%Qexcel Tackles the Challenges of Converting Data to SAS® from Excel®

Jay Zhou, Quintiles, Inc., Kansas City, Missouri

ABSTRACT

One common programming task in the pharmaceutical industry is to transfer external data such as lab, ECG, PK, and other non-case-report-form data to SAS from Excel. Spreadsheets can be read with the SAS import wizard or procedures such as `PROC ACCESS` and `PROC IMPORT`, but often must be modified by adding or editing column names, formatting cells, deleting blank columns or rows, or moving to different platforms. However, the responsibility of the owner for the contents of the original file is lost when any modification to the file is made. Also the audit trail process required by 21 CFR Part 11 cannot be simplified if such data transfers are made with the non-original spreadsheets.

This paper presents a SAS macro, %Qexcel, which has the capability and flexibility to tackle the challenges of handling external data. The macro provides total control over the Excel import with the following features: (1) automatically detecting the worksheet name and cell-range, (2) allowing the user to select any row for data variable names and labels, (3) accepting user-defined variable names and data types (numeric or character), (4) identifying the starting row for the first observation, if not specified, (5) enabling the use of SAS formats and informats to define data fields, (6) changing cell format to 'general' to import the original, unformatted data, (7) transforming out-of-range single cell(s) to be field(s) in a SAS dataset, and (8) uploading the converted SAS data to other platforms (e.g., UNIX) via SAS/CONNECT. This paper, which is suitable for all SAS skill levels, discusses the methodology of %Qexcel design with SAS code, then shows example calls to the macro.

Key Words: Excel, spreadsheet, DDE, function, data transfer

INTRODUCTION

Microsoft Excel is a popular spreadsheet application heavily used to congregate and store external non-case-report-form data in the pharmaceutical industry. In addition to the workbook file (.XLS), Excel is able to read delimited ASCII files (e.g., .TXT, .CSV, .PRN), dBASE files (.DBF), Lotus 1-2-3 files (.WK1 to .WK4, .WKS), HTML files, Quattro Pro files (.WQ1), and data interchange files (.DIF). There are two major ways to handle these external data in clinical trial studies: one to load the data into database management systems (e.g., Oracle Clinical) and the other to transfer the data directly to SAS. For the latter, there are several solutions available in SAS such as the import wizard, procedures (`PROC ACCESS, PROC DBF, PROC DIF, PROC IMPORT`, and `PROC SQL` with ODBC®), and data steps with `INFILE` and `INPUT` statements (with or without filename DDE triplet statement). However, as reviewed and compared by Kuligowski (1999) and Mumma (1999), none of these methods are ideal due to the complexity of external files. Some of these are platform-dependent or version-dependent, some will convert data appropriately only if the data structure or format is suitable, some require specific hard-coding of the worksheet name and cell-range, and others have to supply the variable names, data types, and field lengths.

Because of the limitations of available tools in SAS, the files often have to be pre-processed and modified for the appropriate data transfer by adding or editing column names, formatting cells, deleting blank columns or rows, saving as delimited files, or even moving to different platforms. This results in the loss of responsibility by the original file owner and requires additional audit trail documentation. Also cumbersome is the need for subsequent modifications to the SAS code when the number of records and columns or the name of the spreadsheet is changed, since the cell-range and worksheet name are hard-coded in the programs. In summary, the following challenges cannot be tackled easily when converting external files to SAS datasets with any SAS procedures:

- Making the data transfer without any pre-processing to maintain the owner's responsibility for the external data
- Importing the actual data rather than the formatted values
- Specifying individual cell(s) in Excel as data field(s) in the SAS dataset
- Using the data title or description as the SAS dataset label
- Utilizing the column headers of external data to be SAS variable labels
- Detecting the cell-range and sheet name programmatically
- Importing data in the Windows environment for different platforms to omit the FTP process
- Simplifying the audit trail process required by the FDA regulations entitled 21 CFR Part 11

In this paper, a SAS macro, %Qexcel, will be introduced and discussed to tackle the above challenges. Before discussing the macro design methodology, the usage of the macro parameters will be explained. Several practical calls to the macro will be given before the conclusion to exemplify the macro functionality. The macro, along with all SAS code presented in the paper, was developed and tested with Microsoft Excel 97 and SAS V6.12 and V8.2 under the Windows platform.

