# Paper 7120-2016

# The Combination of SAS® and VBA Makes Life Easier

Yanrong Zhu, Medtronic plc.

#### **ABSTRACT**

VBA has been described as a glue language and has been extensively used in exchanging data between Microsoft products such as Excel and Word or PowerPoint. How to trigger the VBA macro from SAS® via DDE has been widely discussed in recent years. However, using SAS to send parameters to VBA macro was rarely reported. This paper provides a solution for this problem. Copying Excel tables to PowerPoint utilizing the combination of SAS and VBA is illustrated as an example. The SAS program rapidly scans all Excel files contained in one folder, passes the file information to VBA as parameters, and triggers the VBA macro to write PowerPoint files in a loop. As a result, a batch of PowerPoint files can be generated by just one single mouse-click.

#### INTRODUCTION

VBA has functions that SAS does not have. For example, VBA is able to easily exchange data between office products, which unfortunately is not flexibly achieved in SAS. The integration of SAS and VBA extends the functionality of SAS and makes the programming more powerful. One topic has been widely discussed is that how to trigger a VBA macro from SAS *via* DDE. SAS programmers can employ PUT statements within a DATA step to send commands to run a VBA macro in an Office application. The following are the typical SAS statements.

```
options noxwait noxsync;
x '"C:\Program Files\Microsoft Office\Office\excel.exe" ';

/* Sleep for 5 seconds to give Excel time to come up */
data _null_;
    x=sleep(5);
run;

filename cmds dde 'excel|system';

data _null_;
    file cmds;
    /* Open the excel file test.xlsm which contains the VBA macro */
    put '[open("c:\temp\test.xlsm")]';
    /* Run myVBAmacro */
    put '[run("test.xlsm!myVBAmacro")]';
run;
```

As shown above, the first PUT statement is applied to open the Excel file *test.xlsm* which contains the VBA macro, and the second PUT statement is used to send the command to run the VBA macro *myVBAmacro* within *test.xlsm*. They are good enough to run a VBA macro *via* SAS, but it is still not possible for the programmers to modify parameters in *myVBAmacro* from SAS. This paper presented successful modifications in both SAS statements and the VBA macro to achieve the goal that the VBA parameters can be defined *via* SAS.

### **BACKGROUND**

One example from a clinical trial project was used in this paper where over one hundred sites were enrolled. For each site, we had an Excel file to list the performance. Our objective was to transfer each Excel file to a PowerPoint file. Since we had hundreds of the Excel files, it would be extremely time-consuming if we manually copy and paste the Excel data to PowerPoint, so we developed a VBA macro called *ExcelToPPT*(*excelFile*, *pptTemplate*) to do this work. This macro has two parameters, one is *excelFile*, which is the Excel source file, and the other one is *pptTemplate*, which is the PowerPoint template.

Here is the example of how to call this macro. Suppose the Excel file is "C:\myProject\Excelfile1.xlsx" and the PPT template is "C:\myPPTtemplate.PPT", to copy all the data in the excel file to the power point template, we just need to run the following VBA program.

```
Sub myVBAmacro()
   Call ExcelToPPT("C:\myProject\Excelfile1.xlsx", "C:\myPPTtemplate.PPT")
End Sub
```

Compared with manual data copy from Excel to PPT, the above VBA programs have already saved a lot of time, however, another problem to be solved is that the parameters in ExcelToPPT() cannot be altered by program. Only one Excel file was read and populated to the PowerPoint template after the VBA macro *myVBAmacro* was triggered. Since we had hundreds of Excel files in this project, it would require a lot of time to manually change the parameters hundreds of times. Although SAS has the ability to trigger the VBA macro, it does not have the ability to modify the code inside the VBA macro. Therefore, we need a bridge to communicate these two programming languages.

#### SOLUTION

Excel can be the host application of VBA, and in the meanwhile, Excel and SAS can be linked by various methods such as DDE, ExcelXP Tagset, PROC IMPORT, etc. Excel's friendly relationship with VBA or SAS makes it an ideal bridge to inter-communicate these two programming languages. To change the parameters within VBA from SAS, we can firstly store the parameters in an Excel file using SAS. Then VBA reads the parameters from that Excel file. For example, to solve the issue we had for the clinical trial project above, we utilized the following steps.

