

Updating Spreadsheet Data and Graphics Using DDE to Combine SAS ® and Excel

Nigel Mulford, Covance CRU Ltd., Leeds, UK

ABSTRACT

This paper examines possibilities and methods of importing and exporting data from Microsoft Excel spreadsheets and exporting plots to an Excel spreadsheet from within a SAS program by using the DDE triplet accessory. The worked example is a spreadsheet of data that needs to be imported into SAS, analysed, and exported back onto an Excel spreadsheet. Plots also need to be produced from the data and exported onto the spreadsheet using this imported data by creating the plots in SAS and importing them using a macro in Visual Basic Excel run from the SAS program.

The presentation relates to the DDE triplet accessory in SAS 8.2 and includes examples of what code is needed in SAS and Excel (Visual Basic).

INTRODUCTION

A biotech client approached us wanting to closely monitor the results of a first into man study of a new humanized monoclonal antibody in healthy volunteers. The investigator for the study wanted to see a summary of the data as soon as possible after each set of blood tests had been taken to ensure levels of several types of cell in the blood were not approaching dangerous levels. The data were sent from the laboratory in an Excel spreadsheet, the investigator wanted the summary, which contained a list of the data, summary statistics, plots for individual subjects' data and overall data plots, as several worksheets, contained within one Excel workbook.

This can be a common request as the Microsoft Excel format is commonly used to transfer data between departments, sites and companies, as it is a very simple program to enter data onto, easy to use and is widely available. It cannot automate the production of plots and tables, or perform the more complex calculations and analysis that SAS is used for. The DDE triplet tool in SAS allows us to import a specific range of data from an Excel spreadsheet, run calculations and analysis within SAS, then export the data back onto a specific cell or cell range on a spreadsheet.

Often, a spreadsheet summarizing data requires a large amount of figures to be displayed. Using Excel it would be both time consuming and complex to produce a quantity of plots with an identical appearance, something that is relatively straightforward in SAS by using a macro. The DDE triplet allows us to import plots produced and saved with SAS in the form of a Computer Graphics Metafile (.cgm file) into a spreadsheet by running a Visual Basic macro contained either in the spreadsheet or the user's personal macro spreadsheet from within SAS.

The ability to use a macro stored either on the user's personal macro spreadsheet or a previously saved spreadsheet on a new spreadsheet from within SAS can potentially be a very powerful tool as it allows the user to automate repetitive actions, such as assigning different formats to different areas of data, in an identical way on many spreadsheets.

As well as importing and exporting data, the DDE triplet can also be used to execute system commands in Excel such as starting the program, opening a specific file, saving the file with or without a password and closing the program.

THE DDE TRIPLET

The Dynamic Data Exchange (DDE) triplet is a tool within SAS that allows data and commands to be transferred dynamically between applications in the Windows environment in either a single action or updating in real time. It can also run system commands such as opening and closing files, saving a document or running a Visual Basic macro in most Microsoft Office applications. In order to use the DDE tool, the document where the data is being imported from or exported to must be open.

OTHER METHODS OF IMPORTING DATA

Another way of importing and exporting data to and from a spreadsheet is to use the proc import and proc export commands in SAS. The code for these statements is certainly simpler than the code for a DDE triplet statement and they can access documents that are not running, but they do have two major drawbacks when compared to using a DDE triplet:

- The DDE triplet can specify a range of cells to import from and export to instead of the whole sheet. This is a major advantage when SAS is used to calculate descriptive statistics that need to be placed in a specific cell on a summary spreadsheet, or when the required data are amongst a large body of data.
- When using proc import, SAS evaluates the first 20 entries in a column to determine the data type for that variable. This can be a drawback when there is abnormal data after the first 20 entries, e.g. a variable that has 20 entries of numbers less than 1000, then a value of 1000000. SAS will assign a format of 4. when 7. is needed. Also, a flag variable that does not have any entries for the first 20 rows will be left blank. When using the DDE triplet method, the data type is specified when the data are imported.