THE MACRO %Qexcel

Before getting into the technical details, let us take a look at the macro syntax first. %Qexcel is a SAS macro embedded with Microsoft Excel V4.0 macro functions to convert any type of files that can be opened by Excel to SAS datasets. The 14 keyword parameters provide customized specifications to allow full programmatic control over the data flow to SAS from Excel. Except for the **dsout** parameter when specified for UNIX platform, none of the parameters are case-sensitive. The full macro specification is as follows (defaults shown):

```
%qexcel(dsin=,
        dsout=,
        sheet=,
        range=,
        firstobs=,
        vars=1,
        label=1,
        dlabel=,
        add=,
        informat=,
        format=,
        deformat=,
        subset=,
        drop=)
```

The meaning and usage of the macro parameters are thus:

dsin: the only required parameter. It specifies the path and name of the external file, e.g., **`dsin`**`=c:\data\pk_data.xls`.

dsout: specifying the output SAS dataset name and path or libref. By default, the Excel file name and path will be used. This parameter determines the platform where the output SAS dataset will reside based on the structure of the path. For example, **`dsout`**`=/home/mydir/data|pk_data@unix_hp` will upload the SAS data set, `pk_data`, to the Unix platform with the path of `/home/mydir/data` directory at Server `unix_hp`.

sheet: defining the name of the worksheet to be converted. As the name of the active sheet can be detected programmatically by %Qexcel, it is necessary to specify the names of the additional sheets in other calls to the macro if a workbook contains data in more than one sheet.

range: specifying the cell ranges containing the data to be imported into SAS. An attempt will be made to import all observations for the entire worksheet if not defined. The parameter must be specified if importing a block of data rather than all data within a worksheet. This is helpful if multiple datasets need to be made from a single worksheet.

firstobs: specifying the row number of the 1st observation. By default, the data from the 2nd row will be treated as the 1st observation. The parameter will only need to be defined when the 1st observation does not immediately follow the row of column headers (or labels).

vars: determining whether SAS generates variable names from column names in Excel or from user-defined names. As with the SAS import wizard, `PROC ACCESS`, and `PROC IMPORT`, the first row of data will be used by default. There are three possible specifications for this parameter. First, this parameter can be defined with the number of the row which contains the column headers. Second, this parameter accepts user-supplied variable names, e.g., **`vars`**`=ptno gender race age`. Note that the number of names must match the numbers of columns in the range. Third, when **`vars`**`=0` or when there are no data in the specified row, it will take the variable names of C1, C2, C3, …, Cn.

label: specifying a row number where the information to be used for SAS variable labels is located. By default, the 1st row of data will be used with the label length of 40 characters for V6.12 and 200 for V8 if the option `VALIDVARNAME` is not specified. The user can specify the label length with the **label** parameter. For example, **`label`**`=r3/32` specifies the 3rd row of data used for the labels with the length of 32 characters.

dlabel: defining the SAS dataset label. The user can define this parameter with the reference (e.g., **`dlabel`**`=b1` or **`dlabel`**`=r1c2`) of the cell where such information is available (e.g., title or sheet description). Otherwise, a short description can be supplied, e.g., **`dlabel`**`=PK Parameters Data`.

add: allowing individual cell(s) to be specified as additional field(s) in the output SAS dataset, e.g., drug name, project/study number, and investigator's site number. The value of this parameter contains three elements (**x**, **y**, and **z**): **`x`**`|`**`y@z`**, where the symbols '|' and '@' are separators, **x** (required) is the cell reference, **y** (optional) is the variable name, and **z** (optional) is the variable label. For example, **`add`**`=A2|project@Project Number` will create an additional field from Column A, Row 2 with project for the variable name and Project Number for the variable label. If the variable name and label are not specified, the default variable name will be C_{j+1}, where j is the number of the last variable, but the variable label will be missing. When more than one cell reference is specified, a comma must be used to separate the references and the macro quoting function %str must be employed to mask the comma token, e.g., **`add`**`=%str(a1, a2, a3)`.

