Paper 26

Using The SAS® System and Dynamic Data Exchange to Populate Microsoft Word® Documents with Text, Tables, and Graphs

Michael C. Harris, Amgen, Thousand Oaks, CA

Abstract

Dynamic data exchange (DDE) is one of the means by which appropriately written Microsoft® Windows® applications can communicate and share data with one another. DDE is a client/server architecture. The application that controls another via DDE is the client, and the controlled application (which usually provides some kind of functionality not present in the client) is the server. The SAS Data Step Language can encapsulate WordBasic statements and enable SAS programs running on personal computers to be effective clients and control the actions of Microsoft Word for Windows (Word) in ways that are useful for creating, populating, and formatting tables, and placing text strings and graphical elements at predetermined locations within documents.

Introduction

Integrating the results of analyses in text, tabular, and graphical formats into documents created with word processing software is a common activity. SAS software is capable of generating all three types of output mentioned above, but has limited intrinsic features for creating publication quality text and tables, and none at all for directly populating existing word processor files or creating foreign file types without significant programming effort. A library of SAS macros for generating output via DDE can provide the desired flexibility and functionality while maximizing code reuse. Gilmore discussed the use of bookmarks within Microsoft Word to insert small amounts of text at predetermined locations within documents and briefly described how to control the actions of Word[1]. We will focus on techniques for complex data transfer and assume the reader has a working knowledge of Word and the SAS Macro language. Extensive code samples are presented.

Requirements

Specifying what a software system must do is a necessary step in the life cycle of the system. Since anything that can be done via keyboard or mouse can be done via DDE with SAS, it makes sense to identify a subset of Word's functionality that specifically addresses the problem at hand. Note that your problem may not be exactly the same as someone else's, and you may therefore have different requirements. Below is a minimal set of requirements that can serve a starting point for future development. These requirements reflect the functionality used to produce the examples shown in this paper.

Input Requirements

1. The system shall accept SAS data sets as input.
2. The system shall accept user-specified alphanumeric input.
3. The system shall have the ability to open existing Microsoft Word documents.

Processing Requirements

1. The system shall have the ability to launch Microsoft Word without manual intervention.
2. The system shall have the ability to close the current Microsoft Word document.
3. The system shall have the ability to exit Microsoft Word without manual intervention.
4. The system shall have the ability to toggle bold printing on and off.
5. The system shall have the ability to control justification attributes.
6. The system shall have the ability to toggle the superscript attribute on and off.
7. The system shall have the ability to control typefaces and point sizes.
8. The system shall have the ability to control navigation within a document.
9. The system shall have the ability to control text selection within a document.
10. The system shall have the ability to control navigation with tables.
11. The system shall have the ability to control text selection within a table.
12. The system shall provide facilities for formatting tables.
13. The system shall provide facilities for creating tables of user-specified dimensions.

Output Requirements

1. The system shall have the ability to save the current file.
2. The system shall have the ability to save the current file to a new filename.
3. The system shall have the ability to save the current file in any format supported by Word.

Operational Requirements

1. The system shall function with Windows NT 4.0
2. The system shall function with Microsoft Word for Windows version 7.0

Design Decisions

There are two approaches to implementing a DDE system for table generation. The first is to provide ease of use at the expense of flexibility. The second is to provide a high level of control with increased complexity from both the developer's and the user's perspectives. The second approach almost automatically expands functionality beyond table generation into total document control. Since there is a time and place for each, both approaches will be demonstrated.

