Automate the Creation and Manipulation of Word Processor Ready SAS® Output

Izabella Peszek and *Robert Peszek*

Izabella Peszek works as a senior biometrician at Merck and Co. Previously, Iza worked as a senior statistician for Ohmeda, Inc. A SAS user for nine years, her interests include statistical programming, graphical presentation of clinical data, and automation of the report writing. Iza holds a Ph.D. in statistics from the University of Maryland and an M.S. in applied mathematics from the University of Wroclaw, Poland.

Robert Peszek works as a lead analyst at Quality Software Systems Inc. His current areas of expertise and interests are design of software systems and SAS, Java, and PowerBuilder programming. He has a Ph.D. in applied mathematics from the University of Maryland and is a certified developer in Java and PowerBuilder. Robert has been using SAS software for five years. Robert and Iza have been married for 12 years.

Abstract

This paper presents an automated approach to the production and manipulation of word processor ready tables using a combination of SAS and WordBasic macros. The presented SAS macro creates Rich Text Format (RTF) files in a DATA _NULL_ step. Such files can be opened in any word processor equipped with an RTF converter. Manipulation of SAS generated tables and graphs using WordBasic macros is discussed. The use of such techniques eliminates the need for manual word processing of SAS outputs, resulting in cost and resource savings and in improving the quality and accuracy of reports.

Contents

Introduction

There was a time when SAS programmers did not need to worry too much about the formatting of their outputs. Reports were produced with courier font and everybody was happy. Today, the customers are much more demanding and want SAS outputs to be not only accurate and interesting but also eye-pleasing. In many cases, tabulations and graphics produced with SAS software are word processed to become a part of a bigger document. As you read this, someone is probably re-entering numbers from a SAS produced table to create a more appealing one. In many companies, a whole staff of secretaries and proofreaders are employed for just this purpose. This is costly and creates an opportunity for keying errors.

For that reason, people are trying to come up with innovative ways of transforming SAS generated tables into great looking documents with minimal word processing. Microsoft Office, Perfect Office, and Lotus Smart Suite became industry standards for document processing, and it seems natural to use them for this purpose. This can be done in many ways. Seidman and Aster proposed using Microsoft Word templates and WordBasic macros. Their idea was to produce a very simple DATA _NULL_ ASCII output, which can be read into an MS Word template. Special characters are inserted in DATA _NULL_ to mark places where different formatting is to be applied via MS Word macros. Another approach is to use the ODBC interface to read a SAS table into a Microsoft Access database and link this MS Access table to MS Word. We tried these methods and found them useful, yet both methods have some drawbacks. First of all, a considerable programming effort is required for each new table, which makes automation problematic. Second, these methods are not easily implemented by an average SAS programmer because some proficiency is required in VisualBasic, WordBasic, and MS Access programming. There are also some concerns about the system stability because the full automation would require flipping control between SAS and other applications.

We propose here yet another approach, which is versatile and easy to implement. Our method is to use a SAS DATA _NULL_ step to produce a document with Rich Text Format (RTF) specification. The RTF specification is a method of encoding formatted text and graphics for easy transfer between applications. An RTF file consists of unformatted text, control words, control symbols, and groups (readers familiar with $LaTeX$ or TeX will find many similarities between them and the RTF). Most word processors can convert RTF files into their native format, which makes RTF files platform- and application-portable. That is, the same output can be opened in MS Word, WordPerfect, or other word processor equipped with an RTF converter (even on different operating system) with no loss of formatting. Of course, not all SAS programmers are familiar with the RTF language, and we wanted a method that can be widely useful, so we developed a SAS macro to assist us in creation of RTF files. You need only to learn the syntax of this macro to be able to write RTF files. This macro is used within a DATA _NULL_ step in a way very similar to the regular PUT statement, and it is quite easy to learn. The unformatted text is inserted with a PUT statement, while macro variables provide appropriate control words and symbols. Moreover, users familiar with RTF language can easily expand the macro to suit their needs.

We describe the usage of this macro in a simple example in “Creating RTF files in SAS Software”. Appendix A lists the code of the macro and a detailed technical description. During extensive user testing at Merck Research Labs, this macro went through a series of improvements. Several SAS programmers with different levels of SAS skills used it and proposed changes and enhancements. Its present shape reflects their experience.

When we started mass production of MS Word ready SAS outputs (tables and graphs), we learned one lesson: a large number of outputs is difficult to manage manually! If you want to insert 100 tables (some of them in portrait, others in landscape) and 50 graphs into one document, or if you simply want to print them, be prepared for a long and tedious task! Thus, the next natural step for us was to automate this common file manipulation. Again, we tried a couple of methods and decided that using MS Word macro capabilities was the most feasible. MS Word macros can be extremely useful to a SAS programmer. They can range from very simple to very complicated. A few lines of code can accomplish simple tasks, such as printing all files in a certain directory with one mouse click or automatic formatting of standard SAS outputs upon opening in MS Word. The latter operation could amount to applying courier font when *.lis file is opened, or you can get more fancy and try to automatically determine (and set) an appropriate font size and page orientation. Once you start thinking along these lines, you will probably find many ways to cut down on manual labor and make your work more efficient using macro tricks. Some ideas that worked for us are presented in later sections. They include automation of the file inserting and file comparison processes.

Creating RTF files in SAS Software

The following example illustrates the concept behind RTF programming in SAS software. We start with a simple table produced in a traditional way using a PUT statement.

Example 1

```
      /* Create a simple data set to use with the table */
   data test;
   input name trt n mean std median ;
   cards;
   1  1      69    32.8841    12.5047      31
   1  2      67    34.6119    13.7631      33
   1  3     136    33.7353    13.1195      31
   2  1      69    67.2464     4.0959      66
   2  2      67    67.2090     3.7961      67
   2  3     136    67.2279     3.9365      66
   ;

   proc sort; by name trt;

   proc format;
   value namefmt 1 = 'Age (years)'
                 2 = 'Height (in)';
   value trtfmt  1 = 'Control                '
                 2 = 'Experimental Test Drug'
                 3 = 'All                    ';

   %let title1 = Simple Summary Statistics;

   data _null_;
   file 'u:\test.txt';
   set test end=eof ;
     by name trt;

     titel1 = "&title1";
      line1 =   repeat("_", 80);
        /* center titles on the page */
     t1 = (90 - length(titel1))/2;
        /*define starting location for each column */
     c1 = 1;
     c2 = 17;
     c3 = 43;
     c4 = 51;
     c5 = 70;

   if _n_=1 then do;
      put    @t1 titel1;
      put /  @c1 line1;
      put /  @c1 'Variable' @c2 'Treatment' @c3+2 'N' @c4+2 'Mean ± SD'
             @c5 'Median';
      put /  @c1 line1;
   end;

   if last.name then put ;
   if first.name then put @c1 name namefmt. @;
    put @c2 trt  trtfmt.
        @c3  N 3.
        @c4 mean 5.2 + 1 '±' +1 std 5.2
        @c5 median 5. ;
   if last.name then put  @c1 line1;
   run;
```

Figure 1 shows the table produced by this code. To preserve a proper alignment of the columns, when inserting in a MS Word document, the font used for this table has to be fixed size, for example, courier.

Figure 1

Simple Summary Statistics

Variable	Treatment	N	Mean ± SD	Median
Age (years)	Control	69	32.88 ± 12.50	31
	Experimental Test Drug	67	34.61 ± 13.76	33
	All	136	33.74 ± 13.12	31
Height (in)	Control	69	67.25 ± 4.10	66
	Experimental Test Drug	67	67.21 ± 3.80	67
	All	136	67.23 ± 3.94	66

We will now demonstrate how such a table can be produced using the macro %RTF. Although the code looks much more complicated, using this macro allows for a great flexibility in table formatting (as we discuss later).

```
      /* We assume that macro rtf has been compiled. */

0001  data _null_;
0002  file 'c:\test.rtf';
0003  set test end=eof ;
0004    by name trt;
0005
0006  if _n_=1 then do;
0007    %rtf(0); * initialize table;
0008    %rtf(1, b=0); * define a row with 1 column, no borders;
0009      put &bc "&title1" &e;
0010     &par;  * insert empty line;
0011    %rtf(5, 2 3 1 2 1, b=1, h=a, v=a, s=120 120);
0012     * define a row with 5 columns, with borders (vertical and
0013     horizontal);
0014     put &bc 'Variable' &cc 'Treatment' &cc 'N' &cc 'Mean ± SD'
0015         &cc  'Median' &e;
0016  end;
0017
0018  if first.name then do;
0019    %rtf(5, 2 3 1 2 1, b=1, v=a); * horizontal border turned off;
0020  end;
0021
0022  if last.name then do;
0023    %rtf(5, 2 3 1 2 1, b=1, v=a);
0024       put &nl;
0025    %rtf(5, 2 3 1 2 1, b=1, v=a, h=a); * horizontal border turned on;
0026  end;
0027
0028  if first.name then put &bl name namefmt. @;
0029    else put &bl @;
0030
0031   put &cl trt  trtfmt.
0032       &cc  N 3.
0033       &cc mean 5.2 +1 '±' +1 std 5.2
0034       &cc median 5. &e;
0035
0036  if eof then do;
0037     %rtf(100);  * close table;
0038  end;
0039  run;
```

Figure 2 shows that the output from this program, when opened in an ASCII editor, looks formidable.

Figure 2: The RTF File Code

```
{\rtf1\ansi \deff0\deflang1033{\fonttbl{\f0\froman Times New Roman;}
{\f1\froman\fcharset2\fprq2 Symbol;}}
\trowd\trgaph108\trleft0\trqc
\cellx9000\pard
\pard
\intbl\qc\sb30\sa30 Table 2\cell\intbl\row\pard
\intbl\qc\sb30\sa30 Simple Summary Statistics\cell\intbl\row\pard
\pard\par
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx2000\clbrdrb\brdrhair\clbrdrr
\brdrhair\cellx5000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx6000\clbrdrb
\brdrhair\clbrdrr\brdrhair\cellx8000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx9000\pard
\intbl\qc\sb120\sa120 Variable\cell\pard\intbl\qc\sb120\sa120 Treatment
\cell\pard\intbl\qc\sb120\sa120 N\cell\pard\intbl\qc\sb120\sa120 Mean ± SD
\cell\pard\intbl\qc\sb120\sa120 Median\cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrr\brdrhair\cellx2000\clbrdrr\brdrhair\cellx5000\clbrdrr\brdrhair\cellx6000
\clbrdrr\brdrhair\cellx8000\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 Age (years)\cell\pard\intbl\ql\sb30\sa30 Control
\cell\pard\intbl\qc\sb30\sa30  69\cell\pard\intbl\qc\sb30\sa30 32.88 ± 12.50
\cell\pard\intbl\qc\sb30\sa30    31\cell\intbl\row\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\ql\sb30\sa30 Experimental Test Drug\cell
\pard\intbl\qc\sb30\sa30  67\cell\pard\intbl\qc\sb30\sa30 34.61 ± 13.76\cell
\pard\intbl\qc\sb30\sa30    33\cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrr\brdrhair\cellx2000\clbrdrr\brdrhair\cellx5000\clbrdrr\brdrhair\cellx6000
\clbrdrr\brdrhair\cellx8000\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30
\cell\pard\intbl\qc\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30 \cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx2000\clbrdrb\brdrhair\clbrdrr\brdrhair
\cellx5000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx6000\clbrdrb\brdrhair\clbrdrr
\brdrhair\cellx8000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\ql\sb30\sa30 All
\cell\pard\intbl\qc\sb30\sa30 136\cell\pard\intbl\qc\sb30\sa30 33.74 ± 13.12
\cell\pard\intbl\qc\sb30\sa30    31\cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrr\brdrhair\cellx2000\clbrdrr\brdrhair\cellx5000\clbrdrr\brdrhair\cellx6000
\clbrdrr\brdrhair\cellx8000\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 Height (in)\cell\pard\intbl\ql\sb30\sa30 Control
\cell\pard\intbl\qc\sb30\sa30  69\cell\pard\intbl\qc\sb30\sa30 67.25 ±  4.10\cell
```

