Paper 98-25

Using SAS and DDE to execute VBA macros in Microsoft Excel

Christopher A. Roper, Qualex Consulting Services, Inc., Cary, NC. USA

Abstract

Visual Basic for Applications (VBA) is the programming language Microsoft uses for all their desktop products, including Microsoft Excel. VBA allows programmatic control for virtually all the functionality of Excel. Because of the popularity of Excel as an end-user tool and the power and flexibility of the SAS System to deliver information, integration of these two resources is inevitable. One of the most popular and powerful protocols for integrating SAS and Excel is Dynamic Data Exchange (DDE). However, SAS can not send VBA commands to Excel using DDE. This paper will present a technique for circumventing this limitation. A VBA macro can be recorded in an Excel workbook, and then that macro can be executed by a SAS application using DDE and a command from the Excel 4.0 macro language.

Introduction

For many, perhaps most, SAS developers the easiest and fastest way to write VBA code is to first go into a Microsoft application (in this paper Microsoft Excel is used as the application), turn on the macro recorder, perform the steps and functions desired, and then terminate the macro recorder. The end result is a stored VBA program (macro) that can be executed at a later date. This makes it possible for a SAS developer to automate tasks in the Microsoft application, and therefore vastly improve the functionality of an integrated system that takes advantage of the relative strengths of the SAS System and the Microsoft application.

There are two protocols for accomplishing this integration, Object Linking and Embedding (OLE), and Dynamic Data Exchange (DDE). For many developers, DDE is the protocol of choice; if for no other reason than it is the older of the two and the developer is more comfortable using it in an application. However, this presents a problem. As DDE is an older protocol, SAS can not send VBA commands via DDE to Excel, or any other external application. The question then becomes, how does an application developer integrate those stored VBA programs in an Excel file into a SAS application? The answer lies in the backward compatibility of Excel (and other Microsoft applications). All of the “old” or Excel version 4.0 macro functions are still supported under all later releases. Therefore, executing an Excel macro that is written in VBA is no different than executing an Excel macro that is written using the old version 4.0 macro recording language. The DDE command submitted from SAS uses the Excel version 4.0 command RUN to submit the VBA program stored in the Excel application as a macro. SAS doesn’t know that it has caused Excel to execute VBA code, all it knows is that it just asked Excel to execute a macro.

The Task:

Open Microsoft Excel and record a macro to format a range of data cells and create a 3-D pie chart of the data. Save this workbook. Next, write a SAS program to export some data from the SAS examples dataset SASHELP.IAWFIN to the saved Excel workbook. Then continue the SAS program to execute the stored VBA macro to create the chart of the new data.

The Steps:

1. Open Excel
2. Select Tools, Macro, Record New Macro from the pull down menus
3. Select OK
4. Select the range A1:C2 (6 cells total)
5. Select Insert, then Chart from the pull down menus
6. Select Chart Type PIE (on the left)
7. Select Exploded pie with 3-D effect
8. Select Next
9. Select Next (again)
10. Enter some text for the Chart Title
11. Select the Legend tab (at the top of the page)
12. Select radio button Bottom under Placement
13. Select the Data Labels tab (at the top of the page)
14. Select the Show Value radio button
15. Select Next
16. Select Finish
17. Select the Stop Recording Button
18. Click on cell A1 **(VERY IMPORTANT!)**
19. Save As C:\TEST\ SEUGI.XLS
20. Close Excel

Now there is a VBA macro saved in an Excel workbook named C:\TEST\SEUGI.XLS that we will want to execute using SAS and DDE. First, open Excel, but do not load any workbooks. The following code will do everything else.

The fileref EXCEL will be used to send commands to Excel, to open the workbook with the VBA macro, and execute that macro.

