Paper 45-30

A Matter of Presentation: Generating PowerPoint Slides from Base® SAS using Dynamic Data Exchange

Koen Vyverman, SAS Netherlands

ABSTRACT

The creation of PowerPoint slides with SAS content using DDE in a Base SAS environment has long been considered impossible. Unlike MS Word and MS Excel, the PowerPoint application does not come with a scripting language like WordBasic or the Excel 4 macro language that would allow DDE to talk to it in a client/server fashion. The job can be done, though, by using DDE to Excel as an intermediate agent to pull the strings of PowerPoint. A set of easy SAS macros is introduced to perform a number of basic PowerPoint operations. As a sample application, a SAS catalog of graphs is exported to a stand-alone PowerPoint presentation. No specific technical knowledge is required from the reader, at least not beyond a basic understanding of the SAS macro language. A slight degree of familiarity with DDE to Excel concepts should prove enlightening, though.

INTRODUCTION

Quite a lot has been published already about interacting with MS Excel workbooks and MS Word documents from SAS by using the DATA step and Dynamic Data Exchange (DDE): the TS325 document (SAS Institute, 1999) offers a general introduction; a SUGI (Vyverman, 2002) and a SESUG (Vyverman 2003) paper dive into the depths of DDE from/to Excel; another SUGI paper (Viergever & Vyverman, 2003) explores the wonderful world of Word via DDE; and an intriguing applications development paper (Vyverman, 2005) (in these SUGI 30 proceedings) focuses more on certain practical applications of DDE, rather than on the technical aspects. So all this time, the missing part of the MS Office puzzle has been the PowerPoint application.

Why has no one ever written a paper about DDE and PowerPoint? Because the PowerPoint application is not DDE-compliant. Word has the old WordBasic scripting language, Excel still recognizes the Excel 4.0 macro language (X4ML), and both are precursors to the later Visual Basic macro language in the MS Office suite. Both Word and Excel can accept commands from SAS via a DDE fileref. Unfortunately, PowerPoint has no similar tools for applications developers, and so it was long assumed that creating PowerPoint slides straight from a piece of SAS code was impossible.

As you can see from this paper, with a little tinkering, you can create a workaround. Like most workarounds, it is not very aesthetically pleasing, but it works: PowerPoint slides with SAS content can be created from a Base SAS program. The trick is to talk from SAS to the Excel application via DDE and include PUT statements that cause Excel to send the necessary keystrokes to the PowerPoint application.

This paper takes you through a small, yet practical, example of this workaround. First, we set up some data and graphics, and then we start up Excel and PowerPoint from SAS. Finally, we present a DATA step with PUT statements and explain, step-by-step, how the appropriate keystrokes are sent from Excel to PowerPoint.

The example code was developed and tested on a Microsoft Windows 2000 Professional operating environment, running SAS 9.1 in conjunction with MS Office 2000, English version. The language is significant: Excel and PowerPoint keystrokes might vary with different languages and different versions of the software. This paper merely explains the concept. You might need to modify the code for your operating environment. For more information about DDE, see the introductory parts of the paper (Vyverman, 2002).

THE PROBLEM

A common topic on the SAS-L mailing list is this: how to generate business graphs and charts in SAS and automatically put them into PowerPoint slides. The usual advice is to create the presentation in PowerPoint and give links to the images (that is, to the charts you already output from SAS or to blank placeholders). When you generate new charts with SAS, you overwrite the previous files with identically named files, and the next time the presentation is opened, the new images appear. This works fine if the number of charts in the presentation is always the same. If not, further manual work on the presentation is required, either to remove unused slides or to add slides. The challenge is to automate the PowerPoint presentation from SAS and eliminate most of the manual work in PowerPoint.

PREPARATIONS TO EXPORT SAS GRAPHICS TO POWERPOINT

To prepare for our example, we create a SAS catalog that contains a number of GRSEG entries, export those entries to JPEG images, start Excel from SAS, and start PowerPoint from SAS.

DATA PREPARATIONS

For starters, we create a small data set as follows:

```
data wine_consumption;
  length
    year           8
    grape       $ 10
    consumption    8
    ;
  label
    year='Year'
    grape='Grape'
    consumption='Consumption'
    ;
  infile datalines4 dlm='#';
  input
    year        :8.
    grape       :$char10.
    consumption :8.
    ;
  datalines4;
    2002 #Shiraz      #  9
    2002 #Zinfandel   # 18
    2002 #Chardonnay  #  6
    2003 #Shiraz      # 10
    2003 #Zinfandel   # 16
    2003 #Chardonnay  #  8
    2004 #Shiraz      # 10
    2004 #Zinfandel   #  9
    2004 #Chardonnay  # 15
;;;;
run;
```

For three consecutive years, the data set `WORK.WINE_CONSUMPTION` lists the number of bottles imbibed for three types of grapes. We then produce a bar chart for each year, with the grape type as categories on the horizontal axis:

```
proc sort data=wine_consumption;
  by year;
run;

proc gchart data=wine_consumption
            gout=work.sasgraphs
            ;
  by year;
  vbar grape / sumvar=consumption
               sum
               discrete
               autoref
               clipref
               frame
               missing
               name="graph"
               width=20
               coutline=black;
  run;
quit;
```