\pard\intbl\qc\sb30\sa30 66\cell\intbl\row\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\ql\sb30\sa30 Experimental Test Drug\cell\pard
\intbl\qc\sb30\sa30 67\cell\pard\intbl\qc\sb30\sa30 67.21 ± 3.80\cell\pard\intbl
\qc\sb30\sa30 67\cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrr\brdrhair\cellx2000\clbrdrr\brdrhair\cellx5000\clbrdrr\brdrhair\cellx6000
\clbrdrr\brdrhair\cellx8000\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30
\cell\pard\intbl\qc\sb30\sa30 \cell\pard\intbl\qc\sb30\sa30 \cell\intbl\row\pard
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrs\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrs\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx2000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx5000
\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx6000\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx8000
\clbrdrb\brdrhair\clbrdrr\brdrhair\cellx9000\pard
\intbl\ql\sb30\sa30 \cell\pard\intbl\ql\sb30\sa30 All
\cell\pard\intbl\qc\sb30\sa30 136\cell\pard\intbl\qc\sb30\sa30 67.23 ± 3.94
\cell\pard\intbl\qc\sb30\sa30 66\cell\intbl\row\pard
\pard\par }

The same output opened in MS Word is shown in Figure 3.

Figure 3

Simple Summary Statistics

Variable	Treatment	N	Mean ± SD	Median
Age (years)	Control	69	32.88 ± 12.50	31
	Experimental Test Drug	67	34.61 ± 13.76	33
	All	136	33.74 ± 13.12	31
Height (in)	Control	69	67.25 ± 4.10	66
	Experimental Test Drug	67	67.21 ± 3.80	67
	All	136	67.23 ± 3.94	66

We will shortly explain the meaning of the macro variables in the above program. Detailed technical specifications for macro %RTF are in Appendix A. The syntax is very similar to the regular PUT statement, but there are, however, some important differences and rules that must be followed. Each table starts with the table declaration (or initialization) that has the form %RTF(0) (see line 0007). After that, each row is written line-by-line and the format of each row has to be defined. The first positional parameter defines how many columns the row will contain. In line 0008, we define a row with just one column. This column will be centered on the page. If the row has two or more columns, we need to specify the relative widths of the columns. For example, in line 0011, we defined a row of five columns with relative widths 2,3,1,2, and 1. That means that the whole width of the page (minus margins, which are set by default to 1.25" on both sides) is divided proportionally among five columns as follows. The first column is twice as wide as the third one, the second column is three times as wide as the third one, and so on.

The parameter B defines outer borders. We set B=0 if we want a row with no borders (for example, title row) and B=1 if outer borders are desired. The parameters V and H specify inner borders. If H=A, as in line 0011, then the row will have a horizontal border at the bottom of each cell; we could set H=1 3 to get a horizontal border at the bottom of the first and the third cell. The vertical borders are defined similarly. The default line style for the borders is single. Appendix A shows how to specify a double-line border. We could also request borders at the top of the cells.

There is one additional parameter, S, in line 0011. This parameter takes two integers and specifies how much space we want between the text and the top (first integer) and bottom (second integer) of the cell. In most cases, default spacing works just fine. For aesthetic reasons, we decided to space table headers wider than the body of the table.

The row definition stays in effect for each line of output until macro %RTF is invoked again. In our example, we wanted to separate the header from the table and Age from Height using horizontal borders. For that reason, we invoke macro %RTF when first.name condition is true (to turn bottom borders off using a default value H=0 in line 0019) and again when the last.name condition is true (to turn bottom borders on using H=A in line 0025).

After we defined the row appearance, we place data in the cells using a familiar PUT statement. A very important difference from the regular PUT statement is that we use the macro variables &BL , &BC, &BR , &CL , &CC, and &CR at the beginning of each cell. The variables &BL, &BC, and &BR are used only for the **first** cell in the row; the variables &CL, &CC, and &CR are used for the remaining cells (if there are more than one). These variables, besides initializing the cell, define the justification of a text within a cell. The variables &BL, &BC, and &BR request left-justified, centered, and right-justified placement, respectively. The variables &CL, &CC, and &CR are defined similarly. In our example, the text is left-justified in the first two columns (lines 28-31) and centered in the remaining ones (lines 32-34). Sometimes you may prefer a decimal alignment to improve on the appearance of the numbers. The variables &DL1, &DL2, and so on, discussed in the Appendix, serve this purpose. The &B* or &C* variables, as appropriate, ***must*** be used for every cell in a row. If the number of &B*/&C* variables does not match the current number of columns, an error occurs and MS Word crashes upon opening of the document. In line 0029, we demonstrate how to an create empty cell without violating this requirement. Another important feature is that every row ends with a keyword &E.

The variable &PAR inserts a paragraph mark (line break). We used it in line 0010 to separate the borderless row with the title from the table proper. The variable &NL in line 0024 creates a row with all cells empty. An alternative way to create such a row would be

```
put &bc &bl &bl &bl &bl &e;
```

The final point is that we use a mandatory %RTF(100) statement at the end of our table (after writing the last line).

We describe here the most basic features of the RTF programming in SAS. There are many more options available. For example, you may use exotic symbols, format text as italic, bold or underline, subscript or superscript, or vary font size. The wrapping of text within a cell happens automatically — you will never have to worry about overflowing long character variables! The pages can have portrait or landscape orientation and the orientation can vary within the same output.

It should be pointed out that users can easily expand our macro to suit their needs. For example, we defined only two basic fonts (times new roman and symbol) because these are the only fonts that we use for our outputs. More fonts can be added easily to accommodate different needs. Users familiar with the RTF language can also use a PUT statement with direct RTF statements (for example, RTF keywords \b and \b0 turn bold formatting on and off).

Please note that it is not recommended to edit RTF files inside MS Word. This is because MS Word encodes an RTF file in a way different from our macro, and unexpected results may occur when you save your changes. Before you attempt editing, you should save an RTF document as an MS Word document.

The only noticeable drawback of RTF programming within SAS Software is that some errors in the RTF file may cause MS Word to crash, and repeated crashes of MS Word in the same session may eventually crash the system. It is almost impossible for beginners to avoid making errors while writing RTF files, so save your work before opening a newly created RTF output! The admiration in the eyes of customers, when they see your impressive outputs, is well worth these growing pains.

Automating Manipulation of SAS Outputs

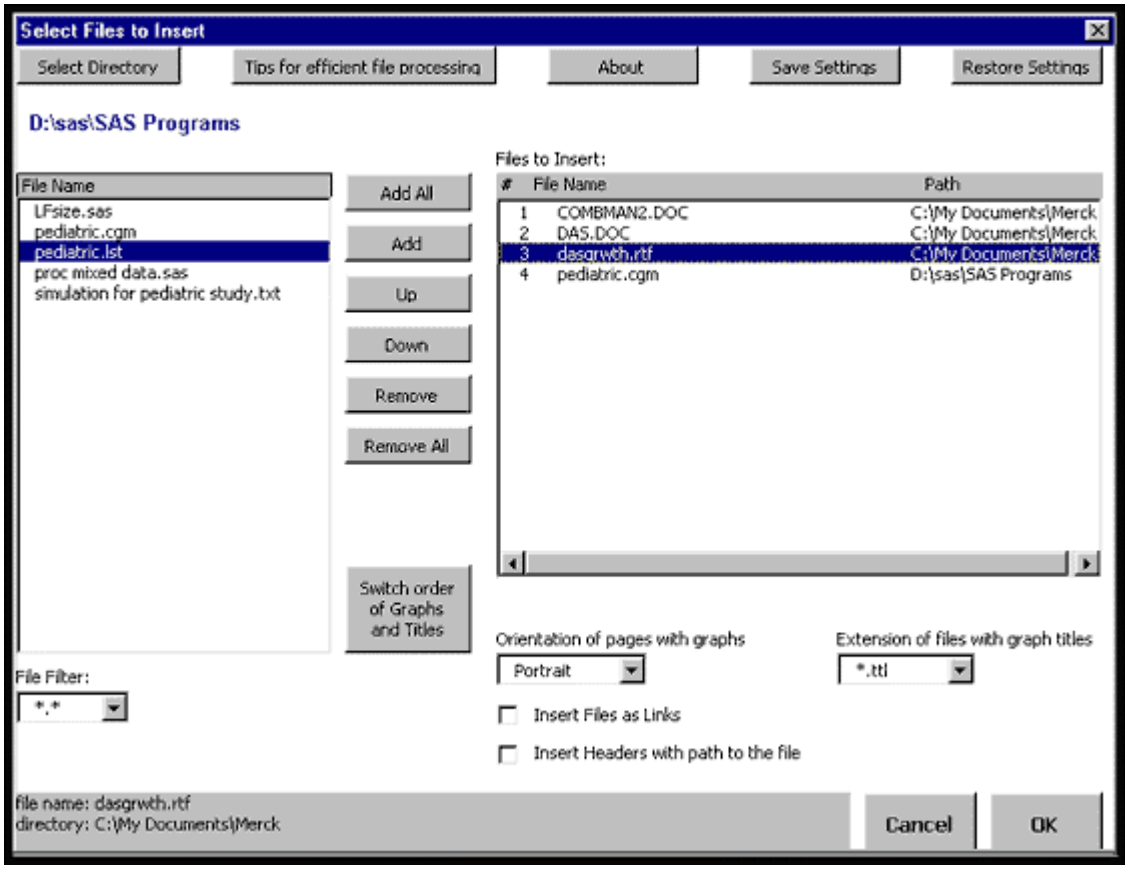
When complex reports are generated, a great number of SAS outputs (graphs, tables, listings, analysis outputs, and so on) needs to be word processed. This is usually a tedious task, especially if a final document consists of different types of SAS outputs. We created a MS Word macro InsertAllFiles that greatly reduces manual effort required to insert multiple files into one document. This macro was created to work with Word documents, *.RTF files, and graphics generated in CGM format. However, it can be easily modified to work with any other types of documents. The installation instructions for this macro are given in Appendix B.

The basic idea is to create a list of files to be manipulated and to insert files from this list into a new MS Word document. The list is created by point-and-click in the displayed dialog box (see Figure 4). Before the file is inserted, its orientation is determined. If necessary, a section break is created in the new document and orientation of the new section is set to match the orientation of the currently processed file. Graphs (that is, *.CGM files) are inserted as pictures in an orientation of the user's choice. All this is handled automatically.