informat: defining SAS informat(s) for the specified column(s) with two elements for each variable: column number (or letter) and SAS informat separated by a slash. A space is added to separate columns when formatting two or more columns, e.g., **`informat`**`=e/date7. f/time5.` or **`informat`**`=5/date7. 6/time5`. Note that the informats specified with this parameter are used directly in the `INFORMAT` statement associated with the `INPUT` and `INFILE` statements to format data from the spreadsheet.

format: supplying user-defined SAS format(s) to the specified column(s) with two elements for each variable: column number (or letter) and the format separated by a slash. The usage of this parameter is similar to **informat**. For instance, **`format`**`=d/$trt.` or **`format`**`=4/$trt`. The formats are used in the `FORMAT` statement for SAS data values.

deformat: changing the format(s) of the specified column(s) in Excel to be 'general' to show the actual, unformatted values. There are at least five scenarios to define this parameter:

(a) All columns: **`deformat`**`=all`;
(b) Columns in range, e.g., for Columns A to D: **`deformat`**`=a:d`;
(c) Individual columns, e.g., for Columns B, D, and G: **`deformat`**`=b d g`.
(d) Cell range, e.g., for Column A and Row 2 to Column G and Row 200: **`deformat`**`=A2:G200` or **`deformat`**`=r2c1:r200c7`;
(e) Combinations: **`deformat`**`=A2:G200 H L M:P X2:Y400`.

subset: specifying a logic condition to subset data, e.g., **`subset`**`=trt ne ' '`.

drop: excluding the specified column(s) from the SAS dataset, e.g., dropping columns from A to D: **`drop`**`=a b c d` or **`drop`**`=a:d`.

METHODOLOGY OF %Qexcel DESIGN

The data transfer to SAS from Excel made by %Qexcel relies on Dynamic Data Exchange (DDE). DDE is a communication method of dynamically exchanging information between Windows applications and establishes a client/server relationship to enable a client application (SAS) to request information from a server application (Excel). The macro establishes a linkage between SAS and Excel via DDE to send Excel V4.0 macro functions enabling a SAS session to take control of Excel. Actions performed interactively through use of mouse-clicks in Excel, such as opening and closing files, selecting and copying data, and reformatting worksheet cells, can be done from within a SAS program. The usage of DDE along with some useful Excel V4.0 macro functions can be found in several SUG papers (Mumma, 1999; Roper, 2000; Vyverman, 2001).

To increase the capability and flexibility of %Qexcel, many advanced techniques are employed by the macro. Let us focus on the following key methods used in %Qexcel:

1. Converting Reference Style

Microsoft Excel has two reference styles to indicate the location of a cell: R1C1 and A1. By default, Excel uses the A1 reference style, which labels columns with letters (A through IV, for a total of 256 columns) and labels rows with numbers (1 through 65,536). In R1C1 style, Excel indicates the location of a cell with an "R" followed by a row number and a "C" followed by a column number. The R1C1 style is useful because it can be used in Excel macros for computing row and column positions. Hence, in the DDE triplet, only the R1C1 style can be used to pass on the range of cells. The column letter has to be converted into the number when using the A1 style. What is the column number for Column BG? Obviously, determining the column number is burdensome if a worksheet in Excel uses the A1 reference style. The good news is that %Qexcel has resolved this issue programmatically, allowing the use of either style to refer the location of a cell or the range of cells. Code to convert the A1 style to R1C1 style follows:

```
%let cell=A1;
%let row=%sysfunc(compress(
          &cell,'ABCDEFGHIJKLMNOPQRSTUVWXYZ'));
%let col=%sysfunc(
          compress(&cell, '1234567890'));


data colname(keep=colname);
  length colname $2;
  do i=1 to 27;
    a=substr(' ABCDEFGHIJKLMNOPQRSTUVWXYZ',i,1);
    do j=1 to 26;
     b=substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',j,1);
     colname=compress(a)||compress(b);
     output;
    end;
  end;
run;
```

```
data _null_;
  set colname;
  if colname="&col" then
     call symput('col', compress(_n_));
run;

%let cell=R&rowC&col;
```

