Paper 076-2010

Using SAS® Software and Visual Basic for Application to Dynamically Manipulate SAS List Files

Zemin Zeng, Forest Research Institute, Inc., Jersey City, NJ
Mei Li, ImClone Systems, Branchburg, NJ

ABSTRACT

This paper demonstrates how to apply Visual Basic for Application (VBA) procedures in the Microsoft Word to dynamically manipulate SAS output list files. In the paper, the author shall show how to convert SAS list files into Rich Text Format (RTF) files with appropriate page setup; extract the first page of each SAS list file; assemble all RTF files into a single file and send print command to Microsoft Word from SAS. All the manipulation tasks shall be accomplished by executing an integrated SAS macro which dynamically interacts with VBA subroutines. The code of both the SAS and VBA macros are included in the paper. The testing of the macros was verified in a Windows Operating System PC with SAS version 9.0 and 9.1.3 and Microsoft Office 2003 and 2007.

INTRODUCTION

In Pharmaceutical industry, it is common for SAS programmers to create customized RTF tables from the SAS output list files. The SAS list files are space-delimited and not tab-delimited, necessary post-process of the list files is needed to create publication quality tables in RTF format for clinical study reports. For the convenience of the quality review and cross checking all the RTF tables and lists, it is highly demanded to assemble the individual RTF files into a single file and print the hard copy of it. In this paper, we will show all the details on how to accomplish these demanding tasks.

We shall apply Visual Basic for Application (VBA) techniques from SAS environment to achieve our goals. Microsoft Visual Basic allows people to define their own VBA projects to programmatically execute tasks in Microsoft Word. Stetz (SUGI 25 Proceedings, 2000) built an elegant bridge between SAS and VBA. Stetz created a VBA macro called Run_VBA in Microsoft Visual Basic system, which allows users to dynamically create new VBA procedures from SAS environment, and execute new Word application as instructed by the SAS statements. In the papers (Zeng & Li, Coders' Corner section, NESUG Proceedings, 2009, and Li & Zeng, Application Big & Small, NESUG Proceedings, 2009), we successfully applied Stetz's VBA techniques to effectively solve some old problems in some new perspectives. In this paper, we would like to explore further and demonstrate how we could utilize powerful VBA macros to manipulate Microsoft Word files in SAS programming.

The organization of the paper would be the following: 1. Briefly review Stetz's Run_VBA macro and our modified version which could be shared for all our three VBA related applications. 2. Introduce the application of converting SAS lists files to RTF files and extracting the first page of each SAS list file. 3. Introduce the application of assembling all RTF file into a single file in the next section. 4. Introduce the application of printing Microsoft Word files. Finally the complete steps to successfully run our integrated SAS macro will be addressed. For the convenience of the interested readers, we present the key ingredients of the VBA macro codes of our applications in Appendix 3 and Appendix 4 so that the readers could easily modify them to fit their needs. At the end of the paper, the Run_VBA Word macro and the integrated SAS macro MANLST for our applications will also be included in the Appendix 1 and Appendix 2, respectively. We hope our paper encourages interested users (experienced SAS programmers or VBA experts) to develop new applications in Microsoft Word through dynamic conversations between SAS software and VBA.

STETZ'S RUN_VBA MACRO

In the paper (SUGI 25 Proceedings, 2000), Stetz provided a VBA macro Run_VBA to automate Word task. To meet author' need for the applications in Microsoft Word, we removed all unnecessary parts for our applications and made it more simple and straightforward as for our applications. It is worth pointing out that the following two changes are necessary for our applications in the next sections (See the Appendix 1 for our version):

1. Delete the 'Kill bas file' statement so that there is enough time to execute the new dynamically generated VBA macro which is defined in the bas file in SAS.
2. Removed the 'Application.Quit' statement for allowing the printing process to be completed before quit the application.

These two modifications are necessary for accommodating our three applications via such a single RUN_VBA macro. We will do the file clean up by issuing an X-command after each application in the Microsoft Word is done.

The following outlines how the Run_VBA works:

- Reads the values of the environment variables
- Creates a new module in the Normal template
- Imports the BAS file defined in SAS to generate a VBA macro in the new module
- Executes the VBA macro
- Removes the new module from the Normal template

The Run_VBA needs to be installed in Microsoft Word before running our integrated application SAS macro MANLST in the Appendix 2. For the detailed setup steps, please refer to the paper (Zeng & Li, Coders' Corner section, NESUG Proceedings, 2009).

