# Data Exchange Between the SAS® System and Microsoft® Excel

Louise Weiler, Pacific Gas and Electric Company
Raymond Wan, Independent Consultant

## INTRODUCTION

SAS users sometimes want to send data from SAS applications into spreadsheet applications, or to bring data from spreadsheet applications into SAS applications. In the past, exchanging data was awkward and tedious. You had to create an intermediate file using one software product, then read the intermediate file using the other software product. Now, the SAS System for OS/2® and the SAS System for Windows™ let you exchange data directly and dynamically, without an intermediate file. The data exchange techniques are powerful and graceful, and are among the most exciting new features in SAS.

This paper will discuss two techniques for exchanging data with Microsoft Excel: Dynamic Data Exchange (DDE), and the Clipboard. DDE lets you open Excel, execute Excel commands, exchange data, and close Excel, all from within a single SAS session. The Clipboard lets you cut and paste text between SAS and Excel. The merits of each technique will be discussed, and tips will be offered that are not mentioned in the SAS manuals. A SAS macro will be presented that converts any SAS data file into an Excel spreadsheet containing column headings. Finally, this paper will discuss the problems of data transfer from Excel to SAS and some of the more advanced DDE features that can make it easier.

## THE CLIPBOARD

The Clipboard provides a way of exchanging text and graphics between different applications. The Clipboard is a common area of memory that can be accessed by any application. Within one application window, you can use your mouse to mark text, then cut (or copy) it to the Clipboard. When you bring up another application window, you can paste the text from the Clipboard. One can cut and paste within the same application as well as between different applications.

Exchanging data between SAS and Excel using the Clipboard is hardly different from cutting a portion of a spreadsheet and pasting it to a word processing application. For example, you could cut a few columns from an Excel spreadsheet and paste them into the SAS Program Editor window following a CARDS statement. Likewise, you can copy the results of a PROC PRINT in a SAS Output window into an Excel spreadsheet. In this case, each row of the SAS output will be inserted into one cell in a single spreadsheet column. You can then perform the Excel command DATA PARSE on the column and divide the single column into as many columns as needed. If any of your SAS variables contained embedded blanks, the DATA PARSE reshaping will require extra effort.

The Clipboard technique is ideal for ad hoc requests as long as the data is manageable — i.e., there's not too much of it and it doesn't need a lot of manipulation after pasting. This technique is not desirable if you need to keep track of what has been done or if you need to do the transfer repeatedly. Under these circumstances, Dynamic Data Exchange (DDE) is a more suitable technique. The rest of this paper will be devoted to discussing DDE.

## DYNAMIC DATA EXCHANGE: FUNDAMENTALS

DDE is defined by Microsoft Corporation as "a means of passing information between Windows applications automatically or semiautomatically." For SAS applications developers, DDE is an exciting new tool that enables you to streamline applications in a way that was never before possible. Now, you can develop systems that retrieve data from SAS files and then automatically update spreadsheet files. Conversely, you can develop systems that retrieve data from spreadsheet files and then automatically update SAS files. DDE is appropriate for updating files in real-time systems as well as for creating files for ad hoc requests.

### Starting an Excel Session

DDE will work only if an Excel session is open on your PC. There are several ways to open an Excel session. Here are three possibilities:

1. Double-click on an Excel icon, either before you open SAS or while your SAS window is minimized.
2. Issue an operating system command from the SAS Program Editor:

   ```
   X 'START EXCEL';
   ```

3. Issue an operating system command using the pull-down menu from any SAS window.

   After selecting Global, Command, Host Command type:

   ```
   EXCEL
   ```

### Exchanging Data

DDE is initiated in SAS by a FILENAME statement that identifies an Excel spreadsheet that is the source or target of the data exchange. Once the FILENAME statement is executed, you can reference the fileref in your code. Thus, you can read from the Excel file using an INPUT statement, or you can write to the Excel file using a PUT statement. DDE cannot currently be initiated in Excel. (Technically, this means that SAS is the client in a client/server relationship.)

The syntax of the FILENAME statement is as follows:

```
FILENAME fileref DDE 'DDE-triplet'
         <DDE-options> ;
```

where ...

**fileref** is a valid SAS fileref

**DDE** is the device-type keyword that tells the SAS System to use DDE

**DDE-triplet** is the DDE external file of the form:

```
'application-name | topic ! item'
```

where ...