2. Opening a File in Excel

In order for a client/server communication link to be established, both SAS and Excel must be running. The first step is programmatically to detect whether an Excel session is running. If not, it is necessary to launch Excel from a SAS session. There are a few techniques available to launch Excel with SAS (Roper, 2000; Vyverman, 2001). %Qexcel uses the technique introduced by Roper (2000) as follows [see Roper (2000) and Vyverman (2001) for a more in-depth explanation of the code]:

```
filename sasexcel dde 'excel|system';
options noxwait noxsync;
data _null_;
  fid=fopen('sasexcel', 's');
  if fid le 0 then do;
    rc=system('start excel');
    time=datetime();
    stop=time + 3;
    do while (fid le 0);
      fid=fopen('sasexcel', 's');
      time=datetime();
      if (time ge stop) then fid=1;
    end;
  end;
  rc=fclose(fid);
run;
```

Once Excel is running, the next step is to open the specified file into Excel. This process in Excel can be easily automated from SAS with the following code:

```
data _null_;
  file sasexcel;❶
  put '[error(false)]';❷
  put '[app.minimize]';❸
  put "[open(""&dsin"")]";❹
run;
```

❶ The `FILE` statement specifies the fileref `sasexcel` defined in the previous doublet-style DDE filename statement for `PUT` statements in the current DATA step. In other words, the Excel macro functions sent by the `PUT` statements will control Excel. ❷ This clears all error-checking and keeps Excel from displaying any messages. Other commands can be continuously sent and executed even if an error is encountered. ❸ This minimizes the Excel window. ❹ This statement opens a data file in Excel. The macro variable `&dsin` is a macro parameter defined in a call to %Qexcel, e.g., `dsin=c:\development\data\pk_data.xls`. Note that the whole command is quoted with double-quotes, which makes it possible to resolve `&dsin`. Using two sets of double-quotes around `&dsin` is necessary because a single set of double-quotes must be kept to quote the resolved macro value after compiled. The command also works if quoted like this: `'[open("' "&dsin" '")]'`.

3. Determining Sheet Name and Cell-Range Programmatically

As required by the DDE triplet, sheet name and a range of cells must be defined to identify the scope of conversation between Excel and SAS. This specifies the data from the 1st row and column to the last row and column within a worksheet. Both sheet name and range need to be hard-coded in a DDE triplet that must be updated or modified if the range is changed in Excel. Ideally, the sheet name and cell-range should be programmatically determined to allow modifications to the spreadsheet without subsequent changes to the SAS code, which %Qexcel does.

In most cases, the reference of the 1st cell is easily determined because the data often starts from the 1st column and the column headers are usually followed by the 1st record of data. The range of cells should then be known if the reference of the last cell can be determined. This task can be done with the help of two Excel V4.0 macro functions: **Select.Last.Cell** and **Copy**. The former, which can be replaced with the **Select.Special**(11) function, highlights the last cell within the cell-range and the latter copies the selected cell to the clipboard. Data are then read from the clipboard by the `INFILE` and `INPUT` statements associated with the `FILENAME` statement for the connection to the clipboard via DDE. The DDE triplet containing the path and name of the file, sheet name, and cell reference is printed to the SAS log window. `PROC PRINTTO` procedures are used to save the SAS log to an external file, which then is converted into SAS data with an `INFILE` statement in a `DATA _NULL_` step to create the SAS macro variables for the sheet name and cell reference. This approach to obtain the sheet name seems much simpler than the one introduced by Vyverman (2001). However, this method obtains only the name of the active sheet, not all of the sheets. Here are the steps of the SAS code:

```
data _null_;
  file sasexcel;
  put '[Select.Last.Cell()]';
  put '[Copy]';
run;

filename temp "c:\temp.log";
filename temp1 dde "clipboard";
proc printto log=temp new;
run;

data _null_;
  infile temp1;
  input v1;
run;

proc printto;
run;

data null_;
  length trip $200;
  infile temp truncover ;
  input @']' trip;
  if indexc(trip,'!') gt 0 then do;
    call symput('sheet',scan(trip,1,'!'));
    call symput
      ('cell',scan(compress(trip,' ,'),2,'!'));
  end;
run;
```