APPLICATION 1: CONVERTING SAS LIST FILES TO RTF AND EXTRACTING THE FIRST PAGE

We may directly open a SAS list file in the Microsoft Word, but the layout of the list file is kind of gets messed up. In order to recover the original display of the SAS list file, in the Microsoft Word we need to set the page orientation, page margins, output page width and height, font and its size and then save it as RTF file. These steps can be done manually, but the best way is to take advantage of the power of the VBA macro. The VBA subroutine in the Figure 1.1 in the Appendix 3 is the key ingredients of our VBA macro of converting a SAS list file to a RTF file, users may modify the page margins, font and its size, and other parameters to fit their needs. The users could save these VBA codes to the Visual Basic Editor with necessary modifications to programmatically convert a SAS list file to RTF.

As for extracting the first page of each SAS list file, we first convert the list file to RTF, then select the range of the first page of the file. There are two cases to handle, one is the multipage files and the other is single page file. By examining the page numbers of the file in VBA to identify which case it belongs to. It is easy to handle the single page file since there is no extraction needed. However, it is a little bit tricky here to correctly select the first page in VBA for multipage files. A natural thought is to refer to the page numbers, and set the start of the selection as the beginning of the file and then move the selection to the next page. But it turns out that it doesn't work well for multipage files since you will result in a blank page after the first one. Here is what we managed to avoid getting the blank page. We set the start of the selection as the beginning of the file, then we move the selection to the page 2, and then we have to move the selection one line back to successfully set the selection to the end of the first page. For details, please see the key codes of our VBA macro to extract the first page in the Figure 1.2 in the Appendix 3.

In the part of the MANLST macro (See Appendix 2) for our converting application, we first get all the SAS list file names and prepare for the loop by file names, then in the loop for each SAS list file we write all the VBA macro codes to an external BAS file. Finally we construct a batch run file and call the system function to run it and invoke the Run_VBA macro in Microsoft Word. During the execution of the Run_VBA, the BAS file will be imported into the Microsoft Visual Basic system and generate a VBA macro like in the Figure 1.1 and Figure 1.2. The application of converting SAS list files to RTF and getting the first page of each list file will be achieved by running the BAS file generated VBA macro.

APPLICATION 2: ASSEMBLING ALLRTF FILES INTO A SINGLE FILE

In the second part of the SAS macro MANLST for setting all RTF files (which were converted from the SAS list files) into a single RTF file, we use the Dir function in VBA to return a string representing the name of a RTF file in the folder containing all the RTF files. Here is the loop process over all the targeted RTF files: Dir returns the first file name that matches pathname (namely the path of the folder in which all RTF files reside and are going to be set into a single file). To get any additional file names that match pathname, call Dir again with no arguments. When no more file names exist, Dir returns a zero-length string (""). Once a zero-length string is returned, the loop is done. During the loop, we insert each RTF file to a new Word document and use the Word page break to separate them. At the end, we save the new document. Make sure don't insert page break when inserting the last file, otherwise a blank page will appear at the end of the assembled file.

The construction of the Application 2 is similar to converting application above, in the BAS file we intent to design the assembling process only for either all page RTF files or first page files. The users can choose the parameter all2one in the SAS macro MANLST to decide either all page RTF files or first page files to set into a single RTF file. Specifying all2one=1 in the MANLST macro means to set the first page files and all2one=2 means to set the all page files into a single RTF file, respectively. While Specifying all2one=0 means no assembling. The users may modify the bas file to meet their needs for their specific applications in Microsoft Word. The key point is that it is so flexible and dynamic to define the Word procedures in SAS.

APPLICATION 3: PRINTING WORD FILES

The printing application is quite straightforward; we open the document and then use the print command to finish the task. There is only one thing needs to take care, since the printing process takes time, we should make sure to quit the application after the printing task is done. It is why we can't issue the Application.Quit in the VBA subroutine for the printing process. Instead, we let the system to rest for several seconds, and then use the Dynamic Data Exchange (DDE) commands to quit the application. It is a good programming habit to quit the application whenever you initiate VBA application and you application is done.

