Paper 034-29

Let SAS® Tell Microsoft Word® to Collate

Haiping Luo, Department of Veterans Affairs, Washington, DC

ABSTRACT

This paper describes the process of developing an application to collate and print Microsoft(MS) Word documents with SAS output tables and charts by region. The user's need was to customize Word documents using SAS generated values, collate the Word document with SAS created tables and charts for individual regions selected by the SAS code, then print and mail the set of Word and SAS files to individual regions. After comparing different possible solutions, the author chose to use SAS sending WordBasic scripts and Word macro names through Dynamic Data Exchange (DDE) links to automate the process. In the resulting application, SAS code was written to prepare data, to send WordBasic scripts, to invoke VBA subroutines, and to loop through regions. Word was driven by SAS to customize Word documents using SAS provided values and to print both Word and SAS output files. This paper presents the design of the application, explains the code of an example, and discusses the problems encountered during the development process. The code of the example application is included in the appendices. The example application was tested in a Windows 2000 platform. PC SAS 8.2 with Base SAS and SAS/Graph installed. The version of MS Office was 2000.

INTRODUCTION

My office regularly needs to mail letters associated by SAS output tables and charts to about 160 hospitals. Those hospitals are usually selected by SAS programs. The manual process was to customize the letter in MS Word with hospital address, attach the tables and graphics created by SAS for a specific hospital, then put the letter and SAS output into the envelope to send to that specific hospital. The manual process was tedious and had the risk of sending materials with patient information to a wrong place. My office wanted to develop a small application to automate the collating and printing process. This paper describes the process of developing such an application, presents the method, and explains an example.

REQUIREMENTS

Based on the manual process, the collating application should meet the following requirements:

1. Incorporate a well-formatted letter created in MS Word.
2. Each letter should have the specific address selected by the SAS program.
3. Each letter should be followed by the table and chart produced by the SAS program for that specific location.
4. The letter, table, and chart should be printed together.
5. The collating and printing process should continue to the next location until completed for all selected locations.
6. The automation should accept input from the user to specify parameters such as paths to files.
7. The whole process should not require intermittent manual interaction and afterward cleanup.

POSSIBLE SOLUTIONS

Three possible solutions were considered for this collating application: 1. Mail merger using MS Word; 2. Use SAS to send WordBasic scripts to drive Word through Dynamic Data Exchange (DDE); 3. Use SAS to write Visual Basic for Application (VBA) code to drive Word to collate and print. The advantages and difficulties of using these three solutions are discussed below.

1. MAIL MERGER IN MS WORD

MS Word allows the merge of mailing addresses and data from different sources with a Word file to create customized Word documents. The merging process can be accomplished using the Mail Merge wizard provided by Word. To use this merging feature, the address file and other data source to be merged should be available first. In our situation, the hospital list is dynamically selected by a SAS program. I will have to run the SAS program to create the mailing list and output tables and charts first. Then from Word I need to run the Mail Merger wizard manually to conduct the merge. To a competent VBA programmer, maybe this whole process, including the SAS process, can be automated in VBA. To a more competent SAS programmer who knows both SAS and Word and their communication means thoroughly, maybe this process can be automated in SAS. But I am not so familiar with VBA and not so strong in SAS to automate this Mail Merge process dynamically.

2. SAS DRIVES WORD THROUGH WORDBASIC AND DDE

Hendricks (Reference 1) developed some SAS macros in 1994 to use Word as a print server to print SAS produced tables. This method involves sending WordBasic code from SAS through DDE. WordBasic was the scripting

language used by Word before MS Office 97. Word 97 and later still accept WordBasic for backward compatibility. WordBasic statements can be sent from SAS through a DDE session like below:

```
*** Execute a Windows command to open Word before setting up DDE;
X '"C:\Program Files\Microsoft Office\Office\WINWORD.EXE"';
*** Set up DDE connection between SAS and MS Word to send Word scripting commands;
filename cmds dde 'WinWord|System';
data _null_;
    file cmds;
    *** Send a WordBasic statement to open a Word document;
    put '[FileOpen.Name = "' "&wfile" '"]';
    *** Set-up Landscape format ***;
    put '[FilePageSetup.PageWidth = "11" + Chr$(34)]'; ...
```

Hendricks used Macro Recording in Word 6 to obtain those WordBasic statements. Note that a SAS macro variable &wfile was embedded in the FileOpen statement to instruct Word to open the file at the path of &wfile.

