Paper 1702-2016

Introduction to SAS® ODS ExcelXP Tagset: Exporting Formulas and Tables

Veronica Renauldo, Grand Valley State University, Allendale, MI

ABSTRACT

SAS® ODS ExcelXP Tagset offers users the ability to export a SAS® dataset, with all of the accompanying functions and calculations, to an Excel worksheet. For industries, this is particularly useful as not everyone is a SAS® programmer but would like to have access to and manipulate data from SAS®. The ExcelXP tagset is one of several built in templates for exporting data in SAS®. The tagset gives programmers the ability to export functions and calculations into the cells of Excel. There are several options within the tagset which enable the programmer to customize the Excel file. Some examples of options allow the programmer to name the worksheet, style each table, embed titles and footnotes, and export multiple data tables to the same Excel worksheet.

INTRODUCTION

What is a tagset? An ODS tagset is “a type of template that defines how to generate a markup language output type from SAS® output (SAS, n.d.)”. Users can create tagsets using the TEMPLATE procedure. In other words, tagsets are predefined templates to output SAS® data. If a predefined tagset does not encompass all of the qualities desired by the user, the user is able to modify the tagset via PROC TEMPLATE. The code below outputs a list of all of the built in tagsets available on a user’s version of SAS®.

```
PROC TEMPLATE;
   LIST TAGSETS;
QUIT;
```

The ExcelXP tagset is used to create Microsoft® SpreadsheetML XML files. The use of a tagset requires at least SAS® 9.1 and Microsoft® Excel 2002 (Vanam, Karidi, & Dodlapati, 2010). Microsoft® SpreadsheetML XML files can be thought of as a programmer’s version of an Excel file. The XML file that is created by the ExcelXP tagset is a Microsoft® file type that was “designed for managing and sharing structured data in a human-readable text file…Using XML application designers can create their own customized tags, data structures, and schemas (Vanam, Karidi, & Dodlapati, 2010)”. This file can be opened in Excel (or if Excel is the default on the computer then it will be automatically open using Excel). Instead of opening a workbook like one usually would when using Excel, the XML file is a coded version of a spreadsheet. The programming code for creating an Excel spreadsheet using an XML format can be rather tedious, especially for those who do not do this often. Thus, the tagset ExcelXP provides a bridge between programming in SAS® and using Excel. The tagset is able to take the SAS® dataset, its attributes, and various Excel options to create a XML file that can be opened and used like any ordinary Excel file. Not everyone in a business is SAS® savvy, but many people are familiar with Excel, making this bridge between the two programs extremely beneficial.

BASIC EXCELXP TAGSET SYNTAX

The basic syntax using REPORT procedure (PROC REPORT) for the ExcelXP tagset is shown below with explanations. The tagset can also be used with PRINT procedure (PROC PRINT) which will be illustrated in the examples section.

```
❶ODS tagsets.ExcelXP
PATH='user path'
❷FILE='filename.xml'
STYLE=style_of_XML_output ❸options(user_options='value');

PROC REPORT ❹DATA=sas_dataset;
      ❺COLUMNS <list of variables to be outputted >;
      ❻DEFINE ❼new_variable_name /
❼STYLE={tagattr="formula:❽calculations to create formula"};
RUN;
❾ODS tagsets.ExcelXP CLOSE;
```

❶The *ODS* statement turns on the output delivery system. The *tagsets.ExcelXP* tells SAS® that the ExcelXP tagset will be used as the template for the outputted data. Additionally, this tells SAS® to create an XML file.

❷ The *FILE=* statement tells SAS® the file location, name, and extension (XML). The file location/name/extension has to be in either single or double quotes. The FILE statement can either be a full direct file name or, when used with the *PATH=* statement, it can just be the file name with the XML extension.

❸ The ExcelXP tagset has several options available to the user. By specifying the *ODS tagsets.ExcelXP*, the FILE statement, and *OPTIONS(*`DOC="help")`, a list of all available options for the tagset will be outputted to the SAS® log. The list will contain option names, default values, and a description of what the option does. An example of this is shown below.

```
ODS tagsets.ExcelXP FILE='full_direct_file_name.xml'

options(doc="help");

ODS tagsets.ExcelXP CLOSE;
```

A few of the available options for this tagset, along with descriptions, can be found in the following *ExcelXP Tagset Options* section.

❹The *DATA=* defines the SAS® dataset that the user would like to output.

❺The *COLUMNS* statement within PROC REPORT defines all of the variables that the user would like outputted to the XML file. If any variables are going to be created within PROC REPORT they need to also be listed in the columns statement.

❻The *DEFINE* statement describes how a variable will be displayed or computed within PROC REPORT (SAS, n.d.). For this article the DEFINE statement is used to construct formulas for variables in the SAS® dataset for use in Excel.

❼The *STYLE={TAGATTR=}* style attributes define the formulas for variables in the XML file. The *TAGATTR* is a style attributed which is comprised of the variable calculations when using the keyword *FORMULA*. This is explained in greater detail in the following paragraphs.

❽The formulas exported from SAS® to the XML file need to be written in a row column (RC) format along with Excel syntax for any functions. This is explained in more details in the following paragraphs.

❾This statement closes the ODS and outputs the XML file to the desired destination.

To export formulas to Excel using the ExcelXP tagset, the *STYLE* option and *TAGATTR* style attribute are required. The *STYLE* option “specifies the style element to use for the specified locations (SAS, n.d.)” for PROC REPORT. This option is used within the *define* statement of PROC REPORT. For PROC PRINT, the *STYLE* option specifies one or more style overrides to use for all columns that are created by a VAR statement (SAS, n.d.)”.

The *TAGATTR* style attribute is used to send instructions to the cells within the Excel spreadsheet. "The syntax to export a formula using the TAGATTR= style attribute is *"formula: "* after the equals with the formula you want exported placed after the colon before the closing quotation mark (Skopic, n.d.)". There should be no space between the *formula:* and the Excel formula. All formulas used should be valid in Excel. To create formulas for Excel, the row and column (RC) reference style is used. Excel formulas consist of cell references such as A1 or B21 but this reference style is not utilized by this tagset. The cell A1 would be written as R[1]C[1], where R[1] is the first row in the spreadsheet and C[1] is the first column in the spreadsheet (Skopic, n.d.). If there are brackets after R and after C then that specific cell is used in all calculations for the variable (an absolute reference) (Skopic, n.d.). This is similar to the *$* absolute reference style within Excel (Skopic, n.d.). For example, a formula containing R[1]C[1] would use that cell for all calculations. Alternatively, if brackets fall after R and C, RC[1], then that column is used as a relative reference (Skopic, n.d.). This is similar to writing a formula in a cell in Excel and copying it down a column so that the reference cell changes with the row. The user needs to be aware of the order in which the columns are being outputted to the Excel file, as this will affect each variable's formula within the *TAGATTR* style attribute. The examples section will go through how cell/column references are constructed within formulas.

The ExcelXP tagset can be used with both PROC REPORT and PROC PRINT. Formulas and calculations can be connected with a specific variable using the *define* statement in PROC REPORT and the *var* statement in PROC PRINT.

EXCELXP TAGSET OPTIONS

There are over 60 different options that can be specified in the *options(option='value')* statement for the ExcelXP tagset. These options control the appearance of the Excel workbook. Below is a list of some of the commonly used options from the tagset.

