Hide/Unhide window will display the “hidden” scroll box on the left side containing hidden variables (which will be empty initially) and a list of all variables in their positional order on the right side in the “displayed” scroll box. Simply click the left double-arrow button to move all variables onto the hidden scroll box. Voila---this list is now alphabetical rather than positional and it should now be easier to identify the variables you really want displayed. Simply navigate to each one and click the single right-arrow to move each back into the displayed scroll box. When done, click “OK”. The viewtable display will now contain just those variables you selected.

CONCLUSION

The SAS System has come a long way since 1976. As it evolves new features pop up and before we can learn them all, new releases with even more new features arrive. It is expected that most of you are familiar with one or more of the topics presented above but doubtful that many of you are familiar with all of them. Hopefully, you will realize that even the “experts” need your help, so as you discover your own valuable tips, please share them with your fellow SAS programmers.

CONTACT INFORMATION

Jeff McCartney and Raymond Hu
Social & Scientific Systems, Inc.
7101 Wisconsin Avenue, Suite 1300
Bethesda, Maryland 20814
Work: 301-986-4870
Fax: 301-652-1749
Email: jmccartney@s-3.com and rhu@s-3.com

TRADEMARK CITATION