The advantage of using a DDE connection to send WordBasic statements is its simplicity and straightforwardness. In a DDE session, all WordBasic scripts are sent by the SAS' PUT statement one statement at a time. SAS macro variables can be wrapped in WordBasic scripts by the PUT statement to accept values from SAS dynamically. If there is a problem during the execution, the DDE error message will pinpoint that which statement has a problem.

I ran into difficulties when I tried to use Hendricks' method to achieve my tasks. I could not obtain WordBasic code through recording macros in Word because I was using Word 2000. The programming language for Word and other MS Office products has become Visual Basic for Applications. The recorded macros comprise of VBA statements. After trying to send the VBA statements from SAS to Word through a DDE session unsuccessfully, I learned that DDE cannot execute VBA statements one by one. That is because, unlike a WordBasic statement, a VBA statement is not executable on its own. Nor can DDE send a VBA subroutine with parameters. This means I cannot let SAS determine values such as file paths, file names, text strings, etc. dynamically and pass them as VBA parameters to Word in a DDE session. These difficulties made it hard to create a WordBasic-through-DDE application or a VBA-through-DDE application to collate and print documents.

3. SAS DRIVES WORD THROUGH DYNAMIC VBA

Stetz (Reference 2) presented a mechanism other than using DDE to let SAS instruct Word to conduct tasks. Stetz used SAS to generate and invoke dynamic VBA macros in Word to conduct tasks specified by SAS programs. The SAS-VBA mechanism has these elements:

1. Word is set up to add a module RUN_VBA in its Normal template.
2. RUN_VBA is a VBA subroutine to be called by Word when the invoking command indicates so. This subroutine can read in Windows environment variables and run Word macros.
3. A SAS macro file run_vba.sas is developed to create and run a batch file dynamically. The batch file sets Windows environment variables, invoke Word with the call to run that RUN_VBA subroutine, and delete the batch file itself after Word starts.
4. Another SAS program is written to carry out the specific tasks for a project:
    a. Generate a Word VBA macro in a data step to conduct the specific tasks in Word.
    b. Run the SAS macro in the run_vba.sas file, set the Windows environment variables to pass SAS generated values to the Word macro created in the data step.

The advantages of using Stetz' method are: 1. VBA macros can be generated through the macro recording tool in newer version of Word. 2. SAS can set any values in VBA dynamically because the whole VBA code is written by SAS on the fly.

The difficulties of applying Stetz' method came from: 1. understanding the complicated interactions among multiple processes and files; 2. preparing a working environment in Word; and 3. writing SAS code to generate VBA on the fly with dynamic SAS output embedded in the VBA code. I tried to run Stetz' example code. I followed all the steps described in Stetz' paper and reached the point that the dynamic Word macro .bas file was generated by the SAS program successfully. But it seemed neither Word was invoked or Word file was created as expected. I tried to diagnose what was wrong in the chain of SAS and VBA code interactions but could not identify the problem.

4. DECISION

After reviewing and testing the above possible solutions, I adopted a combined method: using a DDE session to send both WordBasic and VBA subroutine names from SAS to Word to automate Word tasks. This combined method has the advantage of maintaining DDE's characteristics of simplicity and straightforwardness, while utilizing VBA's strength of richness and sophistication. Because it is basically a DDE method, it still has the limits of needing some

WordBasic code and not sending VBA subroutines with parameters. But with a careful design plus trial and error, this method automated our collating and printing process successfully with pretty light coding.

