Using Dynamic Data Exchange with Microsoft Word

Jodie Gilmore, Freelance Technical Writer, Washougal, WA

ABSTRACT

Dynamic Data Exchange (DDE) enables you to share data between the SAS System and many other Windows applications, including Microsoft Word. In this paper, you'll learn how to prepare your Word documents so that the SAS System can access them, and how to write SAS code that reads data from and writes data to those Word documents. Also, you'll learn how to use DDE to send commands to Word documents from your SAS session.

INTRODUCTION

One of the advantages of using the SAS System under Microsoft Windows is that you can share data between the SAS System and other Windows applications, such as spreadsheets, databases, and word processing applications. The SAS System supports several data-sharing mechanisms, one of which is Dynamic Data Exchange (DDE). Many other Windows applications also support DDE; Microsoft Word is one of those applications.

The key to using DDE with any application is to discover the "hooks" that the SAS System can use to access the other application's data. For Word, the "hooks" are bookmarks. (Another example of "hooks" are the row and column designators in spreadsheet applications such as Excel and Lotus 1-2-3.) So, to use DDE to share data between the SAS System and Word, you must define bookmarks in your Word document and then write SAS code that uses these bookmarks. This paper walks you through some examples of using DDE between the SAS System and Word. Once you understand the basic steps, you can modify the example code to fit your needs.

Not only does DDE enable you to share data between applications, but it also enables you to control other applications from your SAS session. For example, using DDE, your SAS program can open, close, save, print, and otherwise manipulate Word documents. This paper gives you some examples of sending these types of commands to Word via DDE.

THE EXAMPLES

The examples in this paper use the following scenario: You work for the State Agriculture Department, and are in charge of pest management. Several farms are conducting studies of various pest control methods, and you use the SAS System to analyze the data the farmers provide. You also send a quarterly report, formatted with Microsoft Word, to farmers statewide.

DEFINING BOOKMARKS IN WORD

Bookmarks are a way of permanently marking a specific location in a Word document. You can think of bookmarks as analogous to labels in a BASIC program; or, think of them as the on-screen equivalent of a bent-over page or a section underlined in red.

Any data in a Word document can be marked by a bookmark: a few characters, a single word, a whole paragraph, a graphic, or a cell in a table. A Word bookmark has a distinct name with the following characteristics:

- up to 40 characters long
- begins with a letter and contains only letters, numbers, and the underscore (_) character.

Creating a Bookmark

To create a bookmark in a Word document, follow these steps:

1. Highlight the text you want marked by the bookmark. To create an empty bookmark, simply place the cursor where you want the bookmark to be.
2. Choose **Edit** from the Word menu, then choose **Bookmark**. The Bookmark dialog box appears, similar to the one shown in Figure 1.

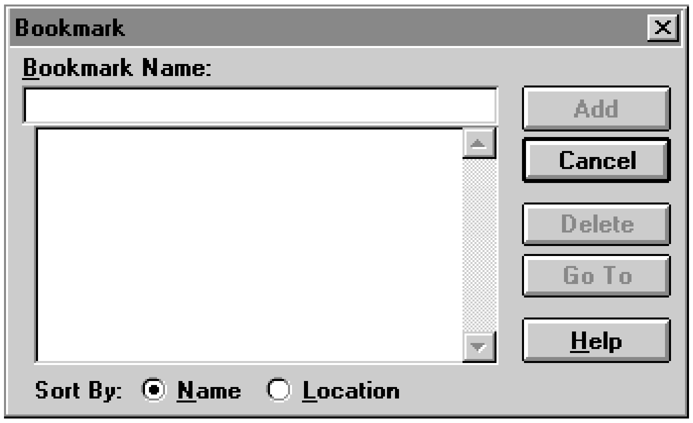


Figure 1.   Sample Bookmark Dialog Box

3. Type the name of the bookmark in the **Bookmark Name** field.
4. Click on the **Add** button.

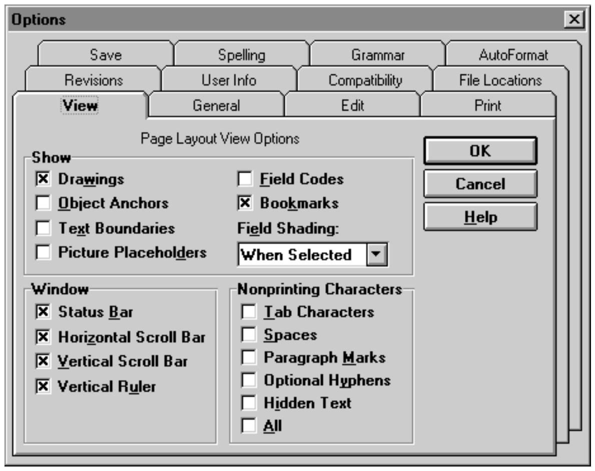
Repeat these steps for each bookmark you want to create.