STEPS TO RUN OUR INTEGRATED SAS MACRO

The users may copy the Run_VBA macro from Appendix 1 and SAS macro MANLST from the Appendix 2 and follow the steps below to test our three applications.
Step 1: Copy macro Run_VBA to the Microsoft Visual Basic Editor under the Normal template. There are many steps to have it set up appropriately in Microsoft Visual Basic system. We refer the users to follow the complete guidelines in the paper (Zeng & Li, Coders' Corner section, NESUG Proceedings, 2009).

Step 2: Assume we will work under the directory C:\VBA, the list files are in the folder C:\VBA\LST and output RTF files to be in the folder C:\VBA\RTF\pageall (for the RTF files from the SAS list files) and C:\VBA\RTF\pageone (for only the first page of the SAS list files) . Once such folder structures set up, we are ready to go to the next step for running the MANLST macro.

Step 3: Copy the SAS codes of the macro MANLST in the Appendix 2 to your SAS editor and run it. Here is the sample call of the macro MANLST:

```
%MANLST(filepath=C:\VBA\LST, outpath=C:\VBA\RTF, lst2rtf=1, all2one=1, isprint=0);
```

The first two macro parameters are the locations of the SAS list files and output RTF files. The assembled single files will be generated in the outpath folder. For the last three macro parameters, please see the below table for the explanation:

Parameters	Explanation
lst2rtf	=1 for converting all SAS list files to RTF; =0 for not converting (if RTF files and their first pages files are available, we usually specify lst2rtf=0 for assembling and printing processes next).
all2one	=1 for assembling only first pages, =2 for all pages, =0 for no assembling.
isprint	=1 for printing the assembled file of all first pages, =2 for the assembled file of all pages, =0 for not printing.

CONCLUSION

The SAS macro MANLST serves as an example to demonstrate how we could dynamically create VBA procedures from SAS environment, and execute Word new applications as instructed by the SAS statements. Meanwhile, the macro shows its power to manipulate Word files.

REFERENCES

Li, Mei and Zeng, Zemin (2009). *Visual Basic for Application with SAS® Software to Dynamically Collate Microsoft Word Documents*, Applications Big & Small Section, Proceedings of the NESUG 2009.

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 Users Group International Conference, Paper 21-25.

Zeng, Zemin and Li, Mei and (2009). *Using SAS® Software and Visual Basic for Application to Automatically Produce Customized RTF Tables from SAS List Files*, Coders' Corner Section, Proceedings of the NESUG 2009.

ACKNOWLEDGMENTS

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.

CONTACT INFORMATION

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

Zemin Zeng, Forest Research Institute, Inc., Jersey City, NJ
Email: *zengzemin@gmail.com / zemin.zeng@frx.com*

Mei Li, ImClone Systems, 33 ImClone Drive, Branchburg, New Jersey
Email: *Jennifer.li@imclone.com*

APPENDIX 1: RUN_VBA WORD MACRO

```
Sub Run_VBA()
'** Adapted from the R_VBA Word Macro, originally     **
'** created by Mark Stetz, SUGI 25, Paper: 21-25, 2000**

'*** Declare local variables ***
Dim vbcs As VBComponents
Dim vbc As VBComponent
Dim vba_filename As String
Dim vba_path As String
Dim vba_pgm As String

'*** Read environment variables (created in RUN_VBA.SAS) ***
vba_path = Environ("VBA_PATH")
vba_pgm = Environ("VBA_PGM")

'*** Create a new Word document ***
 Documents.Add

'*** Get a reference to the Normal template component collection ***
Set vbcs = VBE.VBProjects("Normal").VBComponents

'*** Construct the filename of the SAS generated VBA macro ***
vba_filename = vba_path & "\" & vba_pgm & ".bas"

'*** Add a new module to the Normal template ***
Set vbc = vbcs.Add(vbext_ct_StdModule)

'*** Import the VBA macro code defined in the SAS ***
vbc.CodeModule.AddFromFile FileName:=vba_filename

'*** Run the SAS generated VBA macro ***
Application.Run MacroName:=vba_pgm

'*** Delete the code module from the Normal template ***
vbcs.Remove VBComponent:=vbc

'*** Close the document ***
Documents.Close
End Sub
```