The external file `c:\temp.log` contains the following message:

```
NOTE: PROCEDURE PRINTTO used:
      real time           0.01 seconds
      cpu time            0.00 seconds

56         data _null_;
57            infile temp1;
58            input v1;
59         run;

NOTE: The infile TEMP1 is:
      DDE Session,
      SESSION=Excel|C:\data\[pkdata.xls]pkdata!R450C10,
      RECFM=V,LRECL=256
NOTE: 1 record was read from the infile TEMP1.
      The minimum record length was 0.
      The maximum record length was 0.
NOTE: SAS went to a new line when INPUT statement
reached past the end of a line.
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds
60
61         proc printto;
62         run;
```

One may notice that the 1st column of a cell-range must be specified somewhere if it is not Column A. To avoid this shortcoming, %Qexcel exercises another method to determine the cell-range without using Excel macro functions. It first imports the entire worksheet (R1C1:R65536C256) to SAS and then excludes those columns without any data and creates macro variables for the reference numbers of the1st and last columns and last row. Because it takes more data steps with multiple DO loops, the processing time is a little longer than the method with the Excel macro functions. But it is worthwhile and works well for one table per worksheet. Let's take a look at the code:

```
filename excel0 dde
"excel|&sheet!R1C1:R65536C256";

data varrow;
  infile excel0 dlm='09'x notab dsd missover;
  length
  %do i=1 %to 256;
    c&i $20
  %end; ;
  input
   %do i=1 %to 256;
    c&i
  %end; ;
  %do i=1 %to 256;
    if c&i ne ' ' then l&i=length(c&i);
    else l&i=0;
    drop c&i;
  %end;
run;

data varrow;
  set varrow end=last;
  %do i=1 %to 256;
    retain c&i 0;
    c&i=c&i+l&i;
    drop l&i;
  %end;
  if last then do;
    call symput('row2',compress(_n_));
    output;
  end;
run;

data varrow(keep=col);
  set varrow;
  %do i=1 %to 256;
    if c&i>0 then do;
      col=&i;
      output;
    end;
  %end;
run;

data _null_;
  set varrow end=last;
  if _n_=1 then
     call symput('col1', compress(col));
  else if last then
     call symput('col2', compress(col));
run;
```

4. Obtaining SAS Variable Names, Labels, and Dataset Label from Excel

There are several steps where the FILENAME statements with DDE triplets are defined for DATA steps with INFILE statements to import data from Excel. The purpose of the steps is to create SAS macro variables for variable names, variable labels, data set label, and individual out-of-range cell(s). Since the usage and explanation of DDE triplets within SAS with INFILE statement has been well covered by SAS Institute documentation (SAS Institute Inc., 1996) and SUG papers (Schreier, 1998; Kuligowski, 1999; Mumma, 1999; Vyverman, 2001), no detailed information on this topic will be provided in this paper. The code below illustrates the usage of the DDE triplet and DATA steps in creating macro variables for variable labels. Similar steps are used to obtain macro variables for selecting variable names, data set label, and individual out-of-range cell(s).

```
filename excel2 dde
"excel|&sheet!R&labrow.C&col1:R&labrow.C&col2";❶

data labelrow;❷
  infile excel2 dlm='09'x notab dsd missover;
  informat
  %do i=&col1 %to &col2;
   c&i $&lableng..
  %end; ;
  input
  %do i=&col1 %to &col2;
  c&i
 %end; ;
run;

data labelrow(keep=label n);❸
  set labelrow;
  length label $&lableng;
  %do i=&col1 %to &col2;
    label=left(trim(c&i));
    n=&i;
    output;
  %end;
run;

data _null_;❹
  set labelrow;
  call symput('label'||compress(n),
left(trim(label)));
run;
```