**application-name** name of the DDE server application [e.g., Excel] (required)

**topic** name of the topic of conversation [e.g., name of the spreadsheet or "system"] (required)

**item** range of conversation specified between the client and server applications [i.e., rows and the columns in the spreadsheet]

**DDE-options** HOTLINK, which enables the automatic update of a SAS file when values in an Excel spreadsheet change

NOTAB, which ignores the default tab characters between variables

The following brief example sends 3 variables from the SAS file SASUSER.HOUSES (from the SAS sample library) to the first three columns of the Excel spreadsheet called SHEET1. SHEET1 is the blank spreadsheet that is automatically opened when you start Excel. There are 15 observations in SASUSER.HOUSES.

```
FILENAME EXCEL DDE 'EXCEL |
         SHEET1 ! R1C1:R15C3' ;

DATA _NULL_;
   SET SASUSER.HOUSES;
   FILE EXCEL;
   PUT STYLE SQFEET PRICE;
RUN;
```

Following is the code to read back into a SAS file the data that was just sent to Excel.

```
FILENAME EXCEL DDE 'EXCEL |
         SHEET1!R1C1:R15C3';

DATA HOUSES;
   INFILE EXCEL;
   LENGTH STYLE $8;
   INFORMAT PRICE DOLLAR9.;
   INPUT STYLE SQFEET PRICE;
RUN;
```

The code to receive data from Excel must include the length of character variables and the informats of numeric variables that have been formatted by Excel.

### Sending Commands to Excel

You can also establish a DDE link to Excel that will allow you to execute Excel commands. The syntax is as follows:

```
FILENAME fileref ' EXCEL | SYSTEM ';
```

where fileref is any valid SAS fileref.

The following brief example shows how to open an existing Excel spreadsheet, HOUSES.XLS

```
FILENAME CMDEXCEL ' EXCEL | SYSTEM ';

DATA _NULL_;
   FILE CMDEXCEL;
   PUT '[open("houses.xls")]';
RUN;
```

## DDE TIPS YOU WON'T FIND IN THE MANUALS

The preceding examples are ideal cases; real-life examples will be more complex. DDE users need to be aware of the consequences (sometimes unintended!) of DDE. When you follow the procedure laid out in the manual, you need to steer clear of several traps. The following tips will help to keep you out of trouble.

### Tip #1: Managing Multitasking Environment

DDE functions within a multitasking environment where it is possible to have multiple Excel sessions open simultaneously. If there are multiple sessions of Excel, does SAS takes out its random number generator and roll a virtual die when it establishes a DDE link? Because SAS doesn't handle this situation in a sequential fashion, you could end up exchanging data with one Excel session and sending commands to another! To be safe, you should have only **one** session of Excel open when you execute DDE. This way, there will be no confusion as to who is talking to whom. If you need to have more than one spreadsheet open, you can open multiple spreadsheets as long as you do so within one Excel session.

### Tip #2: Managing Directories

When you open, save, or close an Excel spreadsheet, the current directory is used if no path is specified. When you start Excel from the Excel icon, your current directory is the default Excel directory. But if you start Excel from within SAS, the current directory is the current directory for your SAS session. You may not find your spreadsheet, or your spreadsheet may get saved in an inappropriate directory if you are not aware of this rule. Generally, it is safer to always specify the full path in any written code. For example:

```
FILENAME CMDEXCEL ' EXCEL | SYSTEM ';

DATA _NULL_;
   FILE CMDEXCEL;
   PUT '[SAVE.AS("C:\MYDIR\filename.XLS")]';
RUN;
```

### Tip #3: Avoiding Pop-up Windows

Even though DDE lets you exchange data using a batch SAS program, your program could get stalled by the appearance of a pop-up window that will not go away unless you respond to the window. Here are two ways to avoid pop-up windows.

1. If you start up Excel within your SAS program, first issue the command

   ```
   OPTIONS NOXWAIT;
   ```

This tells the operating system that you want to return to SAS after issuing an X command to the operating system.