DESIGN

The basic design of the combined method has the following aspects:

1. The tasks to be done in Word should be separated into two types: one type of the tasks requires obtaining dynamic values from the SAS side; the other type of the tasks is done in Word without needing SAS input.
2. For the type needing SAS input, WordBasic statements will be used to complete the tasks. SAS macro variables will be embedded in the WordBasic statements to pass the dynamic values.
3. For the type not needing SAS input, Word macros will be recorded beforehand and saved in Word's Normal.dot template as VBA subroutines.
4. SAS statements will be used to start a DDE session with Word, to send WordBasic scripts with dynamic values to Word, and to invoke Word macros by sending VBA subroutines names without parameters.

EXAMPLE

Below is an example to illustrate how this design can meet our collating needs. This example has five types of components:

1. Two input sample datasets. One dataset is from the SAS library called sashelp.shoes. This dataset has the regional data we are supposed to send out. The other dataset is a mailing address list for the regional subsidiaries. The mailing list is contained in an Excel sheet.
2. One pre-written Word document. This document is to be printed before the regional data. The document needs to be customized to include the subsidiary's manager name and address for the corresponding data output.
3. A SAS macro %mxcollate. This macro sends WordBasic scripts and Word macros names to customize, collate and print Word file and SAS output files.
4. Two Word macros, PrintOpen() and SetLandscape(). The Word macros were recorded in Word at the computer where this collating application would run. The macros were saved in Word's normal.dot template to be called by %mxcollate.
5. A main SAS program. This program obtains input dataset and mailing address, selects regions conditionally, loops through the selected regions to alter and print the Word file and prints the regional data output in table and in chart.

The output of the process is a set of printed Word document, a data table, and a data graphic chart for one region, then another set for another region, so on and so forth until all selected regions' materials being printed.

The appendices of this paper contain: the SAS macro %mxcollate, the main SAS program collate-loop.sas, and the mailing address list. With the materials in the appendices, plus the SAS dataset sashelp.shoes, any Word document, and two Word macros PrintOpen() and SetLandscape() being recorded following the instructions in the "How to Test the Example" section, an interested SAS programmer can run the collating application to see how it works. The following sub-sections will explain the code of the two SAS programs, %mxcollate and collate-loop.sas.

%MXCOLLATE

The SAS macro %mxcollate does the following things:

1. Set up the DDE connection from SAS to Word.
2. Send a WordBasic statement to open the Word file.
3. Insert address lines to the Word document through WordBasic statements.
4. Print the Word document using the PrintOpen() VBA subroutine.
5. Delete inserted address lines using WordBasic statements.
6. Save and close the Word file.
7. Open the SAS output file in RTF using a WordBasic statement.
8. Change the page orientation using VBA subroutine SetLandscape().
9. Print the RTF file with SAS created table and chart using the PrintOpen() Word macro.
10. Close the RTF file.

1. Set up the DDE connection from SAS to Word. Before a DDE connection is up, NOXWAIT and NOXSYNC options should be set. NOXWAIT specifies that the command processor automatically returns to the SAS session after the specified Windows command is executed. NOXSYNC allows the operating system command execute asynchronously with the SAS session. With NOXSYNC in effect, one can execute an X command or X statement and return to the SAS session without closing the process spawned by the X command or X statement. The other options

below allow logging macro execution messages to SAS log. The DDE session is started by the FILENAME *fileref* DDE 'WinWord|System' ; statement.

```
options noxwait noxsync macrogen mlogic symbolgen;
filename cmds dde 'WinWord|System';
```

2. Send a WordBasic statement to open the Word file &wfile. The WordBasic statement should be wrapped in single quotes in a PUT statement. Note also that the WordBasic statement itself contains a pair of [ ] quotation marks for the DDE session.

```
data _null_;
  file cmds;
  put '[FileOpen.Name = "' "&wfile" '"]';
```

