

## The REPORT Procedure and ODS Destination for Microsoft Excel: The Smarter, Faster Way to Create First-Rate Excel Reports

Jane Eslinger, SAS Institute Inc.

### ABSTRACT

Does your job require you to create reports in Microsoft Excel on a quarterly, monthly, or even weekly basis? Are you creating all or part of these reports by hand, referencing another sheet containing rows and rows and rows of data? If so, stop! There is a better way!

The new ODS destination for Excel enables you to create native Excel files (XLSX) directly from SAS. Now you can include just the data you need, create great-looking tabular output, and do it all in a fraction of the time!

This paper shows you how to use PROC REPORT to create polished tables that contain formulas, colored cells, and other customized formatting. Also presented in the paper are the destination options used to create various workbook structures, such as multiple tables per worksheet. Using these techniques to automate the creation of your Excel reports will save you hours of time and frustration, enabling you to pursue other endeavors.

### INTRODUCTION

No matter which industry you work in, no matter if your company is big or small, reports in Microsoft Excel are probably a fundamental part of your business life. Microsoft Excel reports seem to be universal and accepted by everyone. Unfortunately, though, many of those reports are still created manually, consuming vast amounts of analysts' time. Well, it is time to change that! You already have SAS® software, so why not use it to create those Excel reports for you? You can create a SAS program to generate each weekly, monthly, and quarterly report, and then spend your time on more important tasks. This paper is going to show you how to do just that. The effort you make now in learning about the ODS destination for Excel will benefit you later.

You will learn about ODS destination for Excel options that can be used to create the desired workbook structure. These options include such things as tab names, tab color, specifying the cell in which the report starts, and specifying the scale to use when printing. All of those options are important to the final look and feel of the report, which, to the consumer of the report, can be just as important as the numbers.

The examples in this paper focus on creating tabular output with the REPORT procedure. Though the ODS destination for Excel is also capable of supporting graphical output, this paper concentrates on tables containing the numbers from your analyses. PROC REPORT provides the capability to mimic the tables created manually in Excel.

Important features of tables include formulas, colors, and customized formatting. Headers and rows of text also play an important role in conveying the information that is inside of the report. Each of these topics is covered through PROC REPORT examples in this paper.

As you read and work through the examples in this paper, please be aware that the ODS destination for Excel did not become production until the third maintenance release of SAS® 9.4 (SAS 9.4 TS1M3). The ODS EXCEL statement was available in 9.4 TS1M1; however, it is best if you use SAS 9.4 TS1M3 or later, especially when creating production-level jobs.

### INVEST TIME IN LEARNING THE ODS DESTINATION FOR EXCEL

Your time is valuable, so you want to create Excel reports as quickly as possible and move on to more interesting or pressing tasks. However, you might wonder if PROC REPORT and the ODS destination for Excel can create what you need. This is an understandable concern. SAS is really good at creating reports, but you need to determine whether it can create **your** report.

The best way to determine whether you can mimic your current manual reports is to explore the options and suboptions available within the ODS destination. These options control the printing, display, and structural components of each workbook. They play an integral role in producing a high-quality Excel report that not only conveys numbers but is also easy to navigate.

## ODS EXCEL STATEMENT OPTIONS FOR DOCUMENT PROPERTIES

First things first. The ODS EXCEL statement has options just like all other ODS statements. A number of options in the ODS EXCEL statement are used to control the document properties. Depending on your business requirements, you might need to use any or all of these options. These options include:

- AUTHOR
- CATEGORY
- COMMENTS
- KEYWORDS
- STATUS
- TITLE

The values for these options must be set in the ODS EXCEL statement that also contains the FILE= option. In other words, they must be set when document creation is started.

**Note:** A few options in the ODS EXCEL statement control graphical output: TITLE | NOGTITLE, GFOOTNOTE | NOGFOOTNOTE, DPI, and IMAGE\_DPI. However, this paper is about tabular output, so please see the *SAS® 9.4 Output Delivery System: User's Guide*, ODS EXCEL Statement for an explanation of these options.