2. Before you save or close a spreadsheet, or before you close an Excel session, send an ERROR(FALSE) command to Excel.

```
FILENAME CMDEXCEL ' EXCEL | SYSTEM ';

DATA _NULL_;
  FILE CMDEXCEL;
  PUT '[ERROR(FALSE)]';
RUN;
```

### Tip #4: Preserving Formatted Values

When you send data from SAS to Excel, Excel will assign an Excel built-in format to every cell. Excel will try to chose the most appropriate format from its set of built-in formats. Amazingly enough, when you send SAS variables with date, datetime and dollar formatted values, Excel will assign its own date, datetime and dollar formats to the columns. All other columns will receive the Excel "general" format. The "general" format treats incoming data as characters strings or numeric data. Therefore, many SAS formatted values will be preserved. The exception to this rule occurs when SAS sends numeric values with leading zeroes (Zn.) or character values with leading blanks ($CHARn.), or character values with trailing blanks. Leading zeroes, leading blanks and trailing blanks are trimmed off in Excel. Also, COMMAn. formatted variables lose their commas when they arrive in Excel.

You can restore commas by assigning the Excel built-in format "#,##0" to the column. You can restore leading zeroes by assigning an Excel custom format to the column. For example, the Z7. format can be restored by creating a custom format "0000000" for the column. There does not appear to be any way to restore leading or trailing blanks to character values.

Appropriate SAS formats should be assigned to variables before transfer to Excel. This is especially important with date and datetime values. Excel stores dates internally as the number of days from Jan. 1, 1900 while SAS uses Jan. 1, 1960. Therefore, if you send unformatted SAS date values to Excel, they may be converted incorrectly.

### Tip #5: Preserving Missing Values and Embedded Blanks

Excel handles data sent from SAS as if it is reading a "flat file" put out by SAS. Excel fits each line into a row and it will put every item separated by a tab character into a cell. When a blank is encountered, however, the blank is replaced with a tab character. Consequently, if you have SAS character values containing missing values or embedded blanks, there will be too many delimiters. Excel will not put the SAS variables into an appropriate column, and your data will end up out of alignment. Numeric missing values, however, are sent as "." as long as you do not use the MISSING=' ' system option.

SAS provides a solution to this problem, although the solution is documented incorrectly in the *SAS® Companion for the OS/2® Environment*. You can use the NOTAB option in the FILENAME statement. This causes DDE to ignore the default tab character (blanks). Now, however, you must insert the hex tab character '09'X between the variables you send to Excel. The following example shows how to safely send the data from SASUSER.HOUSES to Excel.

```
FILENAME EXCEL DDE 'EXCEL | SHEET1 !
         R1C1:R15C3' NOTAB;

DATA _NULL_;
  SET SASUSER.HOUSES;
  FILE EXCEL;
  PUT STYLE   '09'x
      SQFEET  '09'x
      PRICE   '09'x ;
RUN;
```

The *SAS® Companion for the OS/2® Environment* (page 108) provides an example of the use of the NOTAB option to send two variables with embedded blanks to Excel. The example, however, does *not* send the two variables to two separate columns because the code leaves out the insertion of the tab characters. The two variables are actually sent into *one* cell. Do not assume that the NOTAB option *alone* will take care of embedded blanks in the data.

When transferring data from Excel to SAS, missing values and embedded blanks also cause misalignment of the data in the resulting SAS data set. The NOTAB option again provides a solution to this problem. When the NOTAB option is in effect, however, SAS must look for the '09'X delimiter between the columns. You can do this with the DLM option in the INFILE statement.

```
FILENAME EXCEL DDE 'EXCEL |
         SHEET1!R1C1:R15C3' NOTAB;

DATA HOUSES;
  INFILE EXCEL DLM='09'X;
  LENGTH STYLE $8;
  INFORMAT PRICE DOLLAR9.;
  INPUT STYLE SQFEET PRICE;
RUN;
```

### %SAS2EXCL: A Generic SAS Macro to Send Data from SAS to Excel

If you need to create spreadsheets that look like rectangular SAS files, DDE transfer can be accomplished by a generic macro incorporating the above tips. The Appendix contains the SAS macro %SAS2EXCL which will send data from any SAS data file to an Excel spreadsheet. The first row of the spreadsheet will have column headings containing the names (or the labels) of the SAS variables that were sent. The data will be placed from row 2 down. The user may specify:

- the name of the SAS file
- the name of the Excel file
- whether the column headings should contain the SAS variable names or the SAS variable labels
- whether to open an Excel session
- whether to save the Excel file
- whether to close the Excel session.

The macro will calculate how many rows and columns will be needed in the spreadsheet. The macro will also take care of missing values and embedded blanks.

