Creating Text Output Files Automatically with SAS® Macro

S. Patrick Thornton, University of California San Francisco, San Francisco, CA

ABSTRACT

A SAS macro program was created to produce output files using PUT statements that included all variables in a given data set. The program puts data in a chosen output file using formats predefined in the data set or temporary formats chosen just prior to macro execution. The program allows customized control of several aspects of the PUT statement including the choice of data delimiter and the temporary override of data set formats either for specific variables or for all variables of a specific type. The program was developed for use in an environment lacking more sophisticated communication protocals, thus it creates custom production text files as a basis for data exchange with other programs.

INTRODUCTION

The need to automate the output of SAS data sets as text files occurred on a mainframe operating system environment where data presentation was accomplished with desktop software other than the SAS System. The production process effectively ended with dozens of programs producing results on the mainframe that were then manually transcribed to desktop software. The OUTSET macro program was an attempt to extend the production process by modularizing the output functionality, thus quickly allowing the production of downloadable output text files from many different programs. OUTSET also provides some of the custom control necessary for a reusable and scalable solution.

OUTSET PARAMETERS

OUTSET (Figure 1) accepts or requires 6 parameters for execution.

- DAT: resolves to the name of a SAS data set
- PRE: resolves to (put)
- S: resolves to a chosen delimiter, e.g. (",")
- O: resolves to the path of an output file
- FORMA: resolves to a comma delimited list of 6 format arguments, e.g. (0,0,1,z,10,1)
- N: resolves to (y) to request that variable names to be listed in the first row of the output file.

When the parameter FORMA is supplied as input to OUTSET it requires all 6 arguments with specific values supplied to achieve certain results. The values of the six arguments are listed in order, e.g. (0,0,1,z,10,1):

1) 0 lets existing format apply, 1 overrides existing format
2) 0 for global variable formatting, 1-n matching the variable (varnum) in data set to format a specific variable
3) 1 to format numeric(s), 2 to format character(s)
4) a valid SAS Format, e.g. Z
5) a valid variable length, e.g. 10
6) a valid decimal length, e.g. 2

BASIC CONFIGURATIONS

The examples presented in this paper will use the following parameters as set here and the data set TEMP:

```
%let pre = put;
%let s = %str(',');
%let nam = y;
%let outset = c:\projects\SAS Presentation\practice.txt;
option mprint serror merror nosymbolgen;
data temp;
     input num off $ type $ s1-s7 offdat mmddyy8.;
     format offdat mmddyy8.;
     format num comma10.2;
cards;
12345 robbery fel 12.4 6.8 7.2 9.0 12.2 1.7 8.9 3/12/99
76543 burglary mis 11.2 7.4 7.3 11.1 13.8 2.4 9.5 3/11/98
;
run;
```

The most basic functionality of the OUTSET macro puts the data from a data set with a chosen data delimiter to a text file. However by leaving the O parameter blank the following macro execution puts the data in the default SAS log file.

```
%outset(temp,&pre,%quote(&s),o=,forma=,
n=&nam);
```

Also, without a FORMA value the macro program uses the format, length, and decimal meta-data from TEMP to create the PUT statements. In this case COMMA and MMDDYY are used. Also, with the N parameter resolving to (y) the data is outputted with the names of the variables in the first row. The following shows the PUT statements and the resulting data format from the previous %OUTSET statement.

```
MPRINT(OUTSET):  DATA _NULL_;
MPRINT(OUTSET):  SET TEMP;
checking for user format
MPRINT(OUTSET):  IF _N_ = 1 THEN PUT "NUM" ','
"OFF" ',' "TYPE" ',' "S1" ',' "S2" ',' "S3" ','
"S4" ',' "S5" ',' "S6" ',' "S7" ',' "OFFDAT" ',' ;
MPRINT(OUTSET):  PUT NUM COMMA10.2 ',' OFF
',' TYPE ',' S1 ',' S2 ',' S3 ',' S4 ',' S5 ',' S6
',' S7 ',' OFFDAT MMDDYY8.0 ',' ;
MPRINT(OUTSET):  RUN;
```

```
NUM,OFF,TYPE,S1,S2,S3,S4,S5,S6,S7,OFFDAT,
 12,345.00,robbery ,fel ,12.4 ,6.8 ,7.2 ,9 ,12.2 ,1.7 ,8.9
,03/12/99,
 76,543.00,burglary ,mis ,11.2 ,7.4 ,7.3 ,11.1 ,13.8 ,2.4
,9.5 ,03/11/98,
```