This produces three GRSEG entries in a SAS/GRAPH catalog, WORK.SASGRAPHS. The entries are GRAPH (2002 data), GRAPH1 (2003 data), and GRAPH2 (2004 data). We turn these entries into JPEG files with the following code:

```
goptions
  reset=all
  border
  device=jpeg
  gaccess=gsasfile
  xpixels=950
  ypixels=550
  cback=cxfff7ce
  ;

proc greplay nofs igout=work.sasgraphs;
  filename gsasfile 'D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2002.jpg';
  replay graph;
  run;
  filename gsasfile clear;
  filename gsasfile 'D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2003.jpg';
  replay graph1;
  run;
  filename gsasfile clear;
  filename gsasfile 'D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2004.jpg';
  replay graph2;
  run;
  filename gsasfile clear;
quit;
```

EXCEL PREPARATIONS

Because DDE is a client/server protocol, the Excel application must be running. We start Excel with the method introduced by Roper (2000):

```
options
  noxsync
  noxwait
  xmin
  ;

filename sas2xl dde 'excel|system';

data _null_;
  length fid rc start stop time 8;
  fid=fopen('sas2xl','s');
  if (fid le 0) then do;
    rc=system('start excel');
    start=datetime();
    stop=start+10;
    do while (fid le 0);
      fid=fopen('sas2xl','s');
      time=datetime();
      if (time ge stop) then fid=1;
      end;
    end;
  rc=fclose(fid);
run;
```

Now, we must define a macro that will be called later. While troubleshooting some errors, we found a critical point—right after activating the PowerPoint application from Excel—where it is necessary to briefly pause Excel before it is safe to begin sending keystrokes. We don't know why a pause is needed, but if it is not done, nothing works.

To achieve a one-second pause, we define a fairly trivial Excel macro. First, we add an old-style X4ML macro sheet to the DDE fileref SAS2XL:

```
data _null_;
  file sas2xl;
  put '[error(false)]';
  put '[workbook.insert(3)]';
run;
```

The default name of the inserted macro sheet is `Macro1`. We define a cell range on the `Macro1` sheet as a DDE triplet-style fileref `xlmacro`, and write the code there for the macro to pause Excel:

```
filename xlmacro dde "excel|macro1!r1c1:r10c1" notab lrecl=200;

data _null_;
  file xlmacro;
  put '=wait(now()+"00:00:01")';
  put '=halt(true)';
run;

filename xlmacro clear;
```

We determined the one-second waiting period by trial-and-error. Conceivably, a longer pause might be necessary in your operating environment. If you get any errors, you can try modifying the above to `00:00:05` or more.

POWERPOINT PREPARATIONS

Finally, we start the PowerPoint application:

```
data _null_;
  rc=system('start powerpnt');
  rc=sleep(2);
run;
```

Here, `powerpnt` is the name of the PowerPoint executable file, as installed in C:\Program Files\Microsoft Office\Office. The SAS session is put to sleep for two seconds, to allow PowerPoint to start up. As before, it might be necessary to increase the number of seconds. Also, observe that you cannot use the elegant Roper method here: PowerPoint is not a DDE-compliant server, so it will not trigger an escape from the Roper DO WHILE loop after it has been successfully launched. This is a bit of a pain, forcing us to revert to the old method of putting the SAS session to sleep for a while. The Roper method would actually work in terms of firing up PowerPoint, but the DO WHILE loop would continue executing until the maximum waiting time has elapsed. So you're better off determining an optimal `SLEEP` value by means of a few trials.

For this example, we performed an additional, one-time step in PowerPoint. We created a custom template, 'Template for Graphics Catalog.pot', with fewer text boxes than the default templates. This reduces the lines of SAS code needed to demonstrate the technique.

EXCEL X4ML KEYSTROKES FOR POWERPOINT

With the SAS data and graphics ready, and with Excel and PowerPoint ready for directions, we can make Excel talk to PowerPoint. Each `SEND.KEYS` X4ML function is wrapped in a PUT statement to send an individual keystroke, or a combination of keystrokes, from Excel to PowerPoint. The keystrokes work through all the dialog boxes to create a PowerPoint presentation, to apply an existing design template, to complete text boxes for titles, and to add SAS graphics.

This example code is repetitive and could be simplified by using SAS macros to generate keystrokes dynamically. That, however, is left as an exercise for you. To modify the example code, see Table 1 for frequently used keyboard keystrokes and their representation in the argument string of the `SEND.KEYS` function.

Key	Code
BACKSPACE	"{BACKSPACE}" or "{BS}"
BREAK	"{BREAK}"
CAPS LOCK	"{CAPSLOCK}"
CLEAR	"{CLEAR}"
DELETE or DEL	"{DELETE}" or "{DEL}"
DOWN	"{DOWN}"
END	"{END}"
ENTER (numeric keypad)	"{ENTER}"
ENTER	"~" (tilde)
ESC	"{ESCAPE} or {ESC}"
HELP	"{HELP}"
HOME	"{HOME}"
INS	"{INSERT}"
LEFT	"{LEFT}"
NUM LOCK	"{NUMLOCK}"
PAGE DOWN	"{PGDN}"
PAGE UP	"{PGUP}"
RETURN	"{RETURN}"
RIGHT	"{RIGHT}"
SCROLL LOCK	"{SCROLLLOCK}"
TAB	"{TAB}"
UP	"{UP}"

Table 1. Single Keystrokes

Furthermore, to emulate the pressing of keys in combination with the ALT, SHIFT, CTRL, and command keys, key-codes can be preceded by certain modifiers. See Table 2.

