**Paper PO13**

# Methods for Generating Excel Files from SAS Datasets

Xingshu Zhu and Shuping Zhang
Merck Research Laboratories, Merck & Co., Inc., Blue Bell, PA 19422

## ABSTRACT

When processing data in pharmaceutical research, it is often necessary for a SAS programmer to transform a SAS dataset to an Excel file format. Several methods for performing this transformation have been published in previous SAS conference proceedings. In this paper, we briefly introduce each of these methods and offer a critical assessment of the advantages and disadvantages of each approach. We also provide recommendations on the appropriate method to use when a SAS dataset must be transformed into an Excel file under a specific set of circumstances.

## KEYWORDS

SAS dataset, Excel file

## INTRODUCTION

Excel spreadsheets are often used to present the results of data analysis in pharmaceutical companies. It is easier to use Excel to manipulate, organize and format data than to involve complicated programming techniques.
In fact, with over 1200 macro examples in Microsoft Windows Excel, a programmer is able to efficiently find and remove duplicates within an Excel worksheet and to easily create cascade charts.
However, in order for pharmaceutical companies to benefit from the convenience of the Excel format, programmers must confront the task of transforming SAS datasets into Excel files. To accomplish this task, they have several data transformation approaches available to them. The most popular and widely used of these approaches is the Dynamic Data Exchange method, known as DDE, which is explained and evaluated in this paper. Over the past few years, we have also collected, modified and even developed other methods for creating Excel files from SAS datasets. The goal of this paper is to provide guidance for users about these methods by reviewing and assessing the different options available to perform the task of transforming a SAS dataset into an Excel spreadsheet.

## METHOD 1: USING DDE TO CONVERT A SAS DATASET INTO AN EXCEL FILE IN A DATA STEP

In a DATA step, a programmer can use the DDE approach to write a complete dataset to an external file, using all variables or only selected variables. To demonstrate this approach, we begin with an example of a simple and small SAS dataset, RXPK, which is stored in the directory *c:\paper2005\data* and which contains only a few variables. The following is a SAS program that makes use of DDE to automate the transfer of SAS data to Excel. The filename statement is used to create an Excel file called *rx* in the desired directory, and the data null statement is used to write the components of the data to the Excel file.

```
filename rx DDE "Excel|C:\paper2005\DATA\[primerx.xls]sheet1!R1C1:R101C4";
Libname datadir "c:\paper2005\data\";
data _null_;
   file rx NOTAB;
```

```
   set datadir.rxpk;
   if _n_=1 then
   put 'AN'    '09'x  Treatment'  '09'x ' Start Date'      '09'x 'Stop Date'       '09'x;
   put alloc   $ '09'x  treatment $ '09'x  start_dt date9. '09'x  stop_dt date9. '09'x;
run;
```

```
Notes:
NOTAB – allows an entire character string including embedded blanks to be stored in a cell.
"09"x   – means tab delimited.
```

One benefit of this DDE approach is that it is a simple and straightforward way to perform the "one-to-one" conversion from a SAS dataset to an Excel file. In addition, with this method, a user can easily create and modify a variable's label by simply using the PUT statement to generate a user-friendly format.

However, this method also has its limitations. First, it only works well for a dataset with a small number of variables. As the number of variables increases, it becomes tedious and cumbersome to write out all the variables in the DATA step. In addition, b*efore* executing the SAS conversion program with method 1, it is necessary to open and save an empty Excel file in a directory. Conversely, *after* executing the SAS conversion program, it is necessary to save the converted Excel file once again.

## METHOD 2: THE CUSTOM-MADE MACRO %SAS2XLS

The macro %SAS2XLS, published in the proceedings of PharmaSUG 2000, can be used to read in a SAS dataset automatically and convert it to an Excel file. This transformation can take place once the directory and path information of the Excel file are provided in the macro's call parameters. These four parameters are listed below, followed by a description of their usage.

```
%macro sas2xls(
              sasds =
             ,xlsdir=
             ,xlsds =
             ,sheet =
             );
```

sasds : the name of the input SAS dataset (one or two level).
Xlsdir: the physical path of the folder that stores the Excel data file.
xlsds : the name of the Excel data file, and the xls extension should not be included.
sheet : the name of the sheet in the Excel data file.

The key component of macro %SAS2XLS is, again, the use of DDE to output a SAS dataset into an Excel spreadsheet. The following is the set of codes for this key component:

```
filename xlsfile DDE "Excel|&xlsdir&delim.[&xlsds..xls]&sheet!r1c1:r%eval(&n_obs+1)c&n_var";

data _null_;
     set &sasds;
     file xlsfile notab lrecl=&lrecl;
     if _N_=1 then
     put %do i=1 %to &n_var; "&&vname&i"          '09'x %end;;
     put %do i=1 %to &n_var;  &&vname&i &&vfmt&i '09'x %end;;
run;
```

This set of codes is very similar to the codes listed above in the previous section of this paper on Method 1, the DDE approach to transforming SAS datasets into Excel format. The major difference created by using macro %SAS2XLS is that the information from the datasets, such as total number of variables, name and format corresponding to each variable, and total number of observations, is all represented through the corresponding macro parameters. These parameters are filled by retrieving basic dataset information from the PROC CONTENTS of the SAS dataset. In contrast, Method 1 requires manual typing of the dataset information.