### %EXCL2SAS: A Mythical SAS Macro to Move Data From Excel to SAS

%EXCL2SAS is a mythical SAS macro because a generic macro can not exist. The reason is that while SAS datasets are rather well behaved animals, a spreadsheet can be much less well-ordered. Any SAS data set can be fitted into a spreadsheet, but the converse is not true. If, however, the spreadsheet that is being transferred has the "look and feel" of a SAS rectangular file, then a macro could be very easily written along the lines of the %SAS2EXCL macro. Still, SAS is not as flexible as Excel in figuring out how to handle input strings. SAS needs information like informats, length and data types.

If the information that SAS needs is organized in some predefined manner in a spreadsheet, that information can be read into SAS and used to drive a program that would then move the data from Excel.

## EXCEL TO SAS COMMUNICATION: HOT LINKS THAT TALK BACK

DDE can be used to set up dynamic linking of Excel spreadsheets and SAS files, i.e., whenever the cells in a spreadsheet is changed the corresponding SAS file is updated. HOTLINK is an option in the FILENAME statement that can set this up. The option tells SAS to continuously poll specific areas of a spreadsheet as defined by the range in the FILENAME statement. Whenever the spreadsheet data within the range is changed, the link is activated.

The HOTLINK feature can, however, be used in a much more imaginative way. The limitations of client-only-mode DDE means that Excel cannot initiate SAS processes. HOTLINK can be used to partially overcome this limitation, albeit with only a limited vocabulary. For example, a special spreadsheet can be set up with cells that are continuously monitored by SAS. These cells can act as buttons. When the user types a predefined character in a particular cell of the special spreadsheet, SAS will notice the change and instigate a predefined set of actions in the background.

## CONCLUSION

DDE is a powerful tool to enhance applications that share data between SAS and Microsoft Excel. DDE must be used carefully, because unintended errors can easily occur. Generic SAS macros can be used to avoid unintended errors. This paper was developed using OS/2. The techniques are *expected* to work identically under Windows, but have not been tested.


### Acknowledgments

Thanks to Tim Berryhill, Danine Cozzens, Mary Austin and Peter Kretzman who assisted in the development and production of this paper.




### Bibliography


SAS Institute Inc., SAS® Companion for the OS/2® Environment, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990.

SAS Institute Inc., SAS Communications®, Volume XVII, Number 2, Second Quarter 1991.

Microsoft Corporation, Microsoft® Excel User's Guide, Version 3.0, OS/2 Series or Windows Series, 1991