DDE SYNTAX

The DDE triplet to import data from an Excel spreadsheet takes the form:

```
'<program>|<file  
location>[<filename>]<sheetname>!<startrow><startcolumn>:<end><endcolumn>:'
```

e.g. if we want to import the data contained within row 2 column 1 to row 17 column 3 from Sheet1 in 'ExampleIn.xls' stored in Q:\Docs into the filename 'input', the syntax will be:

```
filename input dde 'Excel|Q:\Docs\[ExampleIN.xls]Sheet1!R2C1:R17C3';
```

The easiest way to obtain the triplet is to copy the range of cells containing either the data to be imported, or the area the data will be imported to, to the clipboard. Then within SAS go to Solutions → Accessories → DDE Triplet. This will produce the correct triplet in a small window, which can be copied and pasted into the filename line.

The filename is then put into the following syntax to create a standard work dataset:

```
data <dataset name>;  
  infile <filename> dlm='09'x notab dsd missover;  
  informat <informats of input variables, leave blank for best.>;  
  format <formats of input variables, leave blank for best.>;  
  input <variable1 name> <char/num> <variable2 name> <char/num>;  
run;
```

The final input line needs care as this specifies whether the variable is numeric or character. If the variable is a character variable then there should be a '\$' after the variable name, otherwise this should be left blank. If the cell in Excel has a format attached, e.g. 100000 is displayed as 100,000, this will have to be imported as a character variable then converted into a number after the data have been imported. A similar method is needed if the data within Excel has any other non-numeric characters in the cell, e.g. '<5', 'BLQ', etc.

PhUSE 2006

EXPORTING FIGURES TO EXCEL

Once a figure is produced in SAS and saved into a Windows directory, the plot can be imported into an Excel spreadsheet by setting up a Visual Basic macro within Excel then running the macro from the SAS program. While setting up the macro can be time consuming, once it has been setup the figures can be imported to exactly the same position on the sheet, or any other sheet, every time the program is run.

The macro will have four distinct steps

First we must ensure the correct sheet is activated as there is a possibility the last active worksheet was the sheet the data were imported from, and select the correct worksheet (in this case called 'Figures').

```
Windows("output.xls").Activate  
Sheets("Figures").Select
```

We then delete all existing plots and select the cell location where the first figure will be placed:

```
ActiveSheet.DrawingObjects.Select  
Selection.Delete  
Range("A2").Select
```

Insert the first figure:

```
ActiveCell.Offset(0, 0).Select  
ActiveSheet.Pictures.Insert("<first plot location + name>").Select
```

Select the cell the next plot will be placed and insert the next plot. The offset is the number of rows to move the active cell followed by the number of columns to move it. In this case the next plot is placed four rows lower and seven columns to the right of the previous plot. Negative values in the offset will move the active cell up and/or left:

```
ActiveCell.Offset(4, 7).Select  
ActiveSheet.Pictures.Insert("<next plot location + name>").Select
```

The final step is repeated for all subsequent plots.

To run the macro from within a SAS program we use the DDE with the following syntax that accesses the excel system commands in the filename statement, then runs the macro in the data step:

```
filename CMDS DDE 'EXCEL|SYSTEM';  
data _null_;  
  file CMDS;  
  put '[RUN("<workbook name>!<macro name>")]';  
run;
```

The workbook name is not necessarily the workbook the figures are being exported to, but the workbook where the macro is stored. For example, if the macro is stored on the user's personal macro folder, the following put statement would be used.

```
put '[RUN("PERSONAL.XLS!<macro name>")]';
```

PhUSE 2006

OTHER USES FOR DDE

To open an Excel spreadsheet from SAS we must first set two options:

```
options noxwait noxsync;
```

NOXWAIT ensures the command prompt window disappears after the command has been executed.

NOXSYNC turns off the synchronisation of Excel and SAS. This allows SAS to continue executing commands whilst the system command is still running instead of waiting for the system command to stop before allowing SAS to continue.

To start Excel and open a spreadsheet:

```
data _null_;
    rc=system('start excel');
run;

data _null_;
    x=sleep(1);
run;

/*tells SAS to use an Excel system command whenever CMDS is used*/
filename CMDS DDE 'EXCEL|SYSTEM';

data _null_;
    file CMDS;
    put "[open("&"<spreadsheet name + location>"&")]";
run;
```

The sleep statement is required to make SAS wait for one second before executing the next statement to allow time for the system to start Excel. If this step is not included the system will try to send Excel a command when Excel is not running which can cause unexpected results.

To save the spreadsheet to the same location it was opened from and quit Excel:

```
data _null_;
    file CMDS;
    put "[save()]";
    put "[File.Close()]";
run;

data _null_;
    file CMDS;
    put "[Quit()]";
run;
```

PhUSE 2006

WORKED EXAMPLE

A typical situation that the DDE is very useful for is a lab sending blood test data in a spreadsheet which need presenting on a sheet along with basic summary statistics and figures.