To combine Key with...	Precede Code with...
SHIFT	"+" (plus sign)
CTRL	"^" (caret)
ALT or OPTION	"%" (percent sign)
COMMAND	"*" (asterisk)

Table 2. Combination Keystrokes

EXAMPLE CODE

Each numbered line is explained following the code.

```
data _null_;
  file sas2xl;
  put '[app.activate("microsoft excel - book1")]';
  put '[app.activate("microsoft powerpoint")]';01
  put '[run("macro1!r1c1")]';02
  put '[send.keys("{tab}",true)]'03
      '[send.keys("{tab}",true)]'
      '[send.keys("{tab}",true)]'
      '[send.keys("{return}",true)]'
      '[send.keys("^{n}",true)]'04
      '[send.keys("{return}",true)]'
      '[send.keys("%oy",true)]'05
      '[send.keys("Template for Graphics Catalog",true)]'
      '[send.keys("{return}",true)]'
      '[send.keys("{tab}",true)]'06
      '[send.keys("{return}",true)]'
      '[send.keys("Wine Consumption Statistics",true)]'
      '[send.keys("^{return}",true)]'07
      '[send.keys("Name: Kilo Volt",true)]'
      '[send.keys("{return}",true)]'
```

```
'[send.keys("Company: Dinosaurs R Us",true)]'
'[send.keys("^{m}",true)]'08
'[send.keys("{right}",true)]'
'[send.keys("{down}",true)]'
'[send.keys("{down}",true)]'
'[send.keys("{return}",true)]'
'[send.keys("{tab}",true)]'09
'[send.keys("{return}",true)]'
'[send.keys("2002 Number of Bottles",true)]'
'[send.keys("%ipf")]'10
'[send.keys("D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2002.jpg")]'
'[send.keys("{return}")]'
'[send.keys("%oi")]'11
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("{tab}")]'
'[send.keys("0.5")]'
'[send.keys("{tab}")]'12
'[send.keys("{tab}")]'
'[send.keys("1.40")]'
'[send.keys("^{pgdn}")]'13
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("{tab}")]'
'[send.keys("{tab}")]'
'[send.keys("{tab}")]'
'[send.keys("90")]'
'[send.keys("{return}")]'
'[send.keys("^{m}",true)]'14
'[send.keys("{return}",true)]'
'[send.keys("{tab}",true)]'
'[send.keys("{return}",true)]'
'[send.keys("2003 Number of Bottles",true)]'
'[send.keys("%ipf")]'
'[send.keys("D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2003.jpg")]'
'[send.keys("{return}")]'
'[send.keys("%oi")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("{tab}")]'
'[send.keys("0.5")]'
'[send.keys("{tab}")]'
'[send.keys("{tab}")]'
'[send.keys("1.40")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("^{pgdn}")]'
'[send.keys("{tab}")]'
'[send.keys("{tab}")]'
'[send.keys("{tab}")]'
'[send.keys("90")]'
'[send.keys("{return}")]'
'[send.keys("^{m}",true)]'15
'[send.keys("{return}",true)]'
'[send.keys("{tab}",true)]'
'[send.keys("{return}",true)]'
'[send.keys("2004 Number of Bottles",true)]'
'[send.keys("%ipf")]'
'[send.keys("D:\Koen\SAS\SUGI30 Paper Corders Corner\Pictures\2004.jpg")]'
'[send.keys("{return}")]'
'[send.keys("%oi")]'
```

```
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("{tab}")]'
        '[send.keys("0.5")]'
        '[send.keys("{tab}")]'
        '[send.keys("{tab}")]'
        '[send.keys("1.40")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("^{pgdn}")]'
        '[send.keys("{tab}")]'
        '[send.keys("{tab}")]'
        '[send.keys("{tab}")]'
        '[send.keys("90")]'
        '[send.keys("{return}")]'
        '[send.keys("%fa")]'16
        '[send.keys("D:\Koen\SAS\SUGI30 Paper Corders Corner\Output\PPT from
       SAS.ppt")]'
        '[send.keys("{return}")]'
        '[send.keys("%fx")]'17
        ;
run;
```

DETAILS

01 When using the SEND.KEYS X4ML function in Excel, keystrokes are sent to the active application. Therefore, to ensure that we send keystrokes to PowerPoint, we activate the PowerPoint application.

02 As explained in the section “Preparations to Export SAS Graphics to PowerPoint”, by trial-and-error, it was discovered that we need to pause the Excel application briefly at this point. So we use the RUN function to execute the Excel macro that was prepared previously, and Excel sleeps for a second.

03 Now we begin to send keystrokes to PowerPoint. Note that the SEND.KEYS function has a second argument. It is a Boolean value that, when specified as TRUE, causes Excel to wait for the keys to be processed before continuing. We added the Boolean control to some of the keystrokes and not to others. Our empirical criterion was: Does it work? If not, add TRUE. See Tables 1 and 2 for SEND.KEYS values.

What we do here in the code is equivalent to pressing the TAB key three times and the RETURN key one time. This is done to get rid of the dialog box that popped up when PowerPoint was launched (Figure 1). The dialog box prompts the user to create a new presentation or to open an existing one. We want nothing of it. Three TABs and a RETURN is the same as clicking the Cancel button. Note that this dialog box can be suppressed by default (at least in PowerPoint 2000), so this step might not be necessary.