APPENDIX 2: MANLST SAS MACRO

```
%macro MANLST(filepath=, outpath=, lst2rtf=1, all2one=0, isprint=0);
/***************************************************************************
 * MACRO NAME:    MANLST
 * AUTHOR:        Zemin ZENG and Mei LI
 * DATE:          10/16/2009
 * SAS VERSION:   Version 9.0 or 9.1.3
 * PURPOSE:       To convert List files to customized RTF files, extract
 *                first page, set all files into one and print them
 * PARAMETERS:    filepath-- the path of the folder containing list files
 *                outpath -- the path of the folder to store RTF files
 *                lst2rtf -- 1 for converting lst to rtf, 0 for no
 *                all2one -- 1 for assembling 1st pages, 2 for all
 *                           page, 0 for no assembling
 *                isprint -  1 for printing all 1st pages, 2 for all pages
 *                           0 for not printing
 *
 * CALL SAMPLE:   % MANLST(filepath=C:\VBA\LST, outpath=C:\VBA\RTF,
 *                         lst2rtf=1, all2one=1, isprint=0)
 *----------------------------------------------------------------------------
 * CAUTIONS:     1. Need first set the R_VBA Word macro in Microsoft Word
 *               2. Change the wordpath path if your Winword.exe file is not
 *                  in the folder: C:\Program Files\Microsoft Office\Office12
 *----------------------------------------------------------------------------*/
OPTIONS noxwait noxsync mprint;
%local memnum mem flist wordpath vba_path vba_pgm filename;
%let wordpath=%str(C:\Program Files\Microsoft Office\OFFICE12);
%let vba_path=&filepath;

/*The macro callbat is for the purpose to define a  batch file
  and call the system to run it*/
%macro callbat(fpath=&filepath, batname=, resttime=15, delbas=);
data _null_;
  file "&fpath\&batname..bat" ;
  put "set VBA_PATH=&fpath" ;
  put "set VBA_PGM=&batname" ;
  put "cd &wordpath" ;
  put "start winword.exe /mrun_vba" ;
  put "del ""&fpath\&batname..bat"" " ;
run;
data _null_;
  call system("""&fpath\&batname..bat""") ;
run;

data _null_;
  x=sleep(&resttime);
run;

%if &delbas=1 %then %do;
  x "del &fpath\&batname..bas";
%end;
%mend callbat;

/*-----------------------------------------------------------------*
| Application I: Convert SAS list files to RTF                     |
/*-----------------------------------------------------------------*/
%if &lst2rtf=1 %then %do;
data newfile(keep=nlst nfile fexten);
  length nlst nfile $100;
  rc=filename('id',"&filepath");
  dirid=dopen('id');
  numsel=dnum(dirid);
  do i=1 to numsel;
    nlst=dread(dirid,i);
    x=reverse(strip(nlst));
    nfile=reverse(substr(x, 5));
    fexten=upcase(scan(x, 1, '.'));
    if fexten='TSL' then output;
  end;
  rc=dclose(dirid);
run;

proc sql noprint;
  select nfile into :flist separated by ','
  from newfile;
quit;
%let memnum = 0;
%do %until(%length(%qscan(%quote(&flist), %eval(&memnum+1), %str(,)))=0);
   %let memnum = %eval(&memnum + 1);
   %let mem    = %qscan(%quote(&flist), &memnum,  %str(,));
   %let vba_pgm=mem&memnum;

data _null_;
   file "&vba_path\mem&memnum..bas";
   put "Sub &vba_pgm()" ;
   put 'Dim newdoc As Word.Document' ;
   put 'Dim page1 As Range' ;

  /*page set up the list files and save as rtf*/
   put 'Documents.open' ' "' "&filepath\&mem..lst" '"' ;
   put 'Selection.WholeStory' ;
   put 'With ActiveDocument.PageSetup' ;
   put '    .Orientation = wdOrientLandscape' ;
   put '    .TopMargin = InchesToPoints(1.5)' ;
   put '    .BottomMargin = InchesToPoints(1)' ;
   put '    .LeftMargin = InchesToPoints(1)' ;
   put '    .RightMargin = InchesToPoints(1)' ;
   put '    .PageWidth = InchesToPoints(11)' ;
   put '    .PageHeight = InchesToPoints(8.5)' ;
   put 'End With' ;
   put 'Selection.Font.Size = 7' ;
   put 'Selection.Font.Name = "Courier New"' ;
   put "ActiveDocument.SaveAs FileName:= " '"' "&outpath\pageall\&mem..rtf" '"'
        ", _ " ;
   put ' FileFormat:=wdFormatRTF' ;