 Step1. Using SAS to read and store all the Excel file names to the external Excel file C:\TEMP\Excellnfo.xml.

```
%macro get_filenames(ExcelLocation = );
filename _dir_ "%bquote(&ExcelLocation.)";
data filenames(keep=memname);
handle=dopen( '_dir_' );
if handle > 0 then do;
   count=dnum(handle);
   do i=1 to count;
      memname=dread(handle,i);
   output filenames;
   end;
end;
rc=dclose(handle);
run;
filename _dir_ clear;

data sourceFiles;
set filenames;
```

```
length Excelfile $500.;
if index(memname, '.xlsx ') ^= 0;
Excelfile = "&ExcelLocation.\"||strip(memname);
run;

/*Save the Excel info to C:\TEMP*/
ods listing close;
ods TAGSETS.EXCELXP file= "C:\TEMP\ExcelInfo.xml"
options(sheet_name="Sheet1");
    proc report data= sourceFiles nowd split = "*";
        column Excelfile;
    run;

ods TAGSETS.EXCELXP close;
ods listing;
%mend;

%get_filenames(ExcelLocation = C:\myProject);
```

This example program firstly created a SAS dataset named *filenames* in work library, listing all the file names under *C:\myProject*. Next, SAS created another dataset called *sourceFiles*, in which a new variable *Excelfile* was defended, which was the file path and name of each Excel source file. Finally, the dataset *sourceFiles* was output to external Excel file *C:\TEMP\ExcelInfo.xml* by ExcelXP Tagset.

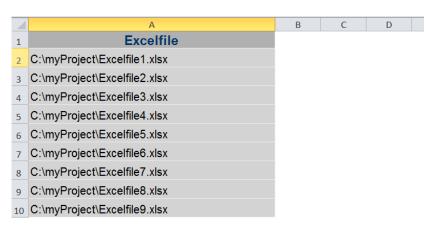


Figure 1. Contents in C:\TEMP\ExcelInfo.xml

As shown in figure 1, after running the above program, all the excel file paths and names were stored in column A in C:\TEMP\ExcelInfo.xml.

• Step2. Modify the VBA program to read the Excel file info from C:\TEMP\ExcelInfo.xml in loop.

```
Sub myVBAmacro()
    Dim ExcelPathApp Excel.Workbook
    Dim irow, k As Integer
    Dim Excelsource As String

'open ExcelInfo file and Activate the Document
    Set ExcelPathApp = Excel.Workbooks.Open("C:\TEMP\ExcelInfo.xml")
    ExcelPathApp.Activate
```

As illustrated above, each value in column A of *C:\TEMP\ExcelInfo.xml* was read as the value of *Excelsource* in loop, and within each loop, the macro *ExcelToPPT* was called *and Excelsource* was read as one of the parameters.

• Step3. Trigger the VBA macro *myVBAmacro()* from SAS using the code in the introduction. After the macro was triggered, the loop was initialized, and PPT files were generated in batch.

The following is the figure to illustrate the whole process.

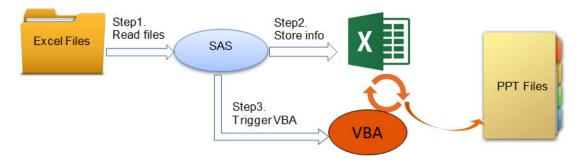


Figure 2. Process of populating batch PPT files by SAS and VBA

# **CONCLUSION**

In this paper a simple, clear, and efficient way was presented to employ Excel as the bridge to send parameters from SAS to VBA. Using this methodology, we significantly compressed our workload time from 65 hours to less than two. We would expect this solution would have wide applicability for the VBA and SAS programmers.

#### **REFERENCES**

Feng, Ying. "Generating Custom Report Tables: Using SAS with DDE and VBA" Available at http://www2.sas.com/proceedings/sugi30/161-30.pdf.

Roper, Christopher. "Using SAS and DDE to execute VBA macros in Microsoft Excel" Available at <a href="http://www2.sas.com/proceedings/sugi25/25/cc/25p098.pdf">http://www2.sas.com/proceedings/sugi25/25/cc/25p098.pdf</a>.

# SAS Support Website,

http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000209687.htm

SAS Support Website,

http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000209538.htm

SAS Support Website,

http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#exittemp.htm

# **CONTACT INFORMATION**

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

Yanrong Zhu Medtronic Inc yanrongzhu@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.