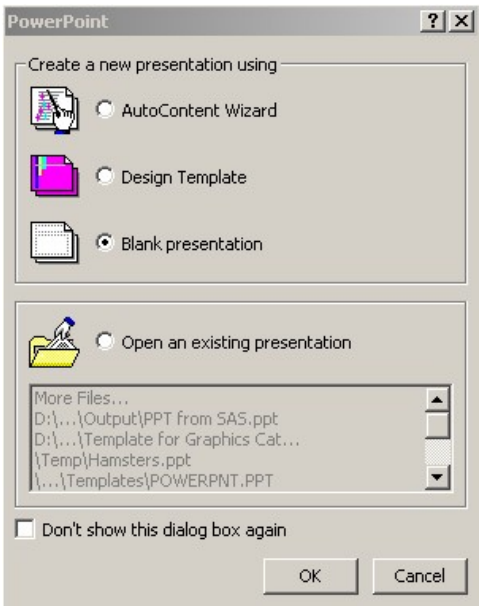

Figure 1. Three TABs and a RETURN Cancel the Startup Dialog Box

04 With the dialog box gone, we perform a CONTROL-N. This creates a new presentation and brings up a dialog box that prompts us to choose a layout for the first slide (Figure 2). Because the default layout is a Title Slide, we merely send a RETURN key to select it. The result is shown in Figure 3.

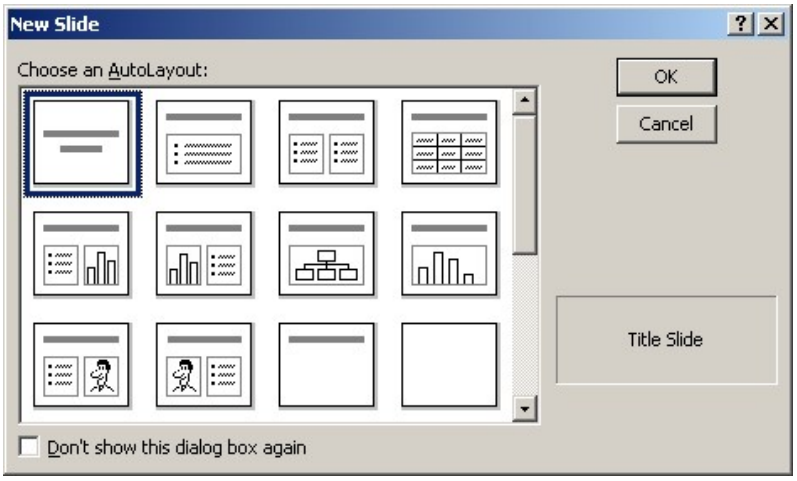

Figure 2. The First New Slide Dialog Box, with the Title Slide Layout Selected by Default

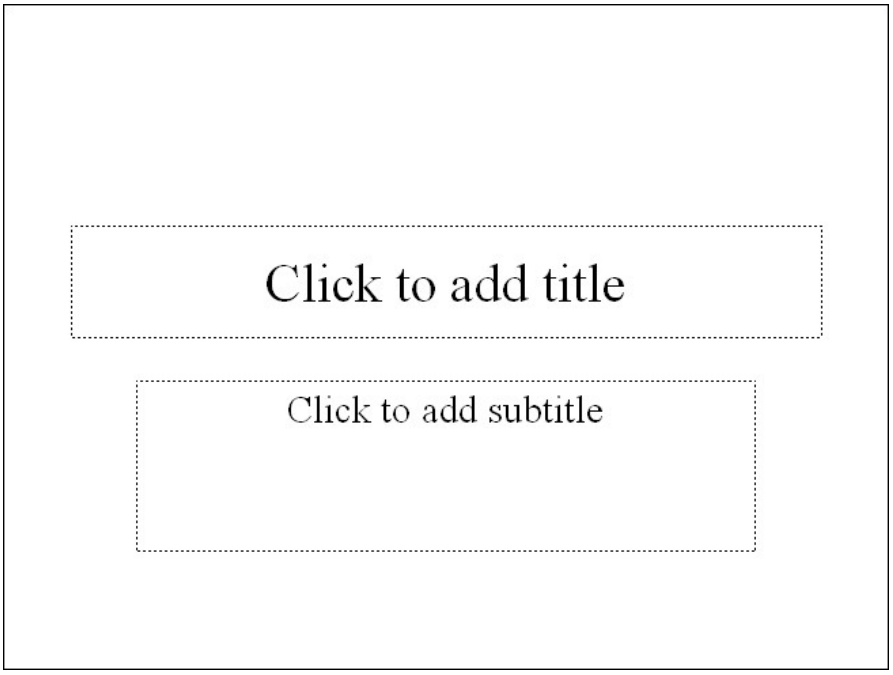

Figure 3. Blank Presentation with a Single Slide of the Title Slide Layout

05 The ALT-O-Y keystroke brings up the dialog box to apply a design template (Figure 4) to the current presentation. For the purpose of exporting a SAS/GRAPH catalog, the next line of code selects a special design template, 'Template for Graphics Catalog.pot'. This template was built manually in PowerPoint, and tailored to suit the problem of exporting SAS graphics to PowerPoint. It consists of a title slide layout and a main slide layout. The title slide layout has two text boxes. The main slide layout has a single text box for a caption or chart title and lots of empty space for a graphic element. The design template is stored in one of the locations where PowerPoint usually stores templates. These locations might vary across systems. To find them, Windows searches for files with the PowerPoint Template extension .pot. The result of applying the design template to our presentation is shown in Figure 5.