The fileref EXPORT will be used to pass data from SAS to Excel.

```
/************************************/
/* Assign two filerefs              */
/*   EXCEL:  Send Excel commands    */
/*   EXPORT: Send data to Excel     */
/************************************/
filename EXCEL  DDE 'EXCEL|SYSTEM';
filename EXPORT DDE
   'EXCEL|Sheet1!r1c1:r2c6' notab ;
```

The XSYNC execution option will cause SAS to pause processing while Excel is processing the commands sent to it by DDE.

```
options xsync;
```

This data _null_ step will open the workbook with the saved VBA macro.

```
/************************************/
/* Open the Excel workbook          */
/************************************/
data _null_;

  file excel;

  put '[open("C:\TEST\SEUGI.XLS")]';

run;
```

Next, create some data to be exported to Excel from the SAS examples dataset SASHELP.IAWFIN.

```
/*******************************/
/* Create data to be exported  */
/* and charted in MS Excel     */
/*******************************/
proc summary data =
sashelp.iawfin nway;
  var budget actual variance;
 output out = iawfin sum=;
run;
```

Now, export the SAS data to Excel. Place the labels that will be used by the chart in the first row, then the data into the second row. The variable TAB is assigned the hexadecimal equivalent of the ASCII tab character (09). The TAB variable will be used to control the columns into which the data are placed. Without the TAB variable, all the data would be placed in the first column.

```
/*******************************/
/* Export the data to Excel    */
/*******************************/
data _null_;

  set IAWFIN;

  tab = '09'x;

  file EXPORT;

  if _n_ = 1 then
   put 'BUDGET' tab 'ACTUAL' tab
'VARIANCE';

  put BUDGET tab ACTUAL tab
VARIANCE;

run;
```

Now that the workbook has something to chart, execute the VBA macro MACRO1 in the Excel workbook.

```
/*************************************/
/* Execute the macro                 */
/*************************************/
data _null_;

  file excel;
  put '[RUN("Macro1")]';

run;

/*************************************/
/* Clean up the filerefs             */
/*************************************/
filename EXCEL  clear;
filename EXPORT clear;
```

Conclusion

This program demonstrates that while the SAS System will not execute VBA commands using DDE, it is possible to use the SAS System and DDE to execute Excel macros written in the VBA language. The general techniques described within this paper can also be applied to other Microsoft applications such as MS Word and MS Access.

Author Contact

Christopher A. Roper
Senior Systems Consultant
Qualex Consulting Services, Inc.

Acknowledgments

The Code:

```
options xsync;
/*----------------------------------------------------------------------*/
/* Assign filerefs:  EXCEL:  Excel commands  EXPORT: Data to Excel      */
/*----------------------------------------------------------------------*/
filename EXCEL  DDE 'EXCEL|SYSTEM';
filename EXPORT DDE    'EXCEL|Sheet1!r1c1:r2c6' notab ;

/*----------------------------------------------------------------------*/
/* Open the Excel workbook                                              */
/*----------------------------------------------------------------------*/
data _null_;

  file excel;
  put '[open("C:\TEST\SEUGI.XLS")]';

run;

/*----------------------------------------------------------------------*/
/* Create some data to be exported and charted in MS Excel              */
/*----------------------------------------------------------------------*/
proc summary data = sashelp.iawfin nway;
 var budget actual variance;
 output out = iawfin sum=;
run;

/*----------------------------------------------------------------------*/
/* Export the data to Excel                                             */
/*----------------------------------------------------------------------*/
data _null_;

  set IAWFIN;
  tab = '09'x;
  file EXPORT;
  if _n_ = 1 then put 'BUDGET' tab 'ACTUAL' tab 'VARIANCE';
  put BUDGET tab ACTUAL tab VARIANCE;

run;

/*----------------------------------------------------------------------*/
/* Execute the macro                                                    */
/*----------------------------------------------------------------------*/
data _null_;

  file excel;
  put '[RUN("Macro1")]';

run;

/*----------------------------------------------------------------------*/
/* Clean up the filerefs                                                */
/*----------------------------------------------------------------------*/
filename EXCEL  clear;
filename EXPORT clear;
```