   put 'ActiveDocument.close' ;

  /* to get the first page of the files*/
   put 'Documents.open' ' "' "&outpath\pageall\&mem..rtf" '"' ;
   put 'If ActiveDocument.BuiltInDocumentProperties(wdPropertyPages) > 1 Then' ;
   put 'Set page1 = ActiveDocument.Range' ;
   put 'Selection.GoTo wdGoToPage, wdGoToAbsolute, 1';
   put 'page1.Start = Selection.Start' ;
   put '   Selection.GoTo wdGoToPage, wdGoToAbsolute, 2' ;
   put '   Selection.GoTo What:=wdGoToLine, Which:=wdGoToPrevious, Count:=1' ;
   put 'page1.End = Selection.Start' ;
   put 'page1.Copy' ;
   put 'Documents.Close SaveChanges:=False' ;
   put 'Set newdoc = Application.Documents.Add' ;
   put 'newdoc.Bookmarks("\EndOfDoc").Range.Paste' ;
   put 'Selection.WholeStory' ;
   put 'With ActiveDocument.PageSetup' ;
   put '  .Orientation = wdOrientLandscape' ;
   put 'End With' ;
   put 'newdoc.SaveAs' ' "' "&outpath\pageone\_1_&mem..rtf" '"'
        ", FileFormat:=wdFormatRTF" ;
   put 'newdoc.Close' ;
   put 'Else' ;
   put '   ActiveDocument.SaveAs FileName:= ' '"' "&outpath\pageone\_1_&mem..rtf"
       '"' ", _ " ;
   put '   FileFormat:=wdFormatRTF' ;
   put 'ActiveDocument.Close' ;
   put 'End If' ;
   put 'Application.Quit SaveChanges:=wdDoNotSaveChanges' ;
   put 'End Sub' ;
run;

%callbat(fpath=&filepath, batname=mem&memnum, resttime=10, delbas=1);
%end;
%end;

/*---------------------------------------------------------------*
| Application II: Set all files to a single file                |
/*--------------------------------------------------------------*/
%if &all2one=1 or &all2one=2 %then %do;
DATA _NULL_;
   file "&filepath\all2one.bas";
   put "Sub all2one()" ;
   put 'Dim Strpath As String' ;
   put 'Dim Strfile As String' ;
   put 'Dim FName As String' ;
   put 'Dim Newdoc1 As Word.Document' ;
   %if &all2one=1 %then %do;
   put 'Strpath = "' "&outpath\pageone\" '"' ;
   %end;
   %if &all2one=2 %then %do;
   put 'Strpath = "' "&outpath\pageall\" '"' ;
   %end;
   put 'Strfile = Dir(Strpath, vbNormal)' ;
   put 'Set Newdoc1 = Application.Documents.Add' ;
   put 'Do Until Strfile = "' '"' ;
   put '  FName = Strpath & Strfile' ;
   put '  Selection.InsertFile FileName:=FName, _' ;
   put '   ConfirmConversions:=False, Link:=False, Attachment:=False' ;
   put '  Strfile = Dir' ;
   put '  If Strfile <> "" Then' ;
   put '   Selection.InsertBreak Type:=wdPageBreak' ;
   put '  End If' ;
   put 'Loop' ;
   put 'Selection.WholeStory' ;
   put 'With ActiveDocument.PageSetup' ;
   put ' .Orientation = wdOrientLandscape' ;
   put 'End With' ;
   %if &all2one=1 %then %do;
   put 'Newdoc1.SaveAs "' "&outpath\one.rtf" '"' ', _' ;
   put '  FileFormat:=wdFormatRTF' ;
   put 'ActiveDocument.Close' ;
   %end;
   %else %if &all2one=2 %then %do;
   put 'Newdoc1.SaveAs "' "&outpath\all.rtf" '"' ', _' ;
   put '  FileFormat:=wdFormatRTF' ;
   put 'ActiveDocument.Close' ;
   %end;
   put 'Application.Quit SaveChanges:=wdDoNotSaveChanges' ;
   put 'End Sub' ;
run;
%callbat(fpath=&filepath, batname=all2one, resttime=10, delbas=1);
%end;

/*-----------------------------------------------------------------*
| Application III: Print files                                     |
/*----------------------------------------------------------------*/
%if &isprint=1 or &isprint=2 %then %do;
DATA _NULL_;
   file "&filepath\isprint.bas";
   put "Sub isprint()" ;
   %if &isprint=1 %then %do;
   put 'Documents.open' ' "' "&outpath\one.rtf" '"' ;
   %end;
   %else %if &isprint=2 %then %do;
   put 'Documents.open' ' "' "&outpath\all.rtf" '"' ;
   %end;
   put 'ActiveDocument.PrintOut' ;
   put 'ActiveDocument.Close' ;
   put 'End Sub' ;
run;
%callbat(fpath=&filepath, batname=isprint, resttime=15, delbas=1);

filename cmds dde 'WinWord|System';
DATA _NULL_;
  file cmds;
  put '[AppClose]';
run;
filename cmds clear;
```

```
%end;
%mend MANLST;
*%MANLST(filepath=C:\VBA\LST, outpath=C:\VBA\RTF, lst2rtf=1, all2one=1, isprint=0);
```

