PharmaSUG2011 - Paper PO06

Get Dynamic Multi-sheet Excel Workbook with *STYLE* using ODS

Niraj J. Pandya, Element Technologies Inc., NJ

ABSTRACT

Working between SAS® and Microsoft Excel® sounds like day to day job in clinical trials data reporting. But it becomes a bit difficult when SAS is not installed on Windows platform. Also generating an Excel file with multiple work sheets becomes a daunting task and demands great set of programming skills and technical knowledge. This paper explains how to generate multi-sheet Excel workbook that contains SAS output and how to control the appearance of data in excel from SAS using styles. This paper also explains briefly adding an excel formula in the resulting workbook from SAS code. This paper explains how to generate such Excel workbook with the use of XML support in SAS 9 using ExcelXP type of ODS tagset. Technique explained here are useful irrespective of platform on which SAS is installed. Method explained in this paper generates Excel workbook that is compatible with Excel version 2002 and later.

INTRODUCTION

SAS gives multiple solutions to Import raw data from Excel file and convert it in to SAS dataset which are easy to use. Even reading data from multi-sheet excel workbook and then playing around the formats and labels and other attributes of data values is not a daunting task for programmer. But when it comes to transform the SAS data and analytical results in to Excel workbook with multiple work sheets and controlling the appearance of workbook, programmer always has to go through lots of hassle. With the latest version of ExcelXP tagset which has undergone many revisions and updates from its initial version and has been updated with many useful features, this daunting task of generating attractive multi-sheet excel workbook can be reduced to less difficult exercise.

SETTING UP THE ENVIRONMENT FIRST TIME

To use the method explained in this paper, you should download the latest version of ExcelXP tagset from ODS Website: http://support.sas.com/rnd/base/ods/odsmarkup/.

On this page, under the entry named '**Updated SAS 9.1 Tagsets to Download Individually"** find ExcelXP. You have to download it and save a copy of this file as SAS code file. Then specify a permanent location to import this tagset and store all other user defined styles. You can do this by submitting following lines.

```
LIBNAME reflib 'Location';

ODS PATH reflib.tmplmst (update) sashelp.tmplmst (read);
```

Libname statement above specifies where to store user defined tagsets and styles. ODS PATH statement specifies where to search for these tagsets and styles when being called later in the code.

After submitting above statement, you can make the tagset available by calling the previously downloaded and stored SAS code file by submitting following statement.

```
%include 'specify the path and the name of SAS code file';
```

You need to do this once and ExcelXP tagset will be imported and stored in the directory specified by REFLIB library reference. Also all the other user defined styles and tagsets will be stored in the file named tmplmst.sas7bitm in the directory associated with REFLIB library reference. All future SAS programs can use this ExcelXP tagset by pointing to the correct libname and path statements.

The new updated ExcelXP tagset has many enhanced features and performance over the released version. This new version has too many new features to list but you can get complete details of all the new features by submitting following statement.

```
ODS tagsets.excelxp file="test.xml" options(doc="help");
```

Above statement will print all the information to the SAS log. A list of supported options can be found at the link given in the conclusion section of this paper.

You also need to have Microsoft Excel 2002 or later version and Base SAS 9.1.3 or later version to make use of method explained in this paper.

CREATION OF USER DEFINED STYLE

ExcelXP tagset generates XML output which turns out to be a multi-sheet workbook when opened in Excel. The layout and formatting is taken care by SAS code only by using STYLE option and you do not require editing the workbook manually. We will see how to get such output later in the paper.

ODS styles control different aspects of the appearance of the output. Each style is composed of different style elements, each of which controls a particular part of the output. Style elements consist of different style attributes like font type, font size, background color etc.

To get the list of all available ODS styles, submit following statements.

```
Proc template;
     List styles;
Run; quit;
```

You will notice by submitting above piece of code that Base SAS V9 and later versions come with more than 50 different styles. Now to get your own style you can create it either from scratch or by using one of the existing default styles which resembles most near to what you want and then make desired changes in style attributes. It is always easier to define your style from one of the existing styles. Now how to find which style fits best to what you need? Following statements will get you the details of different style elements and their default attribute values.

```
Proc template;
     Source styles;
Run; quit;
```

Above piece of code will list all the default styles and their definition in the SAS log. But this way it will become tough to find one style that serves your purpose. For make it little easier, pick any one of the style from the list whose name suggests something related to your purpose and then add it in the source statement as following.

```
Source styles.stylename;
```

This will list only definition of a particular style and then it will be easier to change required attributes to define your own style.

For example purpose in this paper we will pick the style named **"Printer"** and use this style to create our own style.

When we use “Printer” style, following are few of the attributes that we will get in the output.

Table 1:

Style Element	Element Attribute	Value
Header	Font_Face	Thorndale AMT
Header	Font_size	11 point
Header	Background	Gray
Header	Just	Right
Data	Font_Face	Thorndale AMT
Data	Font_size	10 Point
Data	Background	White

We want to change this so that we get following attributes in the output.

Table 2:

Style Element	Element Attribute	Value
Header	Font_Face	Times New Roman
Header	Font_size	12 point
Header	Background	Yellow
Header	Just	Center
Data	Font_Face	Times New Roman
Data	Font_size	11 Point
Data	Background	Light Gray

Following code will generate a style named **“MYXLPrinter”** from **‘Printer”** by changing some of the attributes of some of the style elements.

```
Proc Template;
     Define style styles.MYXLPrinter;
         Parent = styles.Printer;

         style header from header/
              font_face="Times New Roman"
              font_size=12pt
              background=yellow
             just=center;

         style data from data/
             font_face="Times New Roman"
              font_size=11pt
              background=cxB0B0B0;

     end;

Run; Quit;
```

Style **“MYXLPrinter”** will be created and stored in the directory corresponding to the REFLIB library and can be used for all future SAS programs given that correct LIBNAME and PATH statements are applied.

USING ODS TO CREATE WORKBOOK WITH MULTIPLE WORK SHEETS

When we use ExcelXP tagset to create output, by default it creates a new worksheet each time a SAS procedure creates new tabular output. Because of this characteristic, you can create multi-sheet workbook either by having series of procedures that create tabular output or a single procedure that creates tabular output but with BY group processing.

We will use the raw data shown in Figure 1 to create multi-sheet workbook using ExcelXP tageset.

VIEWTABLE: Work.Dmgrphc

	USUBJID	DOB	COUNTRY	RANDDT	WT	HT	BMI
1	1001	06MAR1983	USA	20SEP2008	80	1.5	35.555555556
2	1002	09JUL1984	BRAZIL	07MAY2008	82	1.6	32.03125
3	1003	02JUL1972	USA	17OCT2007	75	1.4	38.265306122
4	1004	02APR1978	USA	13APR2009	76	1.6	29.6875
5	1005	26MAR1975	CANADA	29JAN2009	75	1.7	25.951557093
6	1006	04APR1971	BRAZIL	20NOV2008	79	1.7	27.335640138
7	1007	14JUN1966	CANADA	12AUG2008	79	1.6	30.859375
8	1008	06AUG1975	BRAZIL	04MAY2008	80	1.8	24.691358025
9	1009	05JUN1974	BRAZIL	06JUN2008	65	1.8	20.061728395
10	1010	05FEB1976	USA	14SEP2008	69	1.6	26.953125
11	1011	15AUG1980	CANADA	17OCT2007	73	1.5	32.444444444
12	1012	22JAN1981	CANADA	25AUG2009	76	1.6	29.6875
13	1013	07JUN1980	BRAZIL	02NOV2009	77	1.4	39.285714286
14	1014	21OCT1981	USA	09JUL2007	91	1.6	35.546875
15	1015	20MAR1982	CANADA	24FEB2009	90	1.8	27.777777778
16	1016	11JUN1983	BRAZIL	16NOV2008	86	1.7	29.757785467
17	1017	29JUN1972	USA	18JAN2008	84	1.6	32.8125
18	1018	25JAN1977	CANADA	05MAR2008	94	1.5	41.777777778
19	1019	02MAY1984	BRAZIL	27NOV2008	70	1.6	27.34375
20	1020	22AUG1984	USA	17NOV2008	72	1.6	28.125
21	1021	24MAR1985	BRAZIL	03JAN2008	86	1.7	29.757785467
22	1022	25SEP1975	CANADA	08MAR2008	64	1.5	28.444444444
23	1023	07AUG1975	USA	15APR2007	69	1.8	21.296296296
24	1024	31AUG1981	BRAZIL	12MAR2008	76	1.7	26.297577855
25	1025	19FEB1977	CANADA	24JUL2009	82	1.8	25.308641975
26	1026	11MAR1978	BRAZIL	15APR2009	72	1.6	28.125
27	1027	15JAN1979	CANADA	28JAN2009	79	1.6	30.859375
28	1028	24JAN1978	USA	07MAR2009	76	1.4	38.775510204
29	1029	17FEB1979	USA	21OCT2008	88	1.6	34.375
30	1030	20SEP1979	CANADA	24OCT2008	77	1.7	26.643598616

Figure 1: Raw SAS dataset for Demographic Information

In this dataset, as you can see, we have Demographic information for subjects enrolled in a clinical trial study from different countries. We want to create an Excel Workbook that has a separate sheet for each country. Following code will get such output for us.

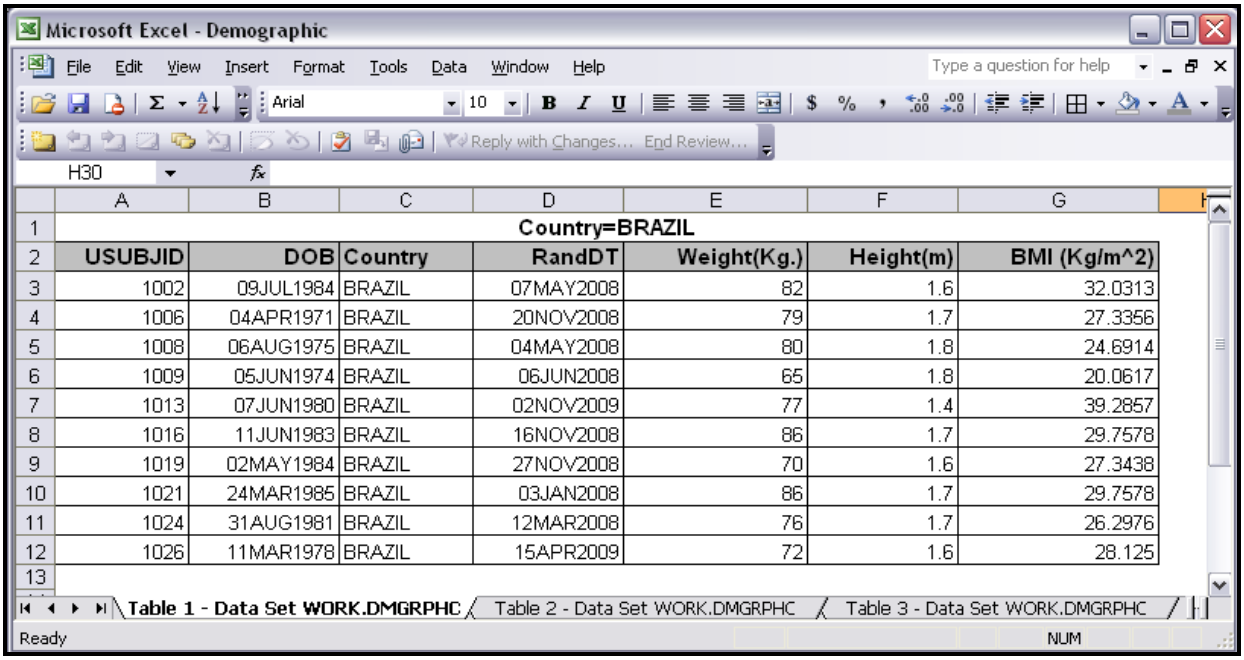
```
ODS listing close;

ODS tagsets.ExcelXP path = 'Output Location' file='Demographic.xml' style = Printer;

Proc sort data = work.dmgrphc;
     By country;
Run;

Proc print data = work.dmgrphc noobs label;
     By country;
     Pageby country;
     Var usubjid dob country randdt wt ht bmi;
Run;
Quit;
ODS tagsets.excelxp close;
```

The Output of above code looks like Figure 2. Notice that here we have used style **"Printer"**. Notice that ExcelXP tagset created 3 different worksheets; one for each value of BY group variable which is country in our case.



Microsoft Excel - Demographic

File Edit View Insert Format Tools Data Window Help

Type a question for help

Arial 10

Reply with Changes... End Review...

H30

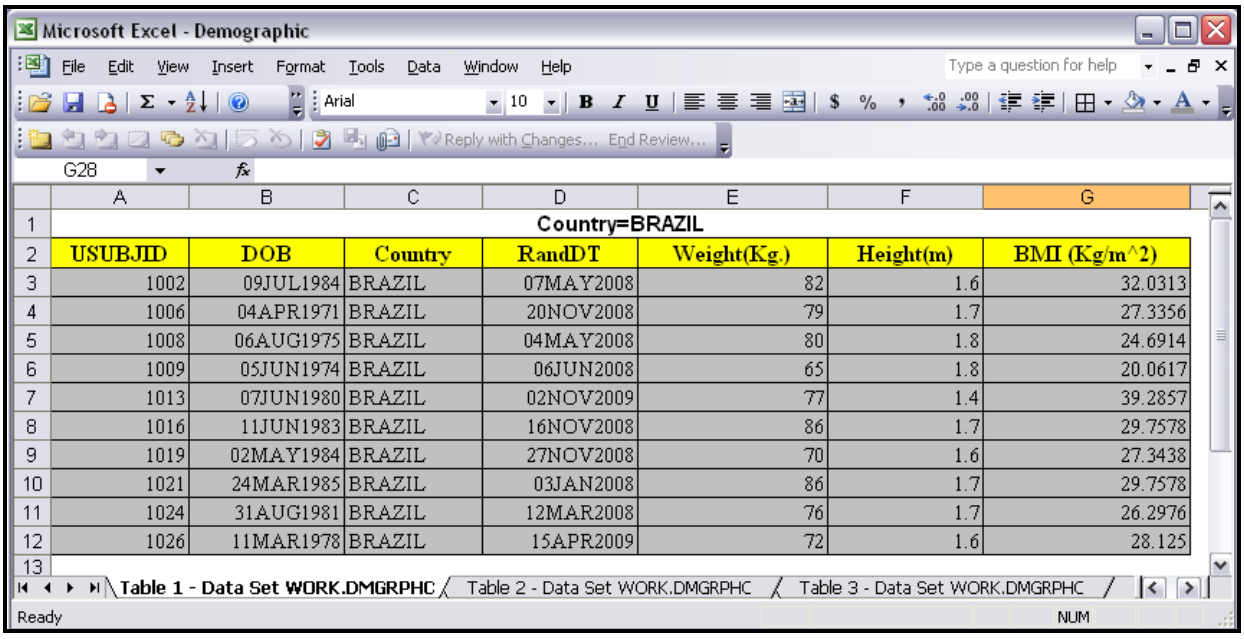
	A	B	C	D	E	F	G
1	Country=BRAZIL						
2	USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m)	BMI (Kg/m^2)
3	1002	09JUL1984	BRAZIL	07MAY2008	82	1.6	32.0313
4	1006	04APR1971	BRAZIL	20NOV2008	79	1.7	27.3356
5	1008	06AUG1975	BRAZIL	04MAY2008	80	1.8	24.6914
6	1009	05JUN1974	BRAZIL	06JUN2008	65	1.8	20.0617
7	1013	07JUN1980	BRAZIL	02NOV2009	77	1.4	39.2857
8	1016	11JUN1983	BRAZIL	16NOV2008	86	1.7	29.7578
9	1019	02MAY1984	BRAZIL	27NOV2008	70	1.6	27.3438
10	1021	24MAR1985	BRAZIL	03JAN2008	86	1.7	29.7578
11	1024	31AUG1981	BRAZIL	12MAR2008	76	1.7	26.2976
12	1026	11MAR1978	BRAZIL	15APR2009	72	1.6	28.125
13							

Table 1 - Data Set WORK.DMGRPHC | Table 2 - Data Set WORK.DMGRPHC | Table 3 - Data Set WORK.DMGRPHC

Ready NUM

Figure 2: Output from ExcelXP Tagset with "Printer" Style

In above code, replacing style **"Printer"** with **"MYXLPrinter"** will change the output as shown in Figure 3. Note the difference in appearance of the output.



Microsoft Excel - Demographic

File Edit View Insert Format Tools Data Window Help

Type a question for help

Arial 10

Reply with Changes... End Review...

G28

	A	B	C	D	E	F	G
1	Country=BRAZIL						
2	USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m)	BMI (Kg/m^2)
3	1002	09JUL1984	BRAZIL	07MAY2008	82	1.6	32.0313
4	1006	04APR1971	BRAZIL	20NOV2008	79	1.7	27.3356
5	1008	06AUG1975	BRAZIL	04MAY2008	80	1.8	24.6914
6	1009	05JUN1974	BRAZIL	06JUN2008	65	1.8	20.0617
7	1013	07JUN1980	BRAZIL	02NOV2009	77	1.4	39.2857
8	1016	11JUN1983	BRAZIL	16NOV2008	86	1.7	29.7578
9	1019	02MAY1984	BRAZIL	27NOV2008	70	1.6	27.3438
10	1021	24MAR1985	BRAZIL	03JAN2008	86	1.7	29.7578
11	1024	31AUG1981	BRAZIL	12MAR2008	76	1.7	26.2976
12	1026	11MAR1978	BRAZIL	15APR2009	72	1.6	28.125
13							

Table 1 - Data Set WORK.DMGRPHC | Table 2 - Data Set WORK.DMGRPHC | Table 3 - Data Set WORK.DMGRPHC

Ready NUM

Figure 3: Output from ExcelXP Tagset with "MYXLPrinter" Style

STYLE OVERRIDE

In above mentioned code to generate multi-sheet workbook using ExcelXP tagest, we can specify multiple VAR statement in PROC PRINT. It enables us to apply different style elements to different variables. We can override the style attributes for one single column and not changing the entire style definition. For this purpose we can generate a

new style element from existing style element and then use this newly generated style element to override the style attributes of any specific column.

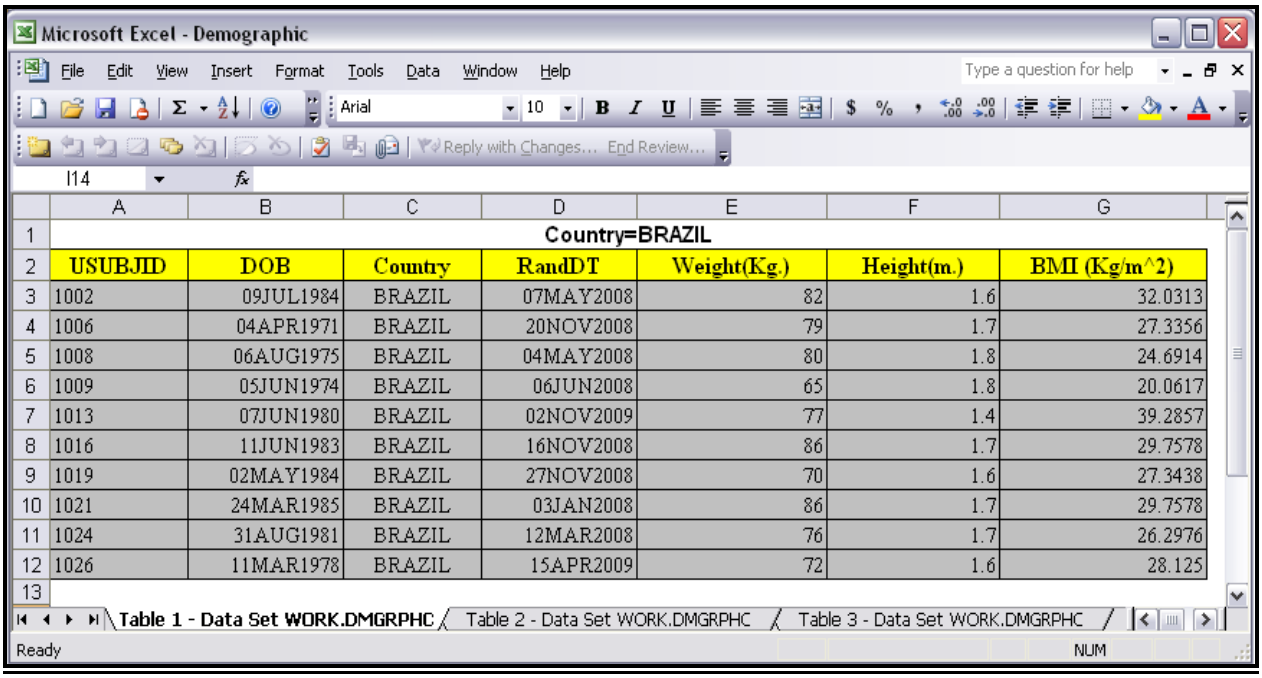
For example we want to change the justification attribute for the data part of first column USUBJID and third column COUNTRY. We want all the data values for USUBJID to be left aligned and data values for COUNTRY to be centered without changing any style attributes of header for these two columns. We can use existing data element of the style **MYXLPrinter** which we defined earlier and can create another data style elements which will be applied individually to USUBJID and COUNTRY variables. This will be applied to data part only without changing the appearance of Header for these columns. Following statements explain how to define these new elements and then how to apply style override. Following piece of code needs to be added in PROC TEMPLATE to the definition of "**MYXLPrinter"** style that we defined earlier.

```
style data_left from data/just=left;
style data_center from data/just=center;
```

Following statements explain how style override will be applied for USUBJID and COUNTRY variables in PROC PRINT.

```
Var usubjid /style(data)= data_left;
Var country/style(data)= data_center;
```

After applying this style override the output from Figure 3 changes and looks like Figure 4. Notice the justification for data values in columns USUBJID and COUNTRY after applying style override.



	A	B	C	D	E	F	G
1	Country=BRAZIL						
2	USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m.)	BMI (Kg/m^2)
3	1002	09JUL1984	BRAZIL	07MAY2008	82	1.6	32.0313
4	1006	04APR1971	BRAZIL	20NOV2008	79	1.7	27.3356
5	1008	06AUG1975	BRAZIL	04MAY2008	80	1.8	24.6914
6	1009	05JUN1974	BRAZIL	06JUN2008	65	1.8	20.0617
7	1013	07JUN1980	BRAZIL	02NOV2009	77	1.4	39.2857
8	1016	11JUN1983	BRAZIL	16NOV2008	86	1.7	29.7578
9	1019	02MAY1984	BRAZIL	27NOV2008	70	1.6	27.3438
10	1021	24MAR1985	BRAZIL	03JAN2008	86	1.7	29.7578
11	1024	31AUG1981	BRAZIL	12MAR2008	76	1.7	26.2976
12	1026	11MAR1978	BRAZIL	15APR2009	72	1.6	28.125
13							

Figure 4: Output from ExcelXP Tagset with "MYXLPrinter" Style after applying style override

DATES IN SAS AND EXCEL

SAS and Excel use different date systems. Hence when SAS dataset is transformed in to Excel or SAS output is read in Excel, often it is observed that format of date values is changed. To solve this problem you have to convert you SAS dates in to Excel Date values by writing a code or create a new dataset with new format of dates. But this solution becomes inefficient as data grows and it is prone to errors.

A better solution is to use combination of both the formats. First specify a SAS format by using FORMAT statement without changing the data values and then specify Excel date format by using a style. SAS format will change the physical value of date which is written to XML file and style override will change the way the Value is displayed in the resulting excel workbook.

For example, we want to change the dates presented in DOB and RANDDT columns from DDMMMYYYY format of SAS in to MM/DD/YYYY format in Excel. To achieve this result, we need to add following statements in PROC TEMPLATE in the definition of "**MYXLPrinter"** style.

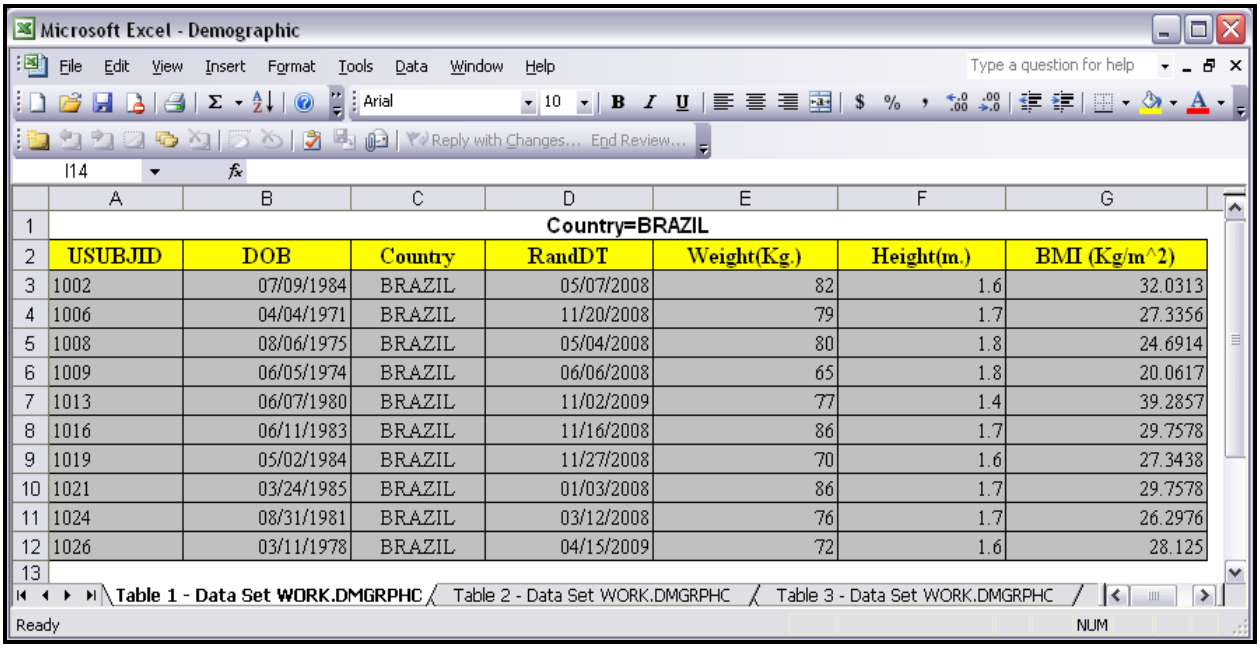
```
style data_date from data/tagattr='format:mm/dd/yyyy type:DateTime';
```

In above statements we defined new style data element named data_date which inherits all attributes from already defined custom data style element except the attribute named 'tagattr'. This attribute is used to tell Excel to interpret the data values as Excel DateTime values and apply the Excel format to the data.

Following statements explain how style override will be applied using this newly define style element for DOB and RANDDT variables in PROC PRINT.

```
Var dob/style(data)=data_date;
Var randdt/style(data)=data_date;
```

Notice the change in date values in Figure 5 from Figure 4.



	A	B	C	D	E	F	G
1	Country=BRAZIL						
2	USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m.)	BMI (Kg/m^2)
3	1002	07/09/1984	BRAZIL	05/07/2008	82	1.6	32.0313
4	1006	04/04/1971	BRAZIL	11/20/2008	79	1.7	27.3356
5	1008	08/06/1975	BRAZIL	05/04/2008	80	1.8	24.6914
6	1009	06/05/1974	BRAZIL	06/06/2008	65	1.8	20.0617
7	1013	06/07/1980	BRAZIL	11/02/2009	77	1.4	39.2857
8	1016	06/11/1983	BRAZIL	11/16/2008	86	1.7	29.7578
9	1019	05/02/1984	BRAZIL	11/27/2008	70	1.6	27.3438
10	1021	03/24/1985	BRAZIL	01/03/2008	86	1.7	29.7578
11	1024	08/31/1981	BRAZIL	03/12/2008	76	1.7	26.2976
12	1026	03/11/1978	BRAZIL	04/15/2009	72	1.6	28.125
13							

Figure 5: Output from ExcelXP Tagset with "MYXLPrinter" Style with Date Override

ADD AN EXCEL FORMULA FROM SAS CODE

Anybody who uses Excel on regular basis knows that formula in Excel always starts with "=" sign and it usually refers to other cells in the Excel work sheet as a part of formula. The reference to other cells in the work sheet works in a little different way while dealing with XML spreadsheets than usual way in Excel spreadsheets. In usual Excel file, reference to the cell is specified by column name and row number (i.e. first cell in the work sheet is referenced as A1). But while dealing with XML spreadsheet, reference should be specified as number of rows up/down to the current cell and number of columns left/right to the current cell. In the formula, Rows are presented by "R" and number of rows are specified in square parenthesis "[ ]".Minus sign "-"in the [ ] specifies up direction and plus sign "+"

specifies down direction. Same way columns are presented by “C” and number of columns are specified in “[ ]”. For columns, “-“sign in [ ] specifies left direction and “+” sign specifies right direction.

With this kind of referencing model and with the use of functions, we can put any formula together and can be applied in resulting XML spreadsheet from SAS code.

In our example, in Figure 5, the values in column G which represents BMI, are static and are coming from SAS dataset directly. We want to add the formula to this column so that if in future either or both of the Weight and Height values are changed in the resulting worksheet by someone, value of BMI is recalculated automatically. Let’s say we also want to apply rounding of 2 decimal and want all the values for BMI variable to be centered in the column.

Using SAS, the formula for BMI would be: BMI **= Round ((WT/ (HT*HT)), *0.2*)**

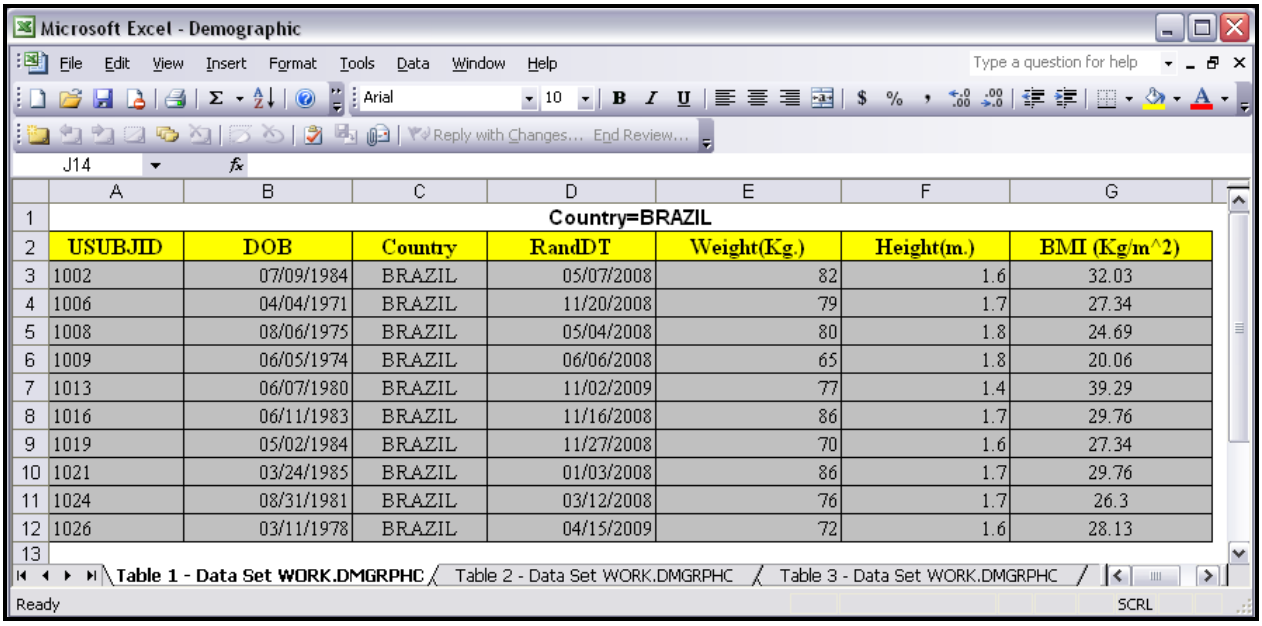
Referring to Figure 5, for column BMI, column WT is 2 columns on the left and column HT is 1 column on the left. We want a formula so that all the calculation occurs within the same row only, and so we don’t need to refer any other row but the current row. In that case no row number is required to specify in [ ]. We also need to use function ROUND. In Excel the formula would be: BMI = **Round (((RC [-2])/ (RC [-1]*RC [-1])), *2*)**

Notice that rounding of 2 is also different in SAS and Excel formula which is highlighted by underline. Now to apply this formula, we will again use style attribute ‘tagattr’ to define new data element which will inherit all the attributes from already defined custom ‘data_center’ element except the attribute ‘tagattr’ and will apply style override for variable BMI. Following statements show how to define this new style data element.

```
Style data_bmi from data_center/
      tagattr='formula:=round(((RC[-2])/(RC[-1]*RC[-1])),2)';
```

Using newly created ‘data_bmi’ style element we can apply style override for variable BMI as per following statement in PROC PRINT.

```
Var bmi/style(data)=data_bmi;
```



	A	B	C	D	E	F	G
1	Country=BRAZIL						
2	USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m.)	BMI (Kg/m^2)
3	1002	07/09/1984	BRAZIL	05/07/2008	82	1.6	32.03
4	1006	04/04/1971	BRAZIL	11/20/2008	79	1.7	27.34
5	1008	08/06/1975	BRAZIL	05/04/2008	80	1.8	24.69
6	1009	06/05/1974	BRAZIL	06/06/2008	65	1.8	20.06
7	1013	06/07/1980	BRAZIL	11/02/2009	77	1.4	39.29
8	1016	06/11/1983	BRAZIL	11/16/2008	86	1.7	29.76
9	1019	05/02/1984	BRAZIL	11/27/2008	70	1.6	27.34
10	1021	03/24/1985	BRAZIL	01/03/2008	86	1.7	29.76
11	1024	08/31/1981	BRAZIL	03/12/2008	76	1.7	26.3
12	1026	03/11/1978	BRAZIL	04/15/2009	72	1.6	28.13
13							

Figure 6: Output from ExcelXP Tagset with “MYXLPrinter” Style with Excel Formula

In above Figure 6, you can notice the change in appearance of column G and formula is also in place for this column so in future, if value of either Weight or Height is changed in resulting XML file, value of BMI will be recalculated automatically.

ExcelXP TAGSET OPTIONS

ExcelXP tagset supports numerous options that control both appearance and functionality of the Excel workbook. These options can be specified in ODS statement using OPTIONS keyword. As with any other ODS options, ExcelXP tagset options remain in effect until they are turned off or their value is changed. We will briefly go through few of such options to explain how they work.

SUPPRESS DEFAULT BYGROUP TITLE

In all the figures above, you can see default byline title that represents the unique value of BY GROUP variable in each work sheet. You can suppress this title by either using global SAS option **NOBYLINE** or by using ExcelXP tagset option **SUPPRESS_BYLINES** and setting its value to 'YES'.

GET USER DEFINED TITLE

Let's say we want some title in each work sheet define by us. We can get this by using ExcelXP tagest option **EMBEDDED_TITLES** and setting its value to 'YES'. After setting this options' value to 'YES', we can specify any title as we do in usual way in SAS and it will come up in each resulting work sheet.

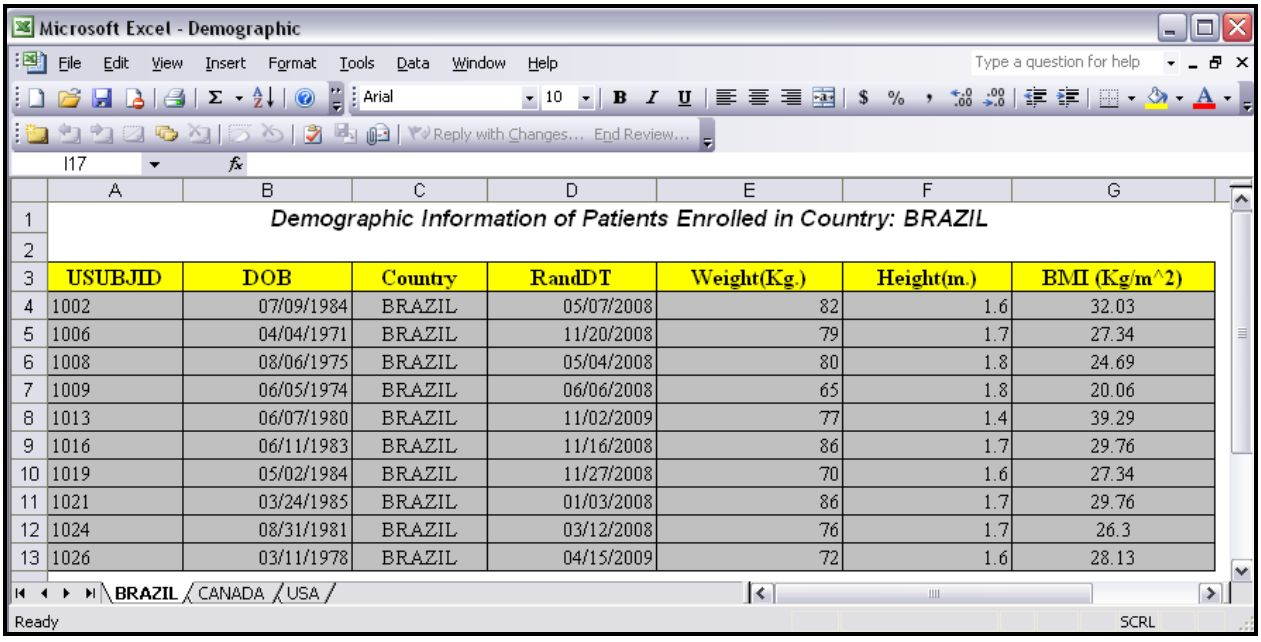
NAME THE WORKSHEETS USING BY GROUP VARIABLE VALUES

ODS gives a unique name to each resulting workbook as per the requirements of Excel. You can see default names of worksheet on all the figures that we have gone through so far. We can overwrite the default name of the worksheet by the current value of BY GROUP variable. For that we have to use **SHEET_INTERVAL** option and set its value to 'BYGROUP' along with the use of **SHEET_LABEL** options and setting its value to missing. (' ').

WRAP TEXT WITH CHANGE IN WIDTH OF CELL

Let's say after the XML output is generated, somebody tries to change the width of the columns manually. In that case, if the data length is more than the width of cell in that column, the data value will be truncated. To prevent this we usually apply wrapping of text in Excel for that column so that the height of each cell is adjusted according to the length of the data value within the cell. We can apply this wrapping of text in resulting XML spreadsheet by using ExcelXP tagset option **AUTOFIT_HEIGHT** and setting its value to 'YES'.

With the use of above discussed options, we can have a final output like Figure 7. Notice the difference in title and workbook names in Figure 7 from Figure 6. How to specify and use above discussed options in SAS is explained in the code given at the end of this paper in Appendix section.



USUBJID	DOB	Country	RandDT	Weight(Kg.)	Height(m.)	BMI (Kg/m^2)
1002	07/09/1984	BRAZIL	05/07/2008	82	1.6	32.03
1006	04/04/1971	BRAZIL	11/20/2008	79	1.7	27.34
1008	08/06/1975	BRAZIL	05/04/2008	80	1.8	24.69
1009	06/05/1974	BRAZIL	06/06/2008	65	1.8	20.06
1013	06/07/1980	BRAZIL	11/02/2009	77	1.4	39.29
1016	06/11/1983	BRAZIL	11/16/2008	86	1.7	29.76
1019	05/02/1984	BRAZIL	11/27/2008	70	1.6	27.34
1021	03/24/1985	BRAZIL	01/03/2008	86	1.7	29.76
1024	08/31/1981	BRAZIL	03/12/2008	76	1.7	26.3
1026	03/11/1978	BRAZIL	04/15/2009	72	1.6	28.13

Figure 7: Final Output from ExcelXP Tagset with "MYXLPrinter" Style

CONCLUSION

ODS ExcelXP tagset provides an elegant solution of creating multi-sheet Excel workbook which is very powerful in feature and functionality, yet easy to learn and implement in real work life. With the use of ODS styles and style overrides, it gives complete control over the appearance of resulting XML spreadsheet which in turn works like a charm with Excel also. There are numerous options available with ExcelXP tagset and discussion on use of those options is out of scope for this paper. But you can get the list of available options, their description and information on their default values from the SAS website: http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_help.html

REFERENCES

- Base SAS ODS MARKUP Resource at:
  http://support.sas.com/rnd/base/ods/odsmarkup/
- SAS 9.1.3 Online Documentation for Proc Template at:
  http://support.sas.com/onlinedoc/913/docMainpage.jsp?%20topic=odsug.hlp/
- Base SAS Quick reference to TAGSETS.EXCELXP at:
  http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_help.html
- DelGobbo, Vincent. 2009. “More Tips and Tricks for Creating Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS®”. Proceedings of NESUG 2009 Conference.
- SAS Usage Note 11206 on Formats and Informats for Date, Time and DateTime at:
  https://support.sas.com/kb/11/206.html

CONTACT INFORMATION

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

Name: NIRAJ J PANDYA
Phone: 201-936-5826
E-mail: npandya@elementtechnologies.com

APPENDIX

SAS CODE WITH MYXLPrinter STYLE DEFINITION:

```
/* Generate Custom Style MYXLPrinter */

 Proc Template;
      Define style styles.MYXLPrinter;
               Parent = styles.printer;

           style header from header/
                font_face="Times New Roman"
                font_size=12pt
                background=yellow
                just=center;
```

```
           style data from data/
                 font_face="Times New Roman"
                 font_size=11pt
                 background=cxB0B0B0;

           style data_left from data/
                 just=left;

           style data_center from data/
                 just=center;

           style data_date from data/
                 tagattr='format:mm/dd/yyyy type:DateTime';

           style data_bmi from data_center/
                 tagattr='formula:=round(((RC[-2])/(RC[-1]*RC[-1])),2)';
       End;

 Run; Quit;

/* Generate separate worksheet for Each Country */

ODS listing close;

ODS tagsets.ExcelXP path = 'Output Location' file='Demographic.xml'
                    style = MYXLPrinter;

ODS tagsets.ExcelXP options (suppress_bylines='YES'
                             embedded_titles='YES'
                             sheet_label=' '
                             sheet_interval='BYGROUP'
                             autofit_height='YES');

Proc sort data = work.dmgrphc;
     By country;
Run;

Title1 "Demographic Information of Patients Enrolled in Country: #BYVAL(Country)";

Proc print data = work.dmgrphc noobs label;
     By country;
     Pageby country;
     Var usubjid /style(data)=data_left;
     Var dob/style(data)=data_date;
     Var country/style(data)=data_center;
     Var randdt/style(data)=data_date;
     Var wt ht;
     Var bmi/style(data)=data_bmi;

     Format dob randdt yymmdd10. ;
Run;
Quit;
ODS tagsets.excelxp close;
```