The dummy raw data appears on a spreadsheet: named Data.xls stored in Q:\PhUSE, the summary data and plots will be presented on the spreadsheet output.xls. The data represents pharmacokinetic data for 8 subjects split into two groups, A and B.

	A	B	C	D
1	Group	Subject	Time (h)	Concentration (ng/mL)
2	A	1	0	0.00
3	A	2	0	0.00
4	B	3	0	0.00
5	B	4	0	0.00
6	A	1	0.25	41.32
7	A	2	0.25	45.19
8	B	3	0.25	17.47
9	B	4	0.25	13.43
10	A	1	0.5	54.37
11	A	2	0.5	63.26
12	B	3	0.5	29.12
13	B	4	0.5	25.52

IMPORTING THE DATA INTO SAS

The following code imports the data from the

```
filename plasma dde 'Excel|Q:\PhUSE\[Data.xls]Sheet1!R2C1:R50C4';
data plasma;
    infile plasma dlm='09'x notab dsd missover;
    input group $ vol ntime conc;
run;
proc sort data=plasma;
    by group ntime;
run;
```

Care must be taken with the cell range in the DDE triplet to ensure all data are imported. If the cell range is too small, e.g. the dde only goes down to row 10, no error or warning will appear in the log but not all data will be present in the dataset. If the cell range extends further than the data, SAS will not include the blank lines as entries or blank columns as variables so it is safe to extend the range much further than is presently needed. However, leaving blank variable names in the input line will result in that variable not being imported, even if the dde cell range covers the appropriate column.

CALCULATING DESCRIPTIVE STATISTICS FOR THE TWO DIFFERENT GROUPS

```
/*Group data*/
proc univariate data=plasma noprint;
    var conc;
    by group ntime;
    output out=gmean
        mean=mean;
run;

proc sort data=gmean;
    by ntime group;
run;

proc transpose data=gmean out=tgmean;    /*create tow columns of data*/
    by ntime;
    id group;
    var mean;
run;
```

PhUSE 2006

```
data prep;                                /*assign format*/
  set tgmean;
  format a b 4.1;
run;
```

The format could be applied on the spreadsheet using a Visual Basic macro described later. This method is useful if the recipient wants a non-SAS format, e.g. 100,000.

The values in the dataset before the data is output to the spreadsheet should look as the user wants the values to appear on the spreadsheet and in the correct order, though the columns do not need to be in the correct order

Outputting the data to the spreadsheet

```
filename eprint dde 'Excel|Q:\PhUSE\[Output.xls]Group!R9C1:R18C3';
```

The dde will output the data to the 'Group' sheet of the workbook 'Output' stored in the directory 'Q:\PhUSE' and in the first three columns, starting at row nine.

A null dataset is then executed containing the dde named 'eprint' to output the data.

```
data _null_;
  set prep;
  file eprint dlm='09'x notab;
  put ntime a b;
run;
```

The dataset prep has the following columns:

	NTIME	_NAME_	_LABEL_	A	B
1	0	MEAN	the mean, CONC	0.0	0.0
2	0.25	MEAN	the mean, CONC	46.7	11.9
3	0.5	MEAN	the mean, CONC	77.1	12.1
4	1	MEAN	the mean, CONC	66.2	10.7
5	2	MEAN	the mean, CONC	30.2	5.3
6	4	MEAN	the mean, CONC	9.2	2.7
7	8	MEAN	the mean, CONC	5.0	1.9
8	24	MEAN	the mean, CONC	2.4	0.8
9	48	MEAN	the mean, CONC	1.9	0.7
10	72	MEAN	the mean, CONC	0.1	0.1

As we do not require the variables _name_ and _label_ they are omitted from the put statement in the null data step so will not be exported to the spreadsheet.