❶ To increase the flexibility of importing data from any worksheet and range of cells, the elements of the DDE triplet are macrotized. The macro variable &sheet defines the topic of the communication between SAS and Excel and tells SAS to locate the specific worksheet within a workbook. The variable &labrow is the row number defined by the macro **label** parameter in a call to %Qexcel. The variables &col1 and &col2 specify the column range of data from the 1st column to the last. ❷ This data step reads data to SAS from the Excel location defined by the DDE triplet to create a SAS dataset called labelrow with only 1 observation. Note that the INFORMAT statement is used to limit the character length defined by the macro variable &lableng. The same results can be obtained by using the LENGTH statement. By default, the maximum number of characters for SAS variable labels for SAS V6 or above, if the option VALIDVARNAME is specified, is 40, otherwise 200. ❸ This step transposes the labelrow dataset to the vertical structure. To keep the logic order of values from the 1st column to the last, the variable n is added to the dataset. ❹ A DATA _NULL_ step creates macro variables with the call symput routine. Note that the variable n is used to mark the order.

5. Changing Cell Formats

Among the external files, only the .XLS file preserves all worksheet and chart data, formatting, macros, and other features available in Excel. Numeric data with decimals are sometimes formatted to reduce the number of decimal places shown in order to narrow the column width. Data transfer made with DDE will import the formatted data. However, it is often desirable to import the actual, unformatted data when converting into SAS data. To alter the cell format manually, one would click on the Format menu, choose Cells, and then select the 'general' format from the Number tab. This is necessary whether using the DDE method or other SAS procedures (except PROC IMPORT). This step can be automated with the Excel macro function **Format.Number** to change the format of the selected cells to be general by default. For example, the following code alters the format for the cell-range, r1c2:r100c2:

```
data _null_;
  file sasexcel;
  put '[error(false)]';
  put '[select("r1c2:r100c2")]';
  put '[Format.Number("General")]';
run;
```

6. Importing Data

To import all data into a SAS dataset, a similar data step to that discussed previously with the `INFILE` and `INPUT` statements associated with the DDE triplet is used. Except for those fields where the informats have been defined in the **informat** parameter, other fields will be imported and treated temporarily as character with the maximum length (using 200 for V6 and 32000 for V8 if option `VALIDVARNAME` is not specified) to avoid any data exclusion or truncation. Once the data are converted into SAS, data type (character or numeric) of a field will be determined intelligently by scanning the entire field. For a character field, the maximum length of data strings will be used for the field length.

Because the number of rows in each Excel worksheet is limited to 65536, data will be excluded if a flat ASCII file contains more observations. To overcome this, %Qexcel uses the `INFILE` statement without the DDE triplet to read the external file. With this approach, the file opened in Excel must be closed to avoid a data transfer abortion due to the file being in use. Two Excel macro functions **Close** and **Quit** can be used to close the file and Excel. This process is triggered when the number of the last row is 65536 (detected by %Qexcel) or larger (defined in the **range** parameter). The sample code to import data from a comma delimited (.CSV) file follows [see SAS Language reference books (SAS Institute Inc., 1990; 1999a) and a SUGI paper (Cody, 1998) for a detailed explanation on the options used in the INFILE statement]:

```
data _null_;
  file excel;
  put '[error(false)]';
  put '[Close]';
  put '[Quit]';
run;

filename cfile "c:\data\pk_data.csv";

data _dsout;
  infile cfile dsd dlm=',' firstobs=2 lrecl=9999
         missover;
  informat pkdate date7. pktime time5.;
  length ptno $8 period 8 day 8 trt $20 values 8;
  input ptno period day trt pkdate pktime values;
run;
```

7. Uploading SAS Data to Other Platforms

Many external files originally arrive in the Windows environment, but often these files need to be converted into SAS datasets residing on different platforms. There are two common practices exercised by programmers. One method is to save files as delimited ASCII files, FTP them to the designated platforms, and then convert them into SAS with SAS procedures or data steps. Another is to read the external files directly with PC SAS, make SAS transport files, and FTP them to the desired platform. Both methods employ several manual steps which not only complicate the audit trail but also can be time-consuming and must be repeated in the event of alterations to the original data.