CUSTOM FORMATTING

Beyond allowing the choices of data set, delimiter, output file, and variable names, OUTSET also accepts your choice of temporary formatting to be applied to the data through the PUT statement. This formatting does not change the format associated with the variable through the data set. The formatting choice can be applied to a specific variable or to all variables of a particular type (numeric or character). Further, OUTSET allows your choice of temporarily overriding ALL formats or allowing saved formats to be applied while overriding ONLY variables of a specific type that do not have saved formatting. This is particularly useful when dates or other numbers require special formatting while all others require a single type of formatting. Using TEMP as an example, the variables NUM and OFFDAT have saved formats of COMMA10.2 and MMDDYY8 respectively. The variables S1-S7 do not have a format specified, thus OUTSET was used to format S1-S7 for output by setting the following arguments to the macro variable F.

```
%let f = 0,0,1,dollar,10,2;
%outset(temp,&pre,%quote(&s),o=,fora=%quote(&f),n=)
```

The following portion of the SAS log file resulted from %OUTSET execution. It shows not only the resulting PUT statements and the formatted data, but also illustrates the decision points and branching that resulted from the 6 arguments of the F macro variable through statements placed by OUTSET in the SAS log file.

```
MPRINT(OUTSET):  DATA _NULL_;
MPRINT(OUTSET):  SET TEMP;
checking for user format
user format found
checking for global request type
Global Format Request
NUM is type 1
type consistent with 1
Do not override existing format
OFF is type 2
TYPE is type 2
S1 is type 1
type consistent with 1
Do not override existing format
for S1 replaced  with dollar
for S1 replaced 0 with 10
for S1 replaced 0 with 2
S2 is type 1
type consistent with 1
Do not override existing format
for S2 replaced  with dollar
for S2 replaced 0 with 10
for S2 replaced 0 with 2
S3 is type 1
type consistent with 1
Do not override existing format
for S3 replaced  with dollar
for S3 replaced 0 with 10
for S3 replaced 0 with 2
S4 is type 1
type consistent with 1
Do not override existing format
for S4 replaced  with dollar
for S4 replaced 0 with 10
for S4 replaced 0 with 2
S5 is type 1
type consistent with 1
Do not override existing format
for S5 replaced  with dollar
for S5 replaced 0 with 10
for S5 replaced 0 with 2
S6 is type 1
type consistent with 1
Do not override existing format
for S6 replaced  with dollar
for S6 replaced 0 with 10
for S6 replaced 0 with 2
S7 is type 1
type consistent with 1
Do not override existing format
for S7 replaced  with dollar
for S7 replaced 0 with 10
for S7 replaced 0 with 2
OFFDAT is type 1
type consistent with 1
Do not override existing format
for OFFDAT replaced 0 with 2
checking for specific request

MPRINT(OUTSET):  PUT NUM COMMA10.2 ',' OFF
',' TYPE ',' S1 DOLLAR10.2 ',' S2 DOLLAR10.2 ',' S3
DOLLAR10.2 ',' S4 DOLLAR10.2 ',' S5 DOLLAR10.2 ','
S6 DOLLAR10.2 ',' S7 DOLLAR10.2 ',' OFFDAT
MMDDYY8.2 ',' ;
MPRINT(OUTSET):  RUN;

 12,345.00,robbery ,fel ,   $12.40,   $6.80,   $7.20,
$9.00,   $12.20,   $1.70,
   $8.90,03/12/99,
 76,543.00,burglary ,mis ,   $11.20,   $7.40,   $7.30,
$11.10,   $13.80,   $2.40,
   $9.50,03/11/98,
```