Figure 4 presents the dialog box that is displayed by the InsertAllFiles macro. Objects on the left side let us display files in a specific directory (in an alphabetical order) and are used in a way similar to Windows' File Manager. Combo-box File Filter is used to filter file extensions. The list is created and displayed in a box on the right side using push-buttons Add All, Add, Up, Down, Remove, Remove All; These buttons also let us rearrange the order of the items on the list. If the alphabetical order of the files matches the order desired in the document, we could create a list with a single click on the button Add All.

The remaining buttons and boxes are used for advanced options. The user may request that a header with a full path and filename is inserted before the file itself. Files can be also inserted as LINKS (readers not familiar with this useful feature are referred to Microsoft Word Help). This comes in handy if the same set of programs generating outputs is run more than once. A master file with all the reports needs to be created only once; when the next set of reports is generated, one needs only to update links in the master document to replace the outputs with the new ones. Note that if the LINKS box is checked, graphs are inserted into document as links with an option that will save a copy in the document. This is done so that the graphs do not disappear when the link is broken.

Figure 4: A Snapshot of the InsertAllFiles Dialog Window



One of the limitations of the MS Word graphic editor is that it corrupts *.CGM pictures containing rotated text. For that reason, it is sometimes convenient to store titles of the graphs produced in SAS in a separate document to allow for their editing (we usually employ the macro %RTF to generate an RTF file with the graph title in the same program as the graph itself). It is convenient to store the graph and its title in the files with the same name but with different extensions, for example, GRAPH1.CGM and GRAPH1.TTl. If the user specifies the extension of files containing graph titles in a combo-box, macro InsertAllFiles recognizes the files with graph titles and does not create page breaks after these files, so that graph and its title are on the same page. The Switch Order of Graphs and Titles push button automatically rearranges the list so that graph titles precede graphs (*.CGM precedes *.TTl when listed alphabetically). This button switches the order of neighboring files if

1. both files have the same name
2. the extension of the first file matches the extension specified in the box
3. the extension of the second file is CGM.

Once MS Word finishes inserting files from the list, the control is returned to MS Word. At this point, the document can be saved, printed, or edited.

Using Save Settings and Restore Settings push buttons, the user can save the information about the order and format of inserted documents in an external file. This way, it is possible to prepare the report ahead of time and rerun it with one push of a button. The user can perform part of the work at one time, save settings needed to reproduce the task, and continue work or modify these settings latter.

It should be noted that in some operating environments memory limitations cause MS Word to lock if you try to insert too many files. If this happens, the message “Error: File could not be found” appears. It is best to limit the size of the list to 40–50 files. By experimenting, you will quickly learn how many files can be safely processed at a time.

Developing Other Time Saving Solutions with Microsoft Office

There are many ways in which SAS users can draw on the power of Microsoft Office. In this section we present some ideas we found useful, leaving the programming details to the creativity of the reader.

Sometimes users need to verify that a the new graph matches a graph generated in the past. The eye-balling technique may not reveal all the subtle differences so some more sensitive methods of comparisons are needed. While MS Word allows for relatively easy comparison of documents, it cannot handle the comparison of graphic objects. There may be specialized graphic software that is capable of comparing graphic objects, but it is not widely available. We use a simple trick in PowerPoint to accomplish this task. The reason for using PowerPoint is that its graphic editor can recolor imported graphic objects. We insert both the old and the new graphs in a blank PowerPoint slide (resizing them if necessary)) and recolor the new one in red while the old one is left black. After we align the graphs, the top graph should cover the bottom one, and we should see only one color if the graphs are identical. The next slide inverts the layer so that the graph that was on the bottom is now on the top. If the two graphs are identical, then the first slide will be monochromatic black and the second one will be monochromatic red. It is fairly easy to spot the differences because they will appear as a two-colored slide.

Of course, you could superimpose two sheets of paper and view them against a source of light to achieve the same trick. However, we can use the described algorithm in a macro that can process a large number of graphs and greatly simplify the work. Office 97 supports PowerPoint Basic, and such a macro program can be implemented directly in PowerPoint. Older versions of PowerPoint do not support macros. Instead, we can write a WordBasic macro and use a SendKeys command to perform tasks in PowerPoint. We find it useful to employ two kinds of macros: the individual comparisons and the list comparisons. The first type compares a graph selected in a MS Word document with the one stored in an external file. The second macro creates the list that consists of the pairs (old graph/new graph) and performs the comparison on each pair. If the list consists of n pairs, the macro will create a PowerPoint set of $2n$ slides (2 slides for each pair) with superimposed pairs of graphs. This set of slides can be visually scanned for the two-colored graphs.

In a similar way, the list of paired MS Word documents or RTF files can be processed and comparisons of documents performed on each pair. In that case, the automation can be carried one step further because WordBasic has the capability to recognize the identical documents, and it is possible to create a macro that deletes from the list all the pairs that are identical and leaves only the pairs with files that differ.

There are probably as many uses for the MS Word macros in conjunction with SAS programming as there are SAS programmers. You can write a macro to search all the files on a disk for the occurrence of a specified string. You can scan saved SAS logs for error messages and warnings. If you still work in Windows for Workgroups, you may want to create a macro to list the total size of the specified directory. We tried to describe here a few applications that saved us a great deal of manual labor. It should be noted that these applications could be developed and improved with VisualBasic (or another application-building tool), which offers more advanced capabilities. However, WordBasic seems to suffice in many situations and is widely available to most PC users without having to invest in another software. Our hope is that we will encourage fellow SAS programmers to try these techniques. The time you spend learning the necessary tools would be well spent — it will save you months of boring work and your company thousands of dollars.

Acknowledgments

The authors would like to thank Mr. Jianmin Long of Schering Plough, Inc., for developing the macro %RTF during his tenure at Merck & Co., and all reviewers for helpful comments and suggestions.

References

Reporting from the Field: SAS Software Experts Present Real-World Report-Writing Applications, SAS Institute Inc., 1994

Microsoft, MS-DOS, Windows, OS/2®, and Apple Macintosh Applications: Rich Text Format (RTF) Specification, Product Support Services Application Note, 1994

Microsoft Word Developer's Kit, Microsoft Press, 1995

Appendix A – Macro %RTF Code and Specifications

```
%macro rtf(n, m, s=30 30, b=1, r=2, o=p, h=0, v=0, hline=b, line=s,
last=0, w=);
%* author: Jianmin Long;
%*  Copyright Merck & Co., 1996;

%if &n=0 %then %do;
    %* initialize rtf document;
    put "{\rtf1\ansi \deff0\deflang1033"@;
    %* define fonts;
    put "{\fonttbl{\f0\froman Times New Roman;}";
    put "{\f1\froman\fcharset2\fprq2 Symbol;}";
    put "{\f2\froman\fcharset2\fprq2 Arial;}" @;
    %* add other fonts using f3, f3 etc...;
    put "}";
    %* define page orientation;
    %if &o=l %then %do;
        put "\paperw15840\paperh12240\landscape ";
    %end;
    %* define shortcuts for cell formatting;
    %global e par newpage;
    %let e='\cell\intbl\row\pard ';
    %* define keyword for new page;
    %let newpage=%str(put '\page \par \pard '; );
    %* define keyword for line break;
    %let par=%str(put '\pard\par '; );
%end;
   %* define closing for a table;
%else %if &n > 31 %then %do;
    put "\pard\par }";
%end;
%else %do;
  %global dbline;
  %* define keyword for double line to use in table formatting;
  %let dbline=%str(put '\sl-20 \slmult0 \par \pard '; );
  %* break complex parameters r,v and h into "words";
  %do i=1 %to %length(&r);
      %if %length(%scan(&r, &i)) %then %do; %let num_of_r=&i; %end;
    %end;
    %do i=1 %to %length(&v);
```

```
      %if %length(%scan(&v, &i)) %then %do; %let num_of_v=&i; %end;
   %end;
    %do i=1 %to %length(&h);
      %if %length(%scan(&h, &i)) %then %do; %let num_of_h=&i; %end;
    %end;
    %* if all vertical borders are requested, calculate how many
columns
       in the table and define borders coding;
    %if &v=a %then %do;
         %do i=1 %to &n;
             %let vl&i=\clbrdrr\brdrhair; %end;
    %end;
    %* define vertical borders coding so only requested cell borders
show;
    %else %do;
       %do i=1 %to &n; %let vl&i= ; %end;
       %if &v ne 0 %then %do;
          %do i=1 %to &num_of_v;
              %let ii=%scan(&v, &i);
              %let vl&ii=\clbrdrr\brdrhair; %end;
       %end;
    %end;

    %* if this is last row in the table, make bottom border double
line;
    %if &last=1 %then %do;
      %do i=1 %to &n;
      %let hl&i=\clbrdr&hline\brdrdb; %end;
    %end;
    %* for other rows, define horizontal borders coding so only
requested cell borders show;
    %else %do;
        %if &h=a %then %do;
            %do i=1 %to &n;
            %let hl&i=\clbrdr&hline\brdrhair; %end;
        %end;
        %else %do;
          %do i=1 %to &n; %let hl&i= ; %end;
          %if &h ne 0 %then %do;
             %do i=1 %to &num_of_h;
               %let ii=%scan(&h, &i);
               %let hl&ii=\clbrdr&hline\brdrhair; %end;
           %end;
        %end;
     %end;

    %* define the width of the table in pixels ;
    %* default width is 9000 pixels for portrait and 12240 for
landscape;
    %if %length(&w)=0 %then %do;
        %if &o=p %then %do;
            %let tw=9000;
        %end;
        %else %if &o=l %then %do;
            %let tw=12240;
        %end;
    %end;
    %* for requested table width in inches, convert width to pixels;
    %else  %do;
        %let tw=%eval(1440*&w/(10**(%length(&w) - 1));
    %end;
```

```
    %* determine if the position of all decimal alignments is the
same (r=1);
    %if &num_of_r = 1 %then %do;
    %* define the tag for the position of the decimal point ;
        %if &r > 32 %then %do;
          %* decimal point of 1st cell will be just left of the cell
middle;
            %let r1=%eval(32 - &r);
        %end;
        %else %do;
            %let r1=&r;
           %* decimal point of 1st cell will be at r/64 of cell
width;
        %end;
      %do i=2 %to &n; %let r&i=&r1; %end;
       %* decimal points for other cells fsame as 1st cell;
    %end;
    %else %do;
       %do i=1 %to &n;
          %let r&i=%scan(&r, &i); %* break parameter r into words;
          %if &&r&i > 32 %then %do;
          %* decimal point of ith cell will be just left of the cell
middle;
             %let r&i=%eval(32 - &&r&i); %end;
       %end;
    %end;
    %global nl bl br bc cl cr cc nc next;
    %* break complex parameter s into distance from top (sb) and from
bottom (sa);
    %let sb=%scan(&s, 1);
    %let sa=%scan(&s, 2);

    %* code tags for 1st cell in a row under left-, center-, and
right-justification;
    %let bl="\intbl\ql\sb&sb\sa&sa ";
    %let bc="\intbl\qc\sb&sb\sa&sa ";
    %let br="\intbl\qr\sb&sb\sa&sa ";
    %* code tags for other cells in a row under left-, center-, and
right-justification;
    %let cl="\cell\pard\intbl\ql\sb&sb\sa&sa ";
    %let cc="\cell\pard\intbl\qc\sb&sb\sa&sa ";
    %let cr="\cell\pard\intbl\qr\sb&sb\sa&sa ";

    %let next="\cell\pard";
    %let nc="\cell\pard";


    %if &n=1 %then %do;
       %global dl1;
        %let nl=%str( "\intbl\cell\intbl\row\pard"; );
    %end;
    %else %do;
        %do temp=1 %to &n;
              %global dl&temp;
        %end;
        %let nl=%str( '\intbl\cell' @; );
        %do i=1 %to %eval(&n-1);
          %let nl=%str( &nl put '\cell' @; );
        %end;
        %let nl=%str( &nl put '\intbl\row\pard'; );
```

```
    %end;
    put "\trowd\trgaph108\trleft0\trqc";
    %* code and print tags if outer borders of the table are
requested;
    %if &b = 1 %then %do;
      put "\trbrdrt\brdr&line\brdrw15\trbrdrl\brdrs\brdrw15";
      put "\trbrdrb\brdr&line\brdrw15\trbrdrr\brdrs\brdrw15"; %end;

    %if &n = 1 %then %do;
        %* code and print tags if the table has only one column;
        put "&hl1&vl1\cellx&tw\pard";
        %* code decimal alignment tags for the table with only one
column;
        %let dt1=%eval((32+(&r1))*&tw/64 - 108);
    %end;
    %else %do;
        %* code and print tags if the table has more than one column;
      %let t=0;
      %if %length(&m) %then %do;
          %do i=1 %to &n;
              %let dd&i=%scan(&m, &i);
              %let t=%eval(&t+&&dd&i);
              %let cw&i =&t; %end;
          %do i=1 %to &n;
              %let w&i=%eval(&&cw&i*&tw/&t);
              %let dt&i=%eval((32+(&&r&i))*&&dd&i*&tw/(64*&t) - 108);
              put "&&hl&i&&vl&i\cellx&&w&i" @; %end;
      %end;
      %else %do;
         %do i=1 %to &n;
             %let w&i=%eval(&i*&tw/&n);
             %let dt&i=%eval(&w1*(32+(&&r&i))/64 - 108);
             put "&&hl&i&&vl&i\cellx&&w&i" @; %end;
    %end;
    %end;
    put "\pard";
    %* code tags for decimal alignment;
    %let dl1="\intbl\sb&sb\sa&sa\tqdec\tx&dt1\tx%eval(&dt1+80)\tab ";
    %do i=2 %to &n;
%let
dl&i="\cell\pard\intbl\sb&sb\sa&sa\tqdec\tx&&dt&i\tx%eval(&&dt&i+80)\
tab ";
    %end;
%end;
%mend rtf;
```