Viewing Bookmarks

Bookmarks are denoted in Word documents by square brackets: [*bookmark-text*]. These square brackets are not always visible. If you do not see the square brackets, follow these steps to make the brackets visible:

1. Choose **Tools** from the Word menu, then choose **Options**.
2. Click on the **View** tab.
3. Under **Show**, click on the **Bookmarks** checkbox.
4. Click on the **OK** button.

Figure 2 shows the **View** tab for your reference.



Page Layout View Options

Show
- [x] Drawings
- [ ] Object Anchors
- [ ] Text Boundaries
- [ ] Picture Placeholders
- [ ] Field Codes
- [x] Bookmarks

Field Shading: When Selected

Window
- [x] Status Bar
- [x] Horizontal Scroll Bar
- [x] Vertical Scroll Bar
- [x] Vertical Ruler

Nonprinting Characters
- [ ] Tab Characters
- [ ] Spaces
- [ ] Paragraph Marks
- [ ] Optional Hyphens
- [ ] Hidden Text
- [ ] All

Figure 2. View Tab in the Options Dialog Box

UNDERSTANDING THE FILENAME STATEMENT USED WITH DDE

To use DDE with the SAS System, you use a special form of the FILENAME statement that creates a fileref for the data you want to transfer. The syntax for this form of the FILENAME statement is as follows:

FILENAME *fileref* DDE '*DDE-triplet*' <*DDE-options*>;

- *Fileref* can be any valid fileref for the SAS System.
- DDE is a keyword that tells the SAS System you are using Dynamic Data Exchange.
- *DDE-triplet* tells the SAS System exactly what data you want to transfer. You can think of the DDE triplet as the ID for the data. Each unique set of data has a unique ID. The following section gives more information on the DDE triplet.
- *DDE-options* are one or more options that control how the data exchange is handled. For example, the NOTAB option is useful when you want to transfer data that does not contain tabs between values. See the *SAS Companion to the Microsoft Windows Environment, Version 6, Second Edition* for a full description of the various DDE options.

Specifying the DDE Triplet for Word Data

To specify the DDE triplet, you must know three things about the application you are writing data to or reading data from:

- the name of the executable file for the application (for Word, WINWORD.EXE)
- the name of the file you want to access (for Word, the document name)
- how to refer to the access point in the file (for Word, the bookmark name).

This information is separated by special characters, as follows:

executable|file-name!access-point

Here is an example DDE triplet:

winword|organic.doc!bookmk1

- winword is the name of the Microsoft Word executable file. You do not have to include the .EXE extension.
- organic.doc is the name of the Word document you want to access.
- bookmk1 is the name of the bookmark in the Word document.

The following FILENAME statement uses this DDE triplet to define the WORD1 fileref:

```
filename word1 dde 'winword|crops.doc!bookmk1';
```

ACCESSING WORD DATA VIA DDE

Now that you know how to define bookmarks in your Word file and how to write FILENAME statements that access these bookmarks, you are ready to write a DDE program.

Example 1: Reading a Numeric Variable from a Word Document

Suppose a county Extension Office has sent you some information on organic methods of controlling crop pests. But the information is in a Word document, not in a SAS data set. The following program defines a fileref, then uses that fileref to read the data from the Word document. For this example to work, Word must be already started and the document must already be open.

```
/* Define the fileref FARMNUM--the number */
/* of farms involved in the study. */
filename farmnum dde
   'winword|organic.doc!NUMBER_OF_FARMS'
   notab;

/* Read the number of farms into the /*
/* variable FARMS. */
data org_info;
   infile farmnum;
   input farms;
run;
```