3. Insert address lines to the Word document through WordBasic statements.

```
  put '[StartOfDocument]';
  put '[Insert "To" + Chr$(9) + "' "&header1" '" + Chr$(13)]';
  put '[Insert Chr$(9) + "' "&header2" '" + Chr$(13)]';
  ...
```

4. Print the Word document using the PrintOpen() VBA subroutine if the &print flag indicates to print. The way to run a VBA subroutine (a Word macro) is to put the Word macro's name in a PUT statement. Note that the subroutine should not require any input parameters. Note that the second ";" is necessary to end the %if statement.

```
%if &print=1 %then
    put '[PrintOpen()]'; ;
```

5. & 6. Delete inserted address lines, save and close the Word file using WordBasic statements. The The the Delete step could be avoided if one wishes to close but not to save the altered Word file. But I could not find the Word statement that allows to close without saving and that will avoid a message box asking the user to confirm. Since the whole collating process should be automated without waiting for a user's input in the middle, these deleting statements are used to avoid inquiring the user. Also since the deleting statements contain dynamic SAS macro values, WordBasic statements are used to embed the SAS values.

```
  put '[StartOfDocument]';
  put '[EditReplace .Find = "To" + Chr$(9) + "' "&header1" '" + Chr$(13), .Replace
= "", .ReplaceAll]';
  ...
  put '[FileSave]';
  put '[FileClose]';
```

7. Open the SAS output file in RTF using a WordBasic statement.

```
  put '[FileOpen.Name = "' "&sasout" '"]';
```

8. & 9. If the &print flag indicates to print the file, change the page orientation using VBA subroutine SetLandscape() then print the SAS created table and chart using the PrintOpen() Word macro.

```
  %if &print=1 %then %do;
    put '[SetLandscape()]';
    put '[PrintOpen()]';
  %end;
```

10. Close the SAS output file and de-assign the command file reference "cmds". This prevents the Word commands in the referred file being carried over to the next iteration in the loop.

```
  put '[FileClose]';
run;
filename cmds clear;
```

COLLATE-LOOP.SAS

Collate-loop.sas is the main program to run the collating and printing process. This program has four major steps:

1. Preparing the regional data.
2. Setting up a looping structure.
3. Calling %mxcollate to collate with regional information.
4. Close the applications.

1. Preparing the regional data.

Collate-loop.sas first sets the options to allow DDE commands being executed asynchronously (noxwait noxsyn), to display macro log (macrogen mlogic symbolgen), and not to show run date at the print out (nodate). The %let statements set up the path to all data, Word, and program files. The shell statement to load Word is placed here for easy changing.

```
%let path=d:\home\paper\sas\sugi29\;
X '"C:\Program Files\Microsoft Office\Office\WINWORD.EXE"';
```

Then PROC SORT is used to select some subsidiaries from the sashelp.shoes dataset. The subsidiaries will be the regions to loop through the collating process.

```
proc sort data=sashelp.shoes
    (where=((substr(region,1,2) eq 'Ca') and
           (substr(subsidiary,2,1) eq 'o')))
    out=shoes;
    by subsidiary product;
```

In a later data step, address data are read in from an Excel sheet and merged with dataset SHOES. Some business statistics are calculated to be sent to the output file.

```
X "D:\home\paper\sas\sugi29\subaddress.xls";
FILENAME addr DDE "EXCEL|sheet1!r2c1:r6c5" ;
DATA address;
    INFILE addr missover DLM='09'X NOTAB DSD ;
    LENGTH
        subsidiary $12
        manager $20
    ...
data shoes;
    merge shoes address;
    by subsidiary;
    ** Calculate some business statistics;
    ros=(returns/sales)*100;
    ...
```