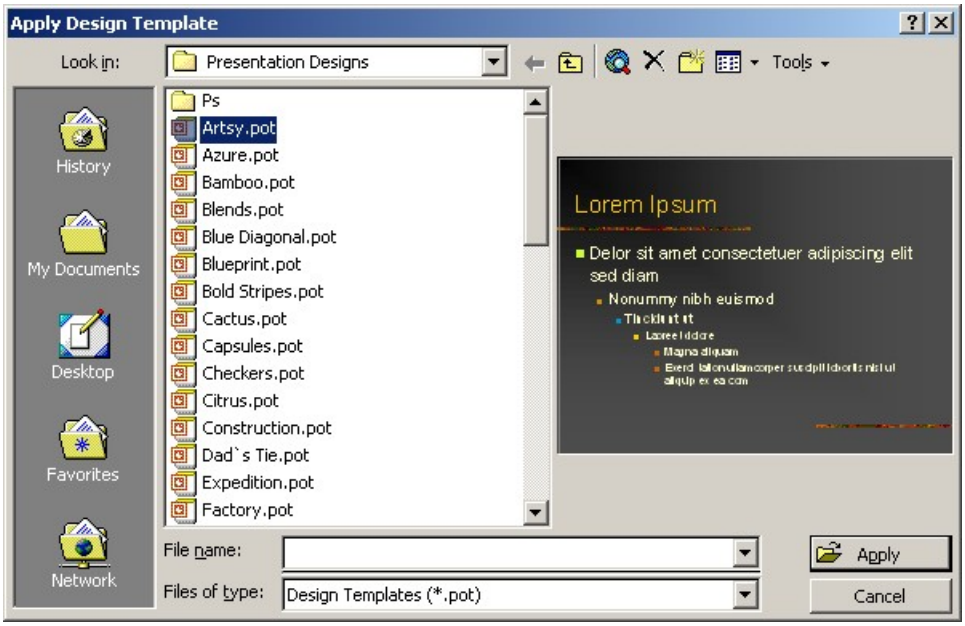

Figure 4. The Apply Design Template Dialog Box

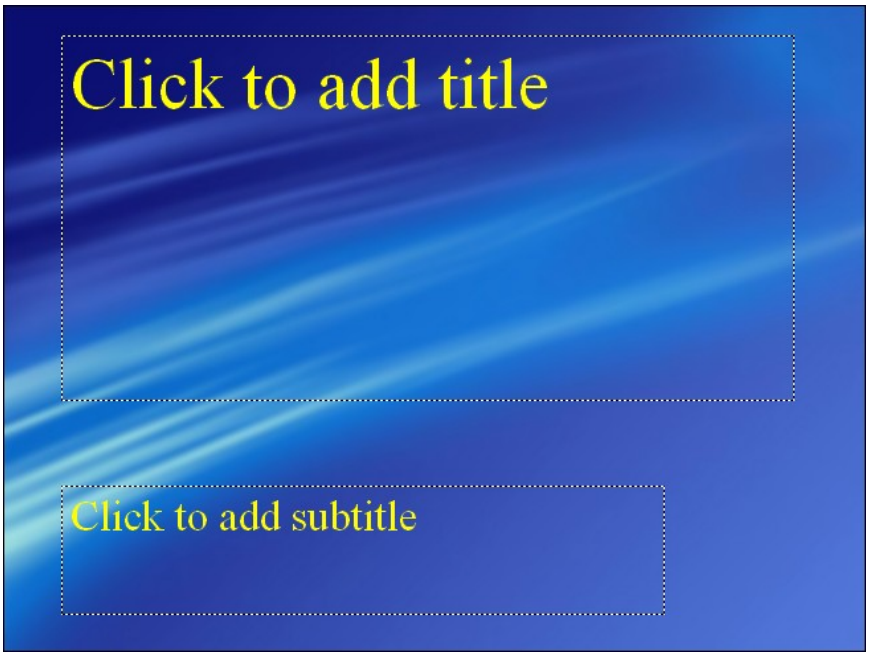

Figure 5. Results of Applying the Design Template to the New Presentation

06 A TAB followed by a RETURN highlights the contents of the first text box on the title slide—where it says 'Click to add title'. Sending a text string replaces the contents of the text box.

07 The CTRL-RETURN combination keystroke moves the focus to the next text box on the slide, and highlights the contents. Sending a text string replaces the contents 'Click to add subtitle'. Pressing RETURN creates a new line within the text string. The result of this step is shown in Figure 6. Our title slide is now complete.

Figure 6. The Title Slide after Customization

08 CTRL-M brings up the New Slide dialog box (Figure 7). Notice that the second layout is highlighted now. PowerPoint remembers that a Title Slide was already used, and now proposes a Bulleted List slide. For the next slides in our presentation, however, we want a slide layout of Title Only. A right-arrow and two down-arrow keystrokes bring us there. A RETURN adds a new slide with the chosen layout.