- *Embed_titles_once*: will embed the titles into the top of each worksheet. The default value for this option is 'No'.
- *Gridlines*: turns on the gridlines in Excel. The default value for this option is 'no'.
- *Frozen_RowHeaders*: freezes the header row. The default value for this option is 'No'
- *Sheet_name*: This option names the workbook sheet (name ≤ 32 characters). The default value for this option is 'None' indicating that SAS® will automatically name the worksheet if no name is specified.
- *Contents*: will create a sheet containing a table of contents for the workbook with a link to each worksheet. The default value for this option is 'no'.

Each option for the tagset has a default value. Options allow the user to customize the output Excel workbook and accompanying worksheets.

EXAMPLES USING EXCELXP TAGSET

All of the following examples will be using the *sales* and *sales2* datasets, which can be found in the Appendix. All of the data in these datasets are for a factitious supermarket. The *sales* dataset contains 5 variables: employee id number (*ID*), department the employee works in (*department*), regular hourly wage of the employee (*wage*), number of hours worked that week (*hours*), and the number of overtime hours worked that week (*over_hours*). The data in *sales2* is comprised of all of the data from *sales* with 3 additional variables: gross regular rate earnings per employee for that week (*earnings*), gross over time rate earning per employee for that week (*over_time*), and the total gross earnings per employee (*total*). All of the following examples were coded using SAS® Enterprise Guide® version 7.1.

EXAMPLE 1: BASIC OUTPUT PROC REPORT

The code below produces the XML file *sales1*. This file contains data from the *sales2* SAS® dataset. No options were specified for the tagset. The data within the Excel workbook will have the color scheme, font type, and other attributes of the *analysis* style. On the PROC REPORT line the *nowd* option is specified which instructs SAS® not to open an interactive report window. Instead, the XML file will just be outputted directly to its file location without giving the user the ability to manually change the contents of the PROC REPORT (SAS, n.d.).

Using PROC REPORT the following variables will be outputted from the *sales2* dataset: *ID*, *department, wage, hours, over_hours, earnings, over_time,* and *total*. In order for formulas to show up in the columns for *earnings, over_time*, and *total*, the *STYLE* option with the *TAGATTR* style attribute needs to be inserted in each *define* statement of the PROC REPORT.

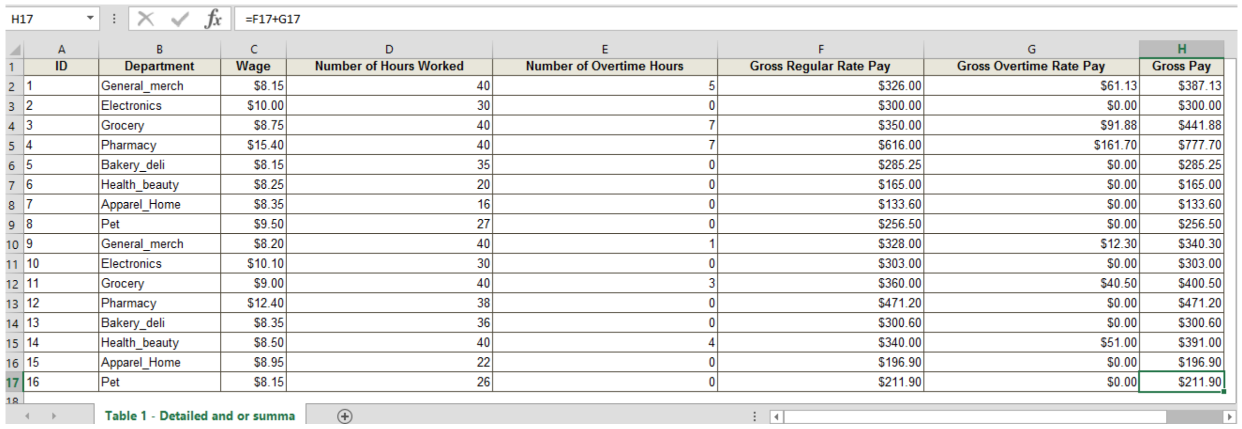
The variable *earnings* in the *sales2* dataset was calculated as *wage*hours*. To transform this calculation into the tagset equivalent, the RC formula style is utilized. The column that is being formulated is considered RC[0]. For the *earnings* formula: columns that proceed the *earnings* column will be expressed as negative RC combinations while columns that follow the *earnings* column will be expressed as positive RC combinations. Since both the *wage* and *hours* columns occur before the *earning* column (in the COLUMNS statement below), negative RC values will be used to represent each variable in the formula. Additionally, since the calculations will be used for the entire *earnings* column (not just for a specific cell) the brackets containing the negative number associated with each variable (column) will occur after the *RC* for each variable. The *wage* column is created 3 columns before the *earnings* column, resulting in a row column expression of RC[-3]. The *hours* column is created 2 columns before the *earnings* column which results in a RC expression of RC[-2]. Hence, the formula to calculate *earnings* within the XML file is RC[-3]*RC[-2]. The formulas for *over_time* and *total* are constructed using the same logic.

```
ODS tagsets.ExcelXP PATH='C:\\File Destination' FILE='Sales1.xml'
STYLE=analysis;

PROC REPORT DATA=sales2 NOWD;
   COLUMNS ID department wage hours over hours earnings over time total;
   DEFINE earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   DEFINE over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   DEFINE total / STYLE={tagattr="formula:RC[-2]+RC[-1]"};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