## ODS EXCEL STATEMENT OPTIONS OPTION

The most important option in the ODS EXCEL statement is called, lamentably, OPTIONS. It enables you to specify a suboptions list that provides greater control over the structure of your Excel workbook and worksheets. You will find the OPTIONS option is the one you need to use the most, so spend some time learning about its uses.

The OPTIONS option is immediately followed by an open parenthesis, unlike other options that are followed by an equal sign. Within the parentheses, you list the suboptions that you would like to apply to the worksheet. The value of each suboption must be enclosed in quotation marks. You can use either single or double quotation marks. It is a little confusing, but once you start to use the ODS EXCEL statement regularly, you will become more comfortable with the syntax and the difference between OPTIONS for the ODS statement and suboptions that go in parentheses.

The suboptions within OPTIONS parentheses can be toggled on and off during the creation of the document, depending on your needs. The suboptions are specific to each individual worksheet. However, once set, the value of a suboption is retained for all subsequent worksheets until it is changed.

The SAS 9.4 TS1M3 documentation lists approximately 45 suboptions! Roughly half of these control various printing aspects of the worksheet. The other half control the visual parts that you notice when viewing the report on the screen. Forty-five suboptions is too many to list in this paper, and too many to focus on. Consequently, only the most important and commonly used suboptions are demonstrated here.

One suboption is important to mention specifically because it can be confusing. This is the GRIDLINES suboption. The GRIDLINES suboption does not affect what you see on the screen when you are viewing the report—it affects only printed output. The “Gray Lines and Borders” section of this paper covers how to include or exclude the standard light gray lines that you see when you open Microsoft Excel. It also demonstrates how to outline the table and cells with more pronounced borders.

You might be familiar with the suboption DOC from other ODS tagsets. The DOC suboption is not available in the ODS destination for Excel. You do not get an error if you include it, but information is not

written to the log. All information about options and suboptions for the Excel destination is contained in the ODS Excel Statement documentation mentioned above.

## Suboptions That Control Worksheet Features

Some worksheet features need to be set no matter what type of output the worksheet will contain. This section discusses the suboptions that are used to control worksheet features.

Example 1, below, includes these suboptions:

- EMBEDDED\_TITLES places the title(s) as part of the worksheet.
- EMBEDDED\_FOOTNOTES places the footnote(s) as part of the worksheet.
- TAB\_COLOR changes the background color of the tab name.
- START\_AT identifies the cell in which to place the first column's header; in SAS 9.4 TS1M3, this must be listed as two numbers in the form *column, row*.
- FROZEN\_HEADERS specifies header rows that should remain static when scrolling up and down.

### Example 1:

```
ods excel file='example1.xlsx' options(embedded_titles='yes'  
    embedded_footnotes='yes' tab_color='purple' start_at='2,4'  
    frozen_headers='yes');  
title 'This title will appear in the worksheet';  
footnote 'This footnote will appear in the worksheet';  
proc report data=sashelp.class;  
run;  
ods excel close;
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

Output 1. Results for Example 1

In Output 1, the output starts at cell B4, the result of using the START\_AT suboption. Since the EMBEDDED\_TITLES suboption is used, this means that the titles are placed within the document, in cell B4. One row is skipped between the titles and the table. The table begins on row 6.

Notice the line under row 6. This is a visual indicator of the scrolling capabilities from using FROZEN\_HEADERS. When you scroll farther down the screen, the first six rows remain static.

The ODS destination for Excel also has a suboption called FROZEN\_ROWHEADERS, which specifies that a column should remain static when you scroll left and right. When FROZEN\_ROWHEADERS is set to YES, which column is frozen depends on the procedure. For example, PROC PRINT has an OBS column, which is considered a row header and is therefore frozen when the FROZEN\_ROWHEADERS suboption is used. PROC REPORT does not have a column designated as a row header. Currently, it is not possible to designate a column generated by PROC REPORT as a row header. For PROC REPORT, you have to specify a number value in the FROZEN\_ROWHEADERS suboption to freeze a column.

Also, notice that the title and footnote wrap. In Output 1, above, the title and footnote are confined to the width of the table. You can alter this behavior using the TITLE\_FOOTNOTE\_NOBREAK suboption. Setting this option to YES, as shown below in Example 2, prevents the title and footnote from wrapping by merging as many cells as necessary to hold the title/footnote.

**Example 2:**

```
ods excel file='example2.xlsx' options(embedded_titles='yes'
embedded_footnotes='yes' tab_color='purple' start_at='2,4'
frozen_headers='yes' title_footnote_nobreak='yes');
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112

**Output 2. Results for Example 2 Using TITLE\_FOOTNOTE\_NOBREAK**

## Suboptions That Control Table Features

The previous section discussed suboptions that control the worksheet features. This section discusses suboptions that control table features. Frequently used suboptions that provide this control are listed below and demonstrated in Example 3.