OTHER CUSTOM FORMATTING

The following OUTSET macro execution was used to produce the output in this section while the 6 arguments of the F macro variable were changed to produce various output. For example, the following configuration lets existing format apply and requests a global variable change for character variables:

```
%let f = 0,0,2,Char,10,2;
%outset(temp,&pre,%quote(&s),o=,fora=%quote(&f),n=)

MPRINT(OUTSET):  PUT NUM COMMA10.2 ',' OFF
CHAR10.2 ',' TYPE CHAR10.2 ',' S1 ',' S2 ',' S3 ','
S4 ',' S5 ',' S6 ',' S7 ',' OFFDAT MMDDYY8.0 ',' ;
```

This configuration lets existing format apply and requests specific variable change for character variable number 2. The output here includes the custom OUTSET macro log statements.

```
%let f = 0,2,2,Char,20,0;

MPRINT(OUTSET):  DATA _NULL_;
MPRINT(OUTSET):  SET TEMP;
checking for user format
user format found
checking for global request type
checking for specific request
Specific change
variable # 2 OFF found
variable OFF is type 2
type consistent with 2
Do not override existing format
for OFF replaced  with Char
for OFF replaced 0 with 20
for OFF replaced 0 with 0
MPRINT(OUTSET):  PUT NUM COMMA10.2 ',' OFF
CHAR20.0 ',' TYPE ',' S1 ',' S2 ',' S3 ',' S4 ','
S5 ',' S6 ',' S7 ',' OFFDAT MMDDYY8.0 ',' ;
MPRINT(OUTSET):  RUN;
```

The next configuration overrides existing formats and requests a global variable change of numeric variables (this is only for example, as the date numeric is formatted as a currency):

```
%let f = 1,0,1,dollar,10,2;

MPRINT(OUTSET):  PUT NUM DOLLAR10.2 ',' OFF
',' TYPE ',' S1 DOLLAR10.2 ',' S2 DOLLAR10.2 ','
S3 DOLLAR10.2 ',' S4 DOLLAR10.2 ',' S5
DOLLAR10.2 ',' S6 DOLLAR10.2 ',' S7 DOLLAR10.2 ','
OFFDAT DOLLAR10.2 ',' ;
MPRINT(OUTSET):  RUN;
```

CONCLUSION

The OUTSET macro program has proven useful in efficient production of text files from SAS data sets to allow quick transfer of data to other software applications. OUTSET is most useful when other more sophisticated communication techniques are not available. OUTSET has not been beta tested with all possible SAS formats. If errors occur in the application of format, length, and decimal combinations please contact the author as such information will be valuable to improving the robustness of the macro program.

REFERENCES

SAS Institute Inc.(1997). SAS macro language: Reference, First Edition. Cary, NC: SAS Institute Inc.

TRADEMARKS

SAS software is a registered trademark of trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

ACKNOWLEDGEMENTS

Thanks to Sarah M. Barry for editing.

CONTACT INFORMATION

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

S. Patrick Thornton
Child Services Research Group
44 Montgomery Street Suite 1450
San Francisco, CA 94104-4602
(415) 502-8004
SPThornton1@juno.com