2. Setting up a looping structure.

The subsidiaries in the selected dataset is counted and assigned a subsidiary number. The total number of the subsidiaries &subnumber will be used by the loop structure to control the end of the loop.

```
proc sort data=shoes(keep=subsidiary) out=shoes0 NODUPKEY;
    by subsidiary;
data snumber;
    set shoes0 end=lastobs;
    by subsidiary;
    sn=_n_;
    if lastobs then call symput('subnumber',sn);
    Label sn = "Subsidiary Number";
run;
```

A %printloop macro is structured to loop through subsidiaries for the collating process. This macro will call the collating macro %mxcollate. The parameters of %printloop contain all parameter values needed by the %mxcollate macro.

```
%macro printloop
    (wdfile=d:\home\paper\sas\sugi29\testword.doc,
           out=d:\home\paper\sas\sugi29\shoes,
           h1=, h2=, h3=, h4=,
           prnt=0);
```

Loop through subsidiaries, produce table and chart and save them in a RTF file.

```
    %do i=1 %to &subnumber;
        *** Get regional data;
        data sales;
            set shoes(where=(sn=&i));
            by subsidiary;
            ** obtain the mailing headers from the regional data;
            if first.subsidiary then do;
                call symput('h1',manager);
                ...
        ods rtf file="&out.&i..rtf";
        *** Print the regional data to a table in the rtf file;
        title1 "Sales Report";
```

```
proc print data=sales
...
proc gchart data=sales;
    vbar product / sumvar=ros patternid=midpoint raxis=axis1;
...
```

3. Call %mxcollate to collate and print within the loop iteration.

```
%mxcollate
    (wfile=&wdfile,
     sasout=&out.&i..rtf,
    %if &h1 ne %then header1=&h1.,;
    %if &h2 ne %then header2=&h2.,;
    %if &h3 ne %then header3=&h3.,;
    %if &h4 ne %then header4=&h4.,;
    print=&prnt
    );
```

4. Close the applications after completing %printloop. Both Word and Excel need to be closed, but I only found the WordBasic statement to close Word successfully.

```
filename cmd2 dde 'WinWord|System';
data _null_;
    file cmd2;
    put '[AppClose]';
run;
filename cmd2 clear;
```

HOW TO TEST THE EXAMPLE

You may copy the files from the appendices and follow these steps to test the example:

1. Assume all data output, Word, and program files will be saved in folder d:\sugi29.
2. Save mxcollate.sas as d:\sugi29\mxcollate.sas.
3. Save collate-loop.sas as d:\sugi29\collate-loop.sas. Change the macro variable &path and the full path to your winword.exe:
   ```
   %let path=d:\sugi29\;
   X '"path to \Microsoft Office\Office\WINWORD.EXE"';
   ```
4. Record two Word macros PrintOpen() and SetLandscape() to your Word normal.dot template by following these steps:
   a. Open Word, click Tools, Macro, Record New Macro.
   b. At the Record Macro window, type PrintOpen as the macro name, make sure the “Store Macro In” field has the value of “All documents (normal.dot)”.
   c. When the recording is on, click File, Print, OK. Then close the macro recording.
   d. Repeat steps a, b, c to record SetLandscape(). At step c, click File, Page Setup, Paper Size, Landscape, then OK as the recorded steps.
5. Create a Word file testword.doc as the Word document. In the document, hit enter first, then make a centered Title and some normal-font content. Save this document in d:\sugi29\.
6. Create an Excel file subaddress.xls and place it in d:\sugi29\ . Copy the address data in Appendix 3 to Sheet1.
7. Now you are ready to test the application:
   a. Open collate-loop.sas from PC SAS’ program editor.
   b. Run the program with the parameter &prnt in macro %printloop set to 0.
   c. Observe testword.doc being opened, altered, changed back, then closed, one subsidiary at a time.
   d. After running the program, review the log, also go to d:\sugi29\ folder to view the two output files, shoes1.rtf and shoes2.rtf, to see if they are correct.
   e. If everything is all right, change %printloop parameter &prnt to “prnt=1” then run the program again. This time the collated files will be printed.