- `HIDDEN_ROWS` hides specific rows in the worksheet.
- `AUTOFILTER` turns on Excel filtering capability; can be applied to all columns or a subset of columns.
- `ABSOLUTE_COLUMN_WIDTH` specifies the column width; one width can be listed for all columns, or a width can be listed for multiple columns. Cannot be overwritten by style attributes within the procedure.
- `ABSOLUTE_ROW_HEIGHT` specifies the row height; one width can be listed for all rows, or a height can be listed for multiple rows.

### Example 3:

```
ods excel file='example3.xlsx' options(hidden_rows='6' autofilter='1-3'
    absolute_column_width='16' absolute_row_height='25');
proc report data=sashelp.cars;
run;
ods excel close;
```

	A	B	C	D	E	F
1	Make	Model	Type	Origin	DriveTrain	MSRP
2	Acura	MDX	SUV	Asia	All	\$36,945
3	Acura	RSX Type S 2dr	Sedan	Asia	Front	\$23,820
4	Acura	TSX 4dr	Sedan	Asia	Front	\$26,990
5	Acura	TL 4dr	Sedan	Asia	Front	\$33,195
7	Acura	3.5 RL w/Navigation	Sedan	Asia	Front	\$46,100
8	Acura	NSX coupe 2dr man	Sports	Asia	Rear	\$89,765
9	Audi	A4 1.8T 4dr	Sedan	Europe	Front	\$25,940

### Output 3. Results for Example 3

Immediately you can see the difference in the table between Output 3 and Output 1. All of the columns are the same width, and each row is tall enough to provide visual separation between the data values.

The `ABSOLUTE_COLUMN_WIDTH` suboption plays a key role in the look of this table. It is one of the last features applied to the column. It always wins. This means that if you set a column width within the procedure, `ABSOLUTE_COLUMN_WIDTH` will override it. This suboption is good to use when you want the column to be a consistent size no matter how many characters are in the data field. Without this setting, the width of the column depends on the data values or a width specification within the procedure.

**Note:** The value specified by `ABSOLUTE_COLUMN_WIDTH` is not the pixel value you see when manually widening a column. It is the value shown by right-clicking the column and selecting **Column Width**.

The drop-down arrows in the first three column headers were created with the `AUTOFILTER` suboption. A person reviewing the report can use these filters to subset the report without having to change the data or rerun the report.

`ABSOLUTE_ROW_HEIGHT` sets the same height, 25, for all rows in the table, including the header row.

In Output 3, above, notice that you cannot see row 6. The row is still present, but you cannot see it because the `HIDDEN_ROWS` suboption was used. The `HIDDEN_ROWS` suboption is especially helpful for rows created by the `BREAK` statement. `PROC REPORT` cannot conditionally execute or print a row generated by a `BREAK` statement; this means that the `BREAK` statement will sometimes have the same values as the detail row above it because the grouping had only one subgrouping value. The `HIDDEN_ROWS` suboption enables you to suppress that repetitive row.

### Suboptions That Affect the Number of Worksheets Created

So far, we have discussed suboptions that affect worksheet features and some of the table's structural features like column widths. You might have noticed that those sections focused on just one table on one worksheet. Chances are, though, the reports you create by hand contain multiple worksheets. They might also contain multiple tables per worksheet.

The default behavior for the destination is to place each table in its own worksheet. Please be aware that the distinction between table and procedure is very important, especially for `PROC REPORT`. By default, `PROC REPORT` creates just one table, so the output from the procedure creates just one worksheet. However, using the `BREAK` statement with the `PAGE` option generates multiple tables. Each one of these tables is written to its own worksheet.

