

SAS® -with-Excel Application Development: Tools and Techniques

LeRoy Bessler, Assurant Health, Milwaukee, USA, bessler@execpc.com

Abstract

The commonest non-SAS tool for data presentation, and working with data, is Microsoft Excel. Other people may want your SAS report formatted as a spreadsheet. Dynamic Data Exchange (DDE) can empower your use of SAS with Excel. You can develop SAS programs to load worksheets, and format them without point-and-click. Almost anything that you can do directly in Excel can also be done using DDE commands from your SAS program. You can access Excel data, even to the level of a specific cell, for writing, reading, or formatting data. To eliminate worry about getting code details right every time, you can simplify and routinize your use of DDE with macros. The presentation shows two different ways to simultaneously deliver data at summary and detail levels in one Excel workbook. Nowhere-else-available technical tidbits are included. E.g., custom indenting of cell contents is unsupported by DDE, but you will learn how to accomplish it. Colour-coding the display of critical data elements, to signal value significance, is easily and often implemented for ODS tables—here a not-obvious solution for Excel is provided to you. The intended audience is all levels of SAS users.

Introduction

Why would you want to run Excel from SAS? You can create a hands-off production job to be automatically scheduled by the computer to access and analyze the data, load the spreadsheet, format it, and attach the report to a SAS-dispatched email message. In any case, if you find yourself preparing the same Excel report over and over, why not automate it?

This paper and the associated slide presentation are an introduction to a self-study collection of sixty macros and over twenty sample programs (with substantial comments), and supporting files, that are available for SAS with Excel application development, via email. To find out more about the Excel commands used here to work with DDE, or to find other commands or command options not used here, you must consult the documentation in Macrofun.hlp. To get this information, you must install Macrofun.hlp by initially downloading Macrofun.exe from the Microsoft Download Center, and then running the .exe file. If you are not experienced using the Download Center, you can find the right part of it by using google.com to search for “macrofun.exe”, or instead you could try this URL: <http://support.microsoft.com:80/support/kb/articles/Q128/1/85.ASP>.

NOTES: To keep the paper compact, screen images have usually been clipped to show only the essential parts of the displayed result. Also, rather than try to explain, incompletely, the options and syntax rules for DDE commands used, I recommend your looking into Macrofun.hlp for the complete story.

DDE: Dynamic Data Exchange

Client-Server Architecture

Your SAS session is a client. It opens Excel, which acts as a server. Your SAS program:

1. Starts/Executes Excel
2. Establishes a channel to talk to Excel
3. Opens a new/existing XLS file (workbook)
4. Sends commands to Excel
5. Saves, or Saves As (new/diff), XLS file
6. Exits/Stops Excel

What Can You Do with DDE?

Anything that you could do in Excel with your keyboard and mouse!!!

Here I cover only a subset of the input/output and formatting possibilities. When DDE can not do it (rare), you can pre-record an Excel macro and run it with DDE. DDE is more flexible than simple Excel macros. With DDE, you programmatically do anything that you can do with a mouse and keystrokes in an Excel window. In fact, if you configure your client PC appropriately, you can watch Excel working as a robot under the direction of your program. The same visual effects that appear on your screen when your hands are doing the work appear when your program is in control.

Some of the Cell Formatting Options

- Font controls available includes: font, style, size, colour, underline.
- Cell background can be filled with colour.
- Cell sizing is possible, and column or row AutoFit and column or row Hide/Unhide.

- Cell alignment, cell merge, text wrap, etc. can be done.
- Borders can be created, e.g., to section the worksheet.
- Cell content can be indented, using two tricks shown in sample programs, using macros.

Some of the Worksheet Controls

- Select worksheet, cells, rows, columns
- Freeze/Unfreeze panes
- Turn AutoFilter On/Off
- Generate subtotals in a worksheet
- Control/Change levels for Excel subtotals (or use multiple sheets for levels of totals)
- Activate a specific worksheet in a workbook
- Create additional worksheets in a workbook
- Delete unused worksheets in a workbook

Some of the Other Manipulation Possibilities

- Insert/Delete columns or rows
- Find, Find and Replace cell contents
- Copy cell, Move cells
- Use formulas to create new column(s)

Simple Example of Using Two Miscellaneous DDE Functions

You can use Zoom, to shrink (or enlarge) a worksheet, and Message, to access the text box in the lower left corner of the window.

For the Original Worksheet:

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Sex	Age	Height	Weight						
2	Alfred	M	14	69	112.5						
3	Alice	F	13	56.5	84						
4	Barbara	F	13	65.3	98						
5	Carol	F	14	62.8	102.5						
6	Henry	M	14	63.5	102.5						
7	James	M	12	57.3	83						
8	Jane	F	12	59.8	84.5						
9	Janet	F	15	62.5	112.5						
10	Jeffrey	M	13	62.5	84						
11	John	M	12	59	99.5						
12	Joyce	F	11	51.3	50.5						
13	Judy	F	14	64.3	90						
14	Louise	F	12	56.3	77						
15	Mary	F	15	66.5	112						
16	Philip	M	16	72	150						
17	Robert	M	12	64.8	128						
18	Ronald	M	15	67	133						
19	Thomas	M	11	57.5	85						
20	William	M	15	66.5	112						
21											
22											
23											
24											
25											
26											
27											
28											
29											
30											
31											
32											
33											
34											
35											

Yields this result:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Name	Sex	Age	Height	Weight								
2	Alfred	M	14	69	112.5								
3	Alice	F	13	56.5	84								
4	Barbara	F	13	65.3	98								
5	Carol	F	14	62.8	102.5								
6	Henry	M	14	63.5	102.5								
7	James	M	12	57.3	83								
8	Jane	F	12	59.8	84.5								
9	Janet	F	15	62.5	112.5								
10	Jeffrey	M	13	62.5	84								
11	John	M	12	59	99.5								
12	Joyce	F	11	51.3	50.5								
13	Judy	F	14	64.3	90								
14	Louise	F	12	56.3	77								
15	Mary	F	15	66.5	112								
16	Philip	M	16	72	150								
17	Robert	M	12	64.8	128								
18	Ronald	M	15	67	133								
19	Thomas	M	11	57.5	85								
20	William	M	15	66.5	112								
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													
35													

Running this code:

```
data _null_;
file DDEcmds; * How this fileref is
    established will be explained later *;
put '[zoom(80)]';
put '[message(true,
"This Sheet reduced to 80% of normal
size")]';
run;
```

The original cell/worksheet size and the default worksheet message ("Ready") are restored with this code:

```
data _null_;
file DDEcmds;
put '[zoom(100)]';
put '[message(false)]';
run;
```

The Mechanics of DDE and the Basic SAS-with-Excel Interaction

Channel for DDE Commands

DDE requires use of special SAS filename statements:

```
filename YourFileRef dde "excel|system";
```

YourFileRef is arbitrary 1-to-8 characters. I like DDEcmds.

How I Handle the DDE Fileref

As initialization code in my sample programs and/or macros you will see

```
filename &ddefileref dde "excel|system";
```

Earlier in the initialization processing, there is always the statement

```
%let ddefileref = DDEcmds;
```

Why &ddefileref? Avoid loss of generality. The “DDEcmds” fileref is not mandatory. Choose your own fileref.

The DDE Triplet for Excel

Identification of the current selection of worksheet and a rectangle of its cells for read or write is accomplished with:

```
EXCEL|YourSheet!RpppppCqqq:RsssssCttt
```

ppppp is 1-to-5-digit start/top row number

qqq is 1-to-3-digit start/left column number

sssss is end/bottom row number

ttt is end/right column number

Row/column numbers with leading zeros are permissible.

You can simplify identification of the certain selections of cells as follows:

RpppppCqqq	- single cell
Rppppp:Rsssss	- range of rows
Rppppp	- single row
Cqqq:Cttt	- range of columns
Cqqq	- single column

Coding the Cell Selection

You are not required to permanently hard code it. You can use a macro variable, and Software Intelligence (SI) in your program can dynamically determine the number of rows or ending row required, etc.

What is “Software Intelligence”? This is a term coined by LeRB in the early 1990’s. SI programs can be more than merely data-adaptive. Full explanation and illustrations of SI Application Development are available in other papers by the author.

The paper and presentation show three ways to code the DDE triplets, etc.: (a) hard-coding; (b) the author’s SAS macros; and SI generation based on data characteristics.

Excel DDE Commands

DDE commands were originally developed by Microsoft for earlier editions(s) of Excel. I have used them with various versions of Windows and Excel. They almost all work for me, but I can offer no guarantees.

The commands are documented in Macrofun.hlp, which you can download from Microsoft as mentioned above. “Macrofun” is an abbreviation for “Macro Functions”, not “Macro Fun”, but these macros functions are fun to use. They do not require use of Excel macros.

Please be aware that not every DDE command works (e.g., rename worksheets, create subtotals, . . .), and some command options may not work. Nevertheless, what *does* work is an enormous tool set!

true / false Values for DDE Commands

These are used to turn options On / Off in DDE command parameter assignments. But be aware that for some commands you are required to use 1 / 0 instead.

The Basic DDE Functions

1. Start Excel: provides a default empty workbook, with three worksheets
2. Open an existing workbook
3. Save As of the current workbook with a new name
4. Exit Excel

Starting Excel from SAS

For other versions of MS Office or Excel than that used for my development, the folder sequence in code below may differ:

```
x "C:\Program Files\Microsoft Office\Office\EXCEL.exe";
data _null_;
z=sleep(3); /* wait 3 seconds for Excel to start */
run;
```

The above program opens a new workbook, with three empty worksheets: Sheet1, Sheet2, and Sheet3.

Open Existing Workbook from SAS

```
data _null_; /* talk to DDE, no output data */
file DDEcmds;
put '[open("c:\YourFolder\YourWorkbook.xls")]';
x=sleep(3); /* wait 3 seconds for it to open */
run;
```

Finished Workbook Save As

```
data _null_;
file DDEcmds;
put '[error(false)]';
put '[save.as ("c:\OtherFolder\DifferentFileName.xls")]';
x=sleep(1);
run;
```

Exiting Excel from SAS

```
data _null_;
file DDEcmds;
put '[error(false)]';
put '[quit()]'; /* empty parenthesis */
run;
```

The error(false) command tells Excel that you do not want any prompts to confirm intention.

Overview of Custom SAS Macro Solutions (You can skip this, permanently, or come back later)

Macros for the Same Scenario As Above

You may not want to have to code so many lines of, frankly, unintuitive code. Available from the author are macros that could be used, to accomplish the same results, as follows:

```
%XLStart;
%XLDDDEcmdsFileName;
%XLOpen(WorkBook=c:\YourFolder\YourWorkbook.xls);
%XLSaveAs(WorkBook=c:\OtherFolder\DifferentFileName.xls);
%XLExit;
```

Macro for Starting Excel

```
%macro XLStart(ExcelPgm=  
C:\Program Files\Microsoft Office\Office\EXCEL.exe,SecondsToWaitForStart=3);  
options noxwait noxsync;  
x %unquote(%str(%'"&ExcelPgm"%'));  
data _null_;  
z=sleep(&SecondsToWaitForStart);run;  
%mend XLStart;
```

Macro for Opening an Existing Workbook

```
%macro XLOpen(WorkBook=,SecondsToWaitForOpen=3);  
data _null_;  
file DDEcmds;  
put %unquote(%str(%'[open("&WorkBook")]%'));  
z=sleep(&SecondsToWaitForOpen);  
run;  
%mend XLOpen;
```

Macro for Saving a Workbook

```
%macro XLSaveAs(WorkBook=);  
data _null_;  
file DDEcmds;  
put '[error(false)]';  
put %unquote(%str(%'[save.as("&WorkBook")]%'));  
x=sleep(1);  
run;  
%mend XLSaveAs;
```

When Argument(s) Is (Are) Macro/Symbolic Variable(s)

The value of argK is put in the symbol table by:

```
%YourMacro(argK=FormattedValue);  
or  
%let argK = FormattedValue;  
or  
call symput('argK',put(value,format.));
```

argK is retrieved from the symbol table with:

```
put %unquote(%str(%'[command(&argK)]%'));
```

OPTIONS MPRINT shows the resolved and executed statement to be:

```
put '[command(FormattedValue)]';
```

My SAS-with-Excel Libraries of Programs and Macros

Available are:

- 26 sample programs
- 5 empty, but specially preformatted, spreadsheets to use in sample programs
- Text files of colour definition information
- 60 macros for your use
- Abundant example code to invoke the macros, which is most educationally run with “OPTIONS MPRINT;”

Running example programs with “OPTIONS MPRINT;” shows generated SAS code in the SAS log. This enables you to understand what any macro does, and to understand what the DDE command content requirements are.

If you wish to avoid use of a macro, you can copy generated code from the SAS log, strip off the MPRINT prefix at the left margin, and adapt that code to other uses without the macro.

Please note these caveats. All macros & programs were tested, but may be incompatible with your software. They are NOT “industrial-strength”. I.e., they assume that you will use valid macro parameter assignments. If you add the needed editing, I will use it, and publish it with attribution to you. You should test the macros before assuming that their results are correct. I can only offer them “As Is”.

To Use a Macro Library

Save the macro in a Windows folder (e.g., c:\MyMacLib). The SAS-Instituted supplied macros are referred to with fileref “SASAUTOS”, which is also the label of a SAS system option. You must identify the location of your macros:

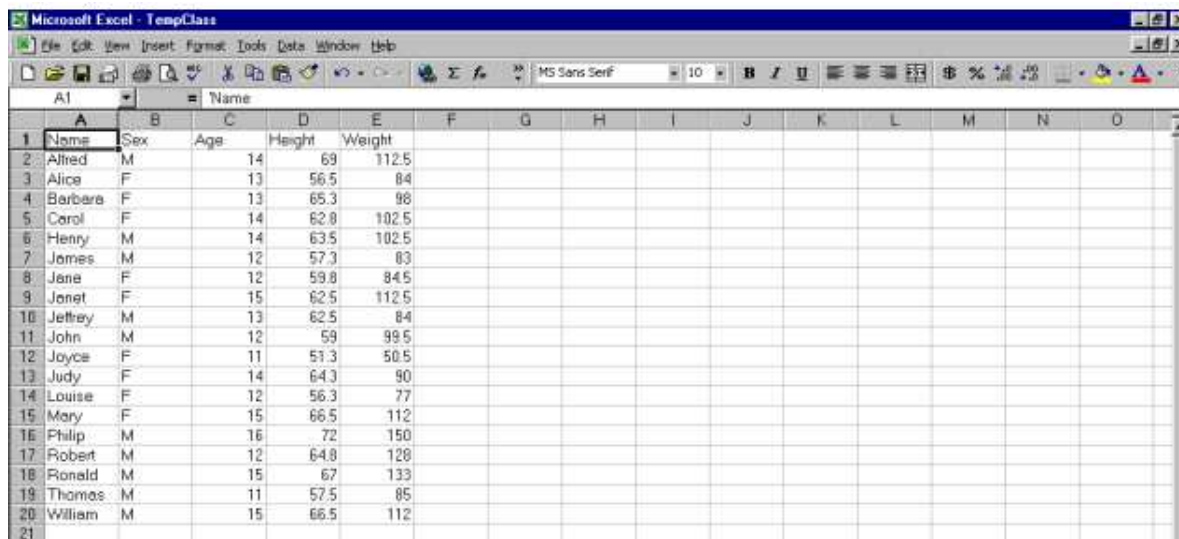
```
filename MyMacros 'c:\MyMacLib';
```

and then tell SAS to search your macro library (if first, with this parameter sequence):

```
options sasautos=(MyMacros sasautos);
```

A Tour of DDE Applications

We will start with a workbook which is loaded with data from SASHELP.CLASS and uses the default Excel formatting.

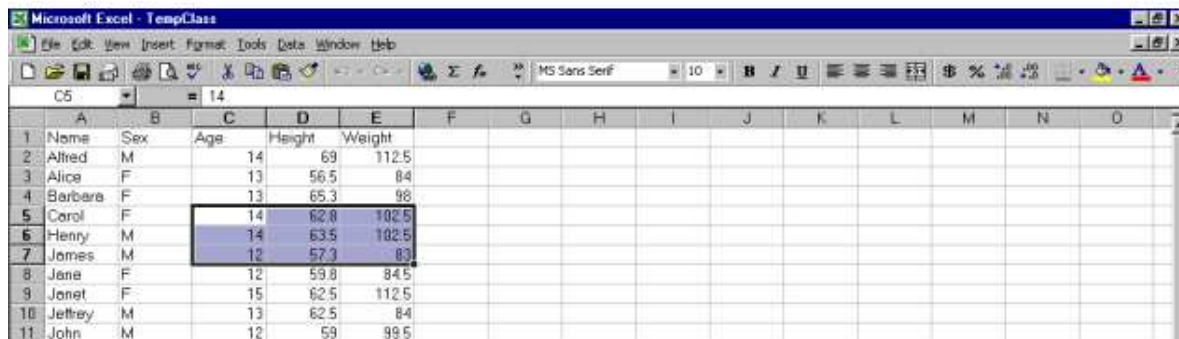


Font Formatting – Step 1

Activate worksheet (if not already there). Not really necessary—this workbook has only one worksheet. Select the cells to be formatted

```
data _null_;
file DDEcmds;
put '[workbook.activate("CLASS")]';
put '[select("R5C3:R7C5")]';
run;
```

Cells to be formatted Selected:



Font Formatting – Step 2

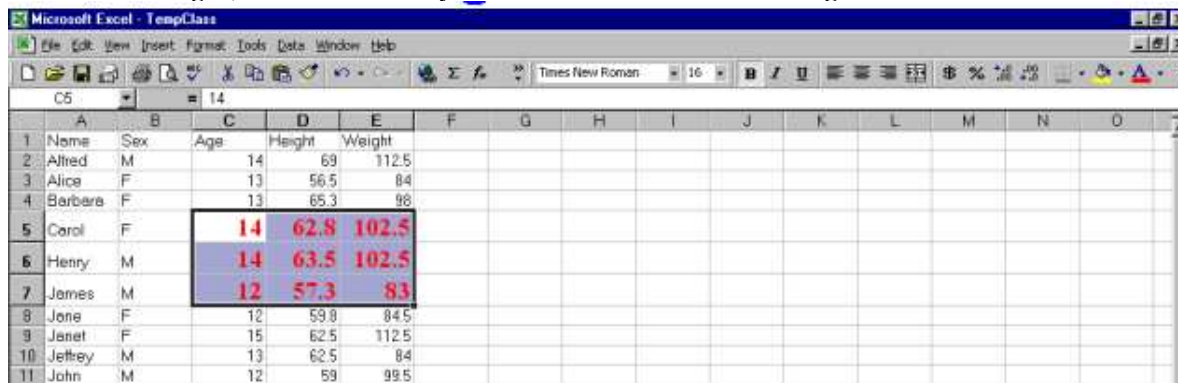
Let's use:

- font Verdana
- style Bold
- size 16 point
- (Excel) colour 3

Here's how (those commas are very important, and represent unused but available options):

```
data _null_;
file DDEcmds;
put '[font.properties("Verdana","Bold",16,,,,,,,,3)]';
run;
```

The font is changed, but is obscured by the cell selection box and shading:



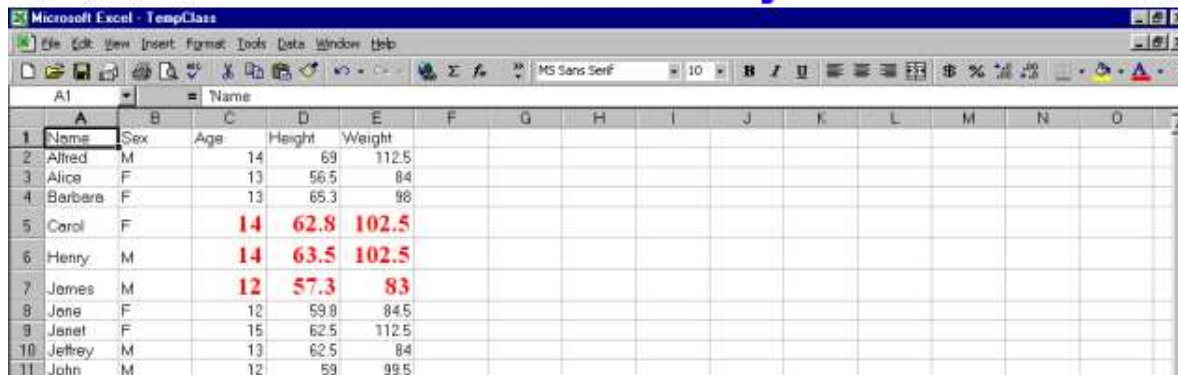
1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69	112.5
3	Alice	F	13	56.5	84
4	Barbara	F	13	65.3	98
5	Carol	F	14	62.8	102.5
6	Henry	M	14	63.5	102.5
7	James	M	12	57.3	83
8	Jane	F	12	59.8	84.5
9	Janet	F	15	62.5	112.5
10	Jeffrey	M	13	62.5	84
11	John	M	12	59	99.5

Font Formatting – Step 3 for Demo

After the block of cells has been formatted, the necessary predecessor `select("R5C3:R7C5")` command leaves them highlighted (just as a mouse select would). But I want to see the new font colour. So, move the focus with:

```
data _null_;
file DDEcmds;
put '[select("R1C1")]';
run;
```

“I can see clearly now.”



1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69	112.5
3	Alice	F	13	56.5	84
4	Barbara	F	13	65.3	98
5	Carol	F	14	62.8	102.5
6	Henry	M	14	63.5	102.5
7	James	M	12	57.3	83
8	Jane	F	12	59.8	84.5
9	Janet	F	15	62.5	112.5
10	Jeffrey	M	13	62.5	84
11	John	M	12	59	99.5

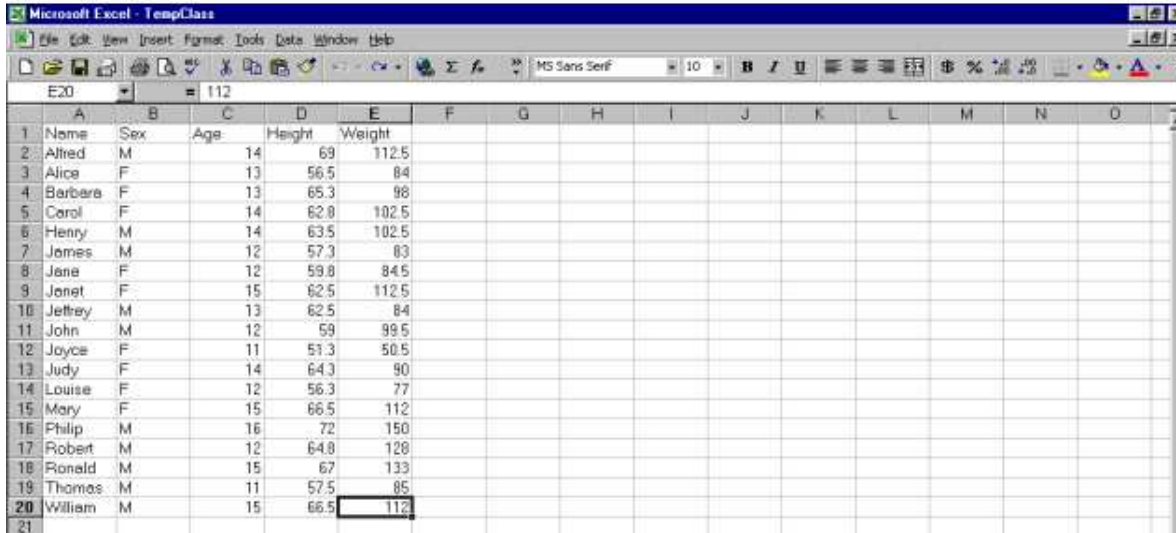
Font Formatting – All In One Step

```
data _null_;
file DDEcmds;
put '[workbook.activate("Class")]';
put '[select("R5C3:R7C5")]';
put '[font.properties("Verdana","Bold Italic",16,,,,,,,,3)]';
put '[select("R1C1")]'; run;
```


Remove the Background Colour

```
data _null_;  
file DDEcmds;  
put '[undo()]';  
run;
```

Colour Undone, Cell Still Selected:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Sex	Age	Height	Weight										
2	Alfred	M	14	69	112.5										
3	Alice	F	13	56.5	84										
4	Barbara	F	13	65.3	98										
5	Carol	F	14	62.8	102.5										
6	Henry	M	14	63.5	102.5										
7	James	M	12	57.3	83										
8	Jane	F	12	59.8	84.5										
9	Janet	F	15	62.5	112.5										
10	Jeffrey	M	13	62.5	84										
11	John	M	12	59	99.5										
12	Joyce	F	11	51.3	50.5										
13	Judy	F	14	64.3	90										
14	Louise	F	12	56.3	77										
15	Mary	F	15	66.5	112										
16	Philip	M	16	72	150										
17	Robert	M	12	64.8	128										
18	Ronald	M	15	67	133										
19	Thomas	M	11	57.5	85										
20	William	M	15	66.5	112										
21															

Prep to Colour-Code the Height Column

We need to be able to read all the data cells from Column 4.

```
filename Column DDE "EXCEL|Class!R2C4:R65536C4" notab;
```

Row 1 contains column headings, but we, a priori, do not know the number of actively used data rows. So, with the **filename** statement, we prepared SAS to read Row 2 through the Excel maximum Row 65536. Now we can read the data from the Height column and compute the average.

```
data AllHeights;  
infile Column;  
input height;  
run;  
  
proc means data=AllHeights noprint mean;  
var height;  
output out=AverageHeight mean=AvgHgt;  
run;
```

We need some criterion for colour-coding, and will arbitrarily designate taller than average as “Bad”, and shorter than average as “Good”.

```
data _null_;  
set AllHeights end=LastCell;  
if _N_ eq 1 then set AverageHeight;  
if height le AvgHgt then ColorIt = 'Good';  
else ColorIt = 'Bad';  
call symput('ColorIt' || trim(left(_N_)),trim(left(ColorIt)));  
run;
```

What Did the Prep Do?

If you submit

```
%put _user_;
```

in the SAS log, you will see this listing of symbol table content:

```

GLOBAL COLORIT1 Bad
GLOBAL COLORIT2 Good
GLOBAL COLORIT3 Bad
.
.
GLOBAL COLORIT18 Good
GLOBAL COLORIT19 Bad
GLOBAL HOWMANY 19

```

For brevity, we will use the author’s macro to do the colour-coding. (In principle, a macro could also be developed to do the colour-coding prep step as well.

```

%XLColorCodeCellsInColumn(
Column=4,StartRow=2,
ColorCodeSymVarPrefix=ColorIt,
ColorCodeCount=&HowMany,
GoodXLBackgroundColorNumber=32, /* Blue */
GoodXLForegroundColorNumber=2, /* White */
BadXLBackgroundColorNumber=3, /* Red */
BadXLForegroundColorNumber=1); /* Black */
* now move Select, to see the new colour *;
%XLFocus;

```

The particular combinations of foreground (text) colour and background (fill) colour are selected to maximize readability of the cell content. The screen print (clipped and magnified) does not demonstrate it well, but White on Blue is OK. Black on Blue would be difficult to read.

Screen Display of White Text OK, Screen Print Not As Good:

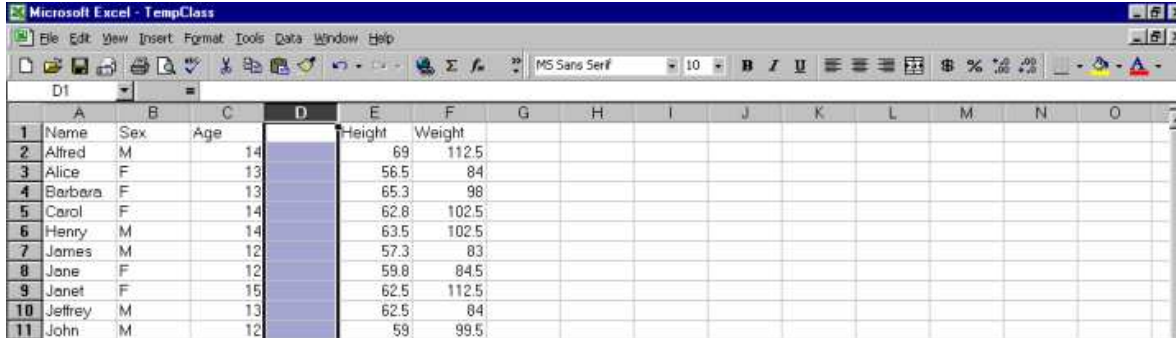
	A	B	C	D	E	F
	Name	Sex	Age	Height	Weight	
2	Alfred	M	14	69	112.5	
3	Alice	F	13	56.5	84	
4	Barbara	F	13	65.3	98	
5	Carol	F	14	62.8	102.5	
6	Henry	M	14	63.5	102.5	
7	James	M	12	57.3	83	
8	Jane	F	12	59.8	84.5	
9	Janet	F	15	62.5	112.5	
10	Jeffrey	M	13	62.5	84	
11	John	M	12	59	99.5	

Using Formulas in Excel

Normally, one could create all the data in SAS pre-spreadsheet-load processing. Suppose, however, we want columns derived from Excel subtotals. For demo only, we will work with simple spreadsheet content, without subtotals. With subtotals present, the use of Excel formulas works the same.

Insert a New Column Left of Column 4

```
data _null_;
file DDEcmds;
put '[select("C4")]';
put '[insert(4)]'; /* 4 specifies type of INSERT */
run;
```



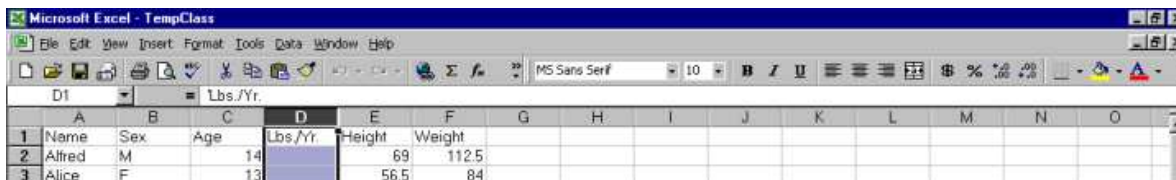
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Sex	Age		Height	Weight									
2	Alfred	M	14		69	112.5									
3	Alice	F	13		56.5	84									
4	Barbara	F	13		65.3	98									
5	Carol	F	14		62.8	102.5									
6	Henry	M	14		63.5	102.5									
7	James	M	12		57.3	83									
8	Jane	F	12		59.8	84.5									
9	Janet	F	15		62.5	112.5									
10	Jeffrey	M	13		62.5	84									
11	John	M	12		59	99.5									

Excel Format the Cells in Column 4 Like SAS Format 4.1

```
/* no need to re-select the column, because the focus is still there */
data _null_;
file DDEcmds;
put '[Format.Number("#0.0")]';
run;
```

Insert a Heading for the New Column 4

```
filename CELL1 DDE "EXCEL|Class!R1C4";
data _null_;
file CELL1;
put 'Lbs./Yr.';
run;
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Sex	Age	Lbs./Yr.	Height	Weight									
2	Alfred	M	14		69	112.5									
3	Alice	F	13		56.5	84									

Prepare for a Division Operation

Column 3 is to be the divisor. For this demo, set some of its cells to null (missing) values, and other cells to zero values.

```
data _null_;
file DDEcmds;
put '[select("C3")]';
put '[formula.replace("16", "", 1, , false, false)]';
put '[formula.replace("12", "0", 1, , false, false)]';
put '[select("R1C3")]'; /* remove highlight */
run;
```


Align Columns 2 and 3

```
data _null_;
file DDEcmds;
put '[select("C2:C3")]'; /* Sex & Age Columns */
put '[alignment(3,false,3,0)]'; /* 3 = Center */
put '[select("C4")]'; /* Lbs. / Yr. Column */
put '[alignment(4, false,3,0)]'; /* 4 = Right */
put '[select("R1C1")]';
run;
```

Need To Fix the Format of Height and Weight:

1	Name	Sex	Age	Lbs./Yr.	Height	Weight
2	Alfred	M	14	8.0	69	112.5
3	Alice	F	13	6.5	56.5	84
4	Barbara	F	13	7.5	65.3	98
5	Carol	F	14	7.3	62.8	102.5
6	Henry	M	14	7.3	63.5	102.5
7	James	M	0	N/A	57.3	83
8	Jane	F	0	N/A	59.8	84.5
9	Janet	F	15	7.5	62.5	112.5
10	Jeffrey	M	13	6.5	62.5	84
11	John	M	0	N/A	59	99.5
12	Joyce	F	11	4.6	51.3	50.5
13	Judy	F	14	6.4	64.3	90
14	Louise	F	0	N/A	56.3	77
15	Mary	F	15	7.5	66.5	112
16	Philip	M		N/A	72	150
17	Robert	M	0	N/A	64.8	128
18	Ronald	M	15	8.9	67	133
19	Thomas	M	11	7.7	57.5	85
20	William	M	15	7.5	66.5	112
21						

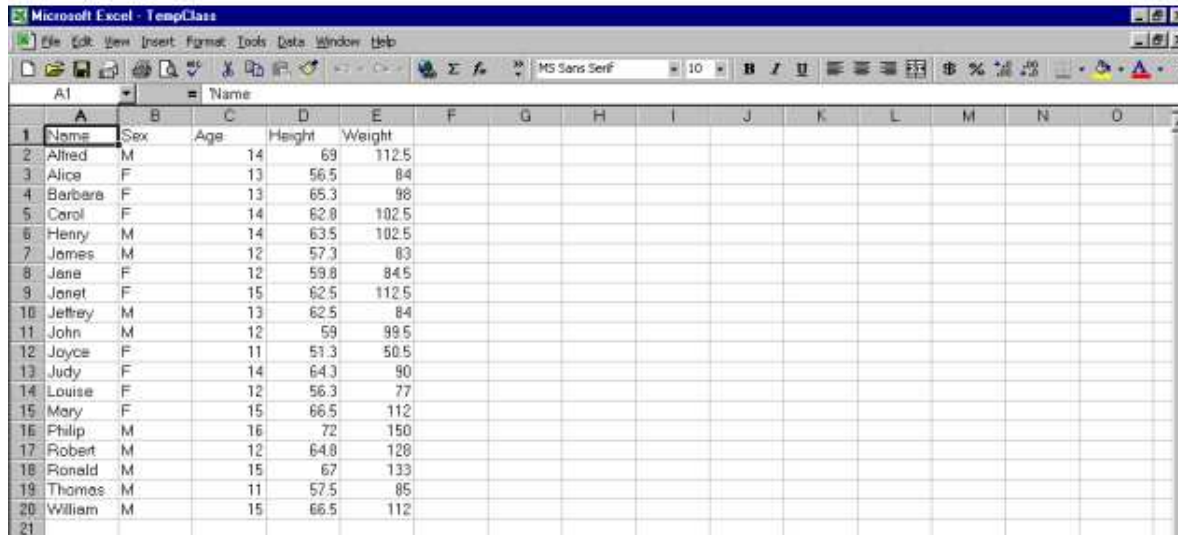
Excel Format the Height and Weight Data Like SAS Format 4.1

```
data _null_;
file DDEcmds;
put '[select("C5:C6")]';
put '[format.number("#0.0")]';
put '[select("R1C1")]'; * remove the column selection highlight *;
run;
```

1	Name	Sex	Age	Lbs./Yr.	Height	Weight
2	Alfred	M	14	8.0	69.0	112.5
3	Alice	F	13	6.5	56.5	84.0
4	Barbara	F	13	7.5	65.3	98.0
5	Carol	F	14	7.3	62.8	102.5
6	Henry	M	14	7.3	63.5	102.5
7	James	M	0	N/A	57.3	83.0
8	Jane	F	0	N/A	59.8	84.5
9	Janet	F	15	7.5	62.5	112.5
10	Jeffrey	M	13	6.5	62.5	84.0
11	John	M	0	N/A	59.0	99.5
12	Joyce	F	11	4.6	51.3	50.5
13	Judy	F	14	6.4	64.3	90.0
14	Louise	F	0	N/A	56.3	77.0
15	Mary	F	15	7.5	66.5	112.0
16	Philip	M		N/A	72.0	150.0
17	Robert	M	0	N/A	64.8	128.0
18	Ronald	M	15	8.9	67.0	133.0
19	Thomas	M	11	7.7	57.5	85.0
20	William	M	15	7.5	66.5	112.0
21						

Customized Excel Worksheet with Subtotals

We start with a worksheet pre-loaded via export of SASHELP.CLASS.



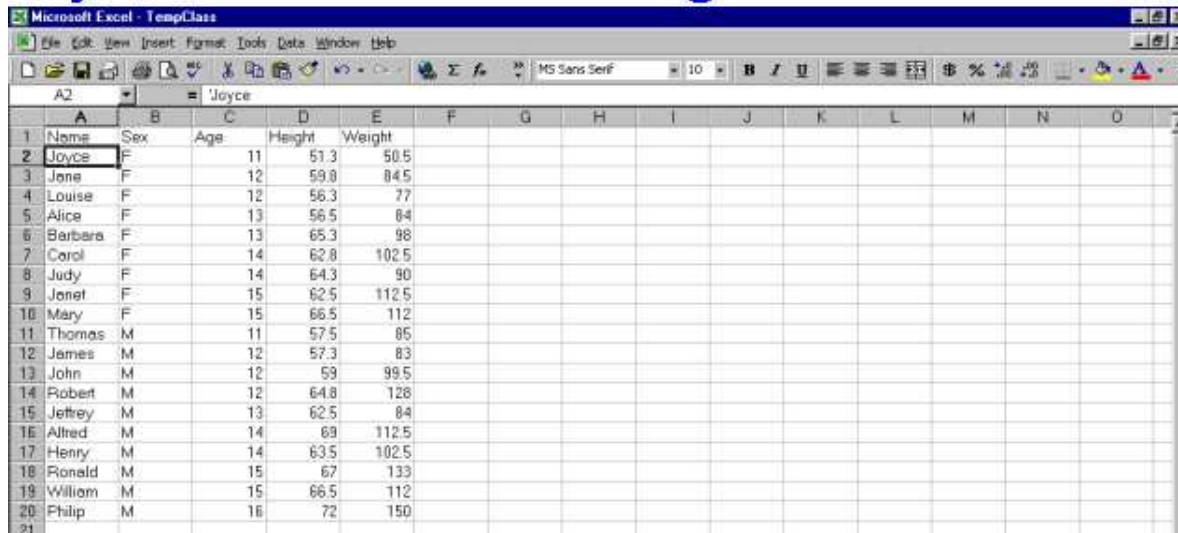
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Sex	Age	Height	Weight										
2	Alfred	M	14	69	112.5										
3	Alice	F	13	56.5	84										
4	Barbara	F	13	65.3	98										
5	Carol	F	14	62.8	102.5										
6	Henry	M	14	63.5	102.5										
7	James	M	12	57.3	83										
8	Jane	F	12	59.8	84.5										
9	Janet	F	15	62.5	112.5										
10	Jeffrey	M	13	62.5	84										
11	John	M	12	59	99.5										
12	Joyce	F	11	51.3	50.5										
13	Judy	F	14	64.3	90										
14	Louise	F	12	56.3	77										
15	Mary	F	15	66.5	112										
16	Philip	M	16	72	150										
17	Robert	M	12	64.8	128										
18	Ronald	M	15	67	133										
19	Thomas	M	11	57.5	85										
20	William	M	15	66.5	112										
21															

Now we sort the data with:

```
data _null_;
file DDEcmds;
put '[select("R1C1:R65536C256")]'; /* all cells */
put '[sort(1,"Sex",1,"Age",1,"Name",1,1,1,0)]';
put '[select("R2C1")]'; run;

/* 1 = sort by rows */
/* , "Sex",1,"Age",1,"Name",1 = sort on Sex, then Age, then Name, all ascending */
/* 1,1,0 = Normal sort, Headers present, Not case sensitive */
```

The worksheet has been sorted by Name within Age within Sex:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Sex	Age	Height	Weight										
2	Joyce	F	11	51.3	50.5										
3	Jane	F	12	59.8	84.5										
4	Louise	F	12	56.3	77										
5	Alice	F	13	56.5	84										
6	Barbara	F	13	65.3	98										
7	Carol	F	14	62.8	102.5										
8	Judy	F	14	64.3	90										
9	Janet	F	15	62.5	112.5										
10	Mary	F	15	66.5	112										
11	Thomas	M	11	57.5	85										
12	James	M	12	57.3	83										
13	John	M	12	59	99.5										
14	Robert	M	12	64.8	128										
15	Jeffrey	M	13	62.5	84										
16	Alfred	M	14	69	112.5										
17	Henry	M	14	63.5	102.5										
18	Ronald	M	15	67	133										
19	William	M	15	66.5	112										
20	Philip	M	16	72	150										
21															

After closing the data workbook, open a workbook of only Excel macros:

```
data _null_;
file DDEcmds;
put '[close(true)]';
x=sleep(1);
put '[open("C:\Folder\ExcelMacros.xls")]';
x=sleep(3); run;
```


Reformat & Save

```
data _null_;
file DDEcmds;
put '[select("R1")]';
put '[font.properties("", "Bold")]';
put '[select("C1:C5")]';
put '[column.width(,,3)]';
put '[error(false)]';
put '[save.as("C:\Folder\DataWithSubtotals.xls")]';
x=sleep(1);
run;
```

Remove Extra Grand Total

Find the last row, and put that row number in the symbol table:

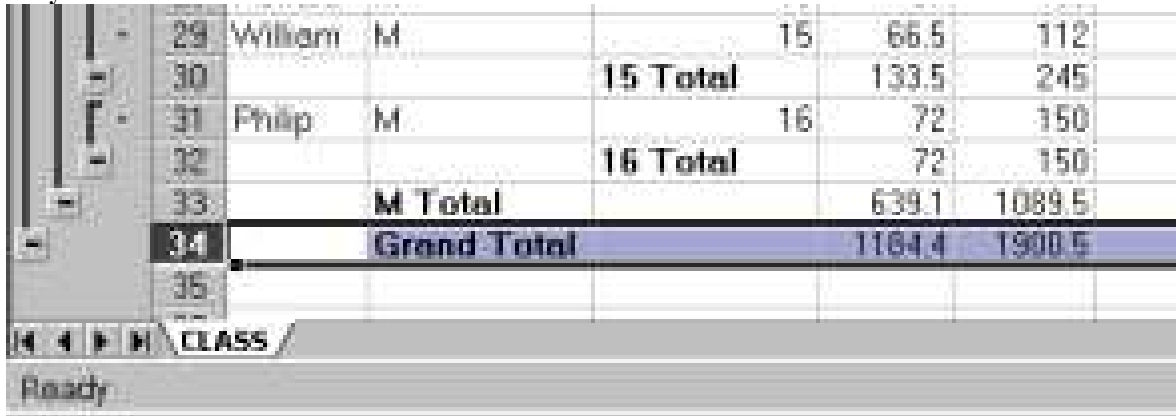
```
filename Column DDE "EXCEL|CLASS!R1C1:R65536C1" notab;
data _null_;
length FirstCharInColumnCell $ 1;
infile Column trunccover end=LastObs;
input FirstCharInColumnCell;
if LastObs;
call symput("LastRow",trim(left(_N_)));
run;
```

Find the second from last row, and delete it:

```
data _null_;
LastRow = &LastRow;
SecondLastGrandTotal = LastRow - 1;
call symput('SecondLastGrandTotal',trim(left(SecondLastGrandTotal)));
run;

data _null_;
file DDEcmds;
put '[select("R' "&SecondLastGrandTotal" '")]';
/* resolves to: put '[select("R34")]'; */
put '[edit.delete(3)]'; run;
```

Only One Grand Total:



29	William	M		15	66.5	112
30				15 Total	133.5	245
31	Philip	M		16	72	150
32				16 Total	72	150
33		M Total			639.1	1089.5
34		Grand Total			1164.4	1900.5
35						

Pseudo-Indent Cell Content

```
data _null_;
file DDEcmds;
put '[select("R2C2:R15C2")]';
put '[formula.replace("F","__Female",1,,false,false)]';
/* change colour of three _ to White
to match cell background colour: */
```

```

put '[font.properties(,,,,,,,,,2,,,1,3)]';
run; /* 2: colour the text string White
      1: start at character 1
      3: for a length of 3 characters */

```

Pseudo-Indents Complete:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Name	Sex	Age	Height	Weight									
2	Joyce	Female		11	51.3	50.5								
3			11 Total		51.3	50.5								
4	Jane	Female		12	59.8	84.5								
28	Ronald	Male		15	67	133								
29	William	Male		15	66.5	112								
30			15 Total		133.5	245								
31	Philip	Male		16	72	150								
32			16 Total		72	150								
33		M Total			639.1	1089.5								
34		Grand Total			1184.4	1900.5								

ReLabel Sex Subtotals (Using Macros Available from the Author)

```

%XLSelectColumns(Column=2);
%XLReplaceCellContents(Find=F Total,Replace=All Females);
%XLReplaceCellContents(Find=M Total,Replace=All Males);
%XLColumnWidth(Column=2,ToColumn=2,Action=AutoFit);
%XLFocus; /* Use %XLFocus to De-Select Column 2 */

```

New Labels for Sex Subtotals:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Name	Sex	Age	Height	Weight									
2	Joyce	Female		11	51.3	50.5								
3			11 Total		51.3	50.5								
13	Janet	Female		15	62.5	112.5								
14	Mary	Female		15	66.5	112								
15			15 Total		129	224.5								
16		All Females			545.3	811								
17	Thomas	Male		11	57.5	85								
18			11 Total		57.5	85								
19	James	Male		12	57.3	83								
31	Philip	Male		16	72	150								
32			16 Total		72	150								
33		All Males			639.1	1089.5								
34		Grand Total			1184.4	1900.5								

Excel subtotals above are showing to Level 4. Let's exercise the subtotals level display controls.

Show Subtotals Only to Level 3

```

data _null_;
file DDEcmds;
put '[show.levels(3,0)]';
run;

```

Excel Subtotals, Showing to Level 3:

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
3			11 Total	51.3	50.5
6			12 Total	116.1	161.5
9			13 Total	121.8	182
12			14 Total	127.1	192.5
15			15 Total	129	224.5
16		All Females		545.3	811
18			11 Total	57.5	85
22			12 Total	181.1	310.5
24			13 Total	62.5	84
27			14 Total	132.5	215
30			15 Total	133.5	245
32			16 Total	72	150
33		All Males		639.1	1089.5
34		Grand Total		1184.4	1900.5

Excel Subtotals, Showing to Level 2:

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
16		All Females		545.3	811
33		All Males		639.1	1089.5
34		Grand Total		1184.4	1900.5

Excel Subtotals, Showing to Level 1:

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
34		Grand Total		1184.4	1900.5

Another Way To Do Subtotals

You can really “Have It Your Way” with this custom design and construction. It relies on a one-time preparation of multiple pre-formatted empty worksheets, which are loaded via DDE. In an analogue or extension of this application, you may have to use Software Intelligence for post-load formatting. The user moves from “level” to “level” via clicking on worksheet tabs, not Excel “level buttons”. Here we show only the results, not the code (which is available, on request, from the author).

Overview WorkSheet:

	A	B	C	D	E
1	Sex	Height	Weight		
2	F Total	545.3	811.0		
3	M Total	639.1	1089.5		
4	Grand Total	1184.4	1900.5		

Detail + Summary WorkSheet:

	A	B	C	D	E
1	Sex	Age	Name	Height	Weight
2	F	11	Joyce	51.3	50.5
3	F	11 Total		51.3	50.5
4	F	12	Jane	59.8	84.5
5	F	12	Louise	96.3	77.0
6	F	12 Total		116.1	161.5
7	F	13	Alice	66.5	84.0
8	F	13	Barbara	65.3	96.0
9	F	13 Total		121.8	182.0
10	F	14	Carol	62.8	102.5
11	F	14	Judy	64.3	90.0
12	F	14 Total		127.1	192.5
13	F	15	Janet	62.5	112.5
14	F	15	Mary	66.5	112.0
15	F	15 Total		129.0	224.5
16	F Total			545.3	811.0
17	M	11	Thomas	57.5	85.0
18	M	11 Total		57.5	85.0
19	M	12	James	57.3	83.0
20	M	12	John	59.0	99.5
21	M	12	Robert	64.8	128.0
22	M	12 Total		181.1	310.5
23	M	13	Jeffrey	62.5	84.0
24	M	13 Total		62.5	84.0
25	M	14	Alfred	69.0	112.5
26	M	14	Henry	63.5	102.5
27	M	14 Total		132.5	215.0
28	M	15	Ronald	67.0	133.0
29	M	15	William	66.5	112.0
30	M	15 Total		133.5	245.0
31	M	16	Philip	72.0	150.0
32	M	16 Total		72.0	150.0
33	M Total			639.1	1089.5
34	Grand Total			1184.4	1900.5

Detail-Only WorkSheet:

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69.0	112.5
3	Alice	F	13	66.5	84.0
4	Barbara	F	13	65.3	98.0
5	Carol	F	14	62.8	102.5
6	Henry	M	14	63.5	102.5
7	James	M	12	57.3	83.0
8	Jane	F	12	59.8	84.5
9	Janet	F	15	62.5	112.5
10	Jeffrey	M	13	62.5	84.0
11	John	M	12	59.0	99.5
12	Joyce	F	11	51.3	50.5
13	Judy	F	14	64.3	90.0
14	Louise	F	12	96.3	77.0
15	Mary	F	15	66.5	112.0
16	Philip	M	16	72.0	150.0
17	Robert	M	12	64.8	128.0
18	Ronald	M	15	67.0	133.0
19	Thomas	M	11	57.5	85.0
20	William	M	15	66.5	112.0

Summary WorkSheet:

	A	B	C	D	E
1	Sex	Age	Height	Weight	
2	F	11 Total	51.3	50.5	
3	F	12 Total	116.1	161.5	
4	F	13 Total	121.8	182.0	
5	F	14 Total	127.1	192.5	
6	F	15 Total	129.0	224.5	
7	F Total		545.3	811.0	
8	M	11 Total	57.5	85.0	
9	M	12 Total	181.1	310.5	
10	M	13 Total	62.5	84.0	
11	M	14 Total	132.5	215.0	
12	M	15 Total	133.5	245.0	
13	M	16 Total	72.0	150.0	
14	M Total		639.1	1089.5	
15	Grand Total		1184.4	1900.5	

Bibliography (Annotated)

There is an introduction to DDE in the “SAS Companion for the Microsoft Windows Environment” (published by SAS Institute Inc., Cary, NC, USA), even in the Version 6 Edition.

If your needs are very simple (i.e., just transferring data from SAS to Excel, or vice versa), you might find this survey of alternative methods, not just DDE, very useful—

David Rucker, “Not All Fish Eat Worms: A SAS Programmer’s Guide to MS Excel and Other Fish Stories”, *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC, USA), 2003.

For a more sophisticated way to start a SAS-with-Excel application, for which you might have need (I have never needed this method), see—

Christopher A. Roper, “Intelligently Launching Microsoft Excel from SAS, using SCL Functions Ported to Base SAS”, *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC, USA), 2000.

For the work of the most frequent author/presenter on the topic of SAS with Excel via DDE, see—

Koen Vyverman, “Using Dynamic Data Exchange to Pour SAS Data into Microsoft Excel”, *Proceedings of the Eighteenth SAS Users European Users Group International Conference*, 2000.

Koen Vyverman, “Using Dynamic Data Exchange to Export Your SAS Data to MS Excel – Against All ODS, Part I”, *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC, USA), 2001.

Koen Vyverman, “Creating Custom Excel Workbooks from Base SAS with Dynamic Data Exchange”, *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*, SAS Institute Inc. (Cary, NC, USA), 2002.

It could be sufficient to read only the last of Vyverman’s papers. Besides providing an excellent introduction to SAS with Excel, he addresses some difficult problems, in which you might be interested. These include: (a) how to rename worksheets; (b) how to create Excel macros from SAS (rather than pre-record them in Excel for later use by SAS); and (c) how to send keystrokes to Excel—the solution of last resort.

Vyverman does take up some of the same matters that I present, but I cover more of the Excel macro commands, and work on several other topics, without taking up the special problems mentioned above.

Acknowledgements

Thanks to Peter Rusza for technical advice, and to Koen Vyverman and others for sharing DDE knowledge and experience with SAS users.

Notices

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other product and brand names are trademarks or registered trademarks of their respective owners.

Author Information

Your questions, comments, suggestions, and other solutions are always welcome.

LeRoy Bessler PhD
Email: bessler@execpc.com
Phone: 1 414 351 6748 (evenings and weekends—time is six hours earlier than GMT)

Dr. LeRoy Bessler has special interests in: Software-Intelligent application development, which yields SAS solutions that are reliable, reusable, maintainable, and extendable; and communication-effective design and construction of reports, tables, graphs, maps, spreadsheets, and web pages.