Example 2: Reading Variables from a Word Document Using a Macro

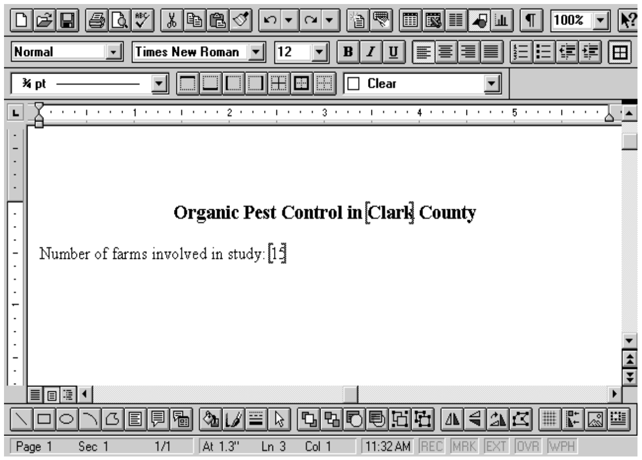
The following program is similar to Example 1, except it reads two variables: FARMS is numeric, while COUNTY is character. This means you need two bookmarks and two FILENAME statements. Also, this example stores the variables in a permanent data set, so a LIBNAME statement is necessary. In addition, this example shows how DDE fits in with the rest of the SAS System by using a macro, GET_DATA, to pass in the bookmark names. For this example to work, Word must be already started and the document must already be open.

```
%macro get_data(mark1=,mark2=);
/* Define the filerefs, one for each */
/* bookmark. */
filename file1 dde
   "winword|organic.doc!&mark1" notab;
filename file2 dde
   "winword|organic.doc!&mark2" notab;

/* Define the libref for the permanent */
/* SAS data library. */
libname cropdata 'c:\sas\cropdata';

/* Read the data. */
data cropdata.org_info;
   length cntyname $45;
   infile file1;
   input farms;
   infile file2;
   input cntyname $;
run;

/* Print the resulting data set. */
proc print;
```

```
run;
%mend get_data;

/* Call the macro with the bookmark names. */
%get_data(mark1=NUMBER_OF_FARMS,
          mark2=COUNTY_NAME)
```

The %MACRO statement uses keyword parameters that allows the user to pass in two bookmark names. The FILENAME statements use double quotation marks, because macro variables will not resolve if single quotation marks are used. The LENGTH statement sets the character variable length to an arbitrary 45--if your data require longer or shorter variables, adjust the code accordingly.

Figure 3 shows a sample Word document from the County Extension office (obviously this is not a full-fledged report, but it gives you the idea), with the square brackets denoting the two bookmarks.



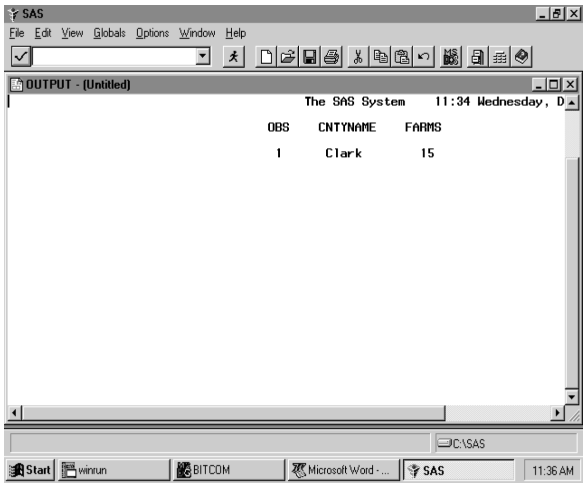
Organic Pest Control in [Clark] County

Number of farms involved in study: [15]

Normal
Times New Roman
12
100%
¾ pt
Clear
Page 1 Sec 1 1/1 At 1.3" Ln 3 Col 1 11:32 AM REC MRK EXT OVR WPH

Figure 3. Sample Word Document with Two Bookmarks

Figure 4 shows the OUTPUT window after the data is pulled into the CROPDATA.ORG_INFO data set and printed.