The code in Example 4 uses the `PAGE` option with the break variable `SEX`. The result, shown in Output 4, is a workbook with two worksheets.

#### Example 4:

```
ODS excel file='example4.xlsx';
proc report data=sashelp.class;
  column sex name age height weight;
  define sex / group;
  break after sex / page;
run;
ods excel close;
```

	A	B	C	D	E	F	G	H
1	<b>Sex</b>	<b>Name</b>	<b>Age</b>	<b>Height</b>	<b>Weight</b>			
2	F	Alice	13	56.5	84			
3		Barbara	13	65.3	98			
4		Carol	14	62.8	102.5			
5		Jane	12	59.8	84.5			
6		Janet	15	62.5	112.5			
7		Joyce	11	51.3	50.5			
8		Judy	14	64.3	90			
9		Louise	12	56.3	77			
10		Mary	15	66.5	112			
11								
12								
13								

Report 1 - Detailed and-or s
Report 2 - Detailed and-or s

#### Output 4. Results for Example 4

The `SHEET_INTERVAL` suboption controls if and when more worksheets are created. It has five possible values, which are explained in the documentation. The value of `NONE` enables you to put multiple pieces of output, generated by one procedure or multiple procedures, in the same worksheet. Specifying either a `SHEET_NAME` or `SHEET_LABEL` suboption controls the name of the worksheet. If `SHEET_NAME` and `SHEET_LABEL` are used together, the `SHEET_NAME` value is used.

Example 5 demonstrates using the `SHEET_INTERVAL` suboption to keep all of the tables generated by `PROC REPORT` in the same worksheet. The `SHEET_NAME` suboption provides a useful name.

### Example 5:

```
ods excel file='example5.xlsx' options(sheet_name='Cars by Make'  
    sheet_interval='none');  
proc report data=sashelp.cars;  
    by make;  
    column model type invoice;  
run;  
ods excel close;
```

	A	B	C
1	<b>Make=Acura</b>		
2			
3	<b>Model</b>	<b>Type</b>	<b>Invoice</b>
4	MDX	SUV	\$33,337
5	RSX Type S 2dr	Sedan	\$21,761
6	TSX 4dr	Sedan	\$24,647
7	TL 4dr	Sedan	\$30,299
8	3.5 RL 4dr	Sedan	\$39,014
9	3.5 RL w/Navigation 4dr	Sedan	\$41,100
10	NSX coupe 2dr manual S	Sports	\$79,978
11	<b>Make=Audi</b>		
12			
13			
14	<b>Model</b>	<b>Type</b>	<b>Invoice</b>
15	A4 1.8T 4dr	Sedan	\$23,508
16	A4 1.8T convertible 2dr	Sedan	\$32,506
17	A4 3.0 4dr	Sedan	\$28,846
18	A4 3.0 Quattro 4dr manual	Sedan	\$30,366
19	A4 3.0 Quattro 4dr auto	Sedan	\$31,388
20	A6 3.0 4dr	Sedan	\$33,129
21	A6 3.0 Quattro 4dr	Sedan	\$35,992
22	A4 3.0 convertible 2dr	Sedan	\$38,325
23	A4 3.0 Quattro convertible 2dr	Sedan	\$40,075
24	A6 2.7 Turbo Quattro 4dr	Sedan	\$38,840
25	A6 4.2 Quattro 4dr	Sedan	\$44,936
26	A8 L Quattro 4dr	Sedan	\$64,740
27	S4 Quattro 4dr	Sedan	\$43,556

### Output 5. Results for Example 5

Please note, if you are using a BY statement, are generating multiple worksheets, and need each sheet to be named the value of the BY value, you can use #BYVAR*n*, #BYVAL*n*, or #BYLINE in the SHEET\_NAME suboption. The biggest advantage to using BY statements to create multiple sheets is that you have control over the sheet names.

### Code for Multiple Tables in Multiple Worksheets

Now it is time to combine the aforementioned techniques. The next logical progression is to create a file that has multiple tables in multiple worksheets. Currently, the ODS destination for Excel does not have a suboption for forcing the creation of a new sheet at a specific place in your program. This ability will likely be added in a future release of SAS, but for SAS 9.4 TS1M3, you must generate a dummy output object in order to trigger a new sheet.