Figure 5: Explanation of the Macro Parameters of the %RTF Macro (default values in parentheses)

Parameter	Type	Description
N	integer	Number of columns in the table. N=0 is used to initiate the table, N=100 is used to close the table.
M	string of integers	Relative width of columns. If used, the number of elements in the string M must be equal to the macro parameter N. If m is not specified, all columns have equal width. (optional)
S (=30 30)	pair of integers	Space (in twips) between the text and the top and the bottom of the cell, respectively.
B (=1)	0 or 1	If B=0, table has no outer border. If B=1, table has an outer border.
R (=2)	string of integers	Defines the position of a decimal point in cells containing numbers. Used if decimal alignment is requested (see global macro variables &DL1, DL2, and so on below). The number of elements in the string defining R must be either 1 or must be equal to the number of columns (N). If R has one element, this value is used for all columns. If the element or the string R is less than 32, the decimal point is placed at (R/64)*(width of the cell) from the right border of the cell (for example, if R=16, then the decimal point is placed at the three-quarters of the cell width). If R>32, then the decimal point is placed just to the left from the middle of the cell.
O (=P)	P or L	O=P requests a portrait orientation for the page; O=L requests a landscape orientation.
H (=0)	0, A, or a string of integers	Specifies which cells have horizontal borders. H=A requests that all cells have horizontal borders. H=1 3 5 specifies that 1st, 3rd, and 5th cells have horizontal borders. H=0 specifies that no cell has a horizontal border. Placement of the border (top or bottom) is determined by the value of a macro parameter hline
V (=0)	0, A, or a string of integers	Specifies which cells have right borders. V=A requests that all cells have right borders. V=1 3 5 specifies that 1st, 3rd , and 5th cells have right borders. If V=0, then no cell has a right border.
HLINE (=B)	B or T	Specifies horizontal border placement, top (T) or bottom (B).
LINE (=S)	S or DB	Defines the line style for outer horizontal border, single (S) or double (DB).
LAST (=0)	0 or 1	LAST=1 is used for the last row of the table to enable the double bottom border.
W	integer	Specifies the width of the table in inches; if decimal number is needed, omit the decimal point (for example, W=5 means 5” and W=55 means 5.5”). Specified width must be less than 10 inches (W=100 means 1”, W=110 means 1.1”)

Figure 6: Explanation of the Global Macro Variables Defined by the Macro %RTF

&BL, &BC, &BR	Used with a PUT statement to indicate the beginning of the first column in a row. Mandatory, unless a &DL1variable is used instead. The last character defines the justification of the text in the first cell (R=right, L=left, C=center)
&BL, &BC, &BR	Used with a PUT statement to indicate the beginning of each column in a row except the first one. Mandatory for each column, unless &DL* is used instead, where * stands for explicit column number. The last character defines the justification of the text in the cell (R=right, L=left, C=Center).
&DL1, &DL2, and so on	Used to define a decimal alignment within a column. If used instead of &BL, &BC, &BR, &CL, &CC or &CR, the number ending &DL* must match the column number. The position of the decimal point is defined by a macro variable R.
&E	Indicates the end of a row. Mandatory.
&PAR	Inserts a paragraph mark. Used to separate two tables from each other (for example, if you define the first table with B=0 option for titles and the second table with B=1). Note: Do not use a "PUT" statement with this variable, because put is a part of its definition.
&NEWPAGE	Inserts a page break. Note: Do not use a PUT statement with this variable, because PUT is a part of its definition.
&NL	Inserts an empty row in a table. Use this variable with a PUT statement.

Figure 7: Other Useful RTF Keywords

\B, \B0	turns bold font on (\B) and off (\B0)	PUT&BC "\B Title \B0" &E;
\I , \I0	turns italic font on (\I) and off (\I0)	PUT &BC "\I Title \I0" &*e*;
\UL, \UL0	turns underline on (\UL) and off (\UL0)	PUT &BC "\ul Title \ul0" &e;
\F0	turns times new roman font on (default)	See below
\F1	turns Symbol font on	"\f1 b\f0 -Agonist" will produce "β-Agonist"
\SUB, \SUPER	turns subscript and superscript on	"1 {\super st} and" will produce "1st and"
\NOSUPERSUB	turns both subscript and superscript off	"1\super st\nosupersub and" will produce "1st and"
\FSXX	used to define the font size (in 2*points)	\FS20 defines a font size 10 "\fs28 Note" will produce "Note"

Appendix B - About the Word Macro InsertAllFiles

The macro InsertAllFiles can be installed by downloading the proper Word template. There are two versions of the macro InsertAllFiles provided with this article. One is for Word 97; the other is for Word 6.0/95. Inst_97.dot is the file to download for use with Word 97; the text for this interface is shown in Appendix C.2. Inst_6.dot is the file to download for use with Word 6.0/95; the text for this interface is shown in Appendix D.2.

The version for Word 97 has all the functionality described in this article; however, the version for Word 6.0/95 is more limited. Two of the most important limitations of this code follow:

1. The user must hard-code all drive letters to be used in the macro code.

2. There is no provision for saving or restoring the settings.

From either template you can view the code of the macro without installing it on your system. If you choose to install the macro, you can run it by selecting its name from the Tools/Macro window and clicking on "Run". Alternatively, you can place a button invoking the macro on the toolbar. Refer to MS Help for instructions to do so.

Follow these steps to download the **Word 97** version of InsertAllFiles macro:

1. Download the Word 97 template from the following web location:
   http://www.sas.com/techsup/download/observations/obswww13/inst_97.dot

2. Open the template document in Word 97. Depending on your browser configuration, the template may open automatically when you download it; if not, you must save it on your hard drive and and open it manually.

3. Follow the instructions provided in the template to install the macro or to view the code. The code is also available in this article as Appendix C.1.

Follow these steps to download the **Word 6.0/95** version of InsertAllFiles macro:

1. Download the Word 6.0/95 template from the following web location:
   http://www.sas.com/techsup/download/observations/obswww13/inst_6.dot

2. Open the template document in Word 6.0/95. Depending on your browser configuration, the template may open automatically when you download it; if not, you must save it on your hard drive and and open it manually.

3. Follow the instructions provided in the template to install the macro or to view the code. The code is also available in this article as Appendix D.1.