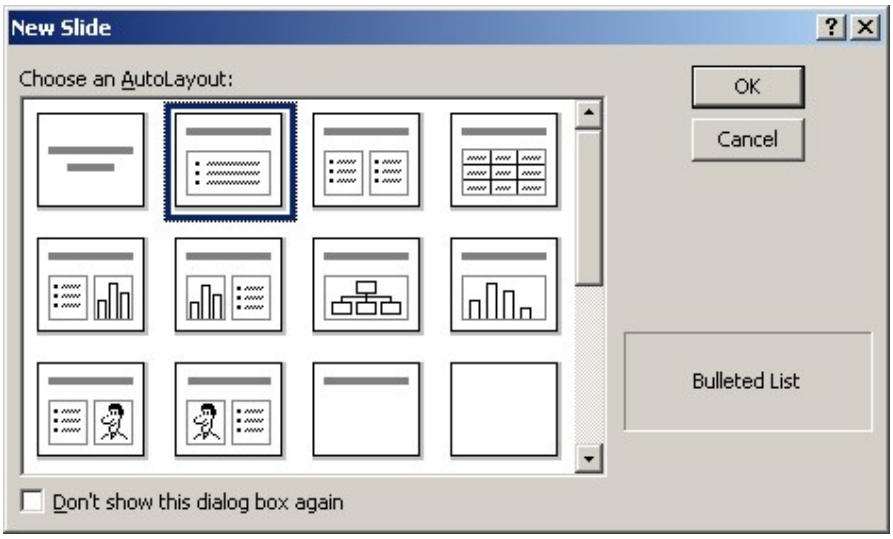

Figure 7. The New Slide Dialog Box Revisited

09 As in point 6, a TAB, a RETURN, and some text customize the content of the text box on the slide. The result is shown in Figure 8.

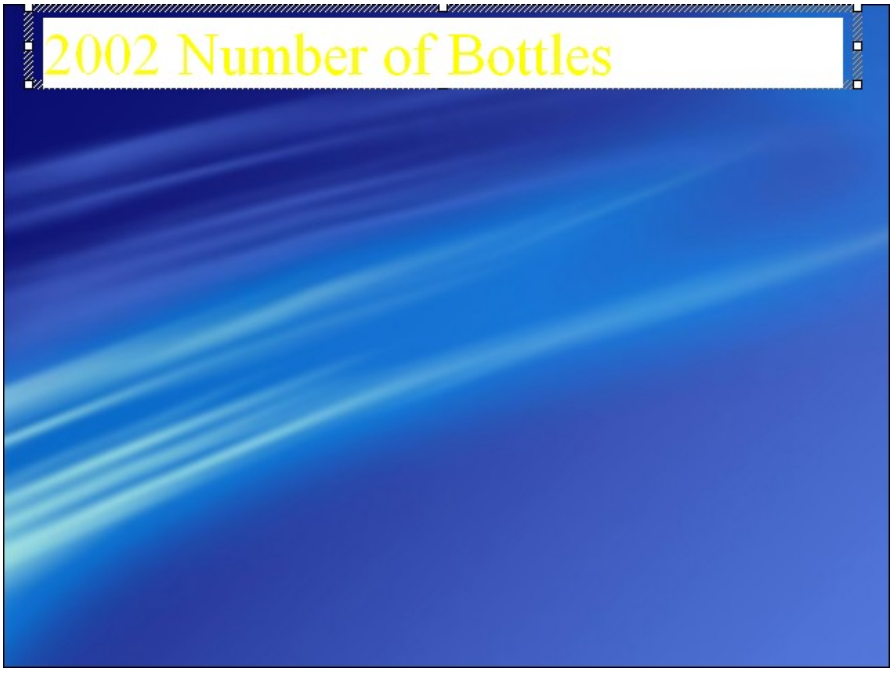

Figure 8. The Text-Box on the First Chart after Customization

10 We use an ALT-I-P-F keystroke to bring up the Insert Picture dialog box (Figure 9). Then we select the first SAS chart that we created earlier. Spelling out the full directory path in the File Name field locates the JPEG image. The result of this action is shown in Figure 10.