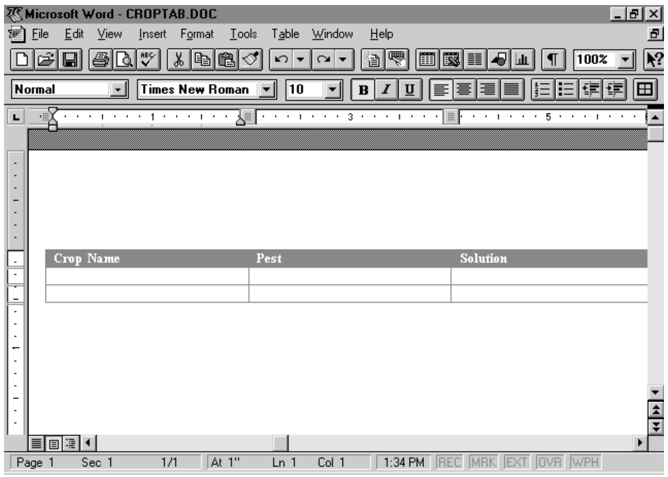
```
                              The SAS System    11:34 Wednesday, D

                         OBS    CNTYNAME    FARMS

                          1      Clark        15
```

Figure 4. OUTPUT Window Showing the Word Data

Example 3: Writing Data to a Word Document with DDE

DDE enables you not only to read Word data into your SAS programs, but also to write data from your SAS session into a Word document. In the agriculture example, suppose a template exists for a quarterly report that is sent to all farmers in the state. The Word document might consist of a table that includes several bookmarks, which are empty until a SAS program writes data into them.

Figure 5 shows the empty Word table. Each table cell has a separate bookmark (the brackets are not visible in this figure).



Crop Name	Pest	Solution

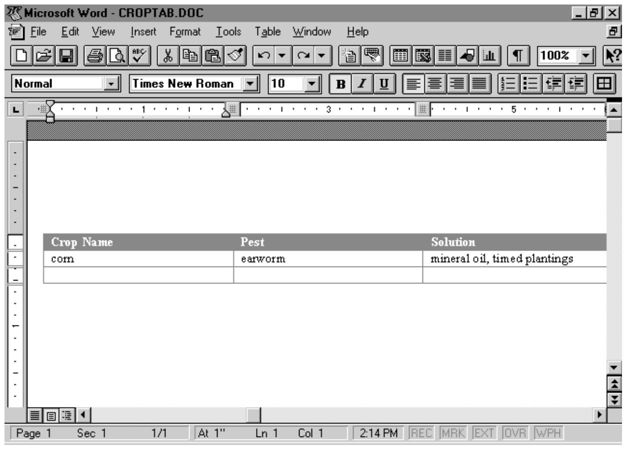
Microsoft Word - CROPTAB.DOC
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
10
100%
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 1:34 PM REC MRK EXT OVR WPH

Figure 5. Empty Word Table

Here is a program that writes a data set into the Word table. The three variables, NAME, PEST, and FIX were gained by some SAS processing of the original data and are stored in a permanent data set named CROPDATA.QTR1. Since the table contains three bookmarks, you need three FILENAME statements. For this example to work, Word must be already started and the document must already be open.

```
/* Define filerefs for each bookmark. */
filename cropname dde
   'winword|croptab.doc!CropName1'
   notab;

filename croppest dde
   'winword|croptab.doc!Pest1'
   notab;

filename solution dde
   'winword|croptab.doc!Fix1'
   notab;

/* Define a libref for the permanent SAS */
/* data library. */
libname cropdata 'c:\sas\cropdata';

/* Write the 3 variables to the 3 bookmarks. */
data _null_;
   set cropdata.qtr1;
   file cropname;
   put name;
   file croppest;
   put pest;
   file solution;
   put fix;
run;
```

Figure 6 shows the table after the example has run.


Microsoft Word - CROPTAB.DOC
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
10
100%
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 2:14 PM REC MRK EXT OVR WPH

Crop Name	Pest	Solution
corn	earworm	mineral oil, timed plantings

Figure 6. Word Table Showing Data from the SAS System

Caution: If you select the entire table cell as the bookmark, this example does not work. Instead, simply place the cursor in the table cell, then add the bookmark.

In this example, there is a one-to-one correspondence to the number of observations in the QTR1 data set and the number of rows the DDE DATA step writes to the table. If you have many rows and observations, you may want to use the macro facility and other features of the SAS System to make your code more efficient.

SENDING COMMANDS TO WORD VIA DDE

As you saw in Examples 1 through 3, using DDE to transfer data between your SAS session and Word documents requires that the Word document be open. While you can manually open the Word documents, you can also programmatically open the documents by using DDE to send a command to Word.