First, in Example 6, the SHEET\_INTERVAL suboption is set to OUTPUT. The dummy output object is generated by a DATA\_NULL\_ step. After the DATA step, the SHEET\_INTERVAL suboption is set back to NONE. Notice that the ODS EXCEL statements used to toggle the SHEET\_INTERVAL suboption do not have the FILE= option; they contain only the OPTIONS option. Also, notice the ODS EXCLUDE

statements. These statements prevent the results of the DATA \_NULL\_ step from being included in the workbook. Remember, the output object does not need to be seen, just used to start a new worksheet.

**Example 6:**

```
ods excel file='example6.xlsx' options(sheet_interval='none'
    sheet_name='Class');
proc report data=sashelp.class;
    column age height weight;
    define age / group;
    define height / mean;
    define weight / mean;
    rbreak before / summarize;
run;

proc report data=sashelp.class;
    column sex height weight;
    define sex / group;
    define height / mean;
    define weight / mean;
    rbreak before / summarize;
run;

ods excel options(sheet_interval='output');

ods exclude all;
data _null_;
    declare odsout obj();
run;
ods select all;

ods excel options(sheet_interval='none' sheet_name='Heart');

proc report data=sashelp.heart;
    column bp_status height weight;
    define bp_status / group;
    define height / mean;
    define weight / mean;
    rbreak before / summarize;
run;

proc report data=sashelp.heart;
    column sex height weight;
    define sex / group;
    define height / mean;
    define weight / mean;
    rbreak before / summarize;
run;
ods excel close;
```

	A	B	C		A	B	C
1	Age	Height	Weight	1	Blood Pressure Status	Height	Weight
2		62.336842	100.02632	2		64.813185	153.08668
3	11	54.4	67.75	3	High	64.713687	161.84636
4	12	59.44	94.4	4	Normal	65.005841	149.16768
5	13	61.433333	88.666667	5	Optimal	64.578947	138.7202
6	14	64.9	101.875	6			
7	15	65.625	117.375	7	Sex	Height	Weight
8	16	72	150	8		64.813185	153.08668
9				9	Female	62.572586	141.38864
10	Sex	Height	Weight	10	Male	67.567374	167.46615
11		62.336842	100.02632	11			
12	F	60.588889	90.111111	12			
13	M	63.91	108.95	13			
14				14			
15				15			
16				16			
17				17			
18				18			
19				19			
20				20			

**Output 6. Results for Example 6**

### Macro Code for Multiple Worksheets

You might have noticed that the code in Example 6 runs almost exactly the same PROC REPORT step four times. Often, Excel reports consist of the same table structure used multiple times, only the values within the table change based on account number, company, school, and so on. The code can be converted into a macro program, which is a great tool for repetitive steps.

Statements needed to start a new worksheet can also go inside of a macro program. Then you can use one statement, the macro call, wherever you need it without it clogging up the main part of your program.

Example 7 creates the exact same file as Example 6. The difference is the code that is used. Example 7 places the PROC REPORT code inside of a macro program called %PRCODE. It also places the DATA \_NULL\_ step inside of a macro program called newsheet. Each macro program is called and executed the appropriate number of times in order to generate one workbook containing two worksheets, each with two tables.

#### Example 7:

```

%macro prcode(sheetname, dsname, grpvar);
ods excel options(sheet_interval='none' sheet_name="&sheetname");
proc report data=&dsname;
  column &grpvar height weight;
  define &grpvar / group;
  define height / mean;
  define weight / mean;
  rbreak before / summarize;
run;
%mend prcode;

%macro newsheet;
ods excel options(sheet_interval='output');

ods exclude all;
data _null_;
  declare odsout obj();
run;

```

```
ods select all;
%mend newsheet;

ods excel file='example7.xlsx';
%prcode(Class, sashelp.class, age);
%prcode(Class, sashelp.class, sex);
%newsheet;
%prcode(Heart, sashelp.heart, bp_status);
%prcode(Heart, sashelp.heart, sex);
ods excel close;
```

One very important aspect to consider when using macros is whether you are generating one file or multiple files. Example 7 generates only one file; therefore, the ODS EXCEL statement with the FILE= option is outside of the macro, so it is only executed once. You might need to generate multiple files, each one for a different division within the company. In that situation, the statement with the FILE= option and the ODS EXCEL CLOSE statement should be placed within the macro program so that it is executed each time the macro is executed.