PhUSE 2006

PRODUCING GROUP MEAN PLOTS

The following macro produces two plots named 'gplota.cgm' and 'gplotb.cgm' stored in a Windows directory called 'plots'

```
%macro gplot(group);
/*reset all previous graph options*/
goptions reset=all;

/*specify location and filename*/
filename gsasfile "Q:\PhUSE\plots\gplot&group..cgm";

/*assign titles*/
title1 j=c a=0 h=.5 f=hwcm002 "Group Curve of Mean Concentration" ls=0.5;
title2 j=c a=0 h=.5 f=hwcm001 "Group &group" ls=0.5;

/*specify graph options*/
goptions noborder cby=black fby=hwcm002 hby=1 csymbol=black ctext=black
ftext=hwcm002 htext=1
htitle=1 rotate=landscape device=cgmof971 gsfmode=replace gaccess=gsasfile vsize=8
cm hsize=8 cm;

/*specify axis ranges and labels*/
axis1 width=1 offset=(1 pct) order=(0 to 100 by 20) label=(h=.3 a=90 r=0 'Plasma
concentration (ng/mL)') value=(h=.3);
axis2 width=1 offset=(1 pct) label=(h=.3 a=0 'Nominal Time post-dose (h)')
value=(h=.3);

/*specify symbol type, symbol size and line type*/
symbol1 c=blue ci=blue v=dot height=.5 cells interpol=join l=1 w=.8;

/*specify data to be graphed and location to store the plot*/
proc gplot data=gmean(where=(group="&group")) gout=work.phuse;

/*plot data and store as plot1*/
plot mean*ntime / name='plot1' caxis = black ctext = black noframe hminor = 0
vminor = 0
vaxis = axis1 haxis = axis2;
run;
quit;
goptions ftext= ctext= htext=;
symbol1;
axis1;

/*output title and footnote to last plot produced*/
title;
footnote;

/*delete all plots in the catalogue so we can use the same plot name*/
proc datasets lib=work mt=catalog; delete phuse; run; quit;

/*end macro and call macro for both groups*/
%mend gplot;
%gplot(A);
%gplot(B);
```

The code saves the two plots in the plots directory but does not put them into the spreadsheet yet as they will be exported along with the individual subjects' plots at the end of the program. This is to allow one Visual Basic macro to import all the plots instead of having to create two separate macros.

PhUSE 2006

INDIVIDUAL SUBJECT DATA

The following code calculates the maximum concentration (Cmax), minimum concentration (Cmin), the time of maximum concentration (Tmax) and time of minimum concentration (Tmin) for each subject.

```
/*sort each volunteer by their concentrations*/
proc sort data=plasma;
    by vol conc;
run;

data max;
    set plasma;
    by vol conc;

    /*put the last concentration for each volunteer as cmax and tmax*/
    if last.vol then do;
        cmax=conc;
        tmax=ntime;
        output max;
    end;
    keep vol cmax tmax;
run;

data min;
    set plasma;
    by vol conc;
    /*put the first concentration for each volunteer as cmin and tmin*/
    if first.vol then do;
        cmin=conc;
        tmin=ntime;
        output min;
    end;
    keep vol cmin tmin;
run;

/*merge the minimum and maximum data together*/
data indstat;
    merge min max;
    by vol;
run;

proc sort data=indstat;
    by vol;
run;

/*transpose the data to get one column per volunteer of the stats*/
proc transpose data=indstat out=tinds prefix=v;
    var cmin tmin cmax tmax;
    id vol;
run;
```

On the final spreadsheet we want the Cmax, Cmin, Tmax and Tmin values to appear underneath the original data. This could be done by exporting the individual data from one dataset and the other values from another dataset, but as we are planning to add additional timepoints later on in the study we will create one dataset with both the individual timepoint values and the Cmin, Cmax, etc. values and export that to the spreadsheet.

```
proc sort data=plasma;
    by ntime;
run;

proc transpose data=plasma out=tindp prefix=v;
    by ntime;
    var conc;
    id vol;
run;
```

PhUSE 2006

```
/*put the original subject data with the stats*/
data allind;
    set tinds tindp;
    if _name_='CMIN' then occ=500;
    else if _name_='CMAX' then occ=600;
    else if _name_='TMIN' then occ=700;
    else if _name_='TMAX' then occ=800;
    else occ=ntime;
run;

/*prepare the dataset for output*/
data prep;
    set allind;
    attrib label format=$8.;
    if ntime=. then label=_name_;
    else label=left(ntime);
    proc sort;
        by occ;
run;

/*output the data*/
filename eprint dde 'Excel|Q:\PhUSE\[Output.xls]Individual!R4C1:R20C5';
data _null_;
    set prep;
    file eprint dlm='09'x notab;
    put label v1 v2 v3 v4;
run;
```