Ease of Use

The idea here is to wrap every piece of functionality in a DATA _NULL_ step packaged in a macro so that each call stands alone. This makes it easy to understand what a program is doing while hiding the details. It is convenient to bundle related macros into one file which is itself a macro that can be invoked through the autocall facility. Invoking the macro sets up the environment required by the system. Below is code for such a library. Space doesn't permit a full description of each macro.

```
%macro ddemac;
  options noxwait;
  filename cmds dde 'winword|system';

/* openword starts Winword and waits for 5
seconds */
  %macro openword;
    data _null_;
      call system('start
c:\progra~1\msoffice\winword\winword.exe');
      t=sleep(5);
    run;
  %mend;
/* opendoc opens the specified document */
  %macro opendoc(doc=);
```

```
    data _null_;
      cmd='[FileOpen.Name =' || '"' || "&doc" ||
'"' || ']';
      file cmds;
      put cmd;
    run;
  %mend;
/* wordsave saves current Winword document */
  %macro wordsave;
    data _null_;
      file cmds;
      put '[FileSave]';
    run;
  %mend;
/* wordexit closes Winword */
  %macro wordexit;
    data _null_;
      file cmds;
      put '[FileExit]';
    run;
  %mend;
/* findtbl positions cursor in row 1 col 1 of
specified table */
  %macro findtbl(table);
    %if &table= %then
      %let table=1;
      data _null_;
      length cmd $ 200;
      cmd='[EditGoTo .Destination = ' || '"' ||
't' || "&table" || '"]';
      file cmds;
      put cmd;
    run;
  %mend;
/* table writes the input data set to the
current table beginning at the current cursor
location. Use findtbl to locate row 1 col 1 of a
particular table */
  %macro table(dsn=, rows=, cols=);
  data _null_;
    %let cells=%eval(&rows*&cols);
    set &dsn;
    file cmds;

    %do r=1 %to &rows;
      %do c=1 %to &cols;
        cmd= '[Insert "' || col&c || '"]';
        put cmd;
        %if %eval(&r*&c) < &cells %then %do;
          put '[NextCell]';;
        %end;
      %end;
      return;
    %end;
  run;

  data _null_;
    file cmds;
    put '[TableDeleteRow]';
  run;
%mend;
/* tblfmt uses the autoformat feature of Word to
format a table */
  %macro tblfmt(fmt);
    %if &fmt= %then
      %let fmt=1;

    data _null_;
      file cmds;
      put "[TableAutoFormat .Format = &fmt,
.Borders = 1, .Shading = 1, .Font = 1, .Color =
0, .AutoFit = 1, .HeadingRows = 1, .FirstColumn
= 1, .LastRow = 0, .LastColumn = 0]";
    run;
  %mend;
/* go2bkmrk movse cursor to the specified
bookmark */
%macro go2bkmrk(bm);
  data _null_;
    file cmds;
    put '[EditGoTo .Destination = "' "&bm" '"]';
  run;
%mend;
/* ins_pict inserts a graphic at the insertion
point */
%macro ins_pict(name=, link=);
  data _null_;
    file cmds;
    cmd='[InsertPicture .Name = "'|| "&name" ||
'", .LinkToFile = "' || "&link"
    || '"]';
    put cmd;
  run;
%mend;
%mend ddemac;
```