The important thing to remember when constructing formulas is that each new column being calculated is considered to be RC[0]. In the formula for *earnings*, RC[0] represents the *earnings* column itself. Subsequently, in the formula for *over_time*, RC[0] represents the *over_time* column itself. The *wage* column occurs 3 before the *earnings* column, resulting in a row column expression of RC[-3] in the *earnings* formula. However, when the *wage* column is used in the *over_time* calculation it occurs 4 columns before the *over_time* column, which results in a row column expression of RC[-4]. Both formulas use the *wage* column in their calculations but the expression for the same column shifts. The column that is being calculated is always RC[0] and references to other columns need to be constructed with this in mind.

Figure 1 shows the Sales1.xml file. The formula bar shows the formula in cell H17 (the *total* for observation 16). The style of the worksheet is the *analysis* styling from SAS®. The worksheet was automatically named by SAS® since the *sheet_name* option was not specified. The sales2 SAS® dataset has *wage*, *earnings, over_time,* and *total* variables formatted to the Dollar7.2 format. These formats are outputted into the Excel worksheet with an equivalent Excel format of currency.

Figure 1. Example 1 Excel Worksheet



H17 =F17+G17

ID	Department	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Pay	Gross Overtime Rate Pay	Gross Pay
1	General_merch	$8.15	40	5	$326.00	$61.13	$387.13
2	Electronics	$10.00	30	0	$300.00	$0.00	$300.00
3	Grocery	$8.75	40	7	$350.00	$91.88	$441.88
4	Pharmacy	$15.40	40	7	$616.00	$161.70	$777.70
5	Bakery_deli	$8.15	35	0	$285.25	$0.00	$285.25
6	Health_beauty	$8.25	20	0	$165.00	$0.00	$165.00
7	Apparel_Home	$8.35	16	0	$133.60	$0.00	$133.60
8	Pet	$9.50	27	0	$256.50	$0.00	$256.50
9	General_merch	$8.20	40	1	$328.00	$12.30	$340.30
10	Electronics	$10.10	30	0	$303.00	$0.00	$303.00
11	Grocery	$9.00	40	3	$360.00	$40.50	$400.50
12	Pharmacy	$12.40	38	0	$471.20	$0.00	$471.20
13	Bakery_deli	$8.35	36	0	$300.60	$0.00	$300.60
14	Health_beauty	$8.50	40	4	$340.00	$51.00	$391.00
15	Apparel_Home	$8.95	22	0	$196.90	$0.00	$196.90
16	Pet	$8.15	26	0	$211.90	$0.00	$211.90

Table 1 - Detailed and or summa

EXAMPLE 2: ADDING OPTIONS AND EXCEL FORMULAS USING PROC PRINT

The ExcelXP tagset can also be used with PROC PRINT. The data outputted in this example will have the *FestivalPrinte*r styling instead of the *analysis* style used in Example 1. The following tagset options were specified: Embedded_Titles= 'no' FitToPage='yes' Gridlines='no' Frozen_RowHeaders= 'no' Sheet_Name ='Sales Data' index=‘yes’ Autofit_height='yes' Absolute_Column_Width = '3.5,16,6.5,10,10,10,10,12'. Subsequently, the XML file will have:

- An initial page of indexed sheets (*index='yes'*)
- The worksheet containing the data table will be named ‘Sales Data’ (*sheet_name = ‘Sales Data’*)
- Gridlines outside of the data table will not be displayed (*gridlines='no'*).
- Titles that are in the SAS® program will not be embedded into the file (*embedded_titles='no'*)
- Page breaks will be set so the data table will fit on one page (*fittopage='yes'*)
- The header row will not be frozen (*frozen_rowheaders='no'*).
- The height of each row will be adjusted automatically (*autofit_height = ‘yes')*
- Columns widths for *ID, department, wage, hours, over_hours, earnings, over_time,* and *total* will be set to 3.5, 16, 6.5, 10, 10, 10, 10, and 12; respectively.

Additionally, the *Sales Data* worksheet in the *sales2*.xml file will not contain observation numbers (*noobs)* and the variables will be labeled (*label*); these options are specified on the PROC PRINT line.

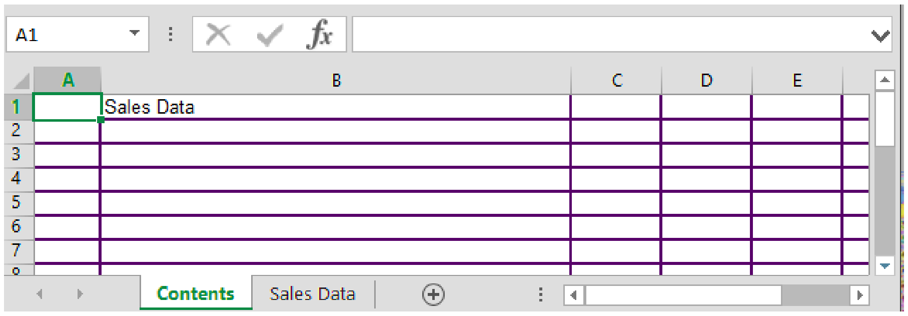
The layout of PROC PRINT is similar to that of PROC REPORT for the tagset. The formulas for *earnings* and *over_time* are the same as the ones used in the previous example. The formula for *total* could have been written in the same fashion as the previous example. Instead, an Excel formula was used. Excel formulas can be used within the formula line. The convention for writing a formula within Excel is starting with an equal sign. This can be omitted or kept in the formula line. If the equal sign is omitted from the formula, when the XML file is opened in Excel the equal sign for the formula is automatically inserted. The *total* column, for this example, uses the SUM Excel function to yield the sum of *earnings* and *overtime* per row.