Placing PROC REPORT inside of a macro program is extremely helpful when generating a table that uses an ACROSS variable and assignment or CALL DEFINE statements are needed for those columns. The macro code makes it very easy to generate `_cn_` references. You might also consider using a macro logic to generate DEFINE statements for multiple analysis variables.

This paper cannot demonstrate all of the possible ways to use macro programs when generating Excel reports. Hopefully, the code above gives you an idea of the possibilities that are available to you.

## INVEST TIME IN LEARNING PROC REPORT

The time you spent learning about the ODS destination for Excel in the previous section is going to pay off. Knowing its capabilities and how it works will help you develop your SAS programs faster. And remember, creating the Excel report via SAS will eventually save you hours of time each month.

Next, you need to spend a little time thinking about the procedure that will help you create that report. As mentioned in the Introduction, this paper focuses on using PROC REPORT. The paper focuses on using PROC REPORT to create the tabular output because it can do the following:

- Calculate statistics
- Create columns using data that is not in the data set
- Add lines of text
- Transpose the data without first restructuring the data set
- Add blank rows
- Apply formatting based on values of other variables

The PRINT and TABULATE procedures have some of these capabilities, but not all. PROC REPORT is the best way to mimic the tables that you currently create manually. No paper can teach you everything you need to know about PROC REPORT, but it can demonstrate how to achieve the important features of the tables that you currently have in Excel.

One of the most important aspects of using PRINT, TABULATE, or REPORT to generate the final report is that you no longer have to dump all of your data into Excel and then manually generate the report. Instead, you use the full data set as input to PRINT, TABULATE, or REPORT and generate the report for Excel using the procedure syntax. In most cases, the amount of data sent to Excel will be much smaller.

## REPORT CREATION

Whether you realize it or not, the Excel report you are creating has a lot of different parts and features. Each one of those individual characteristics has to be thought about and programmed. One way to make sure you do not forget anything is to examine a report that you already have and list all of the features it includes and then use that as a guide.

This section does things a little differently than the previous sections. Instead of showing you the code followed by the output, we start with the results shown in Output 8 and then concentrate on the code.

The key features of this output are called out by the circled numbers and are explained below. The PROC REPORT code follows the explanations. The callout numbers in the code and in the explanations correspond to the numbers in Output 8. The code might seem lengthy, but remember, you did a lot of work in Excel too! Every piece of your report requires a piece of code.

	A	B	C	D	E
1	<b>Year-end Sales Totals</b>				
2					
3	<b>Product Name</b>	<b>Product Number</b>	<b>Unit Cost</b>	<b>Unit Price</b>	<b>Unit Sale</b>
4	<b>Region1</b> ①				
5	Product1	001 ②	\$1,434 ③	\$3,101.39	23,769
6	Product2	002	\$3,141	\$6,831.00	24,742
7	Product3	003	\$1,209	\$1,982.29	20,667
4,5	<b>Total Region1</b> ⑥	⑦	<b>\$5,784</b>	<b>\$11,914.68</b>	<b>69,178</b> ⑧
9	⑨				
10	<b>Region2</b>				
11	Product4	004	\$1,854	\$4,033.26	25,734
12	Product5	005	\$984	\$2,131.20	26,250
13	Product6	006	\$1,332	\$2,884.41	21,572
14	Product7	007	\$1,146	\$2,488.50	30,745
15	Product8	008	\$1,554	\$3,371.33	25,982 ⑩
16	Product9	009	\$4,680	\$10,172.59	29,048
17	Product10	010	\$1,611	\$3,501.65	22,268
18	Product11	011	\$1,782	\$3,866.36	15,339
19	<b>Total Region2</b>		<b>\$14,943</b>	<b>\$32,449.30</b>	<b>196,938</b>
20					
21	<b>Region3</b>				
22	Product12	012	\$4,014	\$8,731.80	26,892
23	Product13	013	\$3,333	\$7,222.40	20,934
24	Product14	014	\$1,596	\$3,142.90	26,971
25	Product15	015	\$3,282	\$7,139.88	24,331
26	Product16	016	\$2,268	\$4,926.05	27,841
27	Product17	017	\$2,199	\$4,777.68	23,642
28	<b>Total Region3</b>		<b>\$16,692</b>	<b>\$35,940.71</b>	<b>150,611</b>
29					
4,11	<b>All Regions</b> ⑫	⑬	<b>\$37,419</b>	<b>\$80,305</b> ⑭	<b>416,727</b> ⑮
31					