FIGURE 1. OUTSET MACRO PROGRAM

```
%MACRO outset(DAT,pre,s,o=,fora=,n=);
DATA DUMM;
 PROC CONTENTS DATA=&DAT OUT=GETFOR;
 PROC SORT DATA=GETFOR;
  BY VARNUM
 DATA _NULL_;
  SET GETFOR;
  CALL SYMPUT('VAR'||TRIM(LEFT(VARNUM)), TRIM(LEFT(NAME)));
  CALL SYMPUT('TYP'||TRIM(LEFT(VARNUM)), TRIM(LEFT(TYPE)));
  CALL SYMPUT('FOR'||TRIM(LEFT(VARNUM)), TRIM(LEFT(FORMAT)));
  CALL SYMPUT('FORI'||TRIM(LEFT(VARNUM)), TRIM(LEFT(FORMATI)));
  CALL SYMPUT('FORd'||TRIM(LEFT(VARNUM)), TRIM(LEFT(FORMATD)));
  CALL SYMPUT('VARCT',VARNUM);
 run;
 data _null_;
   set &dat;
   %put %str(checking for user format);
   %if %quote(&fora) ne  %then %do;
       %put %str(user format found);
       %let k = 0;
       /*parses the format commands to v1-v6 macro variables */
       %do %until (&p = );
          %let k = %eval(&k + 1);
          %let p = %scan(%quote(&fora),%eval(&k));
          %let v&k = &p;
          %let ct = &k;
       %end;
       %put %str(checking for global request type);
       /*condition of a global format request */
       %if &v2 = 0 %then %do;
          %put %str(Global Format Request);
         /* for each variable */
         %do j = 1 %to &varct;
          %put &&var&j %str(is type) &&typ&j;
          %if &&typ&j = &v3 %then %do;
              %put %str(type consistent with) &v3;
            /* if existing format then do not override */
            %if &v1 = 0 %then %do;
             %put %str(Do not override existing format);
             %if &&for&j = %then %do;
                    %put %str(for) &&var&j %str(replaced) &&for&j %str(with) &v4;
               %let for&j = &v4;
             %end;
             %if &&forI&j = 0 %then %do;
                %put %str(for) &&var&j %str(replaced) &&forI&j %str(with) &v5;
                %let forI&j = &v5;
             %end;
             %if &&ford&j = 0 %then %do;
               %put %str(for) &&var&j %str(replaced) &&ford&j %str(with) &v6;
               %let ford&j = &v6;
             %end;
            %end;
            %else %if &v1 = 1 %then %do;
              %put %str(Override existing format);
              %put %str(for) &&var&j %str(replaced) &&for&j %str(with) &v4;
               %let for&j = &v4;
              %put %str(for) &&var&j %str(replaced) &&forI&j %str(with) &v5;
               %let forI&j = &v5;
              %put %str(for) &&var&j %str(replaced) &&ford&j %str(with) &v6;
               %let ford&j = &v6;
            %end;
          %end;
        %end;
      %end;
```

```
        /*condition of a specific variable override */
       %put %str(checking for specific request);
       %if &v2 ne 0 %then %do;
         /* for each variable */
        %put %str(Specific change);
       %do j = 1 %to &varct;
          /* if variable number matches user specified number */
         %if &j = &v2 %then %do;
           %put %str(variable #) &v2 &&var&j %str(found);
          /* condition that variable type is consistent with user specification */
          %put %str(variable) &&var&j %str(is type) &&typ&j;
          %if &&typ&j = &v3 %then %do;
              %put %str(type consistent with) &v3;
             /* if existing format then do not override */
             %if &v1 = 0 %then %do;
              %put %str(Do not override existing format);
              %if &&for&j = %then %do;
               %put %str(for) &&var&j %str(replaced) &&for&j %str(with) &v4;
               %let for&j = &v4;
              %end;
              %if &&forl&j = 0 %then %do;
                %put %str(for) &&var&j %str(replaced) &&forl&j %str(with) &v5;
                %let forl&j = &v5;
              %end;
              %if &&ford&j = 0 %then %do;
               %put %str(for) &&var&j %str(replaced) &&ford&j %str(with) &v6;
               %let ford&j = &v6;
              %end;
            %end;
            %else %if &v1 = 1 %then %do;
              %put %str(Override existing format);
              %put %str(for) &&var&j %str(replaced) &&for&j %str(with) &v4;
              %let for&j = &v4;
              %put %str(for) &&var&j %str(replaced) &&forl&j %str(with) &v5;
              %let forl&j = &v5;
              %put %str(for) &&var&j %str(replaced) &&ford&j %str(with) &v6;
              %let ford&j = &v6;
            %end;
          %end;
        %end;
       %end;
     %end;
   %end;
  %end;
  /* if output file path is specified. */
  %if &o ne  %then %str(file "&o";);
  /* if name of the variables is requested as first row */
  %if &n ne  %then %do;
   %str(if _n_ = 1 then &pre)
   %do k = 1 %to &varct;
    %str("&&var&k" &s)
   %end;
   %str(;)
  %end;
  %str(&pre)
  %do k = 1 %to &varct;
      %str(&&var&k)
      /* if format width is not 0 for variable */
      %if &&forl&k ne 0 %then %do;
        %str(&&for&k)%sysevalf(&&&forl&k%str(.)&&ford&k)
      %end;
      %else %if &&for&k ne %then %do;
        %str(&&for&k).
      %end;
      &s
  %end;
  %str(;)
run;  %mend;
```