

Creating an Autocall Macro to Convert SAS® Datasets to Text Delimited Files

Doyle McDonald, Technology Support, Texas Education Agency

Abstract

The Technology Support division of the Texas Education Agency responds daily to many clients requesting data on diskette in a standard format rather than receive the information on a traditional paper report. Our need to satisfy these data requests quickly and efficiently while maintaining some flexibility demanded an automated solution. The Autocall macro facility of the SAS System provided us with the means by which flexible, generic SAS dataset conversion macros could be easily created, maintained and made available for agency use.

With these macros, SAS datasets or views with hundreds of variables can be converted to text delimited or fixed field sequential files with just a few lines of code. No longer is it necessary to code line after line of tiresome PUT statements just to create a standard text file. Also, the burden of creating an appropriate file layout is also removed, as these macros will create an accompanying descriptor file automatically.

This paper relates only to the macro for SAS dataset conversion to a text delimited file. A similar paper relating to SAS dataset conversion to a fixed field format is available upon request.

The examples contained herein were written for the MVS operating system, but operating system specific code can be easily changed to accommodate other platforms.

Text Delimited Format: The %SAS2TXT Macro

The %SAS2TXT macro was designed to create text files from SAS datasets or views. Delimiters can be either the tab or the comma character. Quoting of variables is optional. Because SAS stores date values as integers, date variables should be converted from numeric to character variables and identified as such prior to macro invocation. This macro directs output to a partitioned data set (PDS) for both the data and the descriptor file.

Syntax for this macro is as follows:

```
%SAS2TXT(SASDSN= libref.dataset(dataset options),  
        PDSREF= fileref,  
        PDSMEM= member name,  
        LAYOUT= member name,  
        QUOTES= value,  
        DELIM = value,  
        PGM   = value);
```

Valid values for these parameters are:

```
%SAS2TXT(SASDSN= any SAS dataset or view,  
        PDSREF= file reference to any previously allocated partitioned dataset,  
        PDSMEM= any partitioned dataset member name- 8 bytes or less,  
        LAYOUT= any partitioned dataset member name- 8 bytes or less,  
        QUOTES= C=character only B=both character and numeric N=none,  
        DELIM = COMMA or TAB,  
        PGM   = any text string 40 bytes or less, excluding reserved characters);
```

Defaults are resolved for any missing parameters as follows:

```
%SAS2TXT(SASDSN= LAST_,
          PDSREF= SYSPC,
          PDSMEM= FILE1,
          LAYOUT= DESC1,
          QUOTES= C,
          DELIM = COMMA,
          PGM   = &SYSJOBID);
```

Output Data Files

Output data files contain nothing but the data itself, formatted in the manner specified. Header records are not included.

The following example was produced by the %SAS2TXT macro from the HOUSES sample dataset, using default values for quotes and delimiter.

```
"RANCH","Sheppard Avenue",1250,2,1,64000
"SPLIT","Rand Street",1190,1,1,65850
"CONDO","Market Street",1400,2,1.5,80050
"TWOSTORY","Garris Street",1810,4,3,107250
"RANCH","Kemble Avenue",1500,3,3,86650
"SPLIT","West Drive",1615,4,3,94450
"SPLIT","Graham Avenue",1305,3,1.5,73650
"CONDO","Hampshire Avenue",1390,3,2.5,79350
"TWOSTORY","Sanders Road",1040,2,1,55850
"CONDO","Jeans Avenue",2105,4,2.5,127150
"RANCH","State Highway",1535,3,3,89100
"TWOSTORY","Fairbanks Circle",1240,2,1,69250
"RANCH","Nicholson Drive",720,1,1,34550
"TWOSTORY","Highland Road",1745,4,2.5,102950
"CONDO","Arcata Avenue",1860,2,2,110700
```

Descriptor Files

Descriptor files are files created by these macros which provide a description and layout of the output data file. Header information includes the name, number of records, estimated number of bytes, references to quoting and type of delimiter used. Variable information includes position in data file, name, type, maximum length and description of variable. Variable order is determined first by type (character variables are always first, when present) and then by the order as defined in the SAS dataset or view.