```
ODS tagsets.ExcelXP PATH=‘C:\\File Destination’ FILE='Sales2.xml'
STYLE=FestivalPrinter options(Embedded_Titles='no' FitToPage='yes'
Gridlines='no' Frozen_RowHeaders= 'no' Sheet_Name ='Sales Data'
index='yes' Autofit_height='yes'
Absolute_Column_Width = '3.5,16,6.5,10,10,10,10,12');

PROC PRINT DATA=sales2 NOOBS LABEL;
   VAR ID department wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
```

```
RUN;

ODS tagsets.ExcelXP CLOSE;
```

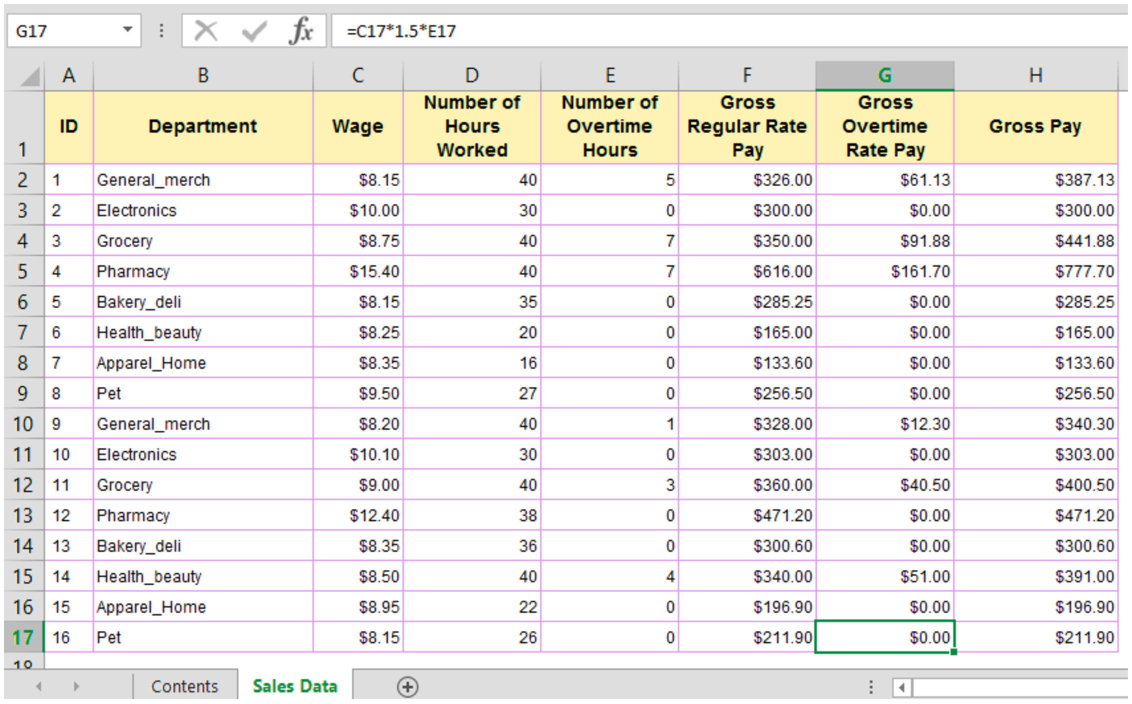
Figure 2 shows the contents page that was created in the Sales2.xml file due to the *index='yes'* tagset option. Since *style=FestivalPrinter* option was added to the ODS line, the gridlines within the index sheet are colored to the specifications of the FestivalPrinter style. The second picture below, Figure 3, shows the *Sales Data* worksheet that houses the sales2 SAS® dataset information. The formula bar shows the formula in cell G17, or the *over_time* value for observation 16.

Figure 2. Example 2 Contents Worksheet



	A	B	C	D	E
1		Sales Data			

Figure 3. Example 2 Sales Data Worksheet



=C17*1.5*E17

ID	Department	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Pay	Gross Overtime Rate Pay	Gross Pay
1	General_merch	$8.15	40	5	$326.00	$61.13	$387.13
2	Electronics	$10.00	30	0	$300.00	$0.00	$300.00
3	Grocery	$8.75	40	7	$350.00	$91.88	$441.88
4	Pharmacy	$15.40	40	7	$616.00	$161.70	$777.70
5	Bakery_deli	$8.15	35	0	$285.25	$0.00	$285.25
6	Health_beauty	$8.25	20	0	$165.00	$0.00	$165.00
7	Apparel_Home	$8.35	16	0	$133.60	$0.00	$133.60
8	Pet	$9.50	27	0	$256.50	$0.00	$256.50
9	General_merch	$8.20	40	1	$328.00	$12.30	$340.30
10	Electronics	$10.10	30	0	$303.00	$0.00	$303.00
11	Grocery	$9.00	40	3	$360.00	$40.50	$400.50
12	Pharmacy	$12.40	38	0	$471.20	$0.00	$471.20
13	Bakery_deli	$8.35	36	0	$300.60	$0.00	$300.60
14	Health_beauty	$8.50	40	4	$340.00	$51.00	$391.00
15	Apparel_Home	$8.95	22	0	$196.90	$0.00	$196.90
16	Pet	$8.15	26	0	$211.90	$0.00	$211.90

EXAMPLE 3: CREATING VARIABLES WITHIN PROC REPORT AND EXPORTING TO XML

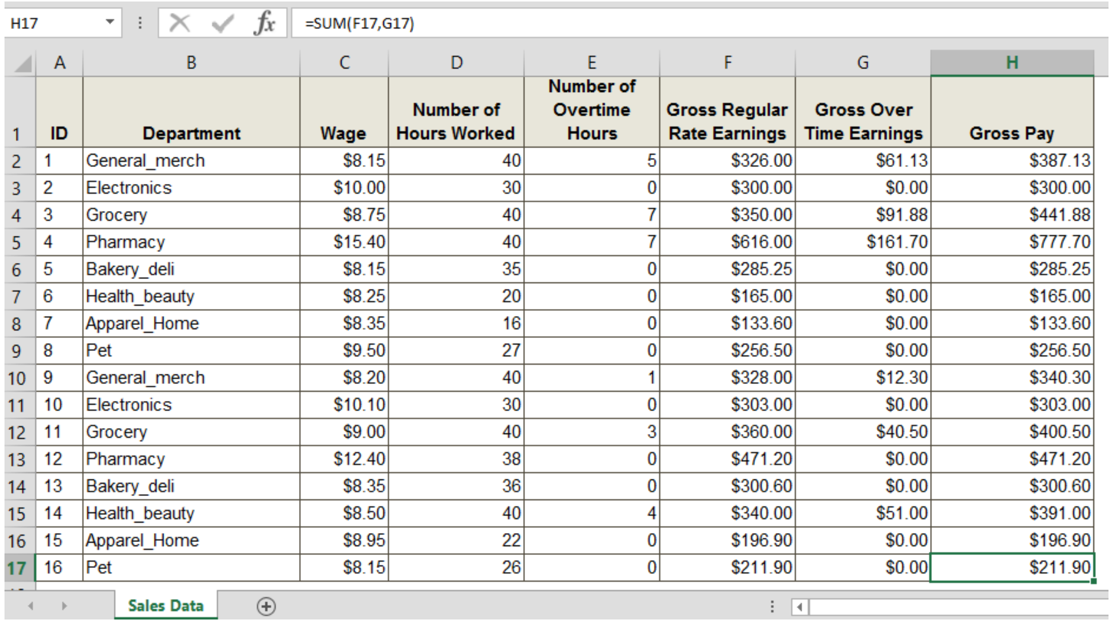
This example will utilize the *sales* dataset which does not contain the variables *earnings, over_time,* or *total*. Using PROC REPORT, these three variables can be calculated using the *compute* statement. These variables have to be defined prior to the computations and appear in the COLUMN statement. The SAS® formats assigned to these computed variables in the *define* statement will be carried over into the worksheet. Additionally, the labels of the computed variables will be exported as the worksheet column names.

```
ODS tagsets.ExcelXP PATH='C:\\File Destination' FILE='Sales3.xml'
STYLE=analysis options(FitToPage='yes' Sheet_Name='Sales Data'
Autofit_height='yes' Absolute_Column_Width = '3.5,16,6.5,10,10,10,10,12');
```

```
PROC REPORT DATA=sales NOWD;
   COLUMN id department wage hours over_hours earnings over_time total;
   DEFINE wage / DISPLAY;
   DEFINE over_hours / DISPLAY;
   DEFINE hours / DISPLAY;
   DEFINE earnings / COMPUTED FORMAT=dollar7.2 'Gross Regular Rate
         Earnings' STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   DEFINE over_time / COMPUTED FORMAT=dollar7.2 'Gross Over Time Earnings'
         STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   DEFINE total / COMPUTED FORMAT=dollar7.2 'Gross Pay'
         STYLE={tagattr="formula:=sum(RC[-2],RC[-1])"};
   COMPUTE earnings;
         earnings = wage*hours;
   ENDCOMP;
   COMPUTE over_time;
         over_time = (wage*1.5)*over_hours;
   ENDCOMP;
   COMPUTE total;
         total = earnings + over_time;
   ENDCOMP;
RUN;

ODS tagsets.ExcelXP CLOSE;
```

Figure 4 shows the resulting Sales3.xml file. The formula bar shows the formula within cell H17 (the *total* for observation 16). The sum Excel function was outputted from SAS® into each cell in the *total* column. The currency format is applied to all cells that were specified with the Dollar7.2 format in either PROC REPORT or during the datastep for the creation of the *sales* dataset.

Figure 4. Example 3 Sales Data Worksheet with Created Variables



ID	Department	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Earnings	Gross Over Time Earnings	Gross Pay
1	General_merch	$8.15	40	5	$326.00	$61.13	$387.13
2	Electronics	$10.00	30	0	$300.00	$0.00	$300.00
3	Grocery	$8.75	40	7	$350.00	$91.88	$441.88
4	Pharmacy	$15.40	40	7	$616.00	$161.70	$777.70
5	Bakery_deli	$8.15	35	0	$285.25	$0.00	$285.25
6	Health_beauty	$8.25	20	0	$165.00	$0.00	$165.00
7	Apparel_Home	$8.35	16	0	$133.60	$0.00	$133.60
8	Pet	$9.50	27	0	$256.50	$0.00	$256.50
9	General_merch	$8.20	40	1	$328.00	$12.30	$340.30
10	Electronics	$10.10	30	0	$303.00	$0.00	$303.00
11	Grocery	$9.00	40	3	$360.00	$40.50	$400.50
12	Pharmacy	$12.40	38	0	$471.20	$0.00	$471.20
13	Bakery_deli	$8.35	36	0	$300.60	$0.00	$300.60
14	Health_beauty	$8.50	40	4	$340.00	$51.00	$391.00
15	Apparel_Home	$8.95	22	0	$196.90	$0.00	$196.90
16	Pet	$8.15	26	0	$211.90	$0.00	$211.90

EXAMPLE 4: TRICKING SAS® OUTPUT FOR EXCEL

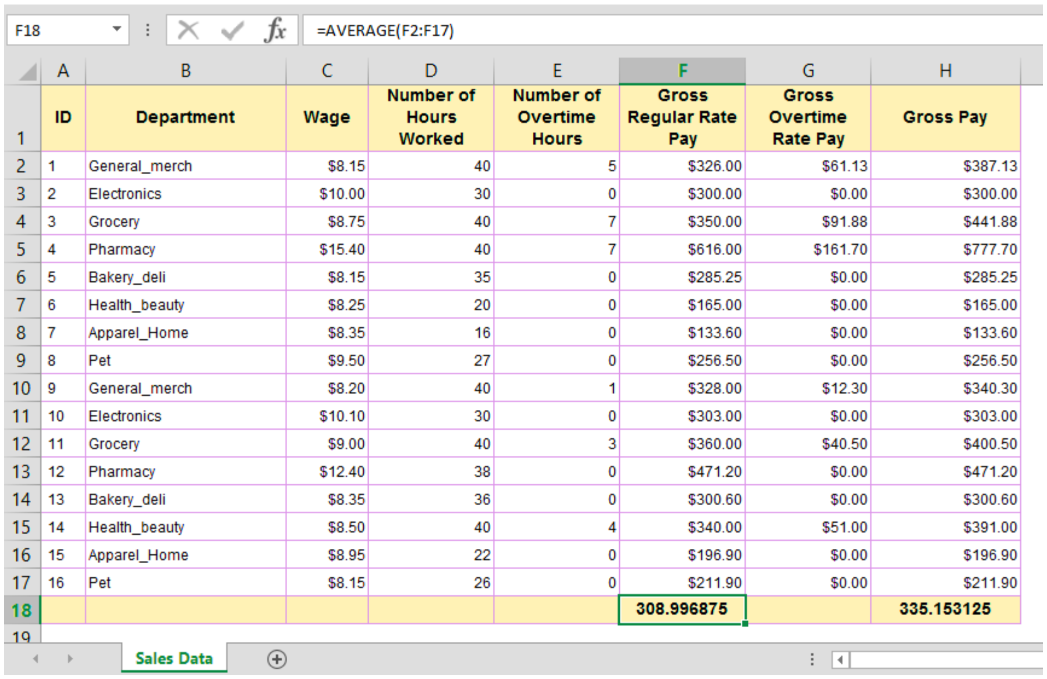
The ExcelXP tagset allows the user to trick SAS® for outputting data into a worksheet. Using PROC PRINT there is no way to explicitly produce an average of a column within the output. However, PROC PRINT does give the user the ability to output the sum of a variable into a table. Given the data in *sales2,* suppose the average *earnings* and *total* need to be outputted into the Excel worksheet. Using the *sum* statement, PROC PRINT will output the sum of these variables in SAS®. Since the average of these variables is desired within the workbook, the AVERAGE function can be written in the formula portion of the *TAGATTR* style attribute so that the average of the cells specified is calculated and displayed. Thus, the Excel file will contain the average of the cells desired columns (*earnings* and *total*) while the SAS® output will still show the sum of the variable specified.

```
ODS tagsets.ExcelXP PATH='C:\\File Destination' FILE='Sales4.xml'
STYLE=FestivalPrinter options(FitToPage='yes' Sheet_Name ='Sales Data'
Autofit_height='yes' Absolute_Column_Width = '3.5,16,6.5,10,10,10,10,12');

PROC PRINT DATA=sales2 LABEL NOOBS;
   VAR id department wage hours over_hours;
   VAR earnings/ STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time /STYLE={tagattr="formula:RC[-4]*RC[-2]*1.5"};
   VAR total / STYLE={tagattr="formula:RC[-2]+RC[-1]"};
   SUM total /STYLE={tagattr="formula:=AVERAGE(R[-16]C[0]:R[-1]C[0])"};
   SUM earnings/STYLE={tagattr="formula:=AVERAGE(R[-16]C[0]:R[-1]C[0])"};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

The sum statements for both *total* and *earnings* output the *average* function into the workbook. Knowledge of where this cell is within the spreadsheet is vital when creating the formula. Since there are 16 observation rows in this dataset, the formula for this cell contains all of the information for the 16 rows prior to the cell of interest within the column of interest (R[-16]C[0]: R[-1]C[0]).

The cell of interest is R[0]C[0] for each cell housing the average of either earnings or *total*. The 16 observation rows of interest fall from one row above the cell of interest (within the same column) to 16 rows above the cell of interest. Hence, the formula for the average of the 16 observations rows prior to the row of interest, within the same column, is *=AVERAGE(R[-16]C[0]: R[-1]C[0])*. Additionally, the cell holding the average function within Excel will not have the formatting of the rest of the column (Figure 5). By double clicking in the cell and then hitting the enter key, the cell will update and adhere to the style of the averaged cells (Figure 6). This is illustrated in the two screen shots below. The first shows the initial output (Figure 5); the second shows the output after the cell updates (Figure 6).

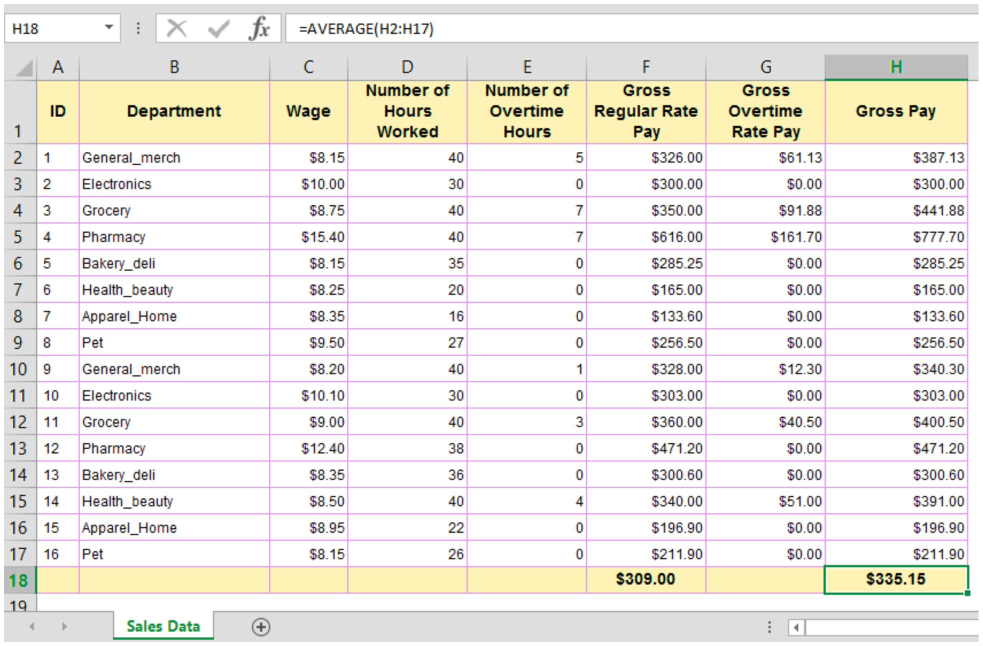
Figure 5. Example 4 Initial Sales Data Worksheet



F18 =AVERAGE(F2:F17)

	A	B	C	D	E	F	G	H
1	ID	Department	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Pay	Gross Overtime Rate Pay	Gross Pay
2	1	General_merch	$8.15	40	5	$326.00	$61.13	$387.13
3	2	Electronics	$10.00	30	0	$300.00	$0.00	$300.00
4	3	Grocery	$8.75	40	7	$350.00	$91.88	$441.88
5	4	Pharmacy	$15.40	40	7	$616.00	$161.70	$777.70
6	5	Bakery_deli	$8.15	35	0	$285.25	$0.00	$285.25
7	6	Health_beauty	$8.25	20	0	$165.00	$0.00	$165.00
8	7	Apparel_Home	$8.35	16	0	$133.60	$0.00	$133.60
9	8	Pet	$9.50	27	0	$256.50	$0.00	$256.50
10	9	General_merch	$8.20	40	1	$328.00	$12.30	$340.30
11	10	Electronics	$10.10	30	0	$303.00	$0.00	$303.00
12	11	Grocery	$9.00	40	3	$360.00	$40.50	$400.50
13	12	Pharmacy	$12.40	38	0	$471.20	$0.00	$471.20
14	13	Bakery_deli	$8.35	36	0	$300.60	$0.00	$300.60
15	14	Health_beauty	$8.50	40	4	$340.00	$51.00	$391.00
16	15	Apparel_Home	$8.95	22	0	$196.90	$0.00	$196.90
17	16	Pet	$8.15	26	0	$211.90	$0.00	$211.90
18						308.996875		335.153125
19								

Figure 6. Example 4 Sales Data Worksheet after Updating Average Cells



H18 =AVERAGE(H2:H17)

	A	B	C	D	E	F	G	H
1	ID	Department	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Pay	Gross Overtime Rate Pay	Gross Pay
2	1	General_merch	$8.15	40	5	$326.00	$61.13	$387.13
3	2	Electronics	$10.00	30	0	$300.00	$0.00	$300.00
4	3	Grocery	$8.75	40	7	$350.00	$91.88	$441.88
5	4	Pharmacy	$15.40	40	7	$616.00	$161.70	$777.70
6	5	Bakery_deli	$8.15	35	0	$285.25	$0.00	$285.25
7	6	Health_beauty	$8.25	20	0	$165.00	$0.00	$165.00
8	7	Apparel_Home	$8.35	16	0	$133.60	$0.00	$133.60
9	8	Pet	$9.50	27	0	$256.50	$0.00	$256.50
10	9	General_merch	$8.20	40	1	$328.00	$12.30	$340.30
11	10	Electronics	$10.10	30	0	$303.00	$0.00	$303.00
12	11	Grocery	$9.00	40	3	$360.00	$40.50	$400.50
13	12	Pharmacy	$12.40	38	0	$471.20	$0.00	$471.20
14	13	Bakery_deli	$8.35	36	0	$300.60	$0.00	$300.60
15	14	Health_beauty	$8.50	40	4	$340.00	$51.00	$391.00
16	15	Apparel_Home	$8.95	22	0	$196.90	$0.00	$196.90
17	16	Pet	$8.15	26	0	$211.90	$0.00	$211.90
18						$309.00		$335.15
19								

EXAMPLE 5: EXPORTING MULTIPLE DATASETS INTO ONE WORKBOOK

Multiple datasets can be exported into different worksheets within one workbook. This process is illustrated in the article *SAS to Excel with ExcelXP tagset* by Mahipal Vanam, Kiran Karidi, and Sridhar Dodlapati. Using the *sales2* dataset, one dataset per department was created which resulted in 8 datasets (See Appendix for code).

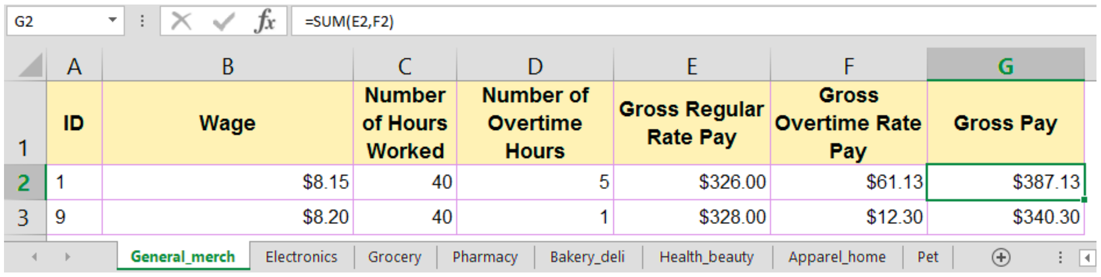
In order to create one worksheet per dataset, there needs to be one ODS statement line per new worksheet. The ODS statement for the first dataset needs to contain a *FILE* statement (or *FILE* and *PATH* statements). To add more specificity to each worksheet, the tagset option *sheet_name*= should be utilized. This allows the user to name each sheet individually in the workbook. If sheet names are not given specifically for each ODS statement then SAS will automatically apply a sheet name. Additionally, if one sheet is named using the *sheet_name* option but none of the following sheets are named then the sheets will numbers added to end of the name of the specified sheet. For example, if one sheet was named 'Sales' and none of the following sheets were explicitly named then the following sheets would become 'Sales2', 'Sales3', etc. Depending on how many sheets are to be outputted, there needs to be the same number of ODS tagset statements. However, only the first ODS statements needs the *FILE* and *STYLE* statements. All ODS statements specified between the initial statement (containing the file name) and the *ODS tagsets.ExcelXP Close* statement will output to the same Excel workbook.

In the example below the Sales5.xml file will contain 8 sheets (see Appendix for the full code). Each sheet was given a specific name based on the department. Since there is no *ODS tagsets.ExcelXP Close* statement until the end of all of the code, all 8 of the ODS tagsets will be exported to one workbook. If formulas are needed for variables, even if the variables have the same formula across the datasets, they need to be defined in each PROC PRINT statement.

```
ODS tagsets.ExcelXP PATH=‘C:\\File Destination’ FILE='Sales5.xml'
STYLE=FestivalPrinter options(FitToPage='yes'
Sheet_Name ='General_merch' Autofit_height='yes' Absolute_Column_Width =
'3.5,16,6.5,10,10,10,10,12');

PROC PRINT DATA=gen NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Electronics');
PROC PRINT DATA=elect NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;
…
ODS tagsets.ExcelXP options(Sheet_Name ='Pet');
PROC PRINT DATA=pet NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP CLOSE;
```

The output from this example is shown in Figure 7. There are 8 worksheets within the Sales5.xml workbook.

Figure 7. Example 5 Multiple Datasets Exported into One Workbook


G2 =SUM(E2,F2)

	A	B	C	D	E	F	G
1	ID	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Pay	Gross Overtime Rate Pay	Gross Pay
2	1	$8.15	40	5	$326.00	$61.13	$387.13
3	9	$8.20	40	1	$328.00	$12.30	$340.30

EXAMPLE 6: OUTPUTTING MULTIPLE TABLES TO THE SAME WORKSHEET

In Example 5 we outputted multiple datasets to the same Excel workbook. The ExcelXP tagset also has the ability to export multiple datasets to the same worksheet. The tagset option *sheet_interval* controls the number of tables outputted to each worksheet. By default this option is set to *table*, indicating that there will be one table outputted to each worksheet. This option value can be changed to *page, bygroup, proc,* or *none.* By changing the value of the option to *none,* several tables can be outputted to the same worksheet.

The ODS TAGSET statement is not required between multiple PROC PRINT or PROC REPORT statements if the tables are going to be outputted to the same worksheet. The ODS TAGSET statement is required when the user would like to change any option for the tagset including changing the *sheet_interval* option. The code below outputs the general merchandise (*gen)* and electronic (*elect)* department datasets to the same worksheet while outputting the grocery (*gro)* data to a separate worksheet. This is done using PROC REPORT.

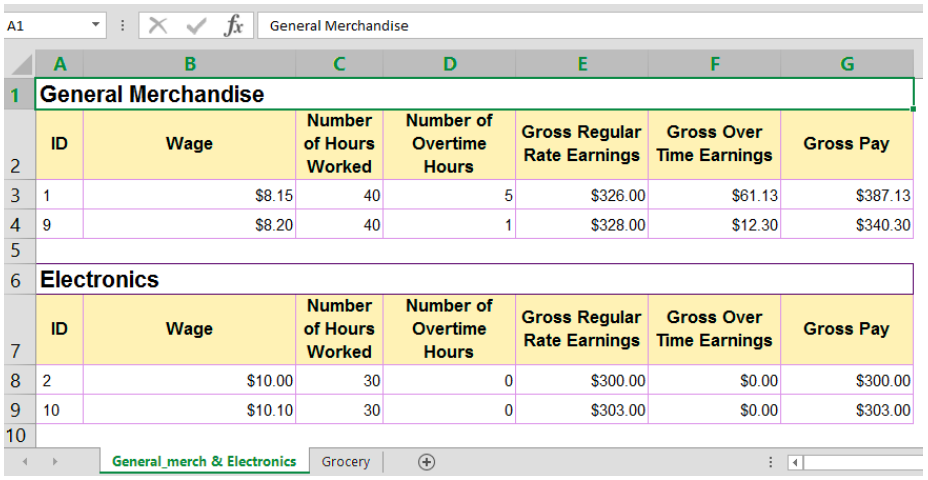
```
ODS tagsets.ExcelXP PATH=‘C:\\File Destination’ FILE='Sales6.xml'
STYLE=FestivalPrinter options(FitToPage='yes'
Sheet_Name ='General_merch & Electronics' sheet_interval = 'None'
Autofit_height='yes' Absolute_Column_Width = '3.5,16,6.5,10,10,10,10,12');

PROC REPORT DATA=gen NOWD;
   COLUMN ID wage hours over_hours earnings over_time total;
   DEFINE ID / DISPLAY;
   DEFINE wage / DISPLAY;
   DEFINE over_hours / DISPLAY;
   DEFINE hours / DISPLAY;
   DEFINE earnings / DISPLAY FORMAT=dollar7.2 'Gross Regular Rate
         Earnings' STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   DEFINE over_time / DISPLAY FORMAT=dollar7.2 'Gross Over Time Earnings'
         STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   DEFINE total / DISPLAY FORMAT=dollar7.2 'Gross Pay'
         STYLE={tagattr="formula:=sum(RC[-2],RC[-1])"};
   COMPUTE BEFORE _PAGE_ / STYLE=[font_size=3 font_weight=bold
         bordercolor=black] CENTER;
         LINE 'General Merchandise';
   ENDCOMP;
RUN;

PROC REPORT DATA=elect NOWD;
   COLUMN ID wage hours over_hours earnings over_time total;
   DEFINE ID / DISPLAY;
   DEFINE wage / DISPLAY;
   DEFINE over_hours / DISPLAY;
   DEFINE hours / DISPLAY;
   DEFINE earnings / DISPLAY FORMAT=dollar7.2 'Gross Regular Rate
         Earnings' STYLE={tagattr="formula:RC[-3]*RC[-2]"};
```

```
    DEFINE over_time / DISPLAY FORMAT=dollar7.2 'Gross Over Time Earnings'
          STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
    DEFINE total / DISPLAY FORMAT=dollar7.2 'Gross Pay'
          STYLE={tagattr="formula:=sum(RC[-2],RC[-1])"};
    COMPUTE BEFORE _PAGE_ / STYLE=[font_size=3 font_weight=bold
          bordercolor=black ] CENTER;
          LINE 'Electronics';
    ENDCOMP;
 RUN;

 ODS tagsets.ExcelXP options(Sheet_Name ='Grocery'
    sheet_interval = 'Table');

 PROC REPORT DATA=gro NOWD;
    COLUMN ID wage hours over_hours earnings over_time total;
    DEFINE ID / DISPLAY;
    DEFINE wage / DISPLAY;
    DEFINE over_hours / DISPLAY;
    DEFINE hours / DISPLAY;
    DEFINE earnings / DISPLAY FORMAT=dollar7.2 'Gross Regular Rate
          Earnings' STYLE={tagattr="formula:RC[-3]*RC[-2]"};
    DEFINE over_time / DISPLAY FORMAT=dollar7.2 'Gross Over Time Earnings'
          STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
    DEFINE total / DISPLAY FORMAT=dollar7.2 'Gross Pay'
          STYLE={tagattr="formula:=sum(RC[-2],RC[-1])"};
    COMPUTE BEFORE _PAGE_ / STYLE=[font_size=3 font_weight=bold
          bordercolor=black] CENTER;
          LINE 'Grocery';
    ENDCOMP;
 RUN;

 ODS tagsets.ExcelXP CLOSE;
```

All of the PROC REPORT statements above contain a compute block of code. Within this block an embedded title is created for each table. A *title* statement would create a title for the table within SAS® but this title does not export into the XML file. To overcome this hurdle, the compute block creates a row above the header row that contains whatever text is included in the *LINE* statement. Since all three datasets contain the same variables, creating an embedded title distinguishes each table from one another. A screenshot of the outputted XML file is below.

Figure 8 and Figure 9 show the worksheets contained in the Sales6.xml file. The general merchandise and electronic departments tables were outputted to the same worksheet labeled *General_merch & Electronics*. This occurred because there was no ODS line separating the PROC REPORTs for the *gen* and *elect* datasets. Conversely, there was an ODS line separating the PROC REPORT for the *elect* data and the PROC REPORT for the *gro* data which resulted in the grocery department table being outputted to its own worksheet (shown below). Additionally, the *sheet_interval* option was reset to 'table' in the ODS line separating the PROC REPORTs for *elect* and *gro*.