This use of DDE also requires a FILENAME statement and a DDE triplet, but the syntax is a little different than for data access. Here is the general form of the FILENAME statement for sending commands to an application:

FILENAME *fileref* DDE '*executable*|SYSTEM';

- *Fileref* can be any valid fileref for the SAS System.
- DDE is a keyword that tells the SAS System you are using Dynamic Data Exchange.
- *Executable* is the name of the application's executable file.
- SYSTEM is a keyword that enables you to send commands to another application.

Here is an example FILENAME statement using the SYSTEM keyword. It creates the fileref CMDS to send commands to Word (whose executable file is named winword).

```
filename cmds dde 'winword|system';
```

Once you have defined a fileref using the SYSTEM keyword, use a null DATA step and the PUT statement to send the commands to Word. The syntax of the commands is application-specific. For Word, use the Word macro language syntax (which is documented by Microsoft, and is also covered briefly in the Word online help). The basic syntax of the PUT statement containing DDE commands is as follows:

PUT '[*DDE-command*]';

Here is the skeleton code for sending DDE commands to an application:

FILENAME *fileref* DDE '*executable*|SYSTEM';
DATA _NULL_;
  FILE *fileref*;
  PUT '[*DDE-command*]';
  .
  .
  .
RUN;

Example 4: Opening a Word Document with DDE

The following example sends a Word macro command to open the ORGANIC.DOC file--but it does not start Word. Word must already be running for this example to work.

```
filename cmds dde 'winword|system';

data _null_;
   file cmds;
   put '[FileOpen.Name = "c:\organic.doc"]';
run;
```

Example 5: Saving a Word Document with DDE

If you have used DDE to write to a Word document and don't want to save it manually (from Word), you can use DDE to save the document. The following example saves the active Word document:

```
filename cmds dde 'winword|system';

data _null_;
   file cmds;
   put '[FileSave]';
run;
```

Example 6: Switching Word Documents with DDE

Any command that you send via the PUT statement applies to the active, or "top" Word document. (Remember, you can have several Word documents open at one time, but only one is active.) But you may want to send a command to a Word document that is open, but not active. Switching active documents requires that you know the order of the documents in the Window List (available from the **Window** menu choice in Word). Luckily, Word lists the files in alphabetical order, so you can figure this number out.

For example, if you have CROPTAB.DOC, MISC.TXT, APP1.DOC, and SURPRISE.DOC open, they are listed in the following order in the Window List:

1. APP1.DOC
2. CROPTAB.DOC
3. MISC.TXT
4. SURPRISE.DOC

Suppose you want MISC.TXT to be the active document. The following program makes MISC.TXT active:

```
filename cmds dde 'winword|system';

data _null_;
   file cmds;
   put '[WindowList 3]';
run;
```

Example 7: Closing a Word Document with DDE

As you might expect, closing a Word document with DDE is simply a matter of sending the right command with the PUT statement. The following example closes the active Word document:

```
filename cmds dde 'winword|system';

data _null_;
   file cmds;
   put '[FileClose]';
run;
```

Example 8: Closing the Word Application with DDE

As with opening and closing documents, shutting Word down via DDE uses the PUT statement with the appropriate command. The following example closes Word:

```
filename cmds dde 'winword|system';

data _null_;
   file cmds;
   put '[FileExit]';
run;
```

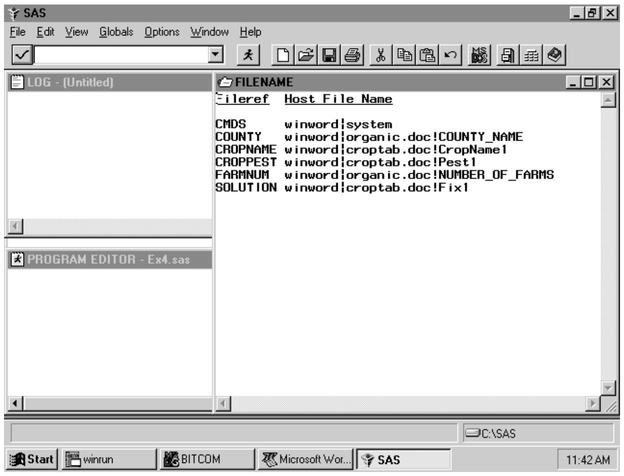
TIPS AND TECHNIQUES

While working with the SAS System, DDE, and Word, you may find the following tips helpful.