The following example was produced by the %SAS2TXT macro from the HOUSES sample dataset.

```
PROGRAM REFERENCE: @DAM1234
RECORDS IN FILE   : 15
APPROX. FILE SIZE: 628 BYTES
QUOTES            : CHARACTER ONLY
DELIMITER         : COMMA
```

POS	NAME	TYPE	MAX LEN	DESCRIPTION
001	STYLE	CHARACTER	008	Style of homes
002	STREET	CHARACTER	016	Street address
003	SQFEET	NUMERIC	015	Square footage
004	BEDROOMS	NUMERIC	015	Number of bedrooms
005	BATHS	NUMERIC	015	Number of bathrooms
006	PRICE	NUMERIC	015	Asking price

The SAS Autocall Macro Library

The SAS Autocall Macro Library is a partitioned dataset that contains various macros supplied by the SAS Institute which are available to all SAS users for a given installation site. These macros require no definition by the programmer, other than the respective macro call. This method of macro invocation is beneficial in that a macro that is of common interest can be maintained in a central location and made available to all users.

Upon invocation of the SAS System, SAS identifies this library from the SASAUTOS DDNAME. Site defined macros can be stored in this library along with those supplied by the SAS Institute. The SAS System option MAUTOSOURCE must be specified in order to use this feature. The NOMAUTOSOURCE option disables this feature. With this option specified, when a macro call is instructed, SAS first searches the SASMACR catalog in the WORK library, and, if not found, then proceeds to search the SASAUTOS library for the macro being called. Once found, the macro code is then included in the code stream.

A alternate method of storage for site defined autocall macros is to create a partitioned dataset of like attributes, and store these macros as members in this PDS. By doing this, maintenance issues are simpler with no loss in functionality. This site defined autocall library can either replace or complement the default autocall library, by either overriding the SASAUTOS DDNAME with the replacement PDS name, or by appending it to the same. JCL examples of both methods are:

To override: //SASAUTOS DD DSN=.SITE.DEFINED.SAS.AUTOLIB,DISP=SHR

To complement: //SASAUTOS DD DSN=SITE.SAS608.AUTOLIB,DISP=SHR
 // DSN=.SITE.DEFINED.SAS.AUTOLIB,DISP=SHR

It is important to note that subsequent calls to the same autocall macro within the same program or session will retrieve this macro from the WORK.SASMACR catalog, and not again from the original autocall library. This is quite important to know when developing autocall macros interactively.

Dissecting the %SAS2TXT macro

To explain the workings of this macro, I have broken it down into the following sections:

1. %MACRO statement and comments
2. Setting QUOTE values
3. Setting the delimiter and other defaults
4. The first DATA step: Creating the data file
5. Capturing CONTENTS of the dataset for the descriptor file
6. The second DATA step: Creating the descriptor file
7. Housekeeping and the %MEND statement

Section 1 - %MACRO statement and comments

Line 1 is the required macro statement with the named macro parameters following. The remainder, lines 2 through 21 are macro comments. Line 22 is somewhat site specific as we often use the macro variable PGMID in other programs as well. The %GLOBAL statement essentially tells this macro to resolve this variable from the global symbol table.

```
0001 %MACRO SAS2TXT(SASDSN=,PDSREF=,PDSMEM=,LAYOUT=,QUOTES=,DELIM=,PGM=);
0002 %*-----*
0003 %* THIS MACRO PROVIDES FOR GENERIC CONVERSION OF SAS DATASET TO A      *;
0004 %* PARTITIONED DATASET IN ASCII DELIMITED FORMAT.  ARGUMENTS ARE:      *;
0005 %*                                                                 *;
0006 %* SASDSN=LIBREF.SASDSN SOURCE DATASET                                (DEFAULT=_LAST_)*;
```

```

0007 %* PDSREF=FILeref      TARGET PDS FILE REFERENCE      (DEFAULT=SYSPC) %*;
0008 %* PDSMEM=MEMBER       TARGET PDS MEMBER              (DEFAULT=FILE1) %*;
0009 %* LAYOUT=MEMBER      TARGET PDS MEMBER LAYOUT        (DEFAULT=DESC1) %*;
0010 %* QUOTES=N,C,B       NONE,CHAR VARS,BOTH CHAR/NUM    (DEFAULT=C) %*;
0011 %* DELIM =COMMA,TAB   DELIMITER                      (DEFAULT=COMMA) %*;
0012 %*
0013 %* THIS MACRO ASSUMES THAT THE TARGET PARTITIONED DATASET HAS BEEN %*;
0014 %* PREVIOUSLY ALLOCATED. %*;
0015 %*-----*
0016 %* 04/13/93                                MCDONALD %*;
0017 %*-----*
0018 %* 04/19/94  REVISED:                                MCDONALD %*;
0019 %*          ADDED PROGRAM IDENTIFICATION, NUMBER OF RECORDS, %*;
0020 %*          QUOTING AND DELIMITER REFERENCES IN DESCRIPTOR FILE %*;
0021 %*-----*
0022 %GLOBAL PGMID;

