You Use SAS®, Your Boss Uses Excel. Guess Where Your Results Are Going to Appear! Using ODS to Create Your Results in Excel

William E Benjamin Jr, Owl Computer Consultancy, LLC, Phoenix, Arizona

ABSTRACT

This paper digs into the use of ODS on the SAS® side to prepare Excel spreadsheets directly. SAS code and macros will be demonstrated to create multi-page Excel spreadsheets, giving the programmer the ability to build tools to exploit data and display information for the end user. As a result, the SAS programmer will be able to create spreadsheets for distribution, automate the standardization of reports and the building of powerful spreadsheet applications with access to information built by offline SAS processes that the end user can refresh.

INTRODUCTION

This paper will help someone beginning to use ODS Tagsets and Style Sheets figure out how to get started. In the five years since this author first described this process many advances have occurred in computer software. SAS Version 8 is no longer the version of choice, and Microsoft Windows XP is no longer the up and coming OS for New PC's. But, programmers are still being told to "Just e-mail me a spreadsheet with the data." And it is happening far more often. In the days when Programmers worked on Mainframes, and Managers worked on PC's, the Programmers never got those requests, and Managers keyed in the data themselves. But, today you get "**THE REQUEST**" for "**THE SPREADSHEET**", on a daily basis, and the files are always too big to re-enter, and it has to look good too.

Now what? Even though the WUSS-11 proceedings from 2003 are not available online, this author will honor e-mail requests for copies of the article sent to the contact address below (for some limited amount of time). Therefore, this version will not include some of the basic parts of that paper from the 2003 SAS Solutions section of WUSS-11. The ODS section of the 2003 paper will be expanded here as the subject of this paper. This paper will explore how to find, set-up, and use one of the tagsets that have been created by SAS, to aid users in transferring data and formatting to Microsoft Excel workbooks. The end result will be learning how SAS code and ODS can be used in conjunction with XML Tagsets to produce Excel Multi-sheet worksheets that look good.

PROCESS ONE (THE TRIVIAL SOLUTION)

But, as with most trivial solutions, you have few options and little control over the output file produced. In this case all variables are output, in the order that they were defined, with the variable name as the title of the variable for Excel. The previous paper describes a method similar to the following to use ODS to move SAS data into Excel. This also showed no way to add any formatting or titles to the spreadsheet, because the output from this process has to be reprocessed by Excel when the file was opened.

```
ods csv body='f:\WUSS_2010\Shoes_1a.csv';
proc print data=SASHELP.SHOES ;
run;
ods csv close;
```

Here is a sample of the output file:

```
"Obs","Region","Product","Subsidiary","Stores","Sales","Inventory","Returns"
"  1","Africa","Boot","Addis Ababa","12","     $29,761","    $191,821","        $769"
"  2","Africa","Men's Casual","Addis Ababa"," 4","     $67,242","    $118,036","      $2,284"
"  3","Africa","Men's Dress","Addis Ababa"," 7","     $76,793","    $136,273","      $2,433"
.
-       - More data -
.
"393","Western Europe","Sport Shoe","Rome","14","      $9,969","     $74,848","        $549"
"394","Western Europe","Women's Casual","Rome"," 2","     $19,964","     $62,256","        $954"
"395","Western Europe","Women's Dress","Rome","16","    $106,676","    $389,861","      $3,160"
```

Now, raise your hand if you have done that! Then, go hang your head and stand in the corner if you liked **"ALL"** of the output results (or know anyone who did). Excel seems to have a mind of its own when it comes to formatting input data fields. Try sending a character value that has all numbers with a leading zero. Using CSV output this way, you have no formatting control.

PROCESS TWO (Make SAS® do something new for you, the "Hello World" project)

The WUSS 2007 (and SGF 2007) conference included a Hands-On-Workshop titled "Creating Multi-Sheet Excel Workbooks the Easy Way With SAS®" presented by Vince DelGobbo. He identified a SAS supplied XML tagset called ExcelXP and created a custom stylesheet called XLsanPrinter. In his paper he described how to use and modify the ExcelXP tagset to create Excel spreadsheets. In February 2008 the tagset was found at: http://support.sas.com/rnd/base/ods/odsmarkup/excltags.tpl

STEP ONE – WHAT DO YOU NEED?

First, find out what you already have installed.

The test system in use for this example is the SAS ® Learning Edition version 4.1. This software uses version 9.1.3 Service Pack 4. (I use this software because I like to have the same platform available as students.) The web site that describes Base SAS ODS Markup resources[1] declares that the 'following ODS statement will provide complete details of all the new features:' Well that command listed on the SAS web site above worked as advertised even on the SAS ® Learning Edition version 4.1. version.

Executing the following command :

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

Produced the following message (proceeded by eight pages of printed help file text):

```
================================================================================
NOTE: This is the Excel XP tagset (SAS 9.1.3, v1.28, 08/29/05). Add options(doc='help') to the
ods statement for more information.
```

Of course, your system may respond differently. But hopefully some message will appear. In 35 years of programming, a message (no matter how cryptic) has always been a sign of hope. Getting the message to change is an indication of progress (regardless of the direction – forward or backward).

The SAS web site above contained a link (in February 2008) that allowed the user to click on a link to ExcelXP. This action opens a file with SAS code to define a tagset. A few lines into the code the following message appears:

```
log_note = "NOTE: This is the Excel XP tagset (SAS 9.1.3, v1.70, 06/05/07). Add
options(doc='help') to the ods statement for more information.";
```

This message indicates that 42 version changes have been made in less than two years. (an average of one every two weeks, but grouped into packages). Changing your computer should always be done carefully, with software from a trusted source, only after making sure your system can support the new programs (and if you are using a company owned computer – after you get permission to make the changes). Finding "SAS 9.1.3" in messages about the installed SAS system, the old tagset, and the new tagset look like messages indicating forward progress.

Second, get the free new software.

One quick way to do that is to click on the link, press the "ctrl"+"A" keys (two keys) and then press the "ctrl"+"C" keys (also two keys) to capture the full file from the web site. Then with SAS running and the curser in a blank editor screen press "ctrl"+"V". This will copy the code (over 14,000 lines) into the editor where it can be saved onto your disk in a file of your choosing. Before doing anything else, you should be advised that new software should only be added to you computer if your system has been "BACKED-UP" to what ever level you are comfortable, because any change could render any computer inoperable. On the target system this ExcelXP code ran in almost no time and left the following messages in the log:

```
14160
14161      end;
NOTE: Overwriting existing template/link: Tagsets.Config_debug
NOTE: TAGSET 'Tagsets.Config_debug' has been saved to: SASUSER.TEMPLAT
14162
14163  run;
NOTE: PROCEDURE TEMPLATE used (Total process time):
      real time           0.64 seconds
      cpu time            0.64 seconds
```

If you do not know what “SASUSER.TEMPLATE” contains, a quick search will lead you from any interactive window by clicking on “VIEW” and then “RESULTS”. Then, from the “RESULTS” window you can click on “VIEW” again, and then click on “TEMPLATES”. This will show you an explorer window that shows paths to the “Sasuser.Templat” and “Sashelp.Tmplmst” data trees. Expanding a data tree will show individual styles or tagsets in the far right pane. Note if you do not have any user defined styles, the SASUSER screen may be blank, but the SASHELP leg of the tree should have some styles or templates to examine. But, back in the editor window by resubmitting the options(doc="help") command from above will show a version of the now familiar message below (and 10 pages of help text):

```
================================================================================
NOTE: This is the Excel XP tagset (SAS 9.1.3, v1.70, 06/05/07). Add options(doc='help') to the
ods statement for more information.
14167  run;
```

Third, Read the instructions.

By downloading and executing the code above for the ExcelXP tagset it is now installed into your SASUSER environment, test this by exiting and reentering SAS and re-running the (doc="help") command from above. Next try replacing “Help” with “Quick”, “Settings”, “Changelog”, or “All” and re-running the (doc=”Help”) command from above. Now before proceeding, read the output. A lot of effort was put into producing it. The implementers must have thought it would be helpful to someone, but only if you read it. You never know what you might find.

Finding this treasure trove of information is neat, but the value is limited until you can figure out what some of it means. Furthermore, what are all of these printouts trying to tell me, and how can I use this stuff? What is a tagset or a stylesheet? Everyone has heard of them, who do you know that can really explain it to you in depth? Well, style sheets or Cascading Style Sheets (CSS) first came on the scene in a limited form in 1996 with Microsoft Internet Explorer 3. CSS is a language used in a markup language (like HTLM or XML) to describe colors, fonts, and layouts for the presentation of the output.

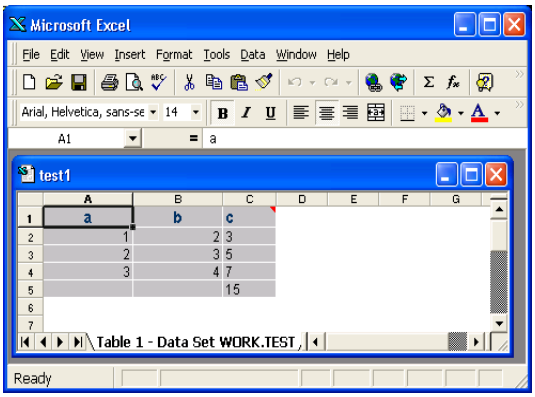
Style sheets and Tagsets have a language and syntax all to their own. Many books have been written about that subject. A quick check of the Base SAS® Help system finds that the code soon to be described below works because the PROC PRINT procedure allows more than one “VAR” statement. Each and every one of them can have its own style sheet option applied so that your output can look different for every variable that you print. Giving each variable a separate “VAR” statement allows you to have a different type of formatting for each variable in the printout. The print procedure below has available a command that allows the specific formatting of individual or groups of variables, in the order they are listed, in the order of the “VAR” statement or statements, with the following syntax. The syntax is listed but not described here in detail.

VAR variable(s) </ STYLE <(location(s))> =<style-element-name><{style-attribute-specification(s)}>>;

Fourth, execute the “Hello World” project.

As mentioned before, messages are a good thing. They show progress, regardless of the direction in which progress is being made. In thirty five years of programming this author has begun using many new languages with a simple technique. The usual first task is simple, and involves the “can I really make this work” concept of programming called a “Hello World” project. Other people call it different things, but the concept is the same, if a known text string can be moved from one place and then found or retrieved in a second place then the project is a success. So, we will start with this simple project, put the words “Hello” and “World” into an Excel spreadsheet, and find them.
The 2003 paper used DDE commands to accomplish this goal. The “Help” text printed above (and read by all) provided the ODS code to accomplish the task here using ODS. (Please note that the help text may have an extra double quote and an extra semicolon on the line that contains the words “Hello World”. But getting rid of the messages generated by those characters is another good thing. The line should end with a “}”). Note, line numbers are included for discussion only, do not use in a SAS program. Once again this code comes directly from the help output of the Excel tageset code downloaded above (DelGobbo V (2007)).

```
(01)      ods tagsets.excelxp file='test1.xml' options(zoom='75');
(02)      data test;
(03)      length a b 8 c $20;
(04)      input a b c $;
(05)      cards;
(06)      1 2 3
(07)      2 3 =RC[-2]+RC[-1]
(08)      3 4 =RC[-2]+RC[-1]
(09)      . . =SUM(R[-3]C:R[-1]C)
(10)      ;
(11)     run;
(12)      Options missing = ' ';  * added to hide the periods in the spreadsheet;
(13)      proc print noobs;
(14)          var a b;
(15)          var c / style(head) = {flyover="Hello World"}
(16)                  style(data) = {cellwidth=50pt};
(17)      run;
(18)      ods tagsets.excelxp close;
(19)      run;
(20)      Options missing = '.';  * added to reset  the default missing display;
```



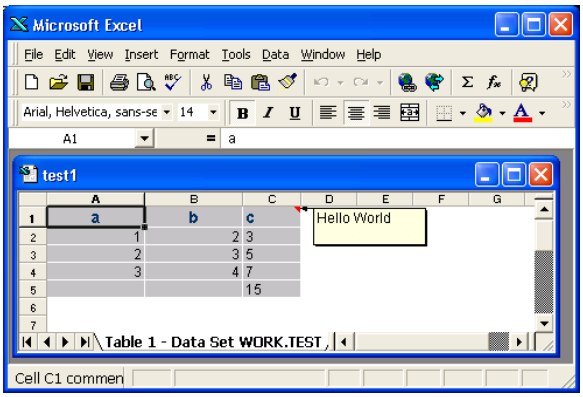


Figure 1. (Left) Excel output from the "Hello World" project code. (before looking for the message).
Figure 2. (Right) Excel output from the "Hello World" project code. (after finding the message with curser over "C1").

The output looks something like the left picture above. But, it looks like something is missing. The words "Hello World" are not visible anywhere on the spreadsheet. So in a fit of frustration you wiggle the mouse all around the screen in an effort to make something happen. Then, WOW!!! all of a sudden your mouse flies over cell "C1" the one with the little red triangle in the upper right corner. Then like magic the yellow box with the words "Hello World" appears, and you know you have succeeded with the project, but do not know how. (see the right picture)

PROCESS THREE (Making this do real work)

The joy and excitement of seeing this first spreadsheet will pass, as soon as your boss walks by and asks what you are doing. You, of course relieve his/her fears by declaring that you are getting their spreadsheet ready, without a clue as to how you will do that. So, let's look at what actions the code processed. For this exercise, line numbers were inserted into the code listed above, and two extra "OPTIONS" statements were added to the original downloaded code.

Line 1: Start the process by opening an ODS output stream to send data to the file "Test1.xml". The "Tagset" called "excelxp" is used to do all of the hard work. A little bit of research will bring to light the fact that tagsets use stylesheets to help them operate. Since none was declared a default stylesheet was used.

Line 2: Start a SAS data step to save some data to output into a SAS work file called TEST.

Line 3: Specifically declare three BASE SAS® DATA STEP variables, A and B as numeric variables with a length of 8 bytes, and C as a character variable with a length of 20 bytes.

Line 4: Read variables A, B, and C from the input list of data values that follows.

Line 5: Declare the beginning of a stream of data lines, with the data values separated by spaces. Once again using the default behavior of the input command.

Line 6-9: Data values, three on a line with 4 lines of data. (we will see more of this later)

Line 10: The end of the data values.

Line 11: This run command builds the SAS Dataset TEST in the work libname location.

Line 12: This line was added so that the output of the variables with missing values (a and b in the third input row) will show spaces instead of periods in the Excel output spreadsheet.

Line 13: Start the Print procedure and suppress the observation numbers on the output listing. This PROC PRINT statement will actually open two output files in the output window. One for the standard output and one for the xml output listing. This code also uses two "VAR" commands.

Line 14: The first "VAR" command controls the output of variables A and B. These are output as numbers in columns 1 and 2, with no special processing.

Line 15/16: The second "VAR" command controls the output of variable C with special formatting, controlled by the style sheet. Furthermore, the data that is output is used to issue commands to Excel. Let us examine the commands closely. Assuming that everyone knows that the "var c" part of the command says write the value of variable a to the output , we will move to the part of the line after the "/". The slash separates the var command and the stylesheet commands from each other. The part on line 19 reads as follows:

```
style(head) = {flyover="Hello World"}
```

Notice that the Excel output has a header row that has the variable names in the header. (Prove it to yourself, change A to "Eat', B to "AT", and C to "JOES" and try it again to see what you get.) Also note the header of Column C is told by the stylesheet that the message "Hello World" should be displayed when the mouse "Flies" over the header cell. (see the code "style(head)") The next line has additional data that looks like this:

```
style(data) = {cellwidth=50pt}
```

This second style command is directed at the data elements of the spreadsheet, changing the size of the cells for column "C" to be 50 points (about .7 inches). Did you notice that the width of column C was smaller than the others? Everyone will also take note of the fact that this command (cellwidth) was described in the help document output above (and read by all).

Line 17: Finishes the PROC Print routine.

Line 18: Closes the ODS output.

Line 19: The final RUN statement, used by the author out of habit.

Line 20: This line was added and resets the way that missing values are displayed by SAS.

Now try these simple changes on your own:
Now lets get back and look at the initial ODS command again. Notice that it was the following :

```
(01)      ods tagsets.excelxp file='test.xml' options(zoom='75');
```

Also notice that the spreadsheet did not appear as a full size spreadsheet. Well that was controlled by the tagset option "ZOOM" by placing other tagset options within the command more changes can be made. Try placing other commands into the parentheses like:

```
sheet_name='New Sheet Name' or row_heights = '40'
```

The last thing to look at is the numbers in column “C”. The data that was read from the input list of cards only included one number for Column “C”. It also had three character strings that translated into Excel formula commands. And a character string that was only a number. Well Excel translated the number “3” (see Line 9) into a numeric value in the new spreadsheet, and the formula entries into commands. Lines 10 and 11 have the same formula (=RC[-2]+RC[-1]) . Which tells Excel to make the current cell equal (=) to the value of the cell on the same row two columns to the left (RC[-2]) plus (+) the value of the cell on the same row one column to the left (RC[-1]). In other words 2 + 3 = 5 and 3 + 4 = 7. The formula on Line 12 totals the values of column “C”, by making the current cell equal (=) to the total (sum) of the values starting in the same column three rows up (R[-3]C) through (:) the values in the same column one call up (R[-1]C). Creating the value 3 + 5 + 7 = 15. Try changing the plus sign (+) to an * to see what the result becomes.

PROCESS FOUR (Using ODS and XML Tagsets to create Multi-sheet Excel Worksheets)

The next thing to do is find out how easy it is to make this do real work you that need done. BY adding a few extra options to the tagset option command list, the spreadsheet shown below was created. Note is has a page for each region, titles on the top of the sheet, and sums on the bottom, with very little extra effort.

```
ods tagsets.excelxp file='test3.xml' options(zoom='75' sheet_interval='bygroup'
sheet_label='By 'row_heights = '40');

proc sort data=sashelp.shoes;
by Region Product Sales ;

proc print noobs;
  by region ;
  sum sales;
     var product;
     var sales  / style(data) = {cellwidth=50pt };
run;
ods tagsets.excelxp close;
run;
```

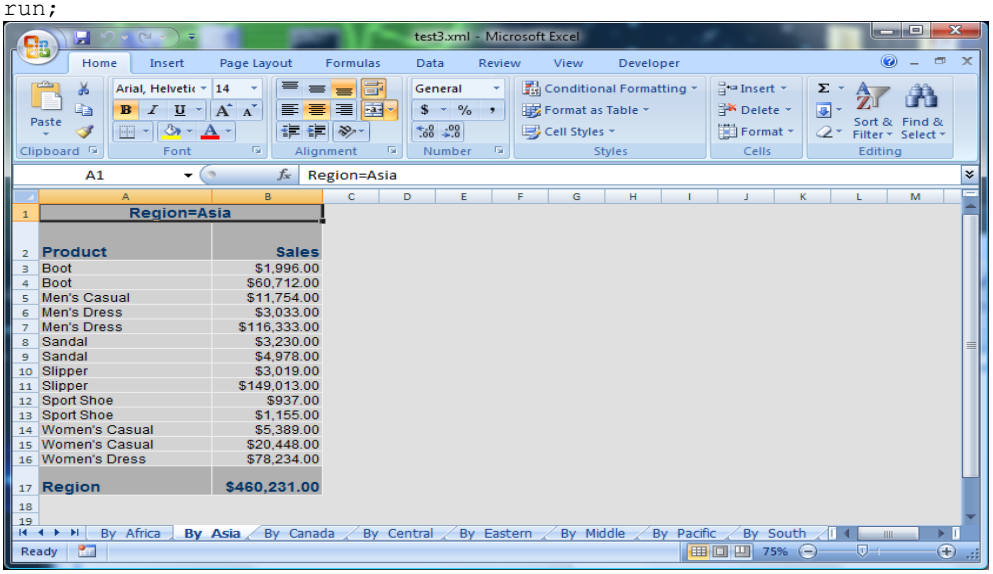


Figure 3. Spreadsheet with multiple pages created with a “BY” statement in PROC PRINT. (Note the “By Asia” page is not the first page, but is shown because it is a small page.)

PROCESS FIVE (Making it pretty – For the advanced student to study)

The next step is to take the extra effort to look at what can be done to set your work apart from the person in the next cube, after all it is your boss that wants the spreadsheet. How are the raises determined in your company? Let’s take a few minutes to expand on the last effort. Figure 3, shows the results of a PROC PRINT output that makes simple work of separating regional data into spreadsheets. The default backgrounds and colors – could be improved.

Step One – Table of Contents with hyperlinks to the pages.

Since you are really trying to impress your boss with a workbook that is useful and pretty, lets add a table of contents to the workbook, with hyperlinks to the pages. (This was only tested with Excel 2007, It may not work with other versions. But let's keep coding.) There is not enough room in this forum to describe the process in detail, But the code has enough comments to help an enterprising programmer.

```
*******************************************************************************;
** Start ODS and identify output file and style                               **;
*******************************************************************************;
ods tagsets.excelxp file='Z:\WUSS_2010\test5.xml' style =Gears ;
*******************************************************************************;
** Pick an input file and sort it into work                                   **;
*******************************************************************************;
proc sort data=sashelp.shoes out=shoes1;
by Region Product Subsidiary Sales ;
run;
*******************************************************************************;
** cleanup the by variables if embedded spaces and special characters        **;
*******************************************************************************;
*remove embedded spaces and special chars from the variable used as the
sheet name (Primary By variable);
data shoes1;
       set shoes1;
       x = 1;
       do until (x eq 0);
          x = index(left(trim(region)),' /') ;
          region = translate(left(trim(region)),'_',' ');
          region = translate(region,'_','/');
       end;
       drop x;
run;
*******************************************************************************;
** Give the first sheet a name                                                **;
*******************************************************************************;
ods tagsets.excelxp options (sheet_label = 'Table of contents') ;
*******************************************************************************;
** set up a title                                                             **;
*******************************************************************************;
title1 'Worldwide Show Sales by Region';
*******************************************************************************;
** Build a table of contents with brute force method.                         **;
*******************************************************************************;
data comments (keep=comments);
retain counter 1;
attrib comments length = $ 90;
comments = '_______________________________'; output;
comments = 'This Excel Workbook contains'; output;
comments = 'several sheets with one for '; output;
comments = 'each region of the world    '; output;
comments = '_______________________________'; output;
comments = '     Regions displayed      '; output;
comments = '_______________________________'; output;
comments = 'Left Click title to see data'; output;
comments = '_______________________________'; output;

* output one record for each region;
do until (eof);
   set shoes1 end=eof;
   by region;
      if first.region then do;
```

```
         counter + 1;
         *Build an Excel hyperlink to a new page in same workbook;
         comments = '=HYPERLINK("[test5.xml]'        ||
                    left(trim(put(region,$char30.))) ||
                    '!A1", "'                        ||
                    left(trim(put(region,$char30.))) ||
                    '")';
         output;
      end;
end;

* write text at bottom of table of contents page;
comments = '_____________________________'; output;
comments = '   CAUTION - CHANGING THE   '; output;
comments = '    NAME OF THE WORKBOOK    '; output;
comments = '     MAY INVALIDATE THE     '; output;
comments = '   HYPERLINKS IN THE FILE   '; output;
comments = '_____________________________'; output;
stop;

*******************************************************************************;
** Send the table of contents page to ODS -- i.e. to the Excel spreadsheet **;
*******************************************************************************;
proc print data=comments noobs label;
run;
*******************************************************************************;
** Reset ODS options to increase fonts and row sizes, and create a page    **;
** for each region. Note Excel seems to ingnore the leading space that is  **;
** generated by the sheet_label option below                               **;
*******************************************************************************;
ods tagsets.excelxp
       options(zoom='75'
       sheet_interval='bygroup'
       sheet_label=' '
       row_heights = '40,20'
       Gridlines = 'Yes'
       )       ;
*******************************************************************************;
** send the real data to ODS -- i.e. the spreadsheet                       **;
*******************************************************************************;
proc print data=shoes1 noobs;
  by region ;
  sum sales;
     var product Subsidiary  / style(data) = {font_size=14pt};
     var sales  / style(data) = {cellwidth=50pt font_size=12pt};
run;
*******************************************************************************;
** Close up and go home.                                                   **;
*******************************************************************************;
ods tagsets.excelxp close;
title;
run;
```

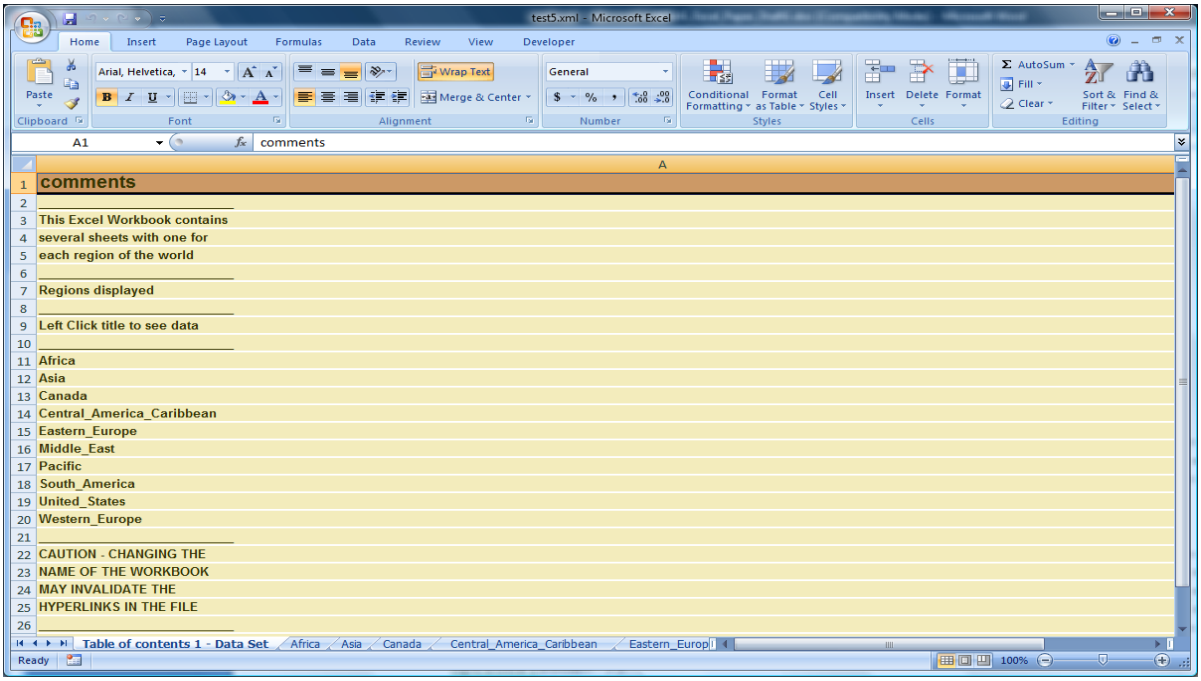


Figure 4. Note the new look of the “GEAR” style and the listing of all of the regions in the file. This type of a first page helps the “Boss” know what you are providing. And gives you a chance to add instructions (see line 9) and warnings like lines 22-25.

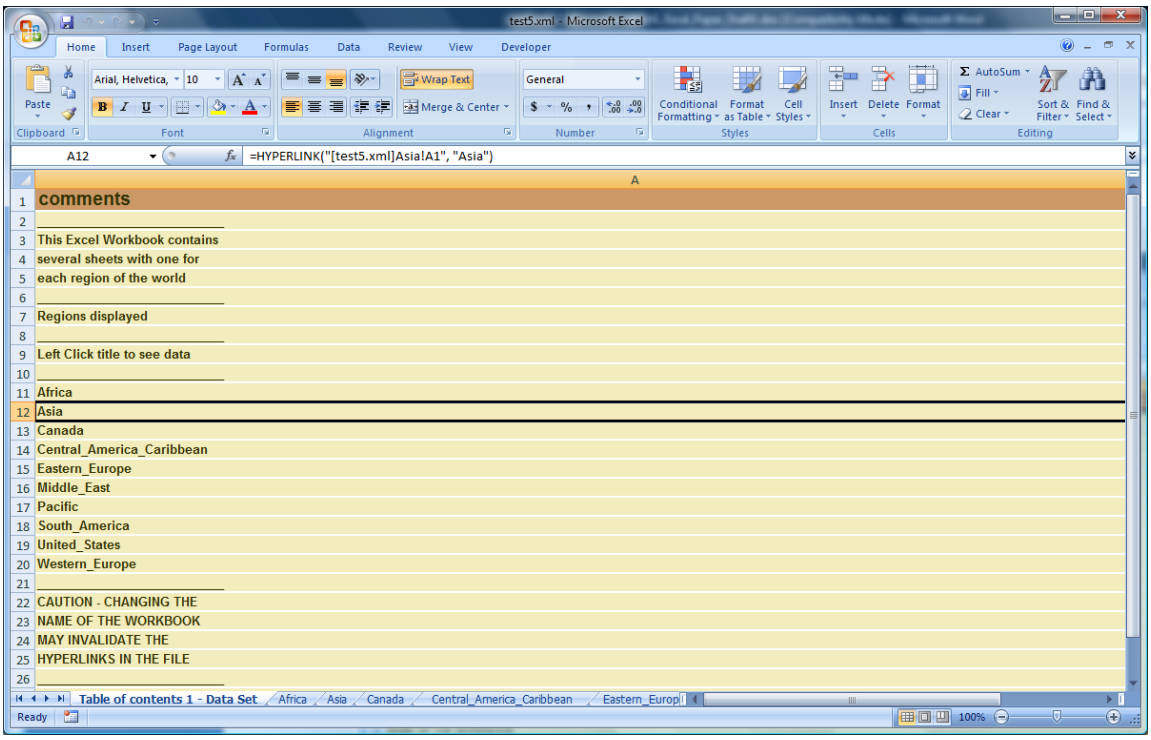


Figure 5. Look at line 12 – it is highlighted – Then look at the function bar, see the hyperlink. ODS wrote that there from a proc print step. A simple left click will get you to the next screen. You can tell me that will not impress your boss, but I am not buying it.

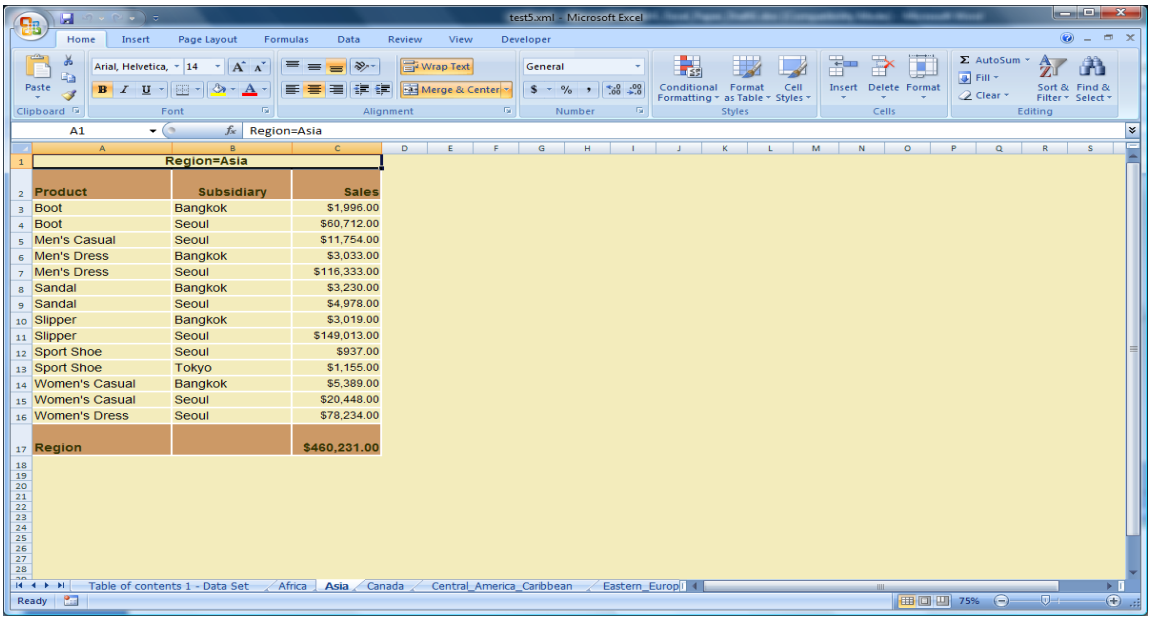


Figure 6. One click and the data is displayed directly from the “Table of Contents” page. Someone that uses Excel a lot will be impressed by the extra time it took you to add the extra features. Because, of course, these take time. But you only you will know how little time it took. (After you get the first one set up of course)

CONCLUSION

The hard part is getting started. The fun part follows, by reading the EXCELXP help listing and experimenting with options, in no time you will be creating spreadsheets that will amaze not only your friends but also your boss. The neat part is that you do not have to do it in Excel, you can still use Base SAS® to do this complicated work. Far too many options and procedures allow access to these Tagsets and style sheets to attempt to describe more here. This paper is intended to help you get started, by showing how to setup your system to use the ExcelXP tagset and how to push just a little father into automating your Excel output routines.

REFERENCES

DelGobbo V (2007), “Creating Multi-Sheet Excel Workbooks the Easy Way With SAS®” Conference Proceedings of the SAS Global Forum, 2007, CD-ROM, Paper number 120. Also available at the SAS Support WEB Site: http://support.sas.com/rnd/papers/sgf07/sgf2007-excel.pdf

ExcelXP tagset (SAS 9.1.3, v1.70, 06/05/07) Available February 15, 2008 at: http://support.sas.com/rnd/base/ods/odsmarkup/excltags.tpl

CONTACT INFORMATION

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

Name William E Benjamin Jr
Enterprise Owl Computer Consultancy, LLC
Address P.O. Box 42434
City, State, ZIP Phoenix, AZ, 85080
Work Phone: 602-942-0370
Fax: 602-942-3204
E-mail: wmebenjaminjr3@juno.com
Web: www.OwlFunding.com

1 - http://support.sas.com/rnd/base/ods/odsmarkup/index.html