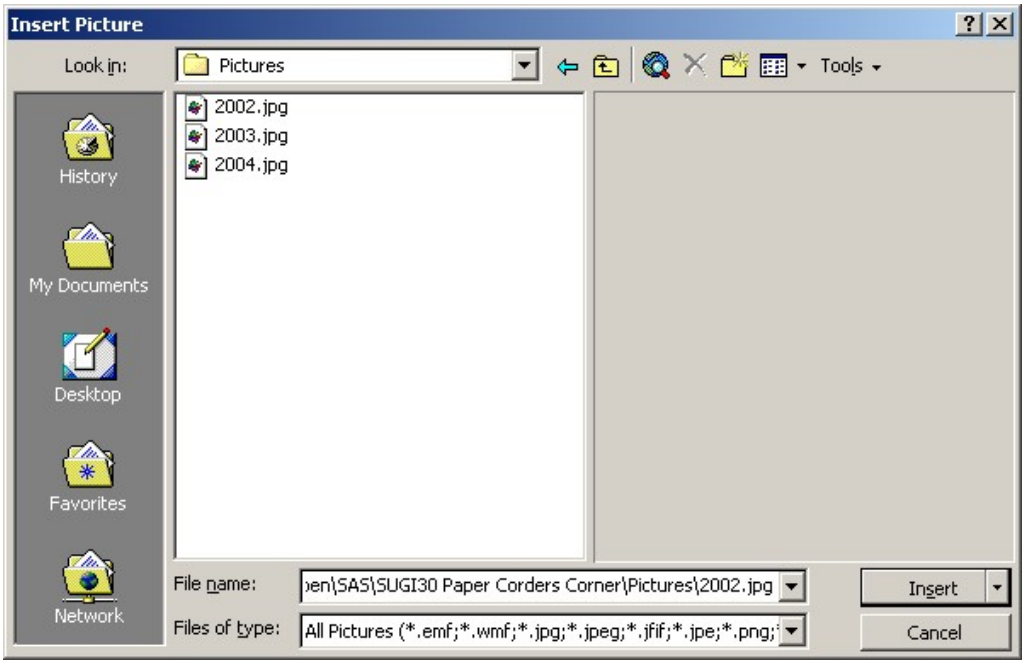

Figure 9. The Insert Picture Dialog Box

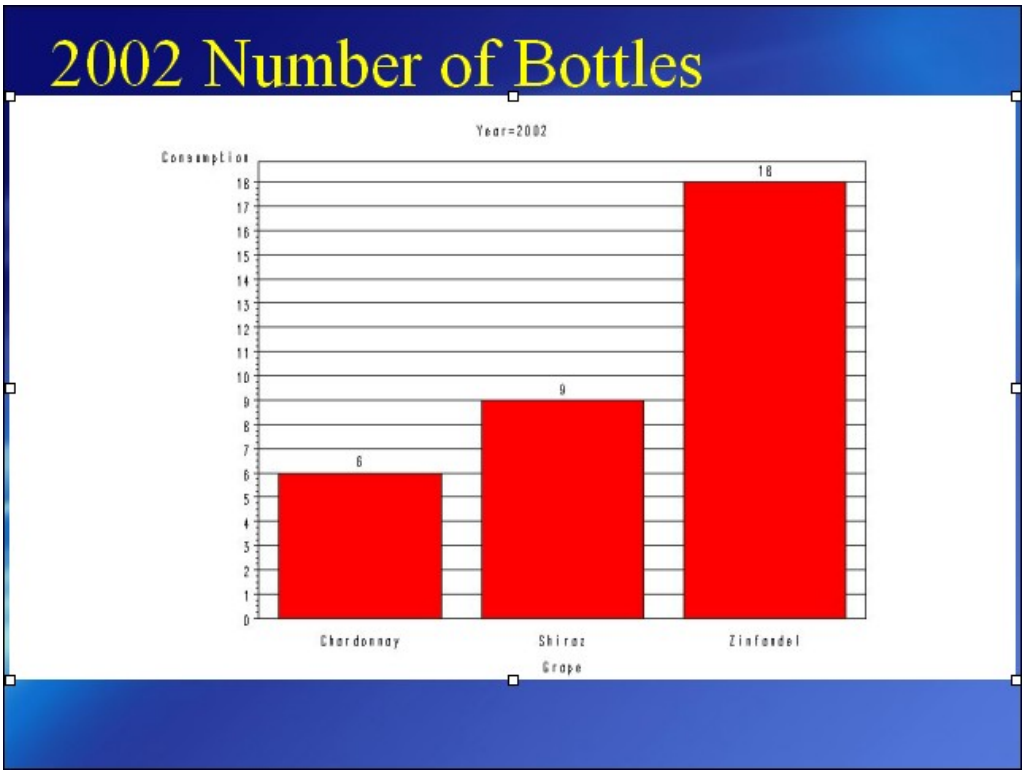

Figure 10. The Result of Inserting a JPEG Image

11 The inserted image is a bit too large, and not ideally positioned on the slide. Using an ALT-O-I keystroke brings up the Format Picture dialog box, as shown in Figure 11. The Picture tab is the active tab, by default. In order to get to the Position tab, we need four CTRL-PAGEDOWN keystrokes. A TAB keystroke takes us to the Horizontal field, where we enter 0.5.

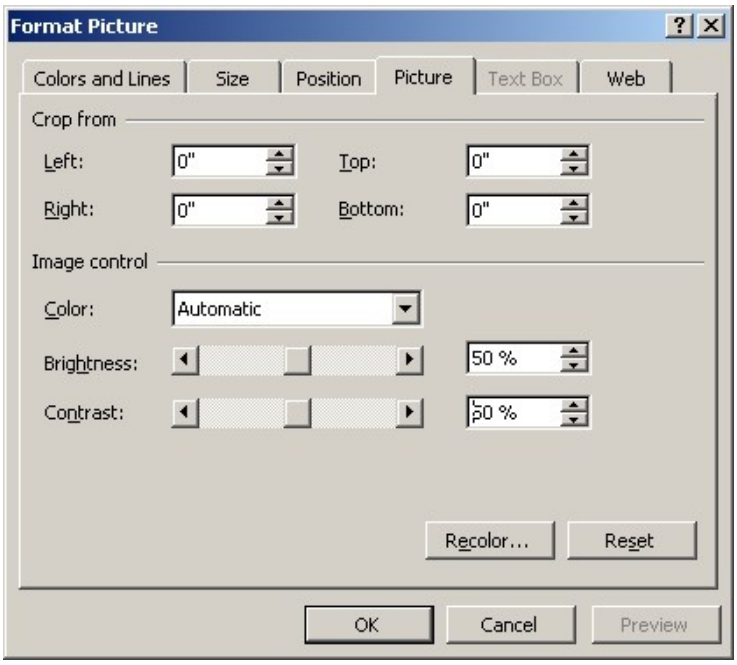

Figure 11. The Format Picture Dialog Box as It Initially Appears

12 Two more TAB keystrokes take us to the Vertical field, and we enter 1.40. Figure 12 shows the Format Picture dialog box at this stage.