```

Section 2 - Setting quote values

Lines 23 through 37 create macro variables that will later set quote values for both character and numeric variables based upon values passed in the macro call.

```

0023 %IF      &QUOTES=N %THEN
0024 %DO;
0025 %LET CQUOTE=;
0026 %LET NQUOTE=;
0027 %END;
0028 %ELSE %IF &QUOTES=B %THEN
0029 %DO;
0030 %LET CQUOTE=QUOTE;
0031 %LET NQUOTE=QUOTE;
0032 %END;
0033 %ELSE
0034 %DO;
0035 %LET CQUOTE=QUOTE;
0036 %LET NQUOTE=;
0037 %END;

```

Section 3 - Setting the delimiter and other defaults

Line 38 tests for values of the PGMID global macro variable, and sets it to the system job identification if not found. Line 39 will set the PGM local macro variable to the PGMID value if a value was not passed in the PGM parameter. Lines 40, 41, 46 and 47 set defaults for the delimiter and the text value which is used later in the descriptor file (This could have been done better). Line 42 sets the default SAS dataset to the last referenced SAS dataset in the program. Line 43 sets the default partitioned dataset file reference, which is likely to be site specific.

```

0038 %IF %LENGTH(&PGMID) =0 %THEN %LET PGMID  =&SYSJOBID;
0039 %IF %LENGTH(&PGM)   =0 %THEN %LET PGM     =&PGMID;
0040 %IF %LENGTH(&DELIM) =0 %THEN %LET DELIMTXT=COMMA;
0041 %ELSE %LET DELIMTXT=&DELIM;
0042 %IF %LENGTH(&SASDSN)=0 %THEN %LET SASDSN  =_LAST_;
0043 %IF %LENGTH(&PDSREF)=0 %THEN %LET PDSREF  =SYSPC;
0044 %IF %LENGTH(&PDSMEM)=0 %THEN %LET PDSMEM  =FILE1;
0045 %IF %LENGTH(&LAYOUT)=0 %THEN %LET LAYOUT  =DESC1;
0046 %IF &DELIM=TAB      %THEN %LET DELIM    ='05'X;
0047 %ELSE %LET DELIM    =',';

```

Section 4 - The first DATA step: Creating the data file

Line 49 and 50 resolve macro variables for SAS dataset, PDS file reference and PDS member name for the data file. Lines 51 and 52 are what allow this entire process to work; arrays for all character and all numeric variables in the dataset. If no character or no numeric variables exist in the dataset, a warning will be written to the log stating 'Warning: defining an array with zero elements'. This has no adverse effect and should be ignored if encountered. Lines 53 and 54 identify temporary variables that are not to be included in the output file, and their position following the array statements are mandatory- otherwise these variables would be included in the output dataset. Line 55 initializes the VPOS (variable position) variable to 1, and DPOS (delimiter position) to zero.

```
0048 DATA _NULL_;
0049 SET &SASDSN END=LAST;
0050 FILE &PDSREF(&PDSMEM);
0051 ARRAY CHARVAR $ _CHARACTER_;
0052 ARRAY NUMVAR _NUMERIC_;
0053 LENGTH TEMPCHAR $200 TEXTBYTE TEXTROWS $15 VPOS DPOS 8;
0054 RETAIN BYTES ROWS 0;
0055 VPOS=1;
0056 DPOS=0;
```

Lines 57 through 72 export the character variables to the output file, calculate the approximate number of bytes written, and count the number of records (ROWS) written. Lines 57 and 58 test for the existence of character variables, and if found, set up the loop for the appropriate number of variables using the DIM function. Line 59 builds the string to export by first compressing out commas and double quotes, and then applying quotes based upon the value of the macro variable CQUOTE, which resolves to become the QUOTE function if specified by the QUOTES parameter in the macro call. Line 60 tests to see if this is the last variable in the dataset, meaning that no numeric variables exist, and if true executes a do group in lines 61 through 65. This do group exports the string to the output file and calculates bytes and rows written. If not true, the ELSE do group executes (lines 66 through 71), meaning that numeric variable do follow. In this instance, (68) delimiter position is calculated, (69) the string is exported to the output file at the position specified by the VPOS variable, the delimiter is exported to the file at the DPOS position (as specified by &DELIM), and then the line is held in anticipation of the numeric variables that will follow. Line 70 recalculates VPOS. The do groups then end (71,72).

```
0057 IF DIM(CHARVAR) > 0 THEN
0058 DO I = 1 TO DIM(CHARVAR);
0059 TEMPCHAR=&CQUOTE(COMPRESS(CHARVAR{I},','));
0060 IF DIM(NUMVAR)=0 AND I=DIM(CHARVAR) THEN /*LAST VARIABLE*/
0061 DO;
0062 PUT @VPOS TEMPCHAR;
0063 BYTES=BYTES+(VPOS+(LENGTH(TRIM(TEMPCHAR)))+1);
0064 ROWS=ROWS+1;
0065 END;
0066 ELSE
0067 DO;
0068 DPOS=DPOS+LENGTH(TRIM(TEMPCHAR))+1;
0069 PUT @VPOS TEMPCHAR @DPOS &DELIM @;
0070 VPOS=DPOS+1;
0071 END;
0072 END;
```