Appendix C - Macro InsertAllFiles for Word 97

Appendix C.1 - Macro Code

```
’_____________________________________________________________
’ The code for the form frmInsertAllFiles.
’_____________________________________________________________

VERSION 5.00
Begin {C62A69F0-16DC-11CE-9E98-00AA00574A4F} frmInsertAllFiles
   Caption         =   "Select Files to Insert"
   ClientHeight    =   7620
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   10425
   OleObjectBlob   =   "frmInsertAllFiles.frx":0000
   StartUpPosition =   1  ’CenterOwner
End
Attribute VB_Name = "frmInsertAllFiles"
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False



’ copyright Iza Peszek, Merck & Co. Inc., 1998
’ All Rights Reserved
Dim currentExt As String
Dim currentTitExt As String
Dim currentPath As String
Dim currentFileName As String

Dim myDialog As Dialog

Dim msg As String
Dim sel_file As String, nextfile As String, prev_file As String
Dim potentialGraphFile As String, potentialGraphPath As String
Dim potentialTitleFile As String, potentialTitlePath As String
Dim pgfname As String, ptfname As String, ptfext As String
Dim tmp As String
Dim sName As String, SFN As String
Dim i As Integer, j As Integer, k As Integer
Dim selected_position As Integer

Dim ext(40)
Dim titext(40)
Dim allFiles() As String

Private Sub GetFilesAndDirs(myPath As String, ext As String)
’ function to populate
’ list box lstDirectoryContent with files in the selected directory
’ input parameters: directory name, extension of files to filter


’print path to current directory in label lblCurrentDirectory
lblCurrentDirectory.Caption = myPath

’populate files list
ChDir myPath
ReDim allFiles(0)
```

```
j = 0
If Right(myPath, 1) <> "\" Then myPath = myPath & "\"
currentFileName = Dir(ext, vbNormal)
Do Until currentFileName = ""
  'Ignore the current directory and the encompassing directory.
  If currentFileName <> "." And currentFileName <> ".." Then
  If (GetAttr(myPath & currentFileName) And vbNormal) < 16 Or _
    (GetAttr(myPath & currentFileName) And vbNormal) >= 32 Then
    ReDim Preserve allFiles(j)
    allFiles(j) = currentFileName
    j = j + 1
   End If
  End If
currentFileName = Dir
Loop
'sort array allFiles
For i = LBound(allFiles) To (UBound(allFiles) - 1)
  For j = (i + 1) To UBound(allFiles)
    If UCase(allFiles(i)) > UCase(allFiles(j)) Then
       tmp = allFiles(i)
       allFiles(i) = allFiles(j)
       allFiles(j) = tmp
    End If
  Next j
Next i
'populate listbox
lstDirectoryContent.List = allFiles
End Sub
Private Sub InsertFileIntoList(PositionFrom As Integer, PositionTo As
Integer)
'positionFrom, positionTo are file numbers starting with 1
   lstFilestoProcess.AddItem Str(PositionTo), PositionTo
   lstFilestoProcess.List(PositionTo, 1) =
lstDirectoryContent.List(PositionFrom)
   lstFilestoProcess.List(PositionTo, 2) = lblCurrentDirectory
    For k = (PositionTo) To (lstFilestoProcess.ListCount - 1)
      lstFilestoProcess.List(k, 0) = Str(k + 1)
    Next
End Sub

Private Sub cbSave_Click()
'allows user to save the settings of the current section
' ask for filename to save
sName = InputBox(prompt:="Enter a unique name for these settings", _
        Title:="Save Settings", _
        Default:="Mylist")
SFN = "C:\InsertAllFiles.txt"
'file name for PrivateProfileString file
'System.PrivateProfileString("C:\InsertAllFiles.txt", "MacroSettings", _
    "LastFile") = ActiveDocument.Fullname
End Sub

Private Sub cbRestore_Click()

End Sub

Private Sub cmbFileFilter_Change()
' after user changes filter for file extension,
' update file list
Call GetFilesAndDirs(lblCurrentDirectory.Caption, cmbFileFilter.Text)
End Sub
```

```
Private Sub cmbTitExt_Change()
’titext_ = cmbTitExt.Text
End Sub

Private Sub cmdAdd_Click()
’check if file is selected in lstDirectoryContent
’if no file selected then do nothing
If lstDirectoryContent.ListIndex < 0 Then Exit Sub


If lstFilestoProcess.ListIndex = -1 Then
’ if no file selected in lstListofFile, append file at the end
Call InsertFileIntoList(lstDirectoryContent.ListIndex,
lstFilestoProcess.ListCount)

Else
  ’ if file is selected in lstFilesToProcess, insert new file below it
  Call InsertFileIntoList(lstDirectoryContent.ListIndex,
lstFilestoProcess.ListIndex + 1)
End If
’set focus in lstDirectoryContent on next file
If lstDirectoryContent.ListIndex < lstDirectoryContent.ListCount - 1
Then
  lstDirectoryContent.ListIndex = lstDirectoryContent.ListIndex + 1
End If
If (lstFilestoProcess.ListIndex < lstFilestoProcess.ListCount - 1 _
      And lstFilestoProcess.ListIndex > -1) Then
  lstFilestoProcess.ListIndex = lstFilestoProcess.ListIndex + 1
End If

End Sub

Private Sub cmdAddAll_Click()

If lstFilestoProcess.ListIndex = -1 Then
   ’if no file selected in lstFilestoProcess, add all new files at the
end

   For i = 0 To (lstDirectoryContent.ListCount - 1)
    Call InsertFileIntoList(i, lstFilestoProcess.ListCount)
    Next
Else
   ’if file is selected in lstFilestoProcess, add all new files below it
   selected_position = lstFilestoProcess.ListIndex
   For i = 0 To (lstDirectoryContent.ListCount - 1)
      Call InsertFileIntoList(i, selected_position + 1)
      selected_position = selected_position + 1
   Next
End If
End Sub

Private Sub cmdCancel_Click()
Me.hide
End
End Sub

Private Sub cmdDown_Click()
’check if file is selected and that it is not the last

i = lstFilestoProcess.ListIndex
```

```
If i > -1 And i < (lstFilestoProcess.ListCount - 1) Then

'move 2nd column
   sel_file = lstFilestoProcess.List(i, 1)
   nextfile = lstFilestoProcess.List(i + 1, 1)
   lstFilestoProcess.List(i + 1, 1) = sel_file
   lstFilestoProcess.List(i, 1) = nextfile
'move 3rd column
   sel_file = lstFilestoProcess.List(i, 2)
   nextfile = lstFilestoProcess.List(i + 1, 2)
   lstFilestoProcess.List(i + 1, 2) = sel_file
   lstFilestoProcess.List(i, 2) = nextfile
'keep focus on previous file
lstFilestoProcess.ListIndex = i + 1

End If
End Sub

Private Sub cmdOK_Click()
'hide form
Me.hide
End Sub

Private Sub cmdRemove_Click()
'check if file is selected in lstFilestoprocess
'if no file selected then do nothing
If lstFilestoProcess.ListIndex < 0 Then Exit Sub

   lstFilestoProcess.RemoveItem lstFilestoProcess.ListIndex
   For i = (lstFilestoProcess.ListIndex) To _
       (lstFilestoProcess.ListCount - 1)
      lstFilestoProcess.List(i, 0) = Str(i + 1)
    Next

End Sub

Private Sub cmdRemoveAll_Click()
For i = 0 To lstFilestoProcess.ListCount - 1
  lstFilestoProcess.RemoveItem 0
Next
End Sub

Private Sub cmdRestore_Click()
'restore previously saved settings
' display a FileOpen dialog box
With Dialogs(wdDialogFileOpen)
  ans = .Display
  SFN = .Name
End With
' add path to file name
If Right(CurDir, 1) = "\" Then
  SFN = CurDir & SFN
Else
  SFN = CurDir & "\" & SFN
End If
If ans <> -1 Then
  'if user cancelled, do nothing
 Exit Sub
Else
    'if user selected file, retrieve the settings
```

```
    ’get list of files
    ’get size of the list
    k = System.PrivateProfileString(SFN, "FileList", "ListSize")
    ’clearlist
    lstFilestoProcess.Clear
    ’get list
    For i = 0 To (k - 1)
    lstFilestoProcess.AddItem Str(i + 1)
    lstFilestoProcess.List(i, 1) = _
      System.PrivateProfileString(SFN, "FileList", "f" & Str(i * 2))
    lstFilestoProcess.List(i, 2) = _
      System.PrivateProfileString(SFN, "FileList", "f" & Str(i * 2 + 1))

    Next i
    ’get current directory
     lblCurrentDirectory.Caption = _
       System.PrivateProfileString(SFN, "CurDir", "CurDir")

    ’get file filter
    cmbFileFilter.Text = System.PrivateProfileString(SFN, "Preferences",
"FileFilter")
    ’get whether Insert files as links
    cbLinkid.Value = System.PrivateProfileString(SFN, "Preferences",
"LinkId")
    ’get whether Insert header
    cbHeadid.Value = System.PrivateProfileString(SFN, "Preferences",
"HeaderId")
    ’get title extension filter
    cmbTitExt.Text = System.PrivateProfileString(SFN, "Preferences",
"TitExt")
    ’get graph orientation preference
    cmbGraphOrientation.ListIndex = System.PrivateProfileString(SFN,
"Preferences", "GO")

    ’populate directory content list
    If cmbFileFilter.Text = "" Then cmbFileFilter.Text = "*.*"
    Call GetFilesAndDirs(lblCurrentDirectory.Caption,
cmbFileFilter.Text)
End If

End Sub

Private Sub cmdSave_Click()
’save current settings
’ display a FileSave dialog box
’ to enable fileSaveAs dialog box, activate word and create a new
document
’ in case none is open
Application.Activate
Documents.Add
With Dialogs(wdDialogFileSaveAs)
  ans = .Display
  SFN = .Name
End With
’ add path to file name
If Right(CurDir, 1) = "\" Then
  SFN = CurDir & SFN
Else
  SFN = CurDir & "\" & SFN
End If
’close created file
```

```
Application.Activate
ActiveDocument.Close savechanges:=wdDoNotSaveChanges

If ans <> -1 Then
  ’if user cancelled, do nothing
 Exit Sub
Else
    ’if user selected file, write the settings

    ’write list of files
    ’write size of the list
    System.PrivateProfileString(SFN, "FileList", "ListSize") =
lstFilestoProcess.ListCount
    ’write list
    For i = 0 To (lstFilestoProcess.ListCount - 1)
      System.PrivateProfileString(SFN, "FileList", "f" & Str(i * 2)) = _
        lstFilestoProcess.List(i, 1)
      System.PrivateProfileString(SFN, "FileList", "f" & Str(i * 2 + 1))
= _
        lstFilestoProcess.List(i, 2)
    Next i
    ’write current directory
    System.PrivateProfileString(SFN, "CurDir", "CurDir") = _
      lblCurrentDirectory.Caption
    ’write file fiter
    System.PrivateProfileString(SFN, "Preferences", "FileFilter") = _
      cmbFileFilter.Value
    ’write whether Insert files as links
    System.PrivateProfileString(SFN, "Preferences", "LinkId") =
cbLinkid.Value
    ’write whether Insert header
    System.PrivateProfileString(SFN, "Preferences", "HeaderId") =
cbHeadid.Value
    ’write title extension filter
    System.PrivateProfileString(SFN, "Preferences", "TitExt") =
cmbTitExt.Value
    ’write graph orientation preference
    System.PrivateProfileString(SFN, "Preferences", "GO") =
cmbGraphOrientation.ListIndex

End If
End Sub

Private Sub cmdSelDir_Click()

’ open word Dialog file open
’ if user selected OK, display the path and populate lstDirectorycontent
 If Dialogs(wdDialogFileOpen).Display <> 0 Then
  lblCurrentDirectory.Caption = CurDir
  ’populate lstDirectoryContent
   Call GetFilesAndDirs(lblCurrentDirectory.Caption, cmbFileFilter.Text)
End If
End Sub
```

```
Private Sub cmdSwitchid_Click()

'check if user selected extension for graph titles
If ((Trim(cmbTitExt.Text) = "") Or (Trim(cmbTitExt.Text) = "*.*")) Then
  MsgBox ("You must specify extension of files with titles of graph")
Else
    ' display explanation
    msg = "Use this button only if" + Chr(13) + _
      "files with graph titles have the same name as graph files but
different extension"
    msg = msg + Chr(13) + "and"
    msg = msg + Chr(13) + "file titles immediately follow corresponding
graphs in the list"
    'if user cancelled, do nothing
    If MsgBox(msg) = 2 Then Exit Sub
    ' otherwise switch order of files and graphs
    For i = 1 To (lstFilestoProcess.ListCount - 1)
       potentialGraphFile = lstFilestoProcess.List(i - 1, 1)
       potentialGraphPath = lstFilestoProcess.List(i - 1, 2)
       potentialTitleFile = lstFilestoProcess.List(i, 1)
       potentialTitlePath = lstFilestoProcess.List(i, 2)
       'separate file name and file extension
       pgfname = Mid(potentialGraphFile, 1, InStr(potentialGraphFile,
".") - 1)
       ptfname = Mid(potentialTitleFile, 1, InStr(potentialTitleFile,
".") - 1)
       ptfext = Right$(Trim(potentialTitleFile), 3)