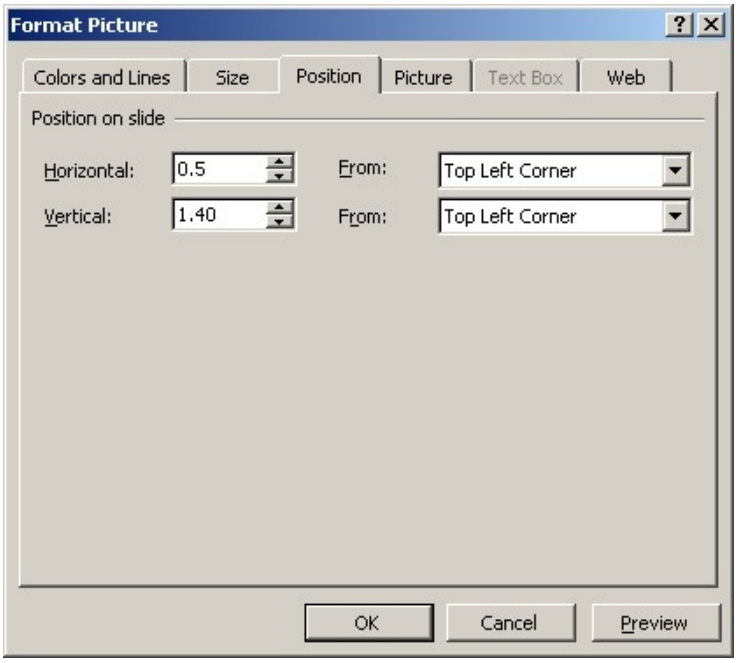

Figure 12. The Format Picture Dialog Box after Customization in the Position Tab

13 In similar fashion, to adjust the size of the inserted image, we use CNTRL-PAGEDOWN keystrokes to move to the Size tab, and we use some TAB keystrokes to set the scaling factor to 90% of the original. After pressing RETURN to confirm the new settings, our first picture slide is now complete (Figure 13).

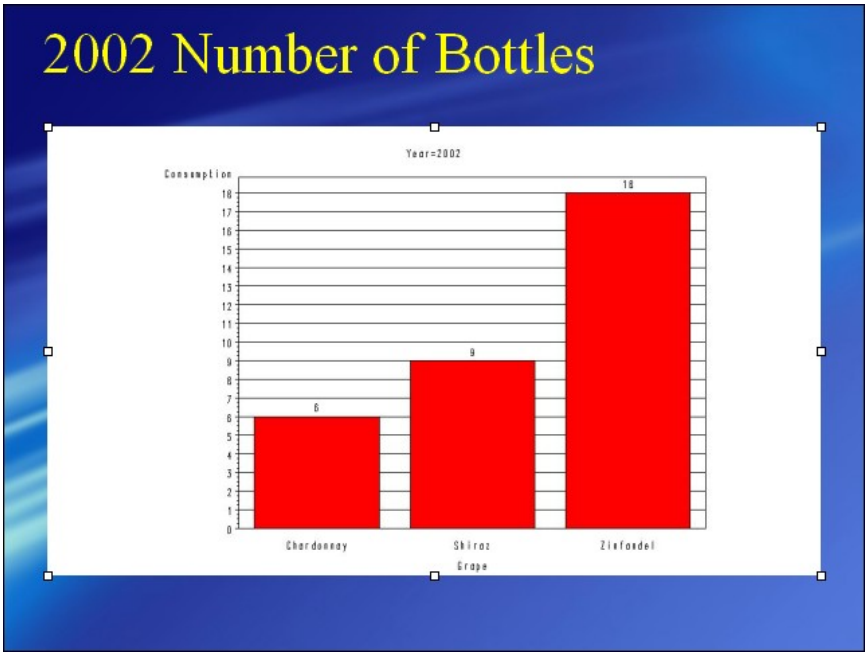

Figure 13. The Inserted Picture, Re-Sized and Re-Positioned

14 **15** As in point 8, we add a new slide using the New Slide dialog box. Note that there's no need for the directional arrow keys now. PowerPoint has remembered that previously we used the Title Only layout and now highlights that same layout. So a simple RETURN suffices to add another Title Only slide. We insert and format the second and third JPEG image in the same way as we did the first image.

16 ALT-F-A brings up the Save As dialog box. We save our finished presentation by writing a full path into the File Name field.

17 Finally, ALT-F-X ends the PowerPoint application.

CONCLUSION

The method shown here to create PowerPoint presentations from a DATA step program in Base SAS is not the most elegant code you could write. It is probably unfit for use in a heavy-duty production environment; but then, so are most DDE-based applications, really. However, if you run a small organization that uses SAS, you can adapt this technique to save time on manual cut-and-paste work, for example, to produce daily management presentations.

REFERENCES

Roper, C. A. "Intelligently Launching Microsoft Excel from SAS, using SCL functions ported to Base SAS". *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, paper 97, 2000.

SAS Institute Inc., "Technical Support Document #325 – The SAS System and DDE". Available http://ftp.sas.com/techsup/download/technote/ts325.pdf, updated 1999.

Viergever, W. W., and Vyverman, K. "Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange—Against All ODS, Part II". *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*, paper 16, 2003.

Vyverman, K. “Using Dynamic Data Exchange to Export Your SAS Data to MS Excel—Against All ODS, Part I”. *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, paper 5, 2002.

Vyverman, K. “Excel Exposed: Using Dynamic Data Exchange to Extract Metadata from MS Excel Workbooks”. *Proceedings of the Tenth South-Eastern SAS Users Group International Conference*, paper 15, 2003.

Vyverman, K. “Publishing Jack Vance: The SAS System as a Tool for Literary Analysis”. *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, paper 25, 2005.

ACKNOWLEDGMENTS

The author would like to thank his late chinchilla, the Lady Madeline, for inspiring him to the discovery of the technique described in the paper.

CONTACT INFORMATION

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

Koen Vyverman
SAS Netherlands
sugi30paper@vyverman.com