LESSONS LEARNED

Although this collating application is a very small SAS-drives-Word application, there were still some lessons learned during the trial and error process. Hope this summary of lessons, difficulties, and tips may save some people’s time in similar work.

1. A DDE link can only send one self-executable script at a time. A statement such as a WordBasic script or a Word

macro name like PrintOpen() is all right. If you try to send a VBA subroutine with a parameter, say, OpenAddPrint("test.sas"), you will run into this error and the debug message:

```
compile error
syntax error
Sub TmpDDE()
    OpenAddPrint__("test.sas"
End Sub
```

2. I tried to record a CloseWithoutSave Word macro to close a revised document without saving. But the action of clicking 'No' to the 'Save As' screen could not be recorded in the macro. So a Save As window will pop up even when the CloseWithoutSave macro is run.

3. I cannot find a VBA statement or record a macro to close Excel from SAS.

4. If you are not satisfied with table and chart produced by SAS, change their font, style, size, color, etc. in SAS rather than trying to let SAS tell Word to change.

5. When recording a macro in Word, I was not able to select a graphic without stopping the macro recording. That means no VBA statements to select a graphic could be recorded into a Word macro.

6. After collate-loop.sas is run, the SAS Results window may not show the right output. But the output RTF files are all right.

7. I tried to use these statements to reset the path to winword.exe:

```
%let wordpath=C:\Program Files\Microsoft Office\Office\WINWORD.EXE;
X '"&wordpath"';
```

Somehow Word will not open when the program is run the first time in a SAS session. The second run in the same SAS session will open Word correctly. I do not know why.

8. To avoid the inserted address strings being placed in the centered Title line of the Word documents, it is wise to place the title line at the second line of the Word file. Also make sure that the first line is left-adjusted and has a correct style for the inserted address lines to use.

9. You may find all WordBasic statements from this link, but there is no information about parameters for each statement:
http://www.powerlan-usa.com/office_converter/Userguide/WordBasic%20Summary.htm#A

CONCLUSION

This collating application automates the process of printing a customized Word document along with SAS generated tables and charts by region. The automation was achieved by SAS sending WordBasic scripts and VBA subroutine names through DDE links. An example was used to demonstrate the application. Any SAS programmer with intermediate SAS skill and a little knowledge of MS Word should be able to try the example code.

REFERENCES

1. Hendricks, Adam. (1994), "Gettable.sas" and "SAS2Word.sas", SAS macros written in August 1994 and published on SAS-L discussion board. http://groups.google.com/groups?q=Hendricks+group:comp.soft-sys.sas&hl=en&lr=&ie=UTF-8&group=comp.soft-sys.sas&safe=off&selm=4fdo3u%24ptb%241%40mhadg.production.compuserve.com&rnum=2 (last accessed on 12/28/2003)

2. Stetz, Mark. (2000), "Using SAS Software and Visual Basic for Applications to Automate Tasks in Microsoft Word: An Alternative to Dynamic Data Exchange", *Proceedings of the Twenty-Fifth Annual SAS User Group International Conference*, paper 21-25 in the CD-ROM.

ACKNOWLEDGEMENTS AND TRADEMARK CITATION

Appreciation goes to Robert Langberg at the Department of Veterans Affiars for his review and comments.