With %Qexcel, the process of converting data to different platforms can be greatly simplified. The macro determines the desired platform where the SAS output dataset is to reside based on the structure (delimiter) of the output path. For example, a slash (/) delimiter is for the UNIX platform and a backslash (\) for Windows. A non-backslash delimiter will trigger PROC UPLOAD to load the data to the desired destination. This process requires SAS/CONNECT® software to be installed on both Windows and UNIX platforms. Here is the sample code for the UNIX platform:

```
libname excldata 'c:\data';
filename rlink
"c:\sas\connect\saslink\tcpunix.scr";
options comamid=tcp remote=unixserver;
signon;

%if &sysver=6.12 %then %do;
  %syslput(outpathp,&outpathp);
  %syslput(outdata,&outdata);
%end;
%if &sysver=8.1 %then %do;
  %lput(outpathp,&outpathp);
  %lput(outdata,&outdata);
%end;

rsubmit;

libname unix "&outpathp";

proc upload inlib=excldata outlib=unix;
  select &outdata / memtype=data;
run;

endrsubmit;
signoff;
```

Note the usage of the two macros, `%syslput` and `%lput`, supplied by the SAS Institute. The macro variables, `&outpathp` and `&outdata`, are defined by %Qexcel in the local server for the path and name of the output dataset. However, they cannot pass through to the remote environment. Hence, `%syslput` and `%lput` are utilized to create the macro variables in the remote UNIX box based on the values in the local environment. Note that a system error message '`ERROR: Expected equals sign not found in %syslput statement`' will appear in the SAS log window when the above code is executed with SAS V8. The reason is that `%syslput` is replaced with `%lput` with the same syntax for SAS V8 and `%syslput` is turned into a SAS macro statement designed syntactically to look like `%sysrput`, a SAS function opposite to `%syslput`. To avoid the error message, one can simply replace the `%syslput` macro with `%lput` for V6.12. Another advantage of using `%lput` is that, unlike the `%syslput` macro statement, `%lput` does not require that both the local and remote hosts run SAS V8 or later. See SAS V8 on-line documentation for more in-depth information (SAS Institute Inc., 1999b).

EXAMPLE

Demonstrations of how %Qexcel may be used are provided. Consider the following practical sample calls to %Qexcel:

Example 1: Assume ECG data are provided in a comma delimited ASCII file with the column headers displayed in the 1st row followed by data in the 2nd row. The entire sheet of data needs to be imported into a SAS dataset with the same file name and location as the external file. The data transfer can be easily completed with %Qexcel by defining the **dsin** parameter with the path and name of the external file. Here is the simplest call to %Qexcel:

```
%qexcel(dsin=c:\development\data\ecg.txt)
```

Example 2: An Excel file, pk_data_for_site1.xls, contains 12 columns of PK data which need to be converted into a SAS dataset. The column descriptions are displayed in the 2nd row followed by data in the 3rd row. The output SAS dataset needs to be saved to the pre-defined SAS library called main with the file name pkdata. The user supplies the variable names with the **vars** parameter, but allows %Qexcel to default the variable labels to the column headers in the spreadsheet. The 6th and 7th columns containing date and time need to be numeric and formatted with SAS informats. The treatment information in the 4th column is required to be formatted with the user-defined format $trt. To accomplish this, the following macro call can be submitted:

```
%qexcel(dsin=c:\data\pk_data_for_site1.xls,
        vars=ptno period trtseq trt day date
             time schdtime stime fromdose
             devtime value,
        label=2,
        informat=6/date7. 7/time5.,
        format=4/$trt.,
        dsout=main.pkdata)
```