More Control

A high level of flexibility can be achieved by wrapping SAS code fragments containing WordBasic statements in macros. The full power of the DATA step is then available to the programmer. The code below fulfills the requirements not met above and demonstrates the kinds of things that can be done.

```
%macro ddemac2;
 /* wordbold toggles bold printing on and off;
valid values for action are 'on' and 'off' */
  %macro bold(action);
     %let action=%upcase(&action);

      %if &action=ON %then %do;
         put '[Bold 1]';
      %end;
      %if &action=OFF %then %do;
         put '[Bold 0]';
      %end;
  %mend;
/* center turns centering on */
  %macro center;
     put '[CenterPara]';
  %mend;
/* leftjust turns on left justification */
  %macro leftjust;
     put '[LeftPara]';
  %mend;
/* ritejust turns on right justification */
  %macro ritejust;
     put '[RightPara]';
  %mend;
/* para inserts a new paragraph */
  %macro para;
     put '[InsertPara]';
  %mend;
/* page inserts a page break */
  %macro page;
     put '[InsertBreak .Type = 0]';
  %mend;
/* fontsize sets font to specified size */
  %macro fontsize(points);
     put "[FontSize &points]";
  %mend;
/* sprscrpt sets the superscript text attribute;
on can be 1 to turn on or 0 to turn off */
  %macro sprscrpt(on);
    %let on=%upcase(&on);
    %if &on=ON %then %do;
       put "[Superscript 1]";
    %end;
    %if &on=OFF %then %do;
       put "[Superscript 0]";
    %end;
  %mend;
/* puttext writes text to the current cursor
location */
  %macro puttext(text);
     put '[Insert ' '"' "&text" '"]';
  %mend;
/* putvar writes a data step variable to the
current cursor location */
  %macro putvar(var);
     cmd= '[Insert ' || '"' || &var || '"]';
     put cmd;
  %mend;
```

```
/* nextcell moves to the next cell in a table
the number of times specified by n */
  %macro nextcell(n);
    %if &n= %then %let n=1;
    %do i=1 %to &n;
      put '[NextCell()]';
    %end;
  %mend;
%mend;

/* merge merges the specified number of cells
beginning at the insertion point */
   %macro merge(count);
      put "[CharRight &count, 1]";
      put '[TableMergeCells]';
   %mend;
/* linedown moves the insertion point down the
specified number of rows in a table */
  %macro linedown(n);
    put "[LineDown &n]";
  %mend;
/* selsent selects the current sentence or cell
contents */
   %macro selsent;
      put '[SelectCurSentence]';
   %mend;
/* newtable creates a table at the current
insertion point */
   %macro newtable(rows=, cols=, format=0);
      put '[TableInsertTable .NumColumns = "'
"&cols" '", .NumRows = ' '"' "&rows" '", .Format
= ' '"' "&format" '"]';
   %mend;
/* seltbl selects entire table for formatting */
   %macro seltbl;
      put '[TableSelectTable]';
   %mend;
/* page inserts a page break into a document */
   %macro page;
      put '[InsertBreak .Type = 0]';
   %mend;
/* borders toggles border attributes on and off.
Valid values for style are top, bottom, left,
right, inside, outside and none. Bool can be 0
to turn off or 1 to turn on */
   %macro borders(style, bool);
      %let style=%upcase(&style);
      %if &style=TOP %then %do;
         put "[BorderTop &bool]";
      %end;
      %else %if &style=BOTTOM %then %do;
         put "[BorderBottom &bool]";
      %end;
      %else %if &style=LEFT %then %do;
         put "[BorderLeft &bool]";
      %end;
      %else %if &style=RIGHT %then %do;
         put "[BorderRight &bool]";
      %end;
      %else %if &style=INSIDE %then %do;
         put "[BorderInside &bool]";
      %end;
      %else %if &style=OUTSIDE %then %do;
         put "[BorderOutside &bool]";
      %end;
      %else %if &style=NONE %then %do;
         put "[BorderNone &bool]";
      %end;
      %else %put Error: Invalid border
specified.;
   %mend borders;
```

Sample Code and Output

The Easy Way

The following code generates a table with a small amount of code using the %table macro. In this example, the input data set has been formatted to match every cell in the table. To conserve space, code used to create the sample data is not shown. The code was used to populate and format Sample Table 1 below.

```
%openword;
%opendoc(doc=h:\My Documents\papers\DDE Abstract.doc);
%findtbl(1);
%table(rows=13, cols=7, dsn=table);
/* cursor is no longer in table, so reposition it to format the table */
%findtbl(1);
%tblfmt(17);

/* add a timestamp to some embedded text via a bookmark */
filename dt dde 'winword|h:\My Documents\papers\DDE Abstract.doc!datestamp';

data _null_;
  dtm=put(datetime(), datetime14.);
  file dt;
  put dtm;
run;

/* add the programmer's name */
%go2bkmrk(programmer);
%wordtext(Mike Harris);
```

Sample Table 1.

Level	Subgroup	Column 3 Value	Column 4 Value	Column 5 Value	Column 6 Value	Column 7 Value
Level 1	Group 1	0.042	9.251	80.12	605.9	2203
	Group 2	0.917	7.537	75.81	405.2	7043
	Group 3	0.637	3.287	93.77	991.6	3359
Level 2	Group 1	0.969	5.828	1.78	472.3	3575
	Group 2	0.920	6.828	61.62	5.8	6055
	Group 3	0.165	5.687	89.06	587.4	8256
Level 3	Group 1	0.219	3.984	61.02	662.3	494
	Group 2	0.824	9.456	17.93	166.9	7343
	Group 3	0.507	9.936	18.23	49.8	9319