### Appendix

```
*************************************************;
*  MACRO: SAS2EXCL                              *;
*                                               *;
*  Purpose: To send all data from a SAS data    *;
*           set to an Excel spreadsheet, and    *;
*           to put the variable names or        *;
*           labels in the first row as column   *;
*           headings.                           *;
*                                               *;
*  Authors:  Louise Weiler                      *;
*            Raymond Wan                        *;
*  Date:     March 10, 1992                     *;
*                                               *;
*  PARAMETERS:                                  *;
*                                               *;
*  START=,   Indicates whether SAS should       *;
*            start an Excel session (Y or       *;
*            anything else). Default is Y.      *;
*                                               *;
*  SASIN=,   Source SAS file.  A one or two     *;
*            level name, already defined in     *;
*            your SAS session.                  *;
*                                               *;
*  EXDIRIN=, Directory of target Excel file,    *;
*            including the final slash, ex.     *;
*            b:\sugi\sugi   (not needed if      *;
*            SHEET1 is the target).             *;
*                                               *;
*  EXCELIN=, Target Excel file. Default is      *;
*            SHEET1, the blank spreadsheet      *;
*            that is opened when Excel is       *;
*            started.                           *;
*                                               *;
*  EXCELOUT=,Saved Excel file, including the    *;
*            complete directory path            *;
*                                               *;
*  CLOSE=,   Indicates whether SAS should       *;
*            close the Excel session (Y or      *;
*            anything else).  Default is Y.     *;
*                                               *;
*  COLHEAD=, Indicates whether the variable     *;
*            name or label should appear in     *;
*            the Excel column heading. (NAME    *;
*            or LABEL).  Default is NAME.       *;
*                                               *;
```

```
*  NOTE:                                          *;
*     If you choose to start an Excel session,    *;
*     you must not have had any previous DDE      *;
*     links in your current SAS session.          *;
************************************************;

OPTIONS NOXWAIT MPRINT;


%macro sas2excl (start=Y,
                 close=Y,
                 colhead=NAME,
                 sasin=,
                 exdirin=,
                 excelin=SHEET1,
                 excelout= );

%*********************************************;
%*  Check macro parameters                   *;
%*********************************************;

%if %upcase(&colhead) ne NAME and
    %upcase(&colhead) ne LABEL    %then
   %do;
      DATA _NULL_;
         PUT 'Parameter error: COLHEAD='
             "&colhead";
         ABORT;
      RUN;
   %end;

%*********************************************;
%*  Start up Excel                           *;
%*********************************************;

%if &START eq Y %then
   %do;
      X "START EXCEL";

      DATA _NULL_;
         X=SLEEP(10);
      RUN;
   %end;

%*********************************************;
%*  Open file to contain Excel commands      *;
%*********************************************;

FILENAME CMDEXCEL DDE 'EXCEL|SYSTEM';

%*********************************************;
%*  Open existing Excel spreadsheet          *;
%*********************************************;

%if %upcase(&excelin) ne SHEET1 %then
   %do;
      DATA _NULL_;
         FILE CMDEXCEL;
         PUT "[open(%bquote(&exdirin&excelin))]";
      RUN;
   %end;

%*********************************************;
%*  Gather information about SAS source file,*;
%*  i.e. the number of rows and columns, and *;
%*  the variable names and labels.  Put this *;
%*  information into macro variables.        *;
%*********************************************;

PROC CONTENTS DATA=&sasin NOPRINT
              OUT=CONTS(KEEP=NOBS NAME LABEL);
RUN;

DATA _NULL_;
   SET CONTS END=EOF;

   IF LABEL EQ '' THEN
      LABEL = NAME;

   CALL SYMPUT('col'||left(_N_),TRIM(NAME));
   CALL SYMPUT('lab'||left(_N_),TRIM(LABEL));

   IF EOF THEN
      DO;
         CALL SYMPUT(columns,TRIM(LEFT(_N_)));
         CALL SYMPUT(rows,TRIM(LEFT(NOBS)));
      END;
RUN;

%*********************************************;
%*  Link to Excel spreadsheet to put column  *;
%*  headings in the first row.               *;
%*********************************************;

FILENAME EXCEL DDE
         "EXCEL| &excelin.!
          R1C1:R1C&columns." NOTAB ;

%*********************************************;
%*  Send column heading to spreadsheet.      *;
%*********************************************;

DATA _NULL_;
   FILE EXCEL;

   HEXTAB = '09'x;

   %if %upcase(&colhead) eq NAME %then
      %do;
         PUT %do i=1 %to &columns;
                "%trim(&&col&i)" HEXTAB
             %end;
             ;
      %end;
   %else %if %upcase(&colhead) eq LABEL %then
      %do;
         PUT %do i=1 %to &columns;
                "%trim(&&lab&i)" HEXTAB
             %end;
             ;
      %end;
RUN;

%*********************************************;
%*  Link to Excel spreadsheet to send SAS data*;
%*  beginning in the second row.             *;
%*********************************************;

FILENAME EXCEL DDE
         "EXCEL|
          &excelin.!
          R2C1:R&rows.C&columns." NOTAB ;

%*********************************************;
%*  Now send the SAS data to the Excel       *;
%*  spreadsheet.                             *;
%*********************************************;

DATA _NULL_;
   SET &sasin;
   HEXTAB = '09'x;
   FILE EXCEL;
   PUT %do i=1 %to &columns;
          &&col&i HEXTAB
       %end;
       ;
RUN;
```

```
%*************************************************;
%*  Save the Excel spreadsheet                   *;
%*************************************************;

%if &excelout ne '' %then
   %do;
      DATA _NULL_;
         FILE CMDEXCEL;
         PUT '[error(false)]' ;
         PUT "[save.as(%bquote("&excelout"))]";
      RUN;
   %end;

%*************************************************;
%*   Close Excel                                 *;
%*************************************************;

%if &close = Y %then
   %do;
      DATA _NULL_ ;
         FILE CMDEXCEL ;
         PUT '[error(false)]' ;
         PUT '[quit()]' ;
      RUN;
   %end ;

%mend sas2excl;
```