Example 3: Assume that an Excel file, pd_pk_data.xls, contains both PD and PK data within the same worksheet. Two separate calls to %Qexcel are needed to create two SAS datasets: one for PD data and the other for PK data. Let us use the PD data to demonstrate how to import data from a specific range in Excel. The PD data, not including the column headers, are in the range of `a6:h1000`. The column headers are listed in the 3rd row and the detailed descriptions in the 4th row. The title of the PD data table is displayed in Cell A1, which needs to be converted into the label of the SAS dataset. Cell A2 contains the study number, required to be populated as a variable in the SAS dataset. The data in Column F needs to be dropped in the SAS dataset. The data in Column G are formatted to reduce the number of the decimal places, but the actual values need to be imported to SAS. The output SAS dataset needs to be saved with a new file name called pddata to the c:\project\study directory. The PD data transfer can be made with the following call to %Qexcel:

```
%qexcel(dsin=c:\data\pd_pk_data.xls,
        vars=3,
        label=4,
        range=a6:h1000,
        deformat=g,
        dlabel=a1,
        drop=f,
        add=a2|study@Study Number,
        dsout=c:\project\study|pddata)
```

Example 4: An Excel file, protviol.xls, contains protocol deviation data which need to be converted into a SAS dataset with the same file name and loaded to the UNIX server unixhp1. This task can be accomplished simply with the following call to %Qexcel:

```
%qexcel(dsin=c:\data\protviol.xls,
        dsout=/client/project/study@unixhp1)
```

CONCLUSION

%Qexcel is a powerful macro which can be executed in both batch and interactive mode to convert those external files that can be opened with Microsoft Excel into SAS datasets. Use of %Qexcel simplifies the audit trail required by 12 CFR Part 11, since a data transfer to SAS from the original external files is made in 'read-only' mode without any modification to the files. The macro allows data transfer to be scheduled to take place during off-hours without maintenance, because the whole transfer process is automated, and worksheet name and cell-range are determined programmatically. %Qexcel reduces programming time by saving the steps of hard-coding labels for datasets and variables (if the information is available in the external files) and manipulating data to subset and drop data. By automating these steps, %Qexcel reduces the likelihood of manual errors and saves repeating steps if the original external file is changed. In conclusion, %Qexcel makes a worthwhile contribution to the area of data transfer.

REFERENCES

Cody, R. (1998), The INPUT Statement: Where It's @. Proceedings of the 23th Annual SAS® Users Group International Conference, paper 61.

Mumma, M. T. (1999), The Redmond to Cary Express – A Comparison of Methods to Automate Data Transfer Between SAS and Microsoft Excel. Proceedings of the 12th Annual Northeast SAS® Users Group Conference, 654-661.

Kuligowski, A. T. (1999), Advanced Methods to Introduce External Data into the SAS System. Proceedings of the 24th Annual SAS® Users Group International Conference, paper 53.

Roper, C. A. (2000), Intelligently Launching Microsoft Excel from SAS, using SCL functions ported to Base SAS. Proceedings of the 25th Annual SAS® Users Group International Conference, paper 97.

SAS Institute Inc. (1990), SAS® Language: Reference, Version 6, First Edition. Cary, NC: SAS Institute Inc., pp. 377–389.

SAS Institute Inc. (1996), Technical Support Document TS-325 – The SAS System and DDE. http://ftp.sas.com/techsup/download/technote/ts325.pdf, updated 1999.

SAS Institute Inc. (1999a), SAS® Language Reference, Version 8. Cary, NC: SAS Institute Inc., pp. 857-872.

SAS Institute Inc. (1999b), SAS® Macro Language Reference: Concepts, Version 8. Cary, NC: SAS Institute Inc.

Shreier, H. (1998), Getting Started with Dynamic Data Exchange. Proceedings of the 6th Annual Southeastern SAS® Users Group International Conference, pp. 207-215.

Vyverman, K. (2001), Using Dynamic Data Exchange to Export Your SAS® Data to MS Excel – Against All ODS, Part I. Proceedings of the 26th Annual SAS® Users Group International Conference, paper 11.

ACKNOWLEDGMENTS

The author would like to thank Dawn DuBois, Michael Eschenberg, and John Morrill for reviewing this manuscript and providing valuable comments.

CONTACT INFORMATION

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

Jay Zhou
Quintiles, Inc.
10245 Hickman Mills Drive
Kansas City, MO 64137
Work Phone: 816-767-6000
Email: jay.zhou@quintiles.com
zhou_jay@yahoo.com