14JUL98:17:00 Programmer: Mike Harris

More Control

Below is a more complex example which demonstrates text insertion, table creation, pagination, superscripts, merged cells, custom table formatting, and text attributes customized at the cell level. The programmer handles all the details, and consequently more code is required than in the first example. To conserve space, pagination was simulated by restricting the number of rows printed and handling control breaks appropriately. Inserting physical page breaks is trivial. Once again, code used to generate sample data is omitted. Word is assumed to be running with the proper document in the current window. The bulk of the processing takes place within a DATA _NULL_ step in which the programmer explicitly controls navigation and control breaks. Page breaks are handled by creating and populating multiple tables and formatting them after all data has been written. Output is shown on the final page of this paper as Sample Table 2.

```
data _null_;
   retain rownum 1 maxrows 15 pagenum 1;
   file cmds;
  set testdata end=eof;
   by level;
   if(_n_=1 or rownum=1) then do;
   /* write title */
      %center %bold(on) %fontsize(12);
      %puttext(Sample Table 2.) %sprscrpt(on)
%puttext(1) %sprscrpt(off);
      %para %bold(off) %fontsize(10) %para;

/* create a table with 1 row and 6 columns */
      %newtable(rows=1, cols=6);

/* write column headers without merged cells */
      %nextcell(2) %center;
      %nextcell(2) %center;
      %nextcell(2) %puttext(Level);
      %nextcell %puttext(Sub-level);
      %nextcell %puttext(Measurement 1);
      %nextcell %puttext(Measurement 2);
      %nextcell %puttext(Measurement 1);
      %nextcell %puttext(Measurement 2);
      %nextcell %lineup(1) %selrow %bold(on)
%center %borders(bottom);
      %linedown(1) %bold(off);
      rownum+1;
   end;

   if(first.level or rownum=2) then do;
      %bold(on) %center;
      %putvar(level);
   end;
   /* write body of table */
   %nextcell() %center %putvar(var1);
   %nextcell %ritejust %putvar(var2);
   %nextcell %ritejust %putvar(var3);
   %nextcell %ritejust %putvar(var4);
   %nextcell %ritejust %putvar(var5);
   %nextcell;
   rownum+1;

   if(last.level) then do;
      %nextcell(6);
      rownum+2;
      %bold(off);
   end;

   if(rownum ge maxrows or eof) then do;
      %linedown %leftjust %sprscrpt(on);
      %puttext(1) %sprscrpt(off);
      %puttext(This is a footnote referred to by
the title.);
      %para; /* %page inserts page break */
      pagenum+1;
      rownum=1;
   end;
   if(eof) then do;
      totpage=put((pagenum-1), 3.);
      call symput('totpage', totpage);
   end;
run;
/* determine column widths using an empirically
determined fudge factor */
data _null_;
   len1=put(ceil(length('level    1')*0.079*72),
3.);
   len2=put(ceil(length('sublevel
10')*0.079*72), 3.);
   call symput('len1', len1);
   call symput('len2', len2);
run;
/* Merge column headers and put outside borders
on all panels. Use a macro to format so we don't
have to know how many pages were written */
%macro fixit;
   %do p=1 %to &totpage;
      %findtbl(&p);
      data _null_;
         file cmds;
         %selcol;
         %colwidth(&len1);
         %nextcell(2);
         %selcol %colwidth(&len2);
         %nextcell(2) %merge(2) %bold(on);
         %puttext(Group A) %borders(bottom);
         %nextcell %merge(2) %bold(on);
         %puttext(Group B) %borders(bottom);
         %seltbl %borders(outside);
      run;
   %end;
%mend;

%fixit;
```

Handling Graphics

Graphics files can be either statically embedded or dynamically linked into a document with the %ins_pict macro. A typical scenario would be to create a bookmark at the desired location, then at runtime use %go2bkmrk to set the insertion point followed by %ins_pict to insert the desired graphics file. Linking and embedding graphics with DDE is trivial.

Developing DDE Macro Libraries

Anything that can be done via mouse or keyboard can be done with WordBasic. It follows that anything that can be done via mouse or keyboard can be done through base SAS by writing WordBasic to a file reference of type DDE. Though technically possible (and not particularly difficult) you probably don't want to write a macro for every function and statement in the WordBasic language. A better approach would be to identify the functionality you need to meet your specific requirements and map that functionality to WordBasic functions and statements. For example, the list of requirements in this paper encompasses only the functionality necessary to produce the sample output shown. As your requirements change it is easy to add new macros to a library.