The code to create individual plots has two differences from the code that produces the group plots; the name of the output file and the way the input data is selected:

```
%macro iplot(v);    /*individual plots*/
    goptions reset=all;
    filename gsasfile "Q:\PhUSE\plots\iplot&v..cgm";

    << identical code to group plots >>

    proc gplot data=plasma(where=(vol=&v)) gout=work.phuse;

    << identical code to group plots >>

%mend iplot;
%iplot(1);
%iplot(2);
%iplot(3);
%iplot(4);
```

INSERTING THE PLOTS ONTO THE SPREADSHEET

We now have two group plots and four individual plots in the plots directory. To import these onto the spreadsheet we use the following Visual Basic macro called Macro1.

```
Sub Macro1()
    'Select the correct sheet
    Windows("output.xls").Activate
    Sheets("Figures").Select

    'Delete existing plots, set start to correct cell
    ActiveSheet.DrawingObjects.Select
    Selection.Delete
    Range("A2").Select
```

PhUSE 2006

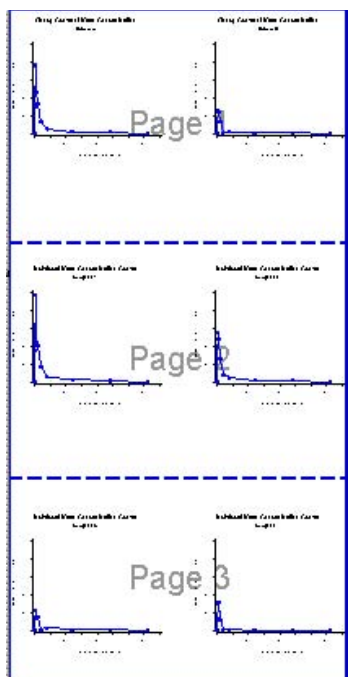
```
'Insert group plots
ActiveCell.Offset(0, 0).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\gplotA.cgm").Select
ActiveCell.Offset(0, 7).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\gplotB.cgm").Select

'Insert individual plots
ActiveCell.Offset(38, -7).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\iplot1.cgm").Select
ActiveCell.Offset(0, 7).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\iplot2.cgm").Select
ActiveCell.Offset(38, -7).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\iplot3.cgm").Select
ActiveCell.Offset(0, 7).Select
ActiveSheet.Pictures.Insert("Q:\PhUSE\Plots\iplot4.cgm").Select
End Sub
```

This macro is saved within the output spreadsheet itself. If this macro is to be used for several different spreadsheets the macro should be saved in the user's personal macro folder. The easiest way to do this in Excel is to open the Visual Basic Editor on the Tools → Macro menu and open the project explorer from the view menu. Then you can drag the macro, which is known as a module in this instance, from the spreadsheet VBA project to the project modules section of PERSONAL.XLS. The macro is now available to use in any of the user's spreadsheets.

To run the macro called Macro1 stored in the output spreadsheet from the SAS program, we use the following code:

```
filename CMDS DDE 'EXCEL|SYSTEM'; /*set the DDE to run Excel system commands*/
data _null_;
  file CMDS;
  put '[RUN("output.xls!Macro1")]';
run;
```



If the macro was stored in the user's personal macro folder and titled 'PhUSE' the put statement would read:

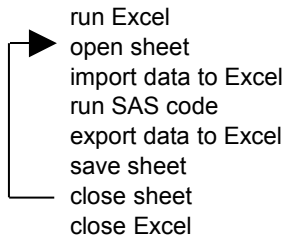
```
put '[RUN("personal.xls!PhUSE")]';
```

The spreadsheet is now complete, containing all the data and plots in the correct location.

LOOKING FORWARD

The DDE is a very powerful tool and there are many possible uses not already discussed.

One use would be the automation of importing data from several different spreadsheets, or sections within spreadsheets, represented on the following diagram:



CONCLUSION

The worked example has hopefully shown how powerful a tool the DDE accessory especially if we are able to combine it with other applications' strongpoints such as the visual basic facility in both Excel and Word.

ACKNOWLEDGMENTS

Thanks to Martin Johnson for technical help with the DDE Triplet.

CONTACT INFORMATION

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

Nigel Mulford
Covance CRU Ltd.
Springfield House
Hyde Street
Leeds
LS2 9LH
Work Phone: 0113 2373500
Fax: 0113 2445600
Email: nigel.mulford@covance.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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