CONTACT INFORMATION

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

Haiping Luo, Phone: 202-273-6271

APPENDIX 1. MXCOLLATE.SAS

```
/* Purpose: This file contains a SAS macro %mxcollate. This macro combines WordBasic
 * and Word Macro to customize, collate & print Word file and SAS output files.
```

```
  * Created by: Haiping Luo      Date: 12/14/2003
  * Macro parameters:
  * wfile: the full path to the Word document;
  * sasout: the full path to the SAS output file in a Word readable format;
  * header1 to header4: the strings to be inserted into the Word document;
  * print: a flag to control if print actually (=1) or just test the code (=0); */
%macro mxcollate (wfile=d:\home\paper\sas\sugi29\testword.doc,
             sasout=d:\home\paper\sas\sugi29\shoes.rtf,
             header1=test, header2=%str( ), header3=%str( ), header4=%str( ),
             print=0);
   options noxwait noxsync macrogen mlogic symbolgen;
   *** set up connection between SAS and MS Word to set VBA commands from SAS to Word;
   filename cmds dde 'WinWord|System';
   data _null_;
       file cmds;
   *** open the Word Document;
     put '[FileOpen.Name = "' "&wfile" '"]';
   *** Insert address;
     put '[StartOfDocument]';
     put '[Insert "To" + Chr$(9) + "' "&header1" '" + Chr$(13)]';
     put '[Insert Chr$(9) + "' "&header2" '" + Chr$(13)]';
     put '[Insert Chr$(9) + "' "&header3" '" + Chr$(13)]';
     put '[Insert Chr$(9) + "' "&header4" '" + Chr$(13)]';
   *** run Word macro to print the Word file;
   %if &print=1 %then put '[PrintOpen()]'; ;
   *** Delete inserted address then save and close the Word file;
   *** This will avoid the "Do you want to save ..." message waiting for input;
     put '[StartOfDocument]';
     put '[EditReplace .Find = "To" + Chr$(9) + "' "&header1" '" + Chr$(13), .Replace
   = "", .ReplaceAll]';
     put '[EditReplace .Find = Chr$(9) + "' "&header2" '" + Chr$(13), .Replace = "",
   .ReplaceAll]';
     put '[EditReplace .Find = Chr$(9) + "' "&header3" '" + Chr$(13), .Replace = "",
   .ReplaceAll]';
     put '[EditReplace .Find = Chr$(9) + "' "&header4" '" + Chr$(13), .Replace = "",
   .ReplaceAll]';
     put '[FileSave]';
     put '[FileClose]';
   *** open the SAS output file in a Word acceptable
     format (txt, cvs, rtf, cmg, ...);
     put '[FileOpen .Name = "' "&sasout" '", .ReadOnly=True]';
   *** Print the SAS output file;
     %if &print=1 %then %do;
       put '[SetLandscape()]';
       put '[PrintOpen()]';
     %end;
   *** Close the SAS output file;
     put '[FileClose]';
   run;
   filename cmds clear;
%mend mxcollate;
```