Figure 8. Example 6 General Merchandise and Electronic Department Data Exported into One Worksheet

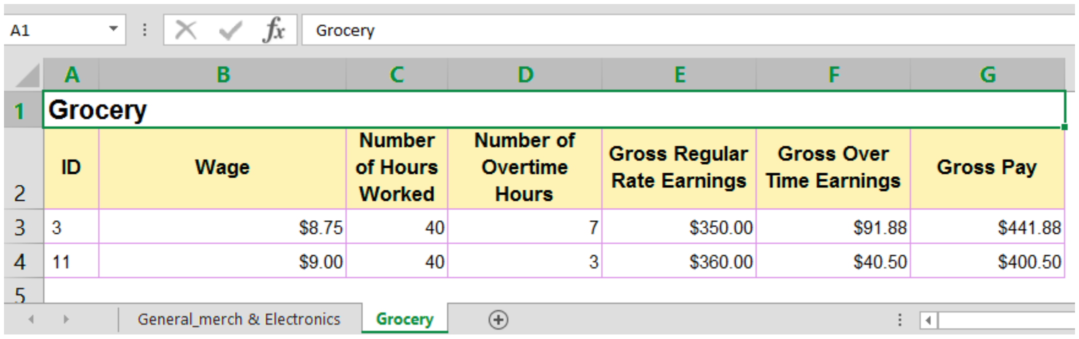


A1 General Merchandise

	A	B	C	D	E	F	G
1	General Merchandise						
2	ID	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Earnings	Gross Over Time Earnings	Gross Pay
3	1	$8.15	40	5	$326.00	$61.13	$387.13
4	9	$8.20	40	1	$328.00	$12.30	$340.30
5							
6	Electronics						
7	ID	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Earnings	Gross Over Time Earnings	Gross Pay
8	2	$10.00	30	0	$300.00	$0.00	$300.00
9	10	$10.10	30	0	$303.00	$0.00	$303.00
10							

General_merch & Electronics | Grocery

Figure 9. Example 6 Grocery Department Data Exported into a Separate Worksheet



A1 Grocery

	A	B	C	D	E	F	G
1	Grocery						
2	ID	Wage	Number of Hours Worked	Number of Overtime Hours	Gross Regular Rate Earnings	Gross Over Time Earnings	Gross Pay
3	3	$8.75	40	7	$350.00	$91.88	$441.88
4	11	$9.00	40	3	$360.00	$40.50	$400.50
5							

General_merch & Electronics | Grocery

CONCLUSION

The ExcelXP tagset allows a SAS® programmer to export data into an Excel spreadsheet. This allows the data to become more universally useful within an organization. In addition to just exporting data, formulas can be written using the tagset language so that the Excel spreadsheet automatically updates cells when the data is modified by the user. More people are familiar with Microsoft® Excel, making this bridge a handy tool in a SAS® programmer's tool bag.

APPENDIX

CODE FOR DATASETS

```
DATA sales;
   INPUT ID $ Department: $13. Wage: 5.2 hours over_hours;
   LABEL hours = 'Number of Hours Worked'
         over_hours = 'Number of Overtime Hours';
   FORMAT wage dollar7.2;
   CARDS;
   1 General_merch  8.15  40 5
   2 Electronics    10.00 30 0
   3 Grocery        8.75  40 7
   4 Pharmacy       15.40 40 7
   5 Bakery_deli    8.15  35 0
   6 Health_beauty  8.25  20 0
   7 Apparel_Home   8.35  16 0
   8 Pet            9.50  27 0
   9 General_merch  8.20  40 1
   10 Electronics   10.10 30 0
   11 Grocery       9.00  40 3
   12 Pharmacy      12.40 38 0
   13 Bakery_deli   8.35  36 0
   14 Health_beauty 8.50  40 4
   15 Apparel_Home  8.95  22 0
   16 Pet           8.15  26 0
   ;
RUN;

DATA sales2;
   SET sales;
   Earnings = wage*hours;
   over_time= (wage*1.5)*over_hours;
   total = earnings + over_time;
   FORMAT earnings dollar7.2 over_time dollar7.2 total dollar7.2;
   LABEL earnings = 'Gross Regular Rate Pay'
         over_time = 'Gross Overtime Rate Pay'
         total = 'Gross Pay';
RUN;
```

CODE FOR DATASETS BY DEPARTMENT

```
DATA gen elect gro phar bake health app pet;
   SET sales2;
   IF department = 'General_merch' THEN OUTPUT gen;
         ELSE IF department = 'Electronics' THEN OUTPUT elect;
         ELSE IF department = 'Grocery' THEN OUTPUT gro;
         ELSE IF department = 'Pharmacy' THEN OUTPUT phar;
         ELSE IF department = 'Bakery_deli' THEN OUTPUT bake;
         ELSE IF department = 'Health_beauty' THEN OUTPUT health;
         ELSE IF department = 'Apparel_Home' THEN OUTPUT app;
         ELSE output pet;
RUN;
```

FULL CODE FOR EXAMPLE 5

```
ODS tagsets.ExcelXP PATH= ‘C:\\File Destination’ FILE='Sales5.xml'
STYLE=FestivalPrinter options(FitToPage='yes'
Sheet_Name ='General_merch' Autofit_height='yes' Absolute_Column_Width =
'3.5,16,6.5,10,10,10,10,12');

PROC PRINT DATA=gen NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Electronics');
PROC PRINT DATA=elect NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Grocery');
PROC PRINT DATA=gro NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Pharmacy');
PROC PRINT DATA=phar NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Bakery_deli');
PROC PRINT DATA=bake NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet Name ='Health_beauty');
PROC PRINT DATA=health NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Apparel_home');
PROC PRINT DATA=app NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS tagsets.ExcelXP options(Sheet_Name ='Pet');
PROC PRINT DATA=pet NOOBS LABEL;
   VAR ID wage hours over_hours;
   VAR Earnings / STYLE={tagattr="formula:RC[-3]*RC[-2]"};
   VAR over_time / STYLE={tagattr="formula:RC[-4]*1.5*RC[-2]"};
   VAR total / STYLE={tagattr="formula:sum(RC[-2],RC[-1])"};
RUN;

ODS TAGSETS.EXCELXP CLOSE;
```

REFERENCES

SAS. (n.d.). *DEFINE Statement*. Retrieved August 18, 2015, from Base SAS(R) 9.2 Procedures Guide: http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473627.htm

SAS. (n.d.). *Overview: ODS Tagsets and the TEMPLATE PROCEDURE*. (n. SAS(R) 9.2 Output Delivery System: User's Guide. SAS(R), Editor, & SAS) Retrieved May 14, 2015, from SAS 9.2 Documentation: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a002565724.htm

SAS. (n.d.). *PROC REPORT Statement*. Retrieved May 22, 2015, from Base SAS(R) 9.2 Procedures Guide: http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473620.htm

SAS. (n.d.). *VAR Statement*. Retrieved August 18, 2015, from Base SAS(R) 9.4 Procedures Guide, Fifth Edition: http://support.sas.com/documentation/cdl/en/proc/68954/HTML/default/viewer.htm#n1cfqafrw9xvoln1sz5oed59lyf3.htm

Skopic, J. K. (n.d.). *Exporting Formulas to Excel Using the ODS ExcelXP Tagset*. Retrieved May 14, 2015, from http://support.sas.com/resources/papers/proceedings14/1854-2014.pdf

Vanam, M., Karidi, K., & Dodlapati, S. (2010). *SAS to Excel with ExcelXP Tagse*. (PharmaSUG2010, Producer) Retrieved May 14, 2015, from http://www.lexjansen.com/pharmasug/2010/CC/CC22.pdf

ACKNOWLEDGMENTS

A special thanks to Dr. Laura Kapitula for her fantastic mentorship.

CONTACT INFORMATION

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

Veronica Renauldo
renauldv@mail.gvsu.edu