Lines 73 through 88 do basically the same operation for numeric variables as was done in the previous group for character variables. One item to note, and possibly to improve upon, is the default numeric to character conversion in Line 75. I found that I could not format generically as well as SAS does, so I let SAS do the conversion. Purists may not like the note that will be found in the log indicating SAS made this conversion.

```

0073 IF DIM(NUMVAR) > 0 THEN
0074 DO I = 1 TO DIM(NUMVAR);
0075   TEMPCHAR=&QUOTE(COMPRESS(TRIM(LEFT(NUMVAR{I})),','));
0076   IF I = DIM(NUMVAR) THEN /*LAST VARIABLE*/
0077     DO;
0078       PUT @VPOS TEMPCHAR;
0079       BYTES=BYTES+(VPOS+(LENGTH(TRIM(TEMPCHAR))+1);
0080       ROWS=ROWS+1;
0081     END;
0082   ELSE
0083     DO;
0084       DPOS=DPOS+LENGTH(TRIM(TEMPCHAR))+1;
0085       PUT @VPOS TEMPCHAR @DPOS &DELIM @;
0086       VPOS=DPOS+1;
0087     END;
0088   END;

```

The first data step concludes by creating two macro variables; BYTES and ROWS. Line 89 tests for the last observation in the dataset by checking the value of the END= variable (see line 49) supplied by the SET statement. If true (89) the do group (90-102) executes. Numeric values for data step variables BYTES and ROWS are converted to character. One byte is added for an end-of-file marker (91) and a text string is created to specify megabytes, kilobytes or bytes for readability. The text strings are stored in these macro variables and are later resolved in the creation of the header portion of the descriptor file. The first data step ends.

```

0089 IF LAST=1 THEN
0090 DO;
0091   BYTES=BYTES+1;
0092   IF BYTES GE 1024000 THEN
0093     TEXTBYTE=PUT((BYTES/1024000),5.1)||' MB';
0094   ELSE
0095     IF BYTES GE 1024 THEN
0096       TEXTBYTE=PUT((BYTES/1024),5.1)||' KB';
0097   ELSE
0098     TEXTBYTE=PUT(BYTES,5.1)||' BYTES';
0099   TEXTROWS=PUT(ROWS,COMMA10.);
0100   CALL SYMPUT('BYTES',LEFT(TEXTBYTE));
0101   CALL SYMPUT('ROWS',LEFT(TEXTROWS));
0102 END;
0103 RUN;

```

Section 5 - Capturing CONTENTS of the dataset for the descriptor file

Lines 104 through 108 run the CONTENTS procedure, creating a temporary dataset containing the variables necessary to create the descriptor file. Because &SASDSN may reference a view, NOBS is not in the KEEP dataset option (Line 106) because it would produce a missing value when referencing a view. This is why ROWS were calculated in the prior data step. Lines 109 through 111 sort this dataset by descending type and (ascending) variable number. Type 2 variables are character, but character variables were first in order of output, hence the descending sort order.

```

0104 PROC CONTENTS
0105 DATA=&SASDSN
0106 OUT =WORK.TEMPCONT(KEEP=NAME TYPE LENGTH VARNUM LABEL)
0107 NOPRINT;
0108 RUN;
0109 PROC SORT DATA=WORK.TEMPCONT;
0110 BY DESCENDING TYPE VARNUM;
0111 RUN;