APPENDIX 2: COLLATE-LOOP.SAS

```
/* Purpose: The main program to loop through regions to collate & print Word & SAS
files.
 * Created by: Haiping Luo       Date: 12/14/2003 */
options noxwait noxsync macrogen mlogic symbolgen nodate;
*** Path to all data, Word and program files;
```

```
%let path=d:\home\paper\sas\sugi29\;
*** Open Word;
X '"C:\Program Files\Microsoft Office\Office\WINWORD.EXE"';
*** include mxcollate.sas which contains %mxcollate;
filename macr1 "&path.mxcollate.sas";
%include macr1/nosource2;
*** Prepare data, choose a subset of subsidiaries;
proc sort data=sashelp.shoes
       (where=((substr(region,1,2) eq 'Ca') and (substr(subsidiary,2,1) eq 'o')))
out=shoes;
    by subsidiary product;
*** Count and assign a number to each selected subsidiary for looping purpose;
proc sort data=shoes(keep=subsidiary) out=shoes0 NODUPKEY; by subsidiary;
data snumber; set shoes0 end=lastobs; by subsidiary;
       sn=_n_;
       if lastobs then call symput('subnumber',sn);
       Label sn = "Subsidiary Number";
data shoes; merge shoes snumber;  by subsidiary; run;
*** Get the address data in Excel format, open .xls first;
X "&path.subaddress.xls";
FILENAME addr DDE "EXCEL|sheet1!r2c1:r6c5" ;
DATA address;
   INFILE addr missover DLM='09'X NOTAB DSD;
   LENGTH subsidiary $12 manager $20 address1 $40 address2 $20 address3 $20;
   INPUT subsidiary $ manager $ address1 $ address2 $ address3 $;
data shoes; merge shoes address; by subsidiary;
   ** Calculate some business statistics;
   ros=(returns/sales)*100;
   rov=(returns/inventory)*100;
   avgsales_sub=sales/stores;
   format ros rov 6.2 avgsales_sub dollar8.;
   Label
       ros="Returns on Sales"
       rov="Returns on Inventory"
       avgsales_sub="Average Store Sales at the Subsidiary";
run;
 /* Macro %printloop loops through each region to print Word and SAS output files;
  * Macro parameters:
  * wdfile: the full path to the Word document;
  * out: the path string to the SAS output file without the file name extension;
  * h1 to h4: the strings to be inserted into the Word document;
  * prnt: a flag to control if print actually (=1) or just test the code (=0);
  * Note that these parameters will be passed to the nested macro %mxcollate;  */
%macro printloop
       (wdfile=&path.testword.doc, out=&path.shoes,
       h1=, h2=, h3=, h4=, prnt=0);
    %put <<< Total number of the selected subsidiaries is &subnumber. >>>;
       %do i=1 %to &subnumber;
           *** Get regional data;
           data sales;
               set shoes(where=(sn=&i));
               by subsidiary;
               ** obtain the mailing headers from the regional data;
               if first.subsidiary then do;
                   call symput('h1',manager);
                   call symput('h2',address1);
                   call symput('h3',address2);
                   call symput('h4',address3);
```

```
            end;
          *** Output to a rtf file;
          ods listing close;
          *** If want to save all SAS output files, name the rft files with &i;
             *ods rtf file="&out..rtf";
             ods rtf file="&out.&i..rtf";
          *** Print the regional data to a table in the rtf file;
          title1 "Sales Report";
          proc print data=sales
              (drop=region manager address1 address2 address3 sn) label noobs
              style(header)={background=white};
              by subsidiary;
          run;
          *** Output the regional data to a chart in the rtf file;
          goptions reset=all;
          proc gchart data=sales;
              vbar product / sumvar=ros patternid=midpoint raxis=axis1;
              title1 "Sales Report";
              axis1 minor=none;
              by subsidiary;
          run;
          quit;
          ods rtf close;
          ods listing;
          *** Use WordBasic and Word Macro to collate;
          %mxcollate
              (wfile=&wdfile,
         /* If want to save all SAS output files, use the rft file name with &I;*/
                    sasout=&out.&i..rtf,  /* sasout=&out..rtf,*/
              %if &h1 ne %then header1=&h1.,;
              %if &h2 ne %then header2=&h2.,;
              %if &h3 ne %then header3=&h3.,;
              %if &h4 ne %then header4=&h4.,;
              print=&prnt,);
    %end;
%mend printloop;
*** Run the loop macro;
%printloop (wdfile=&path.testword.doc, out=&path.shoes,
           h1=, h2=, h3=, h4=, prnt=0 );
*** Close Word;
filename cmd2 dde 'WinWord|System';
data _null_;
    file cmd2;
    put '[AppClose]';
run;
filename cmd2 clear;
```

APPENDIX 3: SUBADDRESS.XLS

Subsidiary	manager	address1	address2	address3
Calgary	Jim Smith	123 M Street		Calgary
Montreal	Brigitte Zuech	321 G Drive	Suite E	Montreal
Ottawa	Larry Watkins	987 T Ave		Ottawa
Toronto	Nancy Green	346 H Lane		Toronto
Vancouver	Don Ali	675 D Way	Suite 3	Vancouver