### Output 8. Results for Example 8

The key features of the above Excel report are as follows:

- ① Section headers with a colored background
- ② Leading zeros for character variables

- ③ Appropriately formatted analysis variables
- ④ Bolded values in all summary rows
- ⑤ Summary row for each region
- ⑥ Customized text in the summary row for each region
- ⑦ Borders on the summary row for each region
- ⑧ Formulas for the SALE variable in the summary row for each region
- ⑨ A blank row between each section
- ⑩ Coloring of the SALE column based on sales criteria
- ⑪ Summary row for the entire report
- ⑫ Customized text in the summary row for the entire report
- ⑬ Borders on the summary row for the entire report
- ⑭ Change the format for the PRICE variable in the summary row for the entire report
- ⑮ Formulas for the SALE variable in the summary row for the entire report

### Example 8:

```

data pricedata;
  length productname $50.;
  set sashelp.pricedata;

  if productname="Product8" then product=8;
  else if productname="Product9" then product=9;
  else if productname="Product10" then product=10;
  else if productname="Product11" then product=11;

  productnum = put(product,z3.);
run;

ods excel file='example8.xlsx' options(embedded_titles='yes'
  frozen_headers='yes');
title "Year-end Sales Totals";
proc report data=pricedata style(summary)=[font_weight=bold]; ④
  column region regionname product productname productnum cost price
  sale;

  define region / group noprint order=internal;
  define regionname / group noprint;
  define product / group noprint order=internal;
  define productname / group;
  define productnum / group 'Product Number' format=$3.; ②
  ③
  define price / style(column)=[tagattr="format:$#,###.00;$-#,###.00"];
  define cost / style(column)=[tagattr="format:$#,###;$-#,###"];
  define sale / style(column)=[tagattr="format:#,###"];

  break after regionname / summarize; ⑤
  rbreak after /summarize; ⑪

  compute before regionname /style=[background=lightblue just=1
    font_weight=bold];

```

```

        line regionname $20.; ①
    endcomp;

    compute after regionname;
        productname = catx(' ', 'Total', regionname); ⑥
        if _break_ ^= '' then call define(_row_, 'style',
            'style=[bordertopstyle=solid bordertopwidth=1pt
            bordertopcolor=black borderbottomstyle=solid borderbottomwidth=1pt
            borderbottomcolor=black]'); ⑦
        line ' '; ⑨
    endcomp;

    compute after;
        productname = "All Regions"; ⑫
        call define(_row_, 'style', 'style=[bordertopstyle=solid
            bordertopwidth=1pt bordertopcolor=black borderbottomstyle=solid
            borderbottomwidth=1pt borderbottomcolor=black]'); ⑬
        call define('price.sum', 'style', 'style=[tagattr="format:$#,###;$-
            #,###"]'); ⑭
    endcomp;

    compute sale;
        if _break_ = '' then do; ⑩
            if sale.sum < 20000 then call define(_col_, 'style',
                'style=[background=red]');
            else if sale.sum > 30000 then call define(_col_, 'style',
                'style=[background=green]');
        end;
        else if upcase(_break_) = 'REGIONNAME' then do; ⑧
            if region = 1 then call define('sale.sum', 'style',
                'style=[tagattr="formula:sum(E5:E7) format:#,###"]');
            else if region = 2 then call define('sale.sum', 'style',
                'style=[tagattr="formula:sum(E11:E18) format:#,###"]');
            else if region = 2 then call define('sale.sum', 'style',
                'style=[tagattr="formula:sum(E22:E27) format:#,###"]');
        end;
        else if _break_ = '_RBREAK_' then do; ⑮
            call define('sale.sum', 'style',
                'style=[tagattr="formula:sum(E8,E19,E28) format:#,###"]');
        end;
    endcomp;
run;
ods excel close;

```

### ① Section headers with a colored background

Each region in the report needs its own section. The first row of that section should indicate the name of the region and have a background color. The COMPUTE BEFORE REGIONNAME statement uses the STYLE= option to apply the background color. The LINE statement within the compute block writes the name of the region.