Keeping Track of Filerefs

If you have many bookmarks, and therefore many FILENAME statements, use the FILENAME window to keep track of your filerefs. This window shows filerefs and their associated physical filenames, which helps you remember what fileref points where.

Issue the FILENAME command from the SAS System's Command bar to open the FILENAME window. Figure 7 shows a sample FILENAME window.



```
Fileref  Host File Name

CMDS     winword|system
COUNTY   winword|organic.doc!COUNTY_NAME
CROPNAME winword|croptab.doc!CropName1
CROPPEST winword|croptab.doc!Pest1
FARMNUM  winword|organic.doc!NUMBER_OF_FARMS
SOLUTION winword|croptab.doc!Fix1
```

Figure 7. Sample FILENAME Window

Opening the Bookmark Dialog Box Quickly

To quickly open the Bookmark dialog box in Word, press CTRL-SHF-F5.

Switching Applications Efficiently

You may find it useful to be able to see your SAS session and your Word document at the same time. Use your mouse to adjust the window sizes so the two applications can share the screen.

Also, you can use ALT-TAB to toggle between the two most-recently-used Windows applications. If you've been using the SAS System and Word, ALT-TAB is a quick way to switch between these two applications.

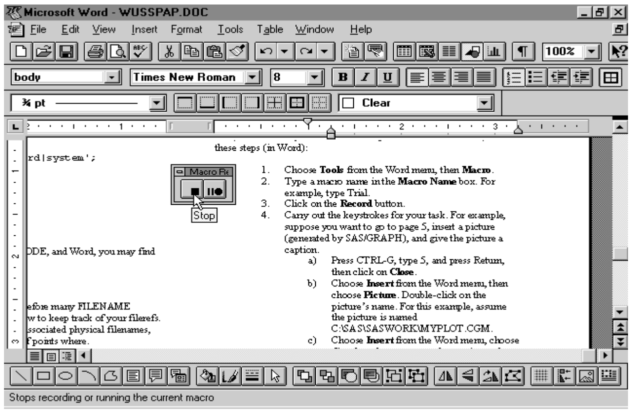
If you are using Windows 95, you can use the Task Bar to quickly move to another application.

Figuring Out the Word Macro Commands

You don't have to be a Word macro wizard to use the SYSTEM keyword to send commands to Word. Instead, use the Word macro recorder and macro editing features to discover what the commands are.

First, record the keystrokes that accomplish the task. To do this, follow these steps (in Word):

1. Choose **Tools** from the Word menu, then **Macro**.
2. Type a macro name in the **Macro Name** box. For example, type `Trial`.
3. Click on the **Record** button.
4. Carry out the keystrokes for your task. For example, suppose you want to go to page 5, insert a picture (generated by SAS/GRAPH), and give the picture a caption.
    a) Press CTRL-G, type 5, and press Return, then click on **Close**.
    b) Choose **Insert** from the Word menu, then choose **Picture**. Double-click on the picture's name. For this example, assume the picture is named C:\SAS\SASWORK\MYPLOT.CGM.
    c) Choose **Insert** from the Word menu, choose **Caption**, then type type the caption and press Return.
5. Now stop recording by clicking on the **Stop** button on the macro recorder icon. Figure 8 shows the macro recorder icon with the mouse pointer positioned over the **Stop** button.


Microsoft Word - WUSSPAP.DOC
File Edit View Insert Format Tools Table Window Help
100%
body
Times New Roman
8
¾ pt
Clear
Macro Re
Stop

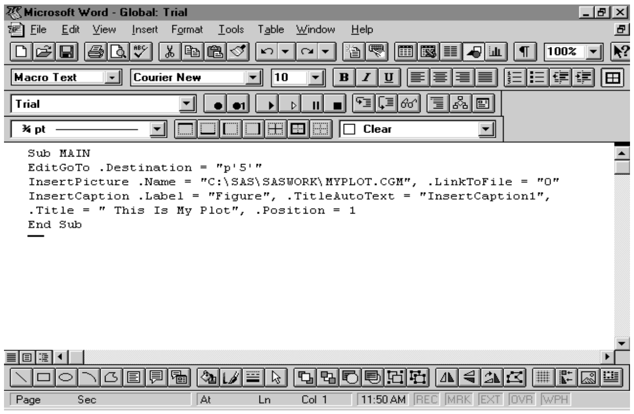
Stops recording or running the current macro

Figure 8. The Word Macro Recorder Stop Button

To see what commands the keystrokes generated, edit the macro by following these steps:

1. Choose **Tools** from the Word menu, then **Macro**.
2. Click on the name of the macro you just recorded (in this case, **Trial**).
3. Click on the **Edit** button.