Clearly, the macro %SAS2XLS used in Method 2 offers a great improvement in efficiency over Method 1. Since the macro is able to convert a SAS dataset directly into an Excel file easily by simply filling in the parameters in the macro call, the need for manual typing of the dataset information is eliminated.

Despite its improvement in efficiency, Method 2 does have some disadvantages. It requires that the location of EXCEL.EXE be specified on the computer. It also involves a waiting time of five seconds or more for opening the MS Excel during the execution of the program.

## METHOD 3: MACRO %SAS2CSV.SAS

The macro %SAS2CSV is an updated version of the macro %SAS2XLS, which was described above in Method 2. Macro %SAS2CSV applies the same methodology used by the previous macro, with modifications for improved efficiency. The macro call for %SAS2CSV is listed below, along with a description of how the macro parameters are used.

```
%macro sas2csv(
              sasds   =
             ,outdir  =
             ,outds   =
             ,delimit =
             );
```

sasds : the name of the input SAS dataset (one or two level).
outdir : the physical path of the folder that stores the output data file.
outds : the name of the output delimit file, and the file extension should not be included.
delimit: comma (extension is csv) or tab (extension is txt).

The first three parameters have the same function as in the previous macro, but the fourth one serves a different purpose. It provides the option for the output file extension to be either a csv or txt file as shown below:

```
%if %length(&delimit) = 0 %then %do; %let delimit=comma; %end;
%if %upcase(&delimit)=COMMA %then %do; %let dlmt=','  ; %let ext=csv; %end; %else
%if %upcase(&delimit)=TAB   %then %do; %let dlmt='09'x; %let ext=txt; %end;
```

The SAS code that was used in Method 2 for the conversion portion is replaced as seen in the following set of codes:

```
%*------------------------------------------------------------------------*;
%* output SAS dataset &sasds into Excel spreadsheet &outdir.&outds.xls    *;
%*------------------------------------------------------------------------*;
data _null_;
     set &sasds;
     file "&outdir&delim&outds..&ext" lrecl=32767;
     if _N_=1 then put %DO i=1 %TO &n_var; %if &i^=1 %then &dlmt; "&&vname&i" %END; ;
     %DO i=1 %TO &n_var;
```

```
    %if &&vtype&i = 1 %then %do;
        if &&vname&i > .z then do;
            _efitmp_=left(put(&&vname&i,&&vfmt&i));
            if index(_efitmp_,&dlmt) or index(_efitmp_,'"') then _efitmp_=put(_efitmp_,$quote200.);
            put %if &i^=1 %then &dlmt; _efitmp_  $ +(-1) %if &i^=&n_var %then @; ;
        end;
        else put %if &i^=1 %then &dlmt;                  %if &i^=&n_var %then @; ;
    %end; %else
    %if &&vtype&i = 2 %then %do;
        if index(&&vname&i,&dlmt) or index(&&vname&i,'"') then do;
            _efitmp_=put(&&vname&i,$quote200.);
            put %if &i^=1 %then &dlmt; _efitmp_  $ +(-1) %if &i^=&n_var %then @; ;
        end;
        else put %if &i^=1 %then &dlmt; &&vname&i $ +(-1) %if &i^=&n_var %then @; ;
    %end;
    %END;
run;
```

The general methodology of the code is similar to that in macro %SAS2XLS in terms of the usage of the dataset information, such as the total number of variables and the name and format corresponding to each variable. As in Method 2, the information from the dataset is read in automatically. However, instead of using DDE to perform the conversion from the SAS dataset to an Excel file, macro %SAS2CSV outputs a comma-delimited file (csv extension) or tab-delimited file (txt extension) directly.

Like the macro in Method 2, macro %SAS2CSV easily performs the conversion to an Excel file by simply filling in the parameters in the macro call. However, by using macro %SAS2CSV, Method 3 also offers some additional advantages. First, this macro has the option of converting a SAS dataset into either a csv (comma delimiter) or a txt (tab delimiter) file directly. Second, the macro enhances efficiency because it can transform more than one SAS dataset into multiple Excel files. Efficiency is also improved because, by dropping DDE, macro %SAS2CSV can eliminate the waiting time of five seconds for opening MS Excel. There is, however, one disadvantage to the use of macro %SAS2CSV: after the conversion to Excel, it is necessary to save the converted file into an xls file in the proper directory.