How does one discover the syntax necessary to accomplish a specific task? The easiest way is to use the online WordBasic Reference provided with version 7.0 of Word. Statements and functions are described in detail and examples are provided. In cases where several arguments are possible, you may choose to default some of them, first ensuring that your specific requirements are met. For example, the %page macro (used to insert page breaks) inserts a break of type 0. You could generalize the macro to insert any type of break supported by Word by passing in a macro parameter specifying the desired break[2].

Another type of defaulting is illustrated by the %nextcell macro. If no argument is specified, it is assumed that you want to move one cell. Otherwise, the insertion point is moved the specified number of cells. This can result in the creation of new table rows, a necessity for data driven programming.

DDE macros written to the version 7 WordBasic application programming interface will work with version 8 of Word even though its native scripting language is Visual Basic®. If you are currently using version 8 of Word, you will need a good WordBasic reference, as the online help documents only Visual Basic, a substantially different language.

Cross Platform Usage

It is not unusual for networks to have a mixture of hardware, such as UNIX servers for large scale data storage and personal computers (PCs) on user's desktops. The SAS System provides tools for doing your processing on the platform best suited for the job. You could summarize large data sets on a UNIX workstation, then through the services of the spawner utility start a remote SAS session on a PC running Windows NT. The UNIX workstation could then remote submit code to the PC to assign remote libnames or download data, start Word, populate and save documents, then exit and return control to the calling UNIX program. This type of client/server computing works well in non-interactive mode.

Tips and Traps

Debugging table creation programs can be tricky, since you are working in a language embedded within another language. The Data Step Debugger can help you understand exactly what your code is doing by stepping through it and seeing the results in your Word document in real time. Performance of DDE applications is processor bound and can also consume large amounts of memory. For production work, sixty-four megabytes of RAM is reasonable. There is a memory leak in Word 7.0 that causes it to consume ever increasing amounts of memory as large jobs run. Exiting and restarting Word reclaims that memory.

Conclusion

Dynamic Data Exchange is a powerful means of integrating SAS output into compliant applications. Developing a library of SAS macros can simplify programming DDE applications and increase productivity.

Sample Table 2.[1]

		Group A		Group B	
Level	**Sub-level**	**Measurement 1**	**Measurement 2**	**Measurement 1**	**Measurement 2**
Level 1	Sublevel 1	0.261	0.475	0.088	0.482
	Sublevel 2	0.100	0.227	0.806	0.009
	Sublevel 3	0.940	0.072	0.741	0.687
	Sublevel 4	0.029	0.011	0.934	0.304
Level 2	Sublevel 1	0.765	0.078	0.126	0.998
	Sublevel 2	0.011	0.707	0.107	0.960
	Sublevel 3	0.376	0.294	0.792	0.026

[1]This is a footnote referred to by the title.

Sample Table 2.[1]

		Group A		Group B	
Level	**Sub-level**	**Measurement 1**	**Measurement 2**	**Measurement 1**	**Measurement 2**
Level 2	Sublevel 4	0.718	0.804	0.310	0.400
Level 3	Sublevel 1	0.148	0.371	0.254	0.591
	Sublevel 2	0.695	0.304	0.246	0.076
	Sublevel 3	0.369	0.440	0.649	0.608
	Sublevel 4	0.783	0.119	0.273	0.556

[1]This is a footnote referred to by the title.

References

[1] Gilmore, Jodie "Using Dynamic Data Exchange with Microsoft Word", SAS Institute, Inc., *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 1997, p 1462

[2] "Help Topics: Microsoft Word", Microsoft Corporation, *Microsoft Word for Windows 95 Version 7.0a Word Basic Reference*, Redmond, WA: Microsoft Corporation, 1996, online reference

Acknowledgements

SAS in a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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

Contacting the Author

Mr. Harris can be reached via snail mail at:

Amgen
One Amgen Center Drive, MS 24-3-A
Thousand Oaks, CA, 91320-1789

or by sending e-mail to mikeh@amgen.com