APPENDIX 3: KEY INGREDIENTS OF THE APPLICATION OF CONVERTING SAS LIST FILES TO RTF AND EXTRACTING THE FIRST PAGE OF THE RTF FILES

Figure 1.1: VBA code for list file to RTF

```
Sub lst2rtf()
Dim newdoc As Word.Document
Dim page1 As Range
Selection.WholeStory
With ActiveDocument.PageSetup
    .Orientation    = wdOrientLandscape
    .TopMargin    = InchesToPoints(1.5)
    .BottomMargin= InchesToPoints(1)
    .LeftMargin      = InchesToPoints(1)
    .RightMargin   = InchesToPoints(1)
    .PageWidth     = InchesToPoints(11)
    .PageHeight    = InchesToPoints(8.5)
End With
Selection.Font.Size = 7
Selection.Font.Name = "Courier New"
End Sub
```

Figure 1.2: VBA code to get page one

```
Sub lst2rtf()
Dim newdoc As Word.Document
Dim page1 As Range
Documents.open "C:\VBA\RTF\pageall\T2_4B.rtf"
If  ActiveDocument.BuiltInDocumentProperties(wdPropertyPages) > 1
Then
Set page1 = ActiveDocument.Range
Selection.GoTo wdGoToPage, wdGoToAbsolute, 1
page1.Start = Selection.Start
   Selection.GoTo wdGoToPage, wdGoToAbsolute, 2
   Selection.GoTo What:=wdGoToLine, Which:=wdGoToPrevious, Count:=1
page1.End = Selection.Start
page1.Copy
Documents.Close SaveChanges:=False
Set newdoc = Application.Documents.Add
newdoc.Bookmarks("\EndOfDoc").Range.Paste
Selection.WholeStory
With ActiveDocument.PageSetup
  .Orientation = wdOrientLandscape
End With
newdoc.SaveAs "C:\VBA\RTF\pageone\_1_T2_4B.rtf",  _
         FileFormat:=wdFormatRTF
newdoc.Close
Else
   ActiveDocument.SaveAs FileName:=
"C:\VBA\RTF\pageone\_1_T2_4B.rtf", _
   FileFormat:=wdFormatRTF
ActiveDocument.Close
End If
Application.Quit SaveChanges:=wdDoNotSaveChanges
End Sub
```

APPENDIX 4: KEY INGREDIENTS OF THE APPLICATION OF SETTING ALL RTF FILES INTO A SINGLE RTF FILE

Figure 2: VBA code to insert all file to one

```
Sub all2one()
Dim Strpath As String
Dim Strfile As String
Dim FName As String
Dim Newdoc1 As Word.Document
Strpath = "C:\VBA\RTF\pageone\"
Strfile = Dir(Strpath, vbNormal)
Set Newdoc1 = Application.Documents.Add
Do Until Strfile = ""
  FName = Strpath & Strfile
  Selection.InsertFile FileName:=FName, _
   ConfirmConversions:=False, Link:=False, Attachment:=False
  Strfile = Dir
  If Strfile <> "" Then
   Selection.InsertBreak Type:=wdPageBreak
  End If
Loop
Selection.WholeStory
With ActiveDocument.PageSetup
 .Orientation = wdOrientLandscape
End With
Newdoc1.SaveAs "C:\VBA\RTF\one.rtf", _
  FileFormat:=wdFormatRTF
ActiveDocument.Close
Application.Quit SaveChanges:=wdDoNotSaveChanges
End Sub
```