## METHOD 4: PROC EXPORT PROCEDURE

Both EXPORT and PROC EXPORT can be applied to perform the SAS-to-Excel conversion. Again, assuming that there is a SAS dataset, RXPK, stored in the directory c:\paper2005\data, the following code could be used to convert rxpk.7bdat into an Excel file called rxpk.xls:

```
Libname datadir "c:\paper2005\data\";
  PROC EXPORT DATA= datadir.rxpk
              OUTFILE= "c:\paper2005\data\rxpk.xls"
              DBMS=EXCEL2000 REPLACE;
  RUN;
```

The DBMS (database management systems) option is used to specify the type and version of the output file to be created. With older versions of Excel, DBMS=excel97 may be specified instead.
The advantage of the PROC EXPORT procedure is that it is simple and easy to use to convert one SAS dataset into a single new Excel worksheet. However, under this procedure, the length of the file name is limited to 64 characters, including the path for the location of the file.

## METHOD 5: USING ODS OUTPUT TO GENERATE EXCEL FILES

It is easy to generate Excel spreadsheets containing SAS datasets using the Output Delivery System (ODS). There are several different options for ODS output. For example, ODS CSV writes a text file of values separated by commas and named with a csv extension, while ODS HTML writes a file with an xls extension. Both file types can be read by Excel as spreadsheets, as explained in a paper published by Chevell Pareker, entitled "Generating Custom Excel Spreadsheets using ODS." This paper also addresses additional details about different options available with the ODS approach.
The following is an example of the type of codes that would be applied when using the HTML destination in ODS to create an .xls file:

```
Libname datadir "c:\paper2005\data\";
ods HTML file="c:\paper2005\DATA\rxpk.xls";
    proc print data=datadir.rxpk;
    run;
ods HTML close;
```

Alternatively, a programmer would use codes like those listed below when using a CSV destination in ODS to create a .csv file:

```
*** output all variables in SAS dataset and using variable name as header;
ods CSV file="c:\paper2005\DATA\rxpk.csv";
    proc print data=datadir.rxpk;
    run;
ods CSV close;

*** to control output variables and using variable label as header;
ods CSV file="c:\paper2005\DATA\rxpk.xls";
    proc print data=datadir.rxpk label;
         var alloc treatment start_dt stop_dt ldate ltime relday;
    label ;
    run;
ods CSV close;
```

One of the benefits of the ODS approach is that it makes the conversion to Excel very simple to perform. It also allows output variables to be controlled with a VAR statement.

On the other hand, there are a couple of minor disadvantages with the ODS method of conversion. For example, in SAS.version 8.2, when a PROC PRINT is specified in ODS HTML with the .xls extension, additional windows may pop up, but they can be ignored, and the file can simply be saved again. In addition, since the CSV destination is experimental with SAS version 8.2 as part of the ODS Markup Language, users receive the following warning message: "CSV is experimental in this release." This message does not, however, impact the final output and can be ignored.

Of greater concern is the fact that by default, the output file generated with the ODS HTML destination begins in row 4 of the Excel file, and the file generated with the ODS CSV destination begins in row 3 of the Excel file. Thus, empty rows are generated with ODS output. To resolve this concern and to customize the ODS output, the extra empty rows can be removed by using the procedure PROC TEMPLATE. The sample code shown below is used before the ODS statement to modify the csv output file so that it starts output data in row 1.

```
Proc template;
Define tagset tagsets.newcsv;
       Parent = tagset.csv;
```

```
      Define event table;
             Finish:
                Put NL;
             End;
      Define event row;
             Finish:
                Put NL;
             End;
End;
Run;
ods tagsets.newcsv file="c:\paper2005\data\rxpk.csv";
    prc print data=datadir.rxpk;
    run;
ods tagsets.newcsv close;
```

## CONCLUSION

This paper briefly described a variety of methods that allow a user to generate an Excel file or a csv file from a SAS dataset. Critical assessments were given on the advantages and disadvantages of each method. The current available methods for performing the SAS-to-Excel conversion were discussed chronologically in order of development, starting with the simple and old-fashioned DATA step method and continuing with SAS macro methods, the PROC EXPORT procedure and finally, the ODS output files. The ODS output is by far the most straightforward method for performing the conversion. However, this method is relatively new and there are some deficiencies with versions earlier than 8.2. We introduced a method to resolve the primary deficiency by using PROC TEMPLATE to generate an ideal output. Of course, users should select the methods that best fit their particular needs for converting from SAS to Excel.

## REFERENCES


1. "Integrating SAS software with Microsoft Visual Basic for applications to Mass-Produce customized powerpoint graphs" by Aileen Yam, PharmaNet, Inc., Princeton, NJ, Proceedings of the PharmaSUG 2000.

2. "Generating Custom Excel Spreadsheets using ODS", by Chevell Pareker, SAS Institute, Cary, NC


## ACKNOWLEDGEMENTS


The authors would like to thank Jodi Benjamin for her thorough review and valuable comments.


## CONTACT INFORMATION

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


Author Name: Xingshu Zhu
Company Merck &Co., Inc.
Address : 10 Sentry Parkway, Blue Bell, PA 19422
Work phone: 484 344 3572
Fax: 484 344 2217
Email: xingshu_zhu@merck.com

Author Name: Shuping Zhang
Company Merck &Co., Inc.

Address : 10 Sentry Parkway, Blue Bell, PA 19422
Work phone: 484 344 3496
Fax: 484 344 2217
Email: Shuping_zhang@merck.com