Paper 119-26

When PROC ACCESS Says NO to Excel 97/2000, DDE says YES Still

Eric T. Sun, IS/Complete Inc., Green Brook, NJ
Eric Kammer, Novartis Pharmaceuticals Corporation, East Hanover, NJ

ABSTRACT

The open connectivity that SAS provides towards major data processing software like Microsoft Excel is always appreciated and heavily used across various industries. A complex but necessary Excel format was pre-designed as a data source template that has a lock mechanism to a block of data conversion settings needed to build up a SAS dataset library. Two major macros were developed to read **vertical** and **horizontal** stretching data blocks that had predefined SAS variable names anchored anywhere inside the template. Several approaches including PROC ACCESS/IMPORT were implemented but two major problems occurred: 1) lacking flexibility to handle 'floating' data blocks mentioned above and 2) SAS does not accept Excel versions like Excel97 or Excel2000 for either SAS v6.12 or the newest v8.

This paper presents the powerful SAS macros with utilization of good old but reliable DDE (Dynamic Data Exchange) technique to overcome the limited function provided by other SAS procedures. Also, DDE still can automate reading multiple Excel workbooks into a SAS dataset library. This program is built on intermediate high SAS Macro skill set and works with SAS612 and SAS801 on Windows95-2000 platforms.

INTRODUCTION

A typical Excel worksheet format having first row with contents to identify each column is very helpful for a SAS programmer to convert that information into SAS variables in a SAS dataset. But this rigid format will force data processing staffs to give up the fundamental popularity of spreadsheet like Excel 97 & 2000 because of limitations of SAS procedures. Flexible SAS programming is the right candidate to overcome this challenge. A cooperative attitude toward worksheet design is needed and pro-action of user groups in prototyping with implantation of clear SAS dataset definition is the key to a successful data conversion project.

DESIGN SCOPES

When SAS reads a data vector in Excel, it naturally reads from top to down in a pair of pre-defined dimensions, like (r5c5:r50c50) for example. This **vertical** stretching data vector that is relatively smooth while extracting contents into a SAS dataset. The other type of data vector presents key data from top to down but expanding data from left to right, so called a **horizontal** stretching data vector. It needs careful consideration since a consequent data transposing is inevitable. Moreover, a combination of these two data vectors could occur in a worksheet with a complexity of one data vector providing key event information while the other presenting multiple records/observations of those key events. It is the flexible format that eases data collection with more 'human style", such as coloring, underlying link_related data cells to eliminate duplicated data entry or certain degree of data validation and so on. It is also the flexibility brings up interesting challenges to fulfill an ideal data collection.

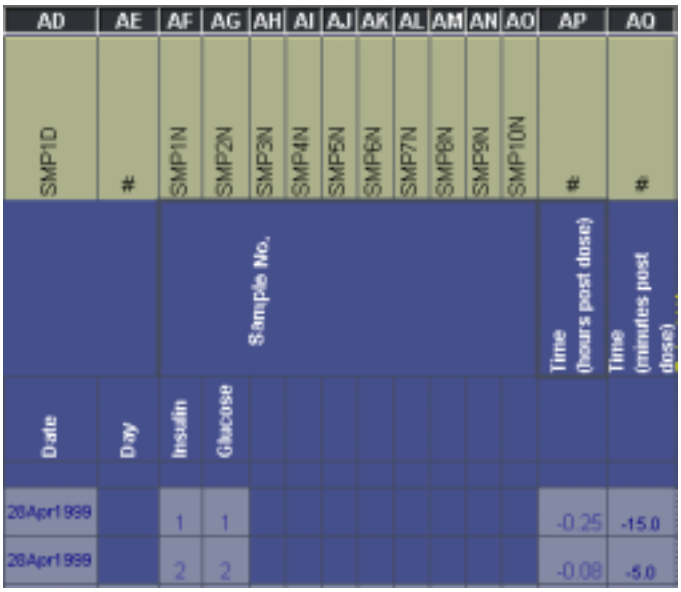
Next, a set of pre-defined SAS variables matching to each stretching vector were assigned according to a naming convention especially to the last letter of a name, for example:

N	Numerical
C	Code
D	Date
T	Time

With this kind of naming convention, a detail control over the data types of target data can be done in macros as described later all at one time. Figure 1 depicts a portion of this mixture of SAS variable definitions with a vertical data vector in an

Excel spreadsheet. The '#' sign in the 'variable definition row' is a tag for unwanted columns and will be discarded by the macro in dataset formation stage.

Below variable definition row is 'description row'. These rows provide a detail title to each column and they are good candidates for label information of SAS variables in the target SAS datasets. This delicate spreadsheet format is necessary to facilitate the macro programming.



AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
SMP1D	#	SMP1N	SMP2N	SMP3N	SMP4N	SMP5N	SMP6N	SMP7N	SMP8N	SMP9N	SMP10N	#	#
		Sample No.										Time (hours post dose)	Time (minutes post dose)
Date	Day	Insulin	Glucose										
28Apr1999		1	1									-0.25	-15.0
28Apr1999		2	2									-0.08	-5.0

Figure 1

Similar strategy is applied to horizontal data vector, see Feature 2, but with 'DummyX' instead of '#' sign due to the requirement imposed by PROC TRANSPOSE. Namely, no repeat variable name shall exist.

SBJINI1A	Initals:	YYY	
DOB1D	Date of birth:	01Jan1950	
SEX1C	Sex:	2	
DUMMY3	Age:	49	years
RCE1C	Race:	1	
EBWBRD2N	Elbow Breadth:	6.5	cm
BDYFRM1C	Body Frame	2	

Figure 2

In each macro there are key columns in which the only function is to determine the end of the target data vector observation. Any variable name added after these key columns will cause no extra programming maintenance and the program can add these new variable names to the data library automatically.

Each SAS dataset definition cell can be grayed out and locked from data entry to maintain the integrity of Excel worksheet layout and data processing.

Due to the implementation scope of the Excel template included several foreign countries and the difference of date formats, a small macro was created to ensure final ddmmmyyyy date format created correctly in English for the final SAS dataset.

PROC ACCESS'S TURN

In finding out the best way to hide and lock SAS dataset definition cells, PROC ACCESS was deployed to perform intermediate reading function. One advantage of PROC ACCESS is that it works perfectly for hidden cells in Excel workbooks. This valuable feature meets the requirements of this project. Any control tags, pre-defined SAS variable names, pre-selected key information can coexist with data points but will not been seen or modified by unauthorized person. Moreover, it reads target Excel files without having the target Excel opened. Unlike DDE, the data won't be exchanged if either one of software (SAS/Excel) is not present. That means PROC ACCESS is more efficient then DDE while processing multiple Excel files.

However, one down side for PROC ACCESS is that it reads Excel underlying numerical values of date and time formats. The date value offset which is 21916 to SAS underlying date value is easy to figure out and compensate but the returned time values are floating point values that reveal no clue for further SAS time conversion. Second, while using PROC ACCESS, if the option like VarName=$30 was not preset, then some text contents may be truncated without pre-scanning maximum string length (30, in this case). Third, lack of flexibility to handle complex formats within a spreadsheet.

PROC ACCESS so far couldn't read Excel versions after Excel95. According to SAS Note: SN-001571

re-check on 1/17/2001, a default read-in rows from Excel to SAS is 16,384 rows. This is a limit for Excel95 but not for Excel97 or Excel2000. A potential data truncation may happen but will not show in SAS log at all. This pending problem in SAS6.12 to SAS8.01 dramatically limited its usage on data conversion. SAS is still working on it with medium priority. Thus, because of Proc Access limitations with Excel 97 it could not be used.

OTHER FEATURES

A batch code section takes care of locating, all open Excel files in specific directory in a descending order of Excel file names to ensure desired conversion sequence.

A group of data export procedures were developed in various kinds of data formats for further data analysis after this dataset library was successfully established.

Each Excel workbook will be closed automatically by the macro once reading process is done. Data accumulation from each specific worksheet is oriented through batch processes in the beginning section and all the Excel files are combined into one dataset.

ENHANCEMENTS

There is a price paid for the flexibilities needed for this task: “Huge Log File”. Typical messages during data importing procedures are like:

```
NOTE: Missing values were generated as a result of
performing an operation on missing values.

NOTE: Invalid third argument to function SUBSTR at
line 835 column 74.

NOTE: Mathematical operations could not be
performed at the following places. The results of
the operations have been set to missing values.
```

All these error like messages spread allover the log file. They can be validated OK but pretty annoy. Certain false data type conversion can be avoided by adding in more filter statements to prevent data conversion from alpha numeric data to numerical value. Some more control macro variables to track dimension of import data to avoid over read that causes excess none-error log lines.

CODE SHOWCASE

Macro for vertical stretching data

```
**** ----------------------------------------------------------------
**** COMMON UTILITY EXCEL-TO-SAS CONVERSION
**** MACRO CALL (Herizontal vars upto 150) ****;
**** exin: input excel sheet name
**** ds: output dataset name
**** keycol: attached after a 'c' to identify key column which
**** contains non-missing value for tracking valid output obs.
**** keytype: key column data type (default is char)
**** prt: test print out output dataset (optional, default is n)
**** strr: start row adjust (optional, default is 1)
**** strc: start column adjust (optional, default is 1);
**** ----------------------------------------------------------------;
%macro convar(exin=,ds=,keycol=1,keytype=char,
                    prt=n,strr=1,strc=1);
 filename excel dde "excel|&exin.!r&strr.c&strc.:r150c150";
run;
 data &ds(drop=ct4flag);
           infile excel dlm='09'x notab dsd missover lrecl=800;
   length c&strc.-c150 $30;
   retain ct4flag 0;
   input c&strc.-c150;

 data &ds; set &ds;
  **** store pre-defined SAS var names in first observation
 **** into macro vars for target dataset setup;
 if _n_=1 then do;
    %do x=&strc %to 150;
       call symput("cc&x",c&x);
    %end;
 end; else
    **** only non-missing key column value can trigger an
    **** output observation;
    %if &keytype=num %then %do;
         if (c&keycol * 1)>0 then do; %end;
    %else %do; if length(trim(c&keycol))>1 then do; %end;
    output;
 end;
 run;

 data &ds(drop=allvar);
     length allvar $200;
   set &ds end=eof;
     retain allvar;
     %do x=&strc %to 150;
            **** pack all valid SAS vars from each macro var
           into
            **** a long string: allvar;
       %if &&cc&x = # or &&CC&x=#CT4NAME or
```

```
        &&cc&x= %then %do; **** invalid SAS var name;
        drop c&x;
      %end;
      %else %do;
       allvar=trim(allvar)||' '||"%trim(&&cc&x)";
            **** put allvar content into a macro var: allvar;
       if eof then call symput('allvar',allvar);
       %let len=%length(&&cc&x);
       %if %eval(%substr(&&cc&x,&len-1,1)+0)>=1 %then
         %do;
           **** all valid SAS var names have a single > 0
           **** digit at 2nd last position if last char of SAS
           **** var name = 'N'(8.), 'D'(date9.), 'T'(time5.)
           **** convert data contents with proper format
           **** under valid SAS var name;
          %if %upcase(%substr(&&cc&x,&len,1))=N %then
          %do;
             &&cc&x=c&x*1; drop c&x;
          %end; %else
          %if %upcase(%substr(&&cc&x,&len,1))=D %then
          %do;
             format &&cc&x date9.;
             %chkmon(c&x);
             &&cc&x=input(c&x,date9.); drop c&x;
          %end; %else
          %if %upcase(%substr(&&cc&x,&len,1))=T %then
         %do;
             format &&cc&x time5.;
             &&cc&x=input(c&x,time5.); drop c&x;
          %end; %else
          %do; rename c&x=&&cc&x;; %end;
          **** replace column names with valid SAS vars;
       %end;
      %end;
    %end;
  **** include study, center and patient ID key vars read from
  **** demography sheet ;
  length ctr1n sbj1n 8. sty1a $30;
  sty1a=symget('study');
  ctr1n=symget('center')*1;
  sbj1n=symget('patid')*1;
run;
*proc print data=&ds;; run;

**** if print function triggered;
%if &prt=y %then %do;
  proc contents;
  title "&ds"; run;
  proc print data=&ds; run;
%end;

**** accumilate datasets to the end of loop (the batch) except
**** demo and lab that need further works;
%if &ds ne labh and &ds ne bcl %then %do;
    %if &i=1 %then %do;
       data out.&ds; set &ds; run;
    %end; %else %do;
       data out.&ds; set out.&ds &ds; run;
    %end;
  %end;
%mend convar;
```

Macro for vertical stretching data

```
**** ----------------------------------------------------------------
**** COMMON UTILITY EXCEL-TO-SAS CONVERSION
****  MACRO CALL (Vertical vars upto 50) ****;
**** exin: input excel sheet name
**** ds: output dataset name
**** keyvarn: identify key variable after data transposing
**** datacol: pinpoint the beginning column of data
**** before data transposing
**** prt: test print out output dataset (optional, default is n)
**** strr: start row adjust (optional, default is 1)
**** strc: start column adjust (optional, default is 1);
**** stpr: stop row adjust (optional, default is 50)
**** stpc: stop column adjust (optional, default is 50);
**** ----------------------------------------------------------------;
%macro convarv(exin=,ds=,keyvarn=,datacol=,prt=n,strr=1,
          strc=1,stpr=50,stpc=50);
 %local i;
 filename excel dde "excel|&exin.!r&strr.c&strc.:
          r&stpr.c&stpc.";  run;
 data a(drop=ct4flag);
          infile excel dlm='09'x notab dsd missover;
  retain ct4flag 0;
  length v1-v&stpc $30;
  input v1-v&stpc;
  **** #CT4NAME is the END POINT of input data rows
  **** reading from Excel;
  if v&strc = '#CT4NAME' then ct4flag=1;
  if ct4flag=1 then delete;
 run;

 data a; set a end=eof;
  retain goodvar 0;
  **** goodvar serves as a counter of good/valid SAS
  **** variables gvar increments a macro variable for further
  **** data conversion in next data step;
  drop gvar v1a len goodvar;

  **** since immediate dataset after proc transpose will not
  **** allow char->num conversion on the same var. ;
  **** all date(D)/time(T)/num(N) conversions need a
  **** substitute var. name (v1a) in transition.  If variable
  **** name not like DUMMYxx then proceed data
  **** conversion;
  if substr(upcase(v1),1,5) ne 'DUMMY' then do;
    len=length(v1);
    if upcase(substr(v1,len,1))='N' then do;
      goodvar=goodvar+1;
      gvar='gvar'||left(put(goodvar,2.));
      v1a=substr(v1,2,len);
      **** pack char->num statements into macro variable;
```

```
      call symput(gvar,v1||'='||v1a||'*1; drop '||v1a||';');
      v1=v1a;
    end; else
    if upcase(substr(v1,len,1))='D' then do;
      goodvar=goodvar+1;
      gvar='gvar'||left(put(goodvar,2.));
      v1a=substr(v1,2,len);
     **** pack date format, input statements into a macro
     **** variable;
      call symput(gvar,  'format '||v1||' date9.;
          %chkmon('||v1a||'); '||v1||'=input('||v1a||',date9.);
           drop '||v1a||';');
      v1=v1a;
    end; else
    if upcase(substr(v1,len,1))='T' then do;
      goodvar=goodvar+1;
      gvar='gvar'||left(put(goodvar,2.));
      v1a=substr(v1,2,len);
     **** pack time format, input statements into a macro
     **** variable;
      call symput(gvar, 'format '||v1||' time5.; '||v1||
                    =input('||v1a||',time5.);  drop '||v1a||';');
      v1=v1a;
    end;
  end;
  **** at end of file, record total number of good/valid
  **** SAS variables;
  if eof then call symput('goodvar',goodvar);
run;
*proc print; run;

proc transpose data=a out=&ds;
  var v&datacol.-v&stpc; id v1;
*proc print; run;

data &ds(drop=dummy1-dummy15  _name_); set &ds;
  **** execute post-transpose data conversion statements
  **** stored in macro variables;
  %do w = 1 %to &goodvar;
     &&gvar&w;;
  %end;
  if &keyvarn=. then delete;
*proc print; run;

**** if print function triggered;
%if &prt=y %then %do;
  proc contents;
  title "&ds"; run;
  proc print data=&ds; run;
%end;

data &ds; set &ds;
  %if %upcase(&ds) = DMG %then %do;
    call symput('study',sty1a);
    call symput('center',ctr1n);
    call symput('patid',sbj1n);
  %end; %else %do;
    length sty1a $30 ctr1n sbj1n 8.;
    sty1a=symget('study');
    ctr1n=symget('center')*1;
    sbj1n=symget('patid')*1;
  %end;
%mend convarv;
```

CONCLUSION

These two macros implement good old DDE technique and successfully fulfill their tasks. Even though, they are far from perfect. With more enhancements, adjustments and maybe combination can bring them to different industries or fields.

ACKNOWLEDGEMENT

Thank Dr. Kenneth Pan's advisory and edit review to ensure the integrity of this paper.

CONTACT INFORMATION

For more information and question regarding these two macros, Please use the main contact:

Eric T. Sun
IS/Complete Incorporation
1000 North Washington Avenue,
Green Brook, NJ08812
Phone: (732) 926-0789
e-mail: eric.sun@iscomplete.com