       If (UCase(ptfext) = UCase(Mid(cmbTitExt.Text, 3, 3))) And _
          (UCase(pgfname) = UCase(ptfname)) Then
         'if current file is a graph title file,
         ' and file name is the same as previous file,
         ' switch order with previous file
          lstFilestoProcess.List(i - 1, 1) = potentialTitleFile
          lstFilestoProcess.List(i - 1, 2) = potentialTitlePath
          lstFilestoProcess.List(i, 1) = potentialGraphFile
          lstFilestoProcess.List(i, 2) = potentialGraphPath
       End If
    Next
End If

End Sub

Private Sub cmdUp_Click()
'check if file is selected and that it is not the first


i = lstFilestoProcess.ListIndex
If i > 0 Then
'move 2nd column
   sel_file = lstFilestoProcess.List(i, 1)
   prevfile = lstFilestoProcess.List(i - 1, 1)
   lstFilestoProcess.List(i - 1, 1) = sel_file
   lstFilestoProcess.List(i, 1) = prevfile
'move 3rd column
   sel_file = lstFilestoProcess.List(i, 2)
   prevfile = lstFilestoProcess.List(i - 1, 2)
   lstFilestoProcess.List(i - 1, 2) = sel_file
   lstFilestoProcess.List(i, 2) = prevfile
'keep focus on previous file
lstFilestoProcess.ListIndex = i - 1
```

```
End If
End Sub

Private Sub CommandButton1_Click()
msg = "To improve the efficiency of file processing, you can:" + Chr(13)
+ Chr(13)
msg = msg + "1. Create files using filenames that follow intended sort
order, i.e., File1.rtf, File2.rf, File3.cgm, File4.rtf" + Chr(13) +
Chr(13)
msg = msg + "2. Store titles of graphs in rtf of text files using the
same name as corresponding graph but different extension" + Chr(13)
msg = msg + "   title extension following graph extension in
alphabetical order i.e., Sales.aaa (title) and Sales.cgm (graph)" +
Chr(13) + Chr(13)
msg = msg + "3. Store all files that you want to insert into one
document in the same folder" + Chr(13) + Chr(13)
msg = msg + "4. Store your list of file if you think you may need to
reuse it" + Chr(13) + Chr(13)
msg = msg + "5. If your files may need updating, insert them as links
and update links when need arises" + Chr(13) + Chr(13)
MsgBox prompt:=msg, Title:="Tips for efficient file processing",
buttons:=vbOKOnly
End Sub

Private Sub CommandButton2_Click()
'display About information
msg = "Macro InsertAllFiles v. 2.0" + Chr(13)
msg = msg + "Copyright Iza Peszek, Merck & Co., Inc., 1998" + Chr(13)
msg = msg + "All Rights Reserved"
MsgBox prompt:=msg, Title:="About InsertAllFiles", buttons:=vbOKOnly
End Sub

Private Sub lstDirectoryContent_Click()
  LblThisFileName.Caption = lstDirectoryContent.Text
End Sub

Private Sub lstDirectoryContent_DblClick(ByVal Cancel As
MSForms.ReturnBoolean)
Call cmdAdd_Click
End Sub


Private Sub lstFilesToProcess_Click()
'display file name and path in status bar
msg = "file name: " +
lstFilestoProcess.List(lstFilestoProcess.ListIndex, 1)
msg = msg + Chr(13)
msg = msg + "directory: " +
lstFilestoProcess.List(lstFilestoProcess.ListIndex, 2)
LblThisFileName.Caption = msg

End Sub

Private Sub lstFilestoProcess_DblClick(ByVal Cancel As
MSForms.ReturnBoolean)
Call cmdRemove_Click
End Sub
```

```
Private Sub UserForm_Initialize()


ext(0) = "*.*"
ext(1) = "*.doc"
ext(2) = "*.rtf"
ext(3) = "*.txt"
'list more extensions if you wish


' define extensions for files with graph titles
titext(0) = "*.tit"
titext(1) = "*.ttl"
titext(2) = "*.*" 'add your extensions
'list more title extensions if you wish

currentPath = CurDir
currentExt = "*.*"
currentTitExt = "*.ttl"

'display current path in label lblCurrentDirectory in the form
lblCurrentDirectory.Caption = currentPath

'populate form controls: list of drives, list of file extensions
' and list of extensions for graph titles with preset values
cmbFileFilter.List() = ext
cmbFileFilter.Text = currentExt
cmbTitExt.List() = titext
cmbTitExt.Text = currentTitExt

' populate directory content
Call GetFilesAndDirs(currentPath, currentExt)
' display options for Graph orientation
cmbGraphOrientation.ColumnCount = 2
cmbGraphOrientation.AddItem "Portrait"
cmbGraphOrientation.List(0, 1) = 0
cmbGraphOrientation.AddItem "Landscape"
cmbGraphOrientation.List(1, 1) = 1
cmbGraphOrientation.BoundColumn = 2
cmbGraphOrientation.Style = fmStyleDropDownList
cmbGraphOrientation.ListIndex = 0
'End With


End Sub

'_____________________________________________________________________
' The code for InsertAllFile macro
' (the part of macro that runs after user closes the form).
' Copyright Iza Peszek, Merck & Co. Inc., 1998
' All Rights Reserved.
'_____________________________________________________________________

Dim i As Integer, tmp As Integer, sizeOfList As Integer
Dim ContOrient As Integer, ContPW As Integer, ContPH As Integer
Dim NFOrient As Integer, NFPW As Integer, NFPH As Integer

Dim titleext As String, Fullname As String
Dim fileextension As String
Dim ContFile As Object
Dim prevFile_wasTitle As Boolean
```

```
Public Sub InsertAllFiles()
’display form
frmInsertAllFiles.Show

’ determine the extension of graph titles
If Len(frmInsertAllFiles.cmbTitExt.Text) < 3 Then
  titleext = "..."
Else
  titleext = LCase(Right$(frmInsertAllFiles.cmbTitExt.Text, 3))
End If
’open new file and assign a name so we can refer to it
Set ContFile = Application.Documents.Add

’insert files from the list

prevFile_wasTitle = False
’used to remember if previously inserted file was a title of a graph
 sizeOfList = frmInsertAllFiles.lstFilestoProcess.ListCount - 1

For i = 0 To sizeOfList

’display message in the status bar showing progress
StatusBar = "Processing file " & Str(i + 1) & " of " & Str(sizeOfList +
1)

  ’ create full names of files (with path)
  If Right$(frmInsertAllFiles.lstFilestoProcess.List(i, 2), 1) <> "\"
Then
     Fullname = frmInsertAllFiles.lstFilestoProcess.List(i, 2) _
        & "\" & frmInsertAllFiles.lstFilestoProcess.List(i, 1)
  Else
     Fullname = frmInsertAllFiles.lstFilestoProcess.List(i, 2) _
       & frmInsertAllFiles.lstFilestoProcess.List(i, 1)
  End If

’ determine the orientation of the last section of the container file
     ContFile.Activate
     ContOrient = ContFile.Sections.Last.PageSetup.Orientation
     ContPH = ContFile.Sections.Last.PageSetup.PageHeight
     ContPW = ContFile.Sections.Last.PageSetup.PageWidth

  ’ determine if file exist
   If Dir(Fullname) = "" Then

       ’if no such file exists, insert page break
       ’and the statement "file FullName was not found"
        ContFile.Activate
        If i > 0 Then Call Insert_PB_at_EOF
        Selection.EndKey Unit:=wdStory
        Selection.InsertAfter "file " & Fullname & " was not found"
        Selection.Collapse Direction:=wdCollapseEnd
   Else
     ’ if file exists
     ’ determine what kind of break is needed and insert break if needed
     ’ then insert file

      ’ determine file extension
      fileextension =
LCase(Right$(frmInsertAllFiles.lstFilestoProcess.List(i, 1), 3))

      Select Case fileextension
```

```
       Case "cgm", "tif", "jpg", "wmf", "bmp", "gif"
          ' check if previous file was a graph title
          ' if so, insert paragraph
          ' if not, check if last section has orientation specified for
graphs
          ' if so, insert page break
          ' if not, insert section break and apply appropriate
orientation
          ContFile.Activate
          If prevFile_wasTitle Then
             With Selection
                .EndKey Unit:=wdStory
                .InsertParagraphAfter
                .Collapse Direction:=wdCollapseEnd
             End With
          Else
              If ContOrient =
frmInsertAllFiles.cmbGraphOrientation.Value Then
                  If i > 0 Then Call Insert_PB_at_EOF 'insert page break
              Else
                tmp = frmInsertAllFiles.cmbGraphOrientation.Value
                 Call Insert_SB_at_EOF(wbPortrait, tmp * 612 + (1 - tmp)
* 792, tmp * 792 + (1 - tmp) * 612, i)
              End If
          End If


       Case titleext
          ContFile.Activate
          ' check if last section was portrait
          ' if so, insert page break
          ' if not, insert section break and apply portrait orientation

          If ContOrient = wbPortrait Then
             If i > 0 Then Call Insert_PB_at_EOF
          Else
             Call Insert_SB_at_EOF(wbPortrait, 792, 612, i)
          End If
          prevFile_wasTitle = True
         ' remember that this file was graph title

       Case "doc", "rtf", "txt"
          'determine page orientation and page size of first section of
this file
          ' if same as last section of the container, insert page break
          ' if different, insert section break and apply settings

           Documents.Open FileName:=Fullname, ReadOnly:=True
             With ActiveDocument.Sections.First.PageSetup
                NFOrient = .Orientation
                NFPH = .PageHeight
                NFPW = .PageWidth
             End With
           ActiveDocument.Close

           ContFile.Activate
           If ((ContOrient = NFOrient) And (ContPH = NFPH) And (ContPW =
NFPW)) Then
              If i > 0 Then Call Insert_PB_at_EOF
           Else
```

```
            Call Insert_SB_at_EOF(NFOrient, NFPH, NFPW, i)
          End If

       Case Else

          ’insert page break and print warning message and skip file
insertion
          ContFile.Activate
          If i > 0 Then Call Insert_PB_at_EOF
          Selection.EndKey Unit:=wdStory
          Selection.InsertAfter "I do not know what to do with file " &
Fullname
          Selection.Collapse Direction:=wdCollapseEnd

     End Select

     ’move to the end of container file
     ContFile.Activate
     With Selection
       .EndKey Unit:=wdStory
       ’insert header with file name if user requested it
       If frmInsertAllFiles.cbHeadid.Value = True Then
              .InsertAfter Fullname
              .EndKey Unit:=wdStory
              .InsertParagraphAfter
        End If
       .Collapse Direction:=wdCollapseEnd
       .EndKey Unit:=wdStory
       .Collapse Direction:=wdCollapseEnd
      End With
  ’insert file : documents as insert file, graphs as insert picture
     Select Case fileextension
        Case "cgm", "tif", "jpg", "wmf", "bmp", "gif"
          ’insert graphs
           ActiveDocument.InlineShapes.AddPicture _
             FileName:=Fullname,
linktofile:=frmInsertAllFiles.cbLinkid.Value, _
             Range:=Selection.Range, savewithdocument:=True
          Selection.EndKey Unit:=wdStory
          Selection.Collapse Direction:=wdCollapseEnd
        Case "doc", "rtf", "txt", titleext
           ’insert recognized documents and graph titles
          Selection.InsertFile FileName:=Fullname, _
            link:=frmInsertAllFiles.cbLinkid.Value
          Selection.Collapse Direction:=wdCollapseEnd
        Case Else
            ’ do nothing with other files
      End Select

   End If


Next
End Sub
Private Sub Insert_PB_at_EOF()
’ inserts page break at the end of active document
          With Selection
           .EndKey Unit:=wdStory
           .Collapse Direction:=wdCollapseEnd
           .Range.InsertBreak Type:=wdPageBreak
           .Collapse Direction:=wdCollapseEnd
```

```
          End With
End Sub
Private Sub Insert_SB_at_EOF(PageOrient, PageHt, PageWdt, SectBreak As
Integer)
' inserts Section break at the end of active document if SectBreak>0
' applies specified settings
Dim NewSection As Object
     If SectBreak > 0 Then
       Set NewSection = ActiveDocument.Sections.Add
     Else
       Set NewSection = ActiveDocument.Sections.Last
     End If

     With NewSection.PageSetup
        .Orientation = PageOrient
        .PageHeight = PageHt
        .PageWidth = PageWdt
     End With
     Set NewSection = Nothing

End Sub
```

Appendix C.2 - Template Text for Word 97

To install: Click on the button below to install the macro InsertAllFiles.

Install

To view macro code:

- Click on Tools/Macro/Visual Basic Editor.
- Double-click on the object frmInsertAllFiles (located in the folder Forms in Project window) to displays the form.
- Double-click anywhere on the form to view the form code.
- Double-click the module InsertAllFiles (located in the folder Modules in Project window) to view code for the part of the macro that processes the list after the form is closed.

Troubleshooting:

If setup fails, read the notes below.

Note: Macros must not be disabled when opening this file

Setup will copy the module InsertAllFiles and the form frmInsertAllFiles to NORMAL.DOT template. If your NORMAL.DOT template already has objects with these names, the setup will fail.

In such a case, do the following:

- Click on Tools/Macro/Organizer
- Make sure that the macros in NORMAL.DOT are visible.
- Look for an object named InsertAllFiles in the NORMAL.DOT window.
- If such an objects exists, rename “InsertAllFiles” objects in INSTR.DOT template window.
- Repeat these steps with the InsertAllFiles object.
- Close the Organizer window.
- Click on the installation button again.

Appendix D - Macro InsertAllFiles for Word 6.0/95

Appendix D.1 - Macro Code

```
'_____________________________________________________________
' Code for the InsertAllFiles macro for Word 6.0/95.
' Copyright Merck & Co., 1996.
' All Rights Reserved.
'_____________________________________________________________

Dim Shared logdir$
Dim Shared titext$
Dim Shared Mylist$(0)
Dim Shared listsize
Dim Shared linkid, headid

Sub MAIN
' change directory to the one used last time
On Error Goto init
startdir$ = \
  GetPrivateProfileString$("InsertAllFiles", "startdir$",
"c:\windows\wrdmacro.ini")
Goto endinit :
init:
startdir$ = Files$(".")
endinit:
On Error Resume Next
ChDir startdir$
On Error Goto 0

Dim subdirs$(0)
Dim filelist$(0)
Dim ListofFiles$(0)

' list all drives that you may use here,
' adjust dimension of drives$ if  necessary
Dim drives$(4)
drives$(0) = "c:\"
drives$(1) = "e:\"
drives$(2) = "q:\"
drives$(3) = "u:\"

' list all file extensions that you may need here,
' adjust dimension of ext$ if  necessary
Dim ext$(3)
ext$(0) = "*.*"
ext$(1) = "*.doc"
ext$(2) = "*.rtf"
```

```
ext$(3) = "*.txt"

' define extensions for files with graph titles
Dim titext$(2)
titext$(0) = "*.*"
titext$(1) = "*.tit"
titext$(2) = "*.ttl"

' initialize variables
linkid = 0
headid = 0
dobreak = 0

' fill subdirs$ with subdirectories of current one
' and filelist with files (pattern=ext$) of current directory
GetFilesAndDirs subdirs$(), filelist$(), ext$(0)

' define a dialog box for user interface
Begin Dialog UserDialog 964, 440, "Pick Files to Insert", .DirList
        Text 10, 22, 100, 13, "Directories:", .dirtxt
        Text 207, 22, 100, 13, "Files:", .filtxt
        Text 10, 8, 371, 13, dirstring$, .mydir
        ListBox 10, 41, 197, 207, subdirs$(), .dir_id
        ListBox 207, 41, 150, 207, filelist$(), .myfiles
        Text 10, 284, 80, 13, "Drive:", .dr
        DropListBox 10, 300, 88, 110, drives$(), .mydrives
        ListBox 498, 40, 452, 328, ListofFiles$(), .Mylist
        Text 207, 284, 100, 13, "File Types", .ft
        ComboBox 207, 300, 110, 92, ext$(), .FileTypes
    Text 350, 274, 150, 25, "Extention of files with GraphTitless:", \
       .Text1
        ComboBox 367, 300, 110, 92, titext$(), .titext
        CheckBox 7, 396, 188, 16, "Insert Files as LINKS", .linkid
    CheckBox 7, 415, 388, 16, "Insert headers with file path", \
       .headid
    PushButton 520, 380, 288, 21, "Swith Order of Graphs/Titles", \
       .Switchid
        PushButton 368, 39, 121, 21, "Add All", .AddAll
        PushButton 368, 225, 121, 21, "Remove All", .RemoveAll
        PushButton 368, 75, 121, 21, "Add", .Add
        PushButton 368, 109, 121, 19, "Up", .Up
        PushButton 368, 139, 121, 21, "Down", .Down
        PushButton 368, 175, 121, 21, "Remove", .Remove
        OKButton 864, 380, 88, 21
        CancelButton 860, 13, 88, 21

End Dialog

' display dialog

Dim mydlg As UserDialog
GetCurValues mydlg
x = Dialog(mydlg)

' after dialog closes, store user selected settings in the INI file
SetPrivateProfileString  "InsertAllFiles", "startdir$", Files$("."), \
   "wrdmacro.ini"

' start processing the list

' open a new file to hold all files from the list
```

```
FileNewDefault
fileout$ = WindowName$()


For i = 0 To listsize - 1
        name$ = nonum$(mylist$(i))
        ext$ = LCase$(Right$(name$, 3))
        Select Case ext$
        Case "doc", "rtf", "txt", "DOC", "RTF", "TXT"
                On Error Goto Warning
   ’ word, rtf and text documents are inserted using InsertFile
   ’ with their orientation preserved
                FileOpen .Name = name$
                Dim dlg As FilePageSetup
                GetCurValues dlg
                orient = dlg.Orientation
                FileClose 2
                Activate fileout$
                EndOfDocument
                Dim dlg As FilePageSetup
                GetCurValues dlg
                oldorient = dlg.Orientation
        ’ if necessary, insert section breaks to allow for both
        ’ landscape and portrait orientation  in one file
                If oldorient <> orient Then
                        If i > 0 Then InsertBreak .Type = 2
                        EndOfDocument
                        Select Case orient
            ’ apply original orientation of the selected file
                        Case 1
                                FilePageSetup .Orientation = 1,
.ApplyPropsTo = 0, \
                                         .PageWidth = "11 in",
.PageHeight = "8.5 in"
                        Case 0
                                FilePageSetup .Orientation = 0,
.ApplyPropsTo = 0, \
                                        .PageWidth = "8.5 in",
.PageHeight = "11 in"
                        End Select
                Else
                        If i > 0 Then InsertBreak .Type = 0
                End If
        ’ insert  file name before the file itself
                ’ if user requested to do so
                If headid = 1 Then      Insert name$
                InsertPara
         ’  insert file as copy or as link according to user request
                InsertFile .Name = name$, .Link = linkid
                dobreak = 0
                Goto getfile
        Case "cgm", "CGM"
                On Error Goto Warning
                Activate fileout$
                EndOfDocument
                Dim dlg As FilePageSetup
                GetCurValues dlg
                oldorient = dlg.Orientation
          ’ graphs will be inserted in pages oriented as portrait
                If oldorient <> 0 Then
                ’ if previous file was landscaped, insert section break
```

```
                        ' and apply portrait orientation
                        If (i > 0 And dobreak <> - 1) Then InsertBreak
.Type = 2
                        EndOfDocument
                        FilePageSetup .Orientation = 0,  .ApplyPropsTo =
0, \
                                .PageWidth = "8.5 in", .PageHeight =
"11 in"
                Else
                        If (i > 0 And dobreak <> - 1) Then InsertBreak
.Type = 0
                End If
                dobreak = 0
         ' insert  file name before the file itself
                 ' if user requested to do so
                If headid = 1 Then Insert name$
                InsertPara
         ' insert graphic file as copy  or as link
                 ' according to user request
                InsertPicture .Name = name$, .LinkToFile = 2 * linkid
                Goto getfile
        Case titext$
          ' files holding titles graphs will be inserted in portrait
          ' pages with no page break after title
                On Error Goto Warning
                Activate fileout$
                EndOfDocument
                Dim dlg As FilePageSetup
                GetCurValues dlg
                oldorient = dlg.Orientation
         ' if previous file was landscaped, insert section break and
         ' apply portrait orientation
        If oldorient <> 0 Then
                        If (i > 0 And dobreak <> - 1) Then InsertBreak
.Type = 2
                        EndOfDocument
                        FilePageSetup .Orientation = 0,  .ApplyPropsTo =
0, \
                                .PageWidth = "8.5 in", .PageHeight =
"11 in"
            Else
                        If (i > 0 And dobreak <> - 1) Then InsertBreak
.Type = 0
                End If
                dobreak = - 1
                If headid = 1 Then Insert name$
                InsertPara
         ' insert file as copy or as link according to user request
                 InsertFile .Name = name$, .Link = linkid
                Goto getfile
        Case Else
        End Select
getfile:
Next
Goto bye

'  warn user if requested file does not exist
   Warning :
        Activate fileout$
        Insert "File  " + name$ + " Does Not Exist"
        InsertPageBreak
```

```
        On Error Goto 0
        Goto getfile
Bye:
   End Sub

' function to list all files with specified extension in a directory
' input parameters: directory name, name of array to hold list files,
' extension  of files

  Sub GetFilesAndDirs(subdirs$(), filelist$(), ext$)
  Redim subdirs$(CountDirectories())
  subdirs$(0) = "[..]"
  For x = 1 To CountDirectories()
        subdirs$(x) = LCase$(GetDirectory$(x))
  Next
  count = 1
  a$ = Files$(ext$)  'first file in current directory
  While Files$() <> ""
        count = count + 1
  Wend
  Redim filelist$(count - 1)
  If Files$(ext$) <> "" Then
        filelist$(0) = LCase$(FileNameInfo$(Files$(ext$), 3))
  '   filename of the  first file
        For x = 1 To count - 1
                filelist$(x) = LCase$(FileNameInfo$(Files$(), 3))
        Next
  End If
  If CountDirectories() > 0 Then SortArray subdirs$()
  If count > 1 Then SortArray filelist$()
  End Sub

'  function used to work with dialog box
   Function DirList(id$, action, wvalue)
   Select Case action
     Case 1             ' The dialog box is displayed
          DlgValue "FileTypes", 0
     ' print the path of the current directory
     ' in the provided text box Mydir
          If Right$(Files$("."), 1) = "\" Then
                DlgText  "mydir", Files$(".")
        Else
                DlgText  "mydir", Files$(".") + "\"
          End If
        Select Case LCase$(Left$(Files$("."), 3))
    ' populate listbox mydrives with preset drive letters
        Case "c:\"
                DlgValue "mydrives", 0
        Case "e:\"
                DlgValue "mydrives", 1
        Case "q:\"
                DlgValue "mydrives", 2
        Case "u:\"
                DlgValue "mydrives", 3
        Case Else
        End Select
        listsize = 0
Case 2          ' The user selects a control
        Select Case id$
        Case "mydrives"
       ' user clicks on drive or directory and all files in this
```

```
        ’ directory with specified extension are displayed
                ChDir DlgText$("mydrives")
        DisplayDir("mydrives", "dir_id", "myfiles", "mydir", "FileTypes")
            DirList = 1
         Case "OK"
            Select Case DlgFocus$()
                    Case "OK"
            ’ user clicked on OK button : store settings
                    ’ and list of files and exit dialog box
                        logdir$ = DlgText$("mydir")
                        linkid = DlgValue("linkid")
                        headid = DlgValue("headid")
                        titext$ = Right$(DlgText$("titext"), 3)
          Case  "FileTypes"
            ’ user requested that only specified file extensions will be
            ’ listed: update display
           displayDir("mydrives", "dir_id", "myfiles", "mydir",
"FileTypes")
                DirList = 1
                Case "dir_id"
         ’  user double clicked on the directory: update display
                        ChangeDir("dir_id", "mydir")
           displayDir("mydrives", "dir_id", "myfiles", "mydir",
"FileTypes")
                        DirList = 1
                Case "myfiles"
             ’ user double-clicked on file name: add file to the list
             ’ right below highlighted file
                        newfile$ = DlgText$("mydir") +
DlgText$("myfiles")
                        selid = DlgValue("Mylist")
                        Dim tmplist$(listsize)
                        If listsize > 0 Then
                                For i = 0 To selid
                                        tmplist$(i) = NoNum$(Mylist$(i))
                                Next
                                tmplist$(selid + 1) = newfile$
                                For i = selid + 2 To listsize
                                        tmplist$(i) = Nonum$(Mylist$(i -
1))
                                Next
                        Else
                                tmplist$(listsize) = newfile$
                                selid = - 1
                        End If

                        Redim Mylist$(listsize)
                        For i = 0 To listsize
                                Mylist$(i) = MS$(i + 1) + tmplist$(i)
                        Next
                        DlgListBoxArray "Mylist", Mylist$()
                        DlgValue "Mylist", selid + 1
                        listsize = listsize + 1
                        DirList = 1
                Case Else
                End Select
         Case "linkid"
    ’  user selected option that files are inserted as links :
         ’  store this info
                dirlist = 1
         Case "headid"
```

```
    ’  user requested that file name will be inserted below the file:
    ’  store this info
               dirlist = 1
        Case "AddAll"
    ’ user requested that all listed files are added to the list: do so
                        selid = DlgValue("Mylist")
                        sizetoadd = DlgListBoxArray("myfiles")
                        Dim addlist$(sizetoadd - 1)
                        size2 = DlgListBoxArray("myfiles", addlist$())
                        If addlist$(0) <> "" Then
                        Dim tmplist$(listsize + sizetoadd - 1)
                        If listsize > 0 Then
                                For i = 0 To selid
                                        tmplist$(i) = Nonum$(Mylist$(i))
                                Next
                                For i = selid + 1 To selid + sizetoadd
                                        tmplist$(i) = \
                                                DlgText$("mydir") +
addlist$(i - selid - 1)
                                Next
                        For i = selid + sizetoadd + 1 To listsize +
sizetoadd - 1
                                        tmplist$(i) = Nonum$(Mylist$(i -
sizetoadd - 1))
                                Next
                                selid = selid + sizetoadd
                        Else
                                For i = 0 To sizetoadd - 1
                                tmplist$(i) = DlgText$("mydir") +
addlist$(i)
                                Next
                                selid = sizetoadd - 1
                        End If
                        listsize = listsize + sizetoadd
                        Redim Mylist$(listsize - 1)
                        For i = 0 To listsize - 1
                                Mylist$(i) = MS$(i + 1) + tmplist$(i)
                        Next
                        DlgListBoxArray "Mylist", Mylist$()
                        DlgValue "Mylist", selid
                        End If
                        DirList = 1
        Case "RemoveAll"
     ’  user requested that all files are removed from the list: do so
                Redim Mylist$(0)
                DlgListBoxArray "Mylist", Mylist$()
                listsize = 0
                DlgValue "Mylist", - 1
                DirList = 1
        Case "Add"
     ’  user requested that selected file is added to the list : do so
                newfile$ = DlgText$("mydir") + DlgText$("myfiles")
                selid = DlgValue("Mylist")
                selid2 = DlgValue("myfiles")
                If selid2 > - 1 Then
                sizefiles = DlgListBoxArray("myfiles") - 1
                Dim tmplist$(listsize)
                If listsize > 0 Then
                        For i = 0 To selid
                                tmplist$(i) = Nonum$(Mylist$(i))
                        Next
```

```
                tmplist$(selid + 1) = newfile$
                For i = selid + 2 To listsize
                        tmplist$(i) = Nonum$(Mylist$(i - 1))
                Next
        Else
                tmplist$(listsize) = newfile$
                selid = - 1
        End If

        Redim Mylist$(listsize)
        For i = 0 To listsize
                Mylist$(i) = MS$(i + 1) + tmplist$(i)
        Next
        DlgListBoxArray "Mylist", Mylist$()
        DlgValue "Mylist", selid + 1
        listsize = listsize + 1
        If selid2 < sizefiles Then
                DlgValue "myfiles", selid2 + 1
        Else
                DlgValue "myfiles", - 1
        End If
        End If
        DirList = 1
  Case "Remove"
’  user requested that selected file is removed from the list:
’ do so
        selid = DlgValue("Mylist")
        listsize = listsize - 1
        If listsize > 0 Then
                Dim tmplist$(listsize - 1)
                For i = 0 To selid - 1
                        tmplist$(i) = Nonum$(Mylist$(i))
                Next
                For i = selid To listsize - 1
                        tmplist$(i) = Nonum$(Mylist$(i + 1))
                Next
                Redim Mylist$(listsize - 1)
                For i = 0 To listsize - 1
                        Mylist$(i) = MS$(i + 1) + tmplist$(i)
                Next
        Else
                Redim Mylist$(0)
        End If
        DlgListBoxArray "Mylist", Mylist$()
        If selid < listsize  Then DlgValue "Mylist", selid
        dirlist = 1
  Case "Up"
’  as user requested, move selected file one position up on the list
        selid = DlgValue("Mylist")
        If selid > 0 Then
                tmp1$ = Nonum$(Mylist$(selid))
                tmp2$ = Nonum$(Mylist$(selid - 1))
                Mylist$(selid) = MS$(selid + 1) + tmp2$
                Mylist$(selid - 1) = MS$(selid)  +  tmp1$
                DlgListBoxArray "Mylist", Mylist$()
                DlgValue "Mylist", selid - 1
        End If
        dirlist = 1
  Case "Down"
’ as user requested, move selected file
’ one position down on the list
```

```
                selid = DlgValue("Mylist")
                If selid < listsize - 1 Then
                        tmp1$ = Nonum$(Mylist$(selid))
                        tmp2$ = Nonum$(Mylist$(selid + 1))
                        Mylist$(selid) = MS$(selid + 1) +  tmp2$
                        Mylist$(selid + 1) = MS$(selid + 2)  + tmp1$
                        DlgListBoxArray "Mylist", Mylist$()
                        DlgValue "Mylist", selid + 1
                End If
                dirlist = 1
        ’  If title extension is specified then rearrange the list
        ’  so titles of graphs are listed  before graphs
        Case "Switchid"
                For i = 1 To listsize - 1
                        name$ = nonum$(mylist$(i))
                        ext$ = LCase$(Right$(mylist$(i), 3))
full$ = nonum$(LCase$(Left$(mylist$(i), InStr(mylist$(i), ".") - 1)))
pfull$ = \
nonum$(LCase$(Left$(mylist$(i - 1), InStr(mylist$(i - 1), ".") - 1)))

If (ext$ = Right$(DlgText$("titext"), 3) And full$ = pfull$) Then
        prev$ = mylist$(i - 1)
        curr$ = mylist$(i)
        mylist$(i - 1) = curr$
        mylist$(i) = prev$
End If
                Next i
                DlgListBoxArray "Mylist", Mylist$()
                Dirlist = 1
        Case Else
        End Select
Case 3
        Select Case id$
        Case "FileTypes"
    displayDir("mydrives", "dir_id", "myfiles", "mydir", "FileTypes")
        dirlist = 1
        Case "titext"
                dirlist = 1
        Case Else
                dirlist = 1
        End Select
Case Else
End Select
End Function

Sub ChangeDir(dir$, label$)
’ function that changes current directory to the selected one
’ first argument is a subdirectory name,
’ second argument is the current directory
        If DlgText$(dir$) <> "[..]" Then
                ChDir DlgText$(label$) + DlgText$(dir$)
            ’ full path=current dir + subdir
        Else
      ’ user clicked on [ ] to (parent directory)
            tmp = Len(DlgText$(label$))
                If tmp > 3 Then
      ’ parent directory is not root, so strip backslash from the path
      ’ to parent directory
                        ChDir Mid$(DlgText$(label$), 1, tmp - 1)
                        tmp$ = Files$(".")
                        ChDir ".."
```

```
                End If
        End If
End Sub


Sub displayDir(drive$, dir$, file$, label$, type$)
' function to populate the label with current directory and listboxes
' with subdirectory list and with file list
                Dim subdirs$(0)
                Dim filelist$(0)
                WaitCursor 1
                GetFilesAndDirs subdirs$(), filelist$(), DlgText$(type$)
                DlgListBoxArray dir$, subdirs$()
                DlgListBoxArray file$, filelist$()
                WaitCursor 0
                dirstring$ = LCase$(Files$("."))
        If Right$(dirstring$, 1) <> "\" Then dirstring$ = dirstring$ +
"\"
                DlgText$ label$, dirstring$
End Sub

Function MS$(number)
' function that formats numbers in the file list
If number < 10 Then
        tmp$ = " " + Str$(number) + "> "
Else
        tmp$ = Str$(number) + "> "
End If
MS$ = tmp$
End Function

Function NoNum$(word$)
' function that strips the numbers from the file list
pos = InStr(word$, ">")
tmp$ = Mid$(word$, pos + 2, Len(word$) - pos + 1)
NoNum$ = tmp$
End Function
```

Appendix D.2 - Template Text for Word 6.0/95

Note: Do not install this macro if you are using Office 97.

To install the macro InstallAllFiles, click on the Install macro InsertAllFiles" button on the toolbar above.

To view macro code:

- Click on Tools/Macro.
- Select the macro InstallAllFiles from the list.
- Click on the Edit button.

Questions and comments should be directed to:

Iza Peszek, PhD.
Merck & Co., Inc.
P. O. Box 2000, RY33-404
Rahway, NJ 07065-0900
E-mail: peszeks@erols.com

Modified code is not supported by the authors or SAS Institute.