```

Section 6- The second DATA step: Creating the descriptor file

Lines 112 through 141 create the descriptor file. Line 117 generically sets the maximum numeric length to 15, however this may be insufficient for some data. We chose 15 as the expected maximum for our purposes; however, your data may necessitate changing this value. Line 118 identifies the descriptor file, resolving values for the PDS file reference and the PDS member. Lines 119 through 135 create the header portion of the descriptor file, resolving such macro variables as ROWS, PGM, BYTES, QUOTES and DELIMTXT. Line 136 writes the relative position of the variable in the dataset, the variable name and then holds the line pointer. Lines 137 through 140 completes writing the variable information, conditionally identifying variable type. Line 141 ends the step.

```

0112 DATA _NULL_;
0113   RETAIN POS 0;
0114   SET WORK.TEMPCONT;
0115   LENGTH NOBTEXT $10;
0116   POS + 1;
0117   IF TYPE=1 THEN LENGTH=15;
0118   FILE &PDSREF(&LAYOUT) NOPRINT;
0119   IF _N_=1 THEN
0120     DO;
0121       NOBTEXT=LEFT("&ROWS");
0122       PUT @01 'PROGRAM REFERENCE:' @21 "&PGM"
0123         / @01 'RECORDS IN FILE   :' @21 NOBTEXT
0124         / @01 'APPROX. FILE SIZE:' @21 "&BYTES"
0125         / @01 'QUOTES           :' @;
0126       IF "&QUOTES"='B' THEN PUT @21 'CHARACTER AND NUMERIC';
0127       ELSE IF "&QUOTES"='N' THEN PUT @21 'NONE';
0128       ELSE PUT @21 'CHARACTER ONLY';
0129       PUT @01 'DELIMITER         :' @21 "&DELIMTXT";
0130     PUT;
0131     PUT @01 'POS' @05 'NAME' @14 'TYPE'
0132       @24 'MAX LEN' @32 'DESCRIPTION'
0133       / @01 '----' @05 '-----' @14 '-----'
0134       @24 '-----' @32 '-----';
0135     END;
0136     PUT @01 POS Z3. @05 NAME @;
0137   IF TYPE=2 THEN
0138     PUT @14 'CHARACTER' @26 LENGTH Z3. @32 LABEL;
0139   ELSE
0140     PUT @14 'NUMERIC ' @26 LENGTH Z3. @32 LABEL;
0141   RUN;

```

Section 7 - Housekeeping and the %MEND statement

Lines 142 through 144 perform housekeeping by dropping the temporary work dataset created by the CONTENTS procedure. Line 145 ends the macro definition.

```

0142 PROC SQL;
0143   DROP TABLE WORK.TEMPCONT;
0144   QUIT;
0145 %MEND SAS2TXT;

```

Precautionary notes for the %SAS2TXT macro

- SAS datasets store date values as numeric; whole integers representing the number of days since January 1, 1960. This macro does not use a date format when exporting the value to the output text file, and therefore will write the date as a whole integer. It is recommended that date variables be converted to character in the date format desired, and dropping the numeric variable when calling this macro. It may be necessary to create an

SQL view of the dataset, excluding date variables and creating character variables representative of the date, in order to export appropriate date values. The select statement can explicitly identify only the variables desired, but can be quite an effort with datasets that have a large number of variables. If this is the case, consider the following example (otherwise see the next example):

```
PROC SQL;
  CREATE VIEW WORK.RETAIL AS
  SELECT *,
    PUT (DATE,MMDDYY8.) AS CHARDATE LABEL='month/day/year'
  FROM SASUSER.RETAIL;
QUIT;

%$SAS2TXT(SASDSN=WORK.RETAIL(DROP=DATE));
```

- The order of creation is always character variables first, numeric second. Within variable type, the order is the same as the order found in the dataset or view. If a specific order within variable type is required, creating an SQL view prior to calling the macro and specifying the order desired in the select statement will produce the desired order. Consider the following example.

```
PROC SQL;
  CREATE VIEW WORK.HOUSES AS
  SELECT STREET, STYLE, PRICE, BEDROOMS, BATHS, SQFEET
  FROM SASUSER.HOUSES;
QUIT;

%$SAS2TXT;
```

- The maximum length of the output record should not exceed the logical record length (LRECL) of the target partitioned dataset, nor should it exceed the value of the SAS linesize option if the partitioned dataset's record format (RECFM) is equal to VBA. If either of these conditions exist, the formatting of the output text file will be corrupted. We recommend the following attributes for the target partitioned dataset:

DSORG=PO RECFM=VB LRECL=9072 BLKSIZE=9076

- This macro assumes that the output partitioned dataset exists and has been previously allocated.

REFERENCES

SAS Language, Version 6, First Edition
SAS Guide to the Macro Processing, Version 6, Second Edition
SAS Guide to the SQL Procedure, Version 6, First Edition
SAS Companion to the MVS Environment, Version 6, First Edition

SAS Institute, Inc., Cary, NC

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

® indicates USA registration.

If you have any questions, comments or problems with the concepts or methods contained within this paper, please feel free to write, FAX or E-Mail the author at:

Doyle McDonald
Mainframe Client Support Manager
Technology Support Division
Texas Education Agency
1701 North Congress Avenue
Austin, Texas 78701

FAX: 512/475-3664

Internet: doylem@tenet.edu

CompuServe: 73612,470