The macro file opens, showing the macro commands generated by the keystroke sequence you followed. For example, Figure 9 shows the Trial macro.


Microsoft Word - Global: Trial
File Edit View Insert Format Tools Table Window Help
Macro Text
Courier New
10
Trial
Clear
Page Sec At Ln Col 1 11:50 AM REC MRK EXT OVR WPH

```
Sub MAIN
EditGoTo .Destination = "p'5'"
InsertPicture .Name = "C:\SAS\SASWORK\MYPLOT.CGM", .LinkToFile = "0"
InsertCaption .Label = "Figure", .TitleAutoText = "InsertCaption1",
.Title = " This Is My Plot", .Position = 1
End Sub
```

Figure 9. Looking at a Word Macro

You are interested in the lines between the `Sub MAIN` and `End Sub` lines--these are the Word macro commands. For example, to go to page 5, the command is `EditGoTo.Destination = "p'5'"`. The insertion of the picture and its caption has a more complex syntax.

Once you see what commands you need, you can cut and paste these commands into PUT statements in your DDE programs to automate Word document manipulation.

Starting Word Programmatically

Although you can use DDE to shut Word down, you cannot use DDE to start Word (because DDE requires an application to be open). However, it is often useful to have your SAS program start Word instead of having to start it manually. You can add an X statement to your SAS program, before your DDE code, that starts Word.

The X statement sends commands to the operating system, and has the following general form:

X O*perating-system-command*;

For example, the following X statement starts Word:

```
x c:\winword\winword.exe;
```

Because it takes Word a while to start up, you may also want to use the SLEEP function in your DATA step, to be sure Word is ready before you start reading and writing data. For example, the following DATA step pauses SAS execution for 5 seconds:

```
data _null_;
   x = sleep(5);
.
. DDE-statements and other code
.
run;
```

Considerations for Batch Mode

The nice thing about DDE is that it enables you to do programmatically what you normally have to do manually. This makes DDE a natural choice for batch mode. Using the X statement and the SYSTEM keyword in the FILENAME statement, your DDE SAS program can run in batch mode with no problem.

A final technique is to have your SAS program run from DOS: it starts Windows, opens Word documents and manipulates them, then when it's done it shuts down Word and, at the end, closes Windows and returns you to the DOS prompt.

Note: This technique is useful only in the Windows 3.1 enviornment--you should not use it if you running SAS under Windows 95 or Windows NT.

To accomplish this, provide the name of the SAS executable file as an argument to the WIN command at the DOS prompt, along with other arguments. Here is an example:

```
WIN C:\SAS\SAS.EXE DDE_1.SAS -CONFIG
  C:\SAS\CONFIG.SAS -EXITWINDOWS
```

- WIN starts Windows.
- C:\SAS\SAS.EXE stipulates that Windows run the SAS System.
- DDE_1.SAS is the name of the program you want the SAS System to execute.
- -CONFIG C:\SAS\CONFIG.SAS gives the location of your SAS configuration file.
- -EXITWINDOWS tells the SAS System to shut down Windows and return control to DOS when the SAS program is finished.

Note: The WIN command must be all on one line when you type it at the DOS prompt.

CONCLUSION

The simple examples presented in this paper have barely scratched the surface of the power of DDE. Much more complex data transfer is possible using more advanced features of the DATA step, SAS macro language, and DDE. Also, any macro command that Word supports can be sent via DDE, so you have extremely detailed control over your Word documents from your SAS session. DDE enables you to harness the data-sharing capabilities of Windows applications, and is a handy tool for any Windows-based SAS programmer.

The author can be reached at: Jodie Gilmore
32601 NE Ammeter Rd.
Washougal, WA 98671
jgilmor@pacifier.com