### ② Leading zeros for character variables

PRODUCTNUM is a character variable that contains numbers starting with zeros. Specifying the format \$3. prevents Microsoft Excel from stripping the leading zeros. Another method for accomplishing the same thing is to specify the type STRING.

```
define productnum / group 'Product Number'  
    style(column)={tagattr="type:String"};
```

In a future release of SAS, it is likely you will be able to specify a format to maintain leading zeros.

```
define productnum / group 'Product Number'  
    style(column)={tagattr="format:@"};
```

With all three of these methods, Excel puts small green triangles in the corner of the cell to indicate a conflict with placing a numeric value in a character column. In a future release of SAS, you will likely be able to specify the following format to both maintain the zero and avoid the triangles.

```
define productnum / group 'Product Number'  
    style(column)={tagattr="format:000"};
```

### ③ Appropriately formatted analysis variables

The DEFINE statements for the analysis variables use the TAGATTR attribute to specify the Microsoft Excel formula for that column. The PRICE and COST variables are given a currency format. The SALE variable is given a numeric format with commas.

### ④ Bolded values in all summary rows

The STYLE(SUMMARY)= option applies bolding to all the summary lines created with either a BREAK or RBREAK statement.

### ⑤ Summary row for each region

The BREAK statement generates a summary row for each value of REGIONNAME.

### ⑥ Customized text in the summary row for each region

It is helpful to delineate the summary row for each region with informative text. For this report, the text should include the region name. The REGIONNAME column is not printed; therefore, the text needs to be placed inside the PRODUCTNAME column. The assignment statement for creating the text is within the COMPUTE AFTER REGIONNAME block.

The DATA step above the PROC REPORT code is required because the original length of the PRODUCTNAME variable was not long enough to hold all of the text. The length had to be changed via a DATA step; it cannot be done within PROC REPORT.

### ⑦ Borders for the summary row for each region

The COMPUTE AFTER REGIONNAME block controls the row generated by the BREAK statement. The CALL DEFINE statement adds borders to those rows. Notice that the BORDERTOPSTYLE= and BORDERBOTTOMSTYLE= attributes are set to solid to ensure that the borders appear with the appropriate style, a solid line.

### ⑧ Formulas for the SALE variable in the summary row for each region

A CALL DEFINE statement using the TAGATTR attribute to apply a formula is needed for each of the regions in the report. The cells that are specified in the SUM formula were determined by first creating the file and determining the row number for each of the region totals.

You will likely need to run the program multiple times to determine the correct column and cell reference needed for the formula.

### ⑨ A blank row between sections

The LINE statement specified with blank text inserts the blank row after each region section.

### ⑩ Coloring of the SALE column based on sales criteria

The background color of the SALE column indicates whether the value is well below or well above the target value. Poor performance is indicated by a red background, and high performance is indicated by a green background. In this example, the target value is a made-up value that is hardcoded. In more robust code, the target value could be a macro variable or a variable in the data set. Also, more than two colors could be applied. The example demonstrates adding the background colors to the cell for one of the analysis variables, but it could be applied to all three if desired.

#### **11 Summary row for the entire report**

The RBREAK statement generates an overall summary row for the entire report.

#### **12 Customized text on the summary row for the entire report**

The COMPUTE AFTER block controls the row generated by the RBREAK statement. The PRODUCTNAME variable is assigned an appropriate label for the overall summary row.

#### **13 Borders on the summary row for the entire report**

The CALL DEFINE statement adds borders to that row. Notice that the BORDERTOPSTYLE= and BORDERBOTTOMSTYLE= attributes are set to solid in order to ensure that the borