PharmaSUG 2012 - Paper CC07

Importing Excel ® File using Microsoft Access ® in SAS ®

Ajay Gupta, PPD Inc, Morrisville, NC

ABSTRACT

In Pharmaceuticals/CRO industries, Excel files are widely use for data storage. There are different methods such as PROC IMPORT\LIBNAME to convert Excel file in SAS dataset. But, due to technical limitations in SAS these traditional methods might create missing values while reading the columns in Excel file containing mixed data types (both numeric and character). Especially, when the Excel file is large then it is hard to debug the loss of data. Also, if the validation programmer is using the same traditional method to read the Excel file then there is a possibility that datasets will pass the validation and further use into production even there is loss of data. This will affect the overall quality of deliverable and also increase the cost. To overcome this issue and provide an alternative method for validation, there is desperate need for new method to read Excel. This paper will introduce a unique method to read Excel using Microsoft Access in SAS. This convenient and reliable solution will help SAS Programmers/Statisticians to have better control over the quality of data and save significant time with minimal coding.

INTRODUCTION

Microsoft Access is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications and is included in the Professional and higher versions for Windows and also sold separately.

While SAS can accept input from a variety of formats, Excel files (XLS) are among the most common. Currently there are many ways to import Excel data into SAS: The Import Wizard, `PROC IMPORT`, using a `LIBNAME` statement. But, due to technical limitations in SAS these traditional methods might create missing values while reading the columns in Excel file containing mixed data types (both numeric and character). Especially, when the Excel file is large then it is hard to debug the loss of data. Also, these methods mostly depend upon SAS to decide the formats of the data and can create different attributes for similar variables when same Excel file is read by different traditional method. So, both programmer and validator might needs to do some extra programming to match the data which can be time consuming. In case, if the header in the excel file is not on the first line then it will require further processing of the data to get the right header which might be time consuming. Due to the above mentioned issues there is a desperate need for new method to read Excel which can read the mixed data properly and programmers can predefined the variables formats with minimal coding.

This paper introduces another useful (and probably simpler) method to import data from any Excel file, into SAS. The entire process involves SAS, Excel, and Access which is automated by a SAS macro called %Convert_Excel2_SAS. It will import the entire contents while reserving the Excel layout and structures, including tables with multiple columns, into SAS data. Since variable formats are predefined in the Microsoft Access table, this will avoid any formatting changes occur to the data during the conversion process. In this macro the user can defined the line number containing the header and the macro will assign the header variables from the given line number.

TECHNIQUE & MECHANISM

The general process to convert an Excel file into a SAS dataset is as follows (note that the first two items are a one-time setup, and the rest can be managed via SAS code):

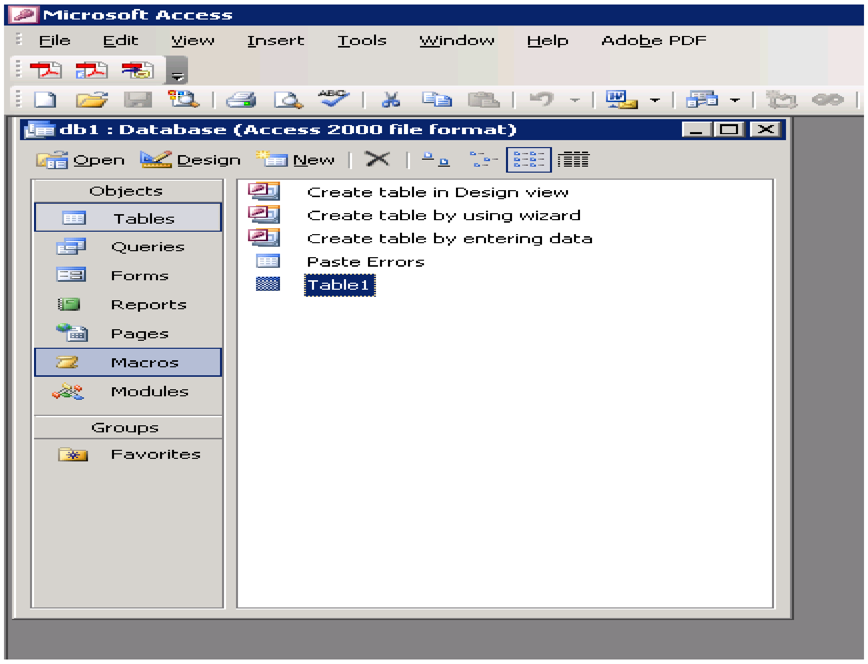
1. Create a new master Microsoft Access Database file (which will act like a template) and define a table having required variables and formats.
2. Create a Auto execution macro in the Master Access Database template which will have the following functions:
    a) Select table
    b) Set all warnings to yes.
    c) Select all records from the table.
    d) Paste the records.
3. Copy the master Microsoft Access Database template to the production location.
4. Open the target file in Excel and copy the content.
5. Open the Microsoft Access Database from production location. Due to the auto execution of the Access macro, the Access table will have all the records inserted automatically into it from the clipboard buffer.
6. Read the content into SAS using `LIBNAME` or `PROC IMPORT` procedure.
7. If require, execute macro `%Excel_Header` to get header from Excel file.
8. Delete the Microsoft Access Database from the production location.

To automate these steps, the DDE solution is leveraged to build the communication bridge between SAS v9.1.3 and Excel 2003 or Access 2003. WordBasic commands (Microsoft Corporation, 1999) can be sent from SAS via DDE to enable SAS to take control of Excel and Access.

CREATE MASTER ACCESS DATABASE AND DEFINE A TABLE:

Using the graphical user interface provided by Microsoft for Access, programmers can easily create a new database (e.g. db1) and define the table (e.g. Table1). The table attribute can be defined in the design view per your requirements. To help avoid unwanted formatting changes, all of the variables are defined as character. For more information on using the Access, please refer to the help tab in Access. As there is no coding involved and in order to have a better understanding of the Microsoft Access Database, the whole process is described using screenshots.

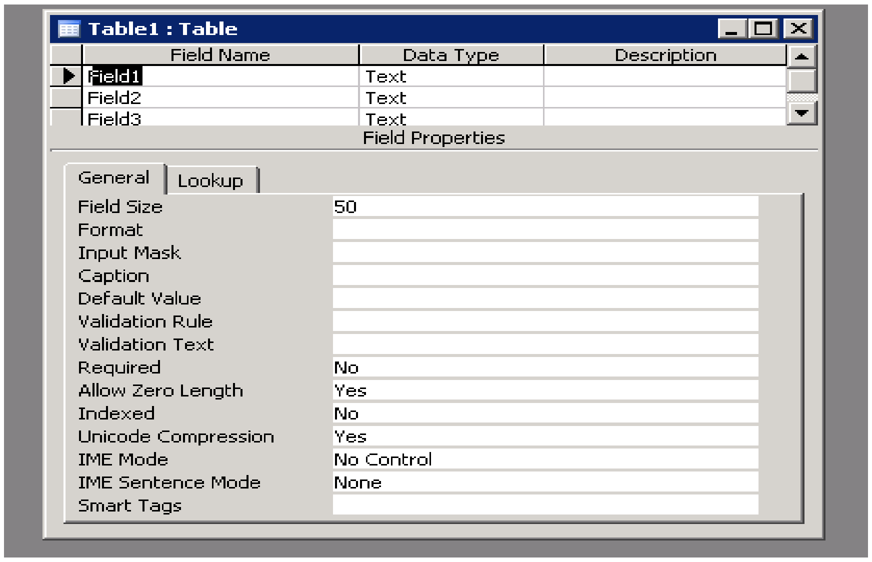
Below is the screenshot of the Master Access Database and table defined as Table1:



Microsoft Access
File Edit View Insert Tools Window Help Adobe PDF
db1 : Database (Access 2000 file format)
Open Design New
Objects
Tables
Queries
Forms
Reports
Pages
Macros
Modules
Groups
Favorites
Create table in Design view
Create table by using wizard
Create table by entering data
Paste Errors
Table1

Display 1. Master Access Database.

Below is the screen shot of Table1 in design view:



Field Name	Data Type	Description
Field1	Text	
Field2	Text	
Field3	Text	

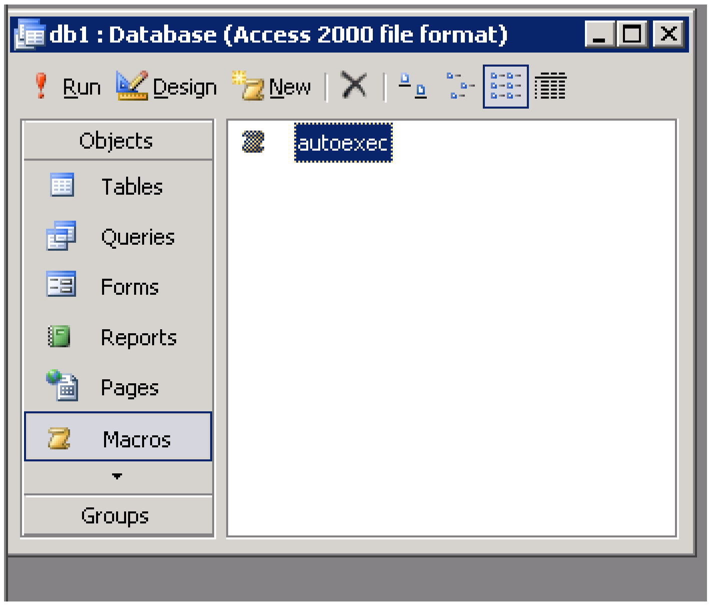
Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

Display 2. Design View of Table1.

AUTO EXECUTION MACRO IN MASTER ACCESS DATABASE:

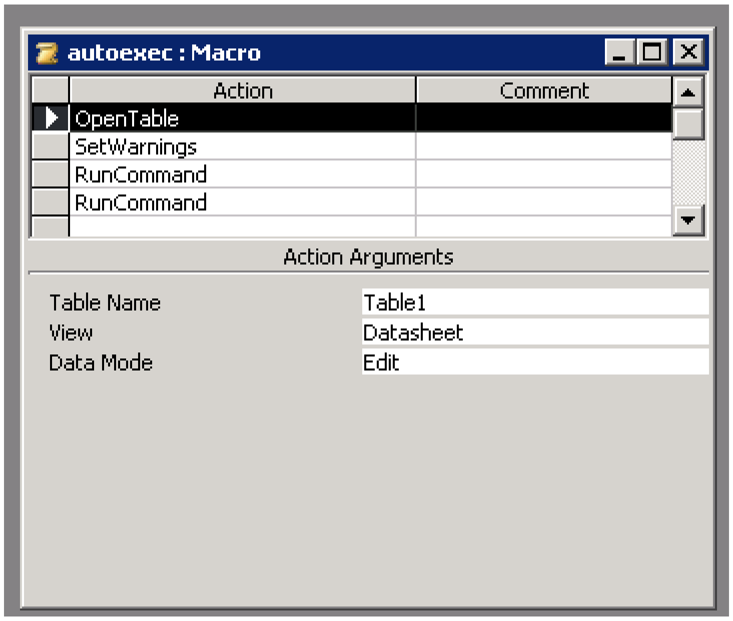
Auto execution macro will be executed at the startup of the Access Database. The new macro is created using the macros tab in the Access Database. Further, different commands are added using the design view.

Below is the screenshot of the Autoexec macro:



Display 3. Autoexec Macro in Master Access Database.

Below is the screenshot of Autoexec macro in the design view:



Action	Comment
OpenTable	
SetWarnings	
RunCommand	
RunCommand	

Table Name	Table1
View	Datasheet
Data Mode	Edit

Display 4. Design View of Autoexec Macro.

In the Autoexec macro the following values are assigned to the different series of actions:

1. For OpenTable insert ‘Table1’ in the Table Name tab.
2. For SetWarnings insert ‘NO’ in the Warnings On tab.
3. For the first RunCommand insert ‘SelectAllRecords’ in the Command tab.
4. For the second RunCommand insert ‘Paste’ in the Command tab.

COPY THE MASTER ACCESS DATABASE TEMPLATE TO THE PRODUCTION LOCATION

Prior to executing these statements, there are two system options needed: NOXWAIT and NOXSYNC. The NOXWAIT option specifies that the DOS command prompt window disappears without you having to type EXIT when the process is finished, while the NOXSYNC specifies that the process should execute asynchronously. That is, control is returned immediately to the SAS System and the command continues executing without interfering with your SAS session.

Since the Access table will have new data, due to execution of the autoexec macro in the Access database, we will always need a fresh copy of the Master Access Database Template.

The following command is used to copy the Master Access Database Template to the production location:

```
X copy "&location\Master\*.mdb" "&location";
```

Note, "`&location`" is your production location.

OPEN THE TARGET FILE IN EXCEL AND COPY THE CONTENT

In order for a client/server communication link to be established, both SAS and Excel must be running. Therefore, for the first iteration, it is necessary to programmatically launch Excel from a SAS session. There are several techniques available to launch Excel from SAS. The simplest one is the following statement:

```
%let rc=%sysfunc(system(start excel));
```

The above command is dependent on the finishing of previous commands. Note, the `SLEEP` function can be used frequently in the SAS command/datastep that is dependent upon previous jobs finishing. This will help avoid errors from occurring due to delays in the execution of previous SAS commands/data steps.

The syntax for the `SLEEP` function is given below:

```
data _null_;
   x=sleep(5);
run;
```

The above step will pause the SAS session for five seconds.

To communicate with Excel from SAS, the following `FILENAME` statement is used to establish the linkage between SAS and Excel via the DDE triplet:

```
filename excel DDE 'Excel|System';
```

The next step is to open the target file "`&in`" by sending the `Open` WordBasic command to Excel with a `data _null_` step. Once the file is open in Excel, select and copies the desire columns (here 50) by sending the `Select` and `Copy` commands as shown in the following:

```
data _null_;
   file excel;
   put '[Error(false)]';
   put '[Open("' "&in" '")]';
   put '[Select("C1:C50")]';
   put '[Copy]';
   put '[Close]';
   put '[Quit]';
run;
```

It is necessary to programmatically close the document with the `Close` command. Further, the `Quit` command will exit Microsoft Excel.

OPEN THE MICROSOFT ACCESS DATABASE FROM PRODUCTION LOCATION

The same approach as invoking Excel is used to start a new Access session:

```
%let rc=%sysfunc(system(start msaccess &location\db1.mdb));
```

The above command will open the Master Access Database "db1.mdb" at production location.

Due to the Auto execution of macro in db1.mdb, the following events will happen:

- Table1 will be selected in edit mode
- All of the Access warnings will be set to 'NO', which has the same effect as pressing 'ENTER' whenever a warning or message box is displayed.
- All of the records in table1 are selected using the 'SelectAllRecords' command.
- All of the records from the Excel file are pasted in table1 using the 'Paste' command.

To communicate with Access from SAS, the following `FILENAME` statement is used to establish the linkage between SAS and Access via the DDE triplet:

```
filename access DDE 'Msaccess|System';
```

The following command will save the data in Table1, and then close and exit the Access session.

```
data _null_;
   file access;
   put '[Save]';
   put '[Close]';
   put '[Quit]';
run;
```

After execution of the above datastep, the data will be stored in Table1.

READ THE CONTENT INTO SAS USING LIBNAME OR PROC IMPORT PROCEDUE

After the text or data is saved into an Access table, it is time to read it into SAS by either using the `LIBNAME` statement or `PROC IMPORT.` If the SAS\ACCESS license is available, the following libname can be used:

```
libname test access "&location\db1.mdb";
libname dat "&location";

data dat.&out.;
   set test.table1;
run;
```

In the above datastep, "`&location`" is the production location and "`&out`" is the output dataset created and saved in library "`dat`".

GET HEADER FROM THE EXCEL FILE BY EXCEUTING MACRO %EXCEL_HEADER

If the Excel file has header then the following macro %Excel_Header (See Appendix) will rename the variables from access with the one from Excel.

There are only two keyword parameters:
Header_obs: Line number for header for e.g., `1.`
Special: Delete any special character from header variable for e.g., `%.`

Below is the simple macro call to `%Excel_Header`.

```
%Excel_Header(header_obs=, special=);
```

DELETE THE MICROSOFT ACCESS DATABASE FROM PRODUCTION LOCATION

To avoid any overwriting of the data, Access database file "db1.mdb" can be deleted as follows:

```
filename db "&location\db1.mdb";
%let rc=%sysfunc(fdelete(db));
filename db clear;
```

"`&location`" is the production location.

%Convert_Excel2_SAS

To facilitate and automate the above discussed steps from opening the Excel file to reading the Access file into SAS, a SAS macro called `%Convert_Excel2_SAS` was developed for SAS v9.1.3 or above (see Appendix for details). The user can easily extend the macro to fit other SAS versions.

There are only six keyword parameters:
In: Define the path and the name of the input Word file, e.g., `C:\Demo\test.xls.`
Out: Define the SAS dataset name, for e.g., `raw`.
Location: Production Location, for e.g., `C:\Demo.`
Header: Need header for e.g., `Y or N`.
Header_obs: Line number for header for e.g., `1`.
Special: Delete any special character from header variable for e.g., `%`.

Below is the simple macro call to `%Convert_Excel2_SAS`.

```
%Convert_Excel2_SAS (in= C:\Demo\test.xls, out=raw, location=C:\Demo,
header=,header_obs=,special=);
```

CONCLUSION

%Convert_ExceL2_SAS can convert the Excel into SAS datasets without any manual pre-process. Since variable formats are predefined in the Microsoft Access table, this will avoid any formatting changes from occurring to the data during the conversion process. Hence, the integrity of keeping the original content (including special characters) and structures of Excel is guaranteed.

REFERENCES

Gupta Ajay, 2011. Reading Title and Footnote from RTF Output into SAS® utilizing Microsoft® Excel. Proceedings of the PharmaSUG 2011 Conference, paper CC11.

Gupta Ajay, Matthew Lesko. 2010. Importing Data from RTF Output into SAS® utilizing Microsoft® Access/Word in an intriguing, efficient way. Proceedings of the PharmaSUG 2010 Conference, paper CC06.

Microsoft Corporation. 2000. *Word 95 WordBasic Help File*. http://www.microsoft.com/downloads/details.aspx?familyid=1a24b2a7-31ae-4b7c-a377-45a8e2c70ab2&displaylang=en

Excel 4.0 Macro Reference. http://www.microsoft.com/downloads/en/details.aspx?FamilyID=00D31943-3AD1-4DF1-9F93-C19C7E84F01C&amp;displaylang=en

ACKNOWLEDGMENTS

Thanks to Kjersten Offenbecker and Karen Easterling for there reviews and comments. Thanks to my wife Neha and my Mother in Law for motivating me to write this paper. Special Thanks to my son Advait for keeping me awake while writing this paper.

CONTACT INFORMATION

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

Name: Ajay Gupta, M.S.
Enterprise: PPD Inc.
Address: 3900 Paramount Parkway
City, State ZIP: Morrisville, NC-27560
Work Phone: (919)-456-6461
Fax: (919)-654-9990
E-mail: Ajay.Gupta@ppdi.com, Ajaykailasgupta@aol.com

APPENDIX

```
%macro Convert_Excel2_SAS(in=,out=,location=,header=,header_obs=,special=);

          options noxwait noxsync;
          X copy "&location\Master\*.mdb" "&location";
          %let rc=%sysfunc(system(start excel));

          data _null_;
                x=sleep(5);
          run;

          filename Excel DDE 'Excel|System';

          data _null_;
            file excel;
                put '[Error(false)]';
                put '[Open("' "&in" '")]';
                put '[Select("C1:C50")]';
                put '[Copy]';
                put '[Close]';
                put '[Quit]';
          run;

          data _null_;
                x=sleep(2);
          run;

          %let rc=%sysfunc(system(start msaccess &location\db1.mdb));

          filename access DDE 'Msaccess|System';

          data _null_;
                x=sleep(5);
          run;

          data _null_;
                file access;
                put '[Save]';
                put '[Close]';
                put '[Quit]';
          run;

          libname test access "&location\db1.mdb";
          libname dat "&location";

          data dat.&out.;
                set test.table1;
          run;

          libname test clear;

          data _null_;
                x=sleep(2);
          run;

          filename db "&location\db1.mdb";

          %let rc=%sysfunc(fdelete(db));

          filename db clear;

    %if &header="Y" %then %do;
```

```
        %macro excel_header(header_obs=&header_obs, special=&special);

            data header;
                set &out. (firstobs=&header_obs obs=&header_obs);
            run;

            proc transpose data=header out=header2;
                var _ALL_;
            run;

            data header2;
                set header2;
                col1="_"||strip(compress(col1,&special.));
                col1=tranwrd(strip(col1),' ' ,'_');
            run;

            proc sql noprint;
                select count(*) into:n from header2;

                select col1 into:file1-:file%cmpres(&n)
                        from header2;

                select _name_ into:file11-:file1%cmpres(&n)
                    from header2;
            quit;

            data &out.;
                set &out.;
                    rename
                        %do i=1 %to &n;
                            &&File1&i=&&file&i
                        %end;;
            run;

        %mend excel_header;
        %excel_header;

    %end;

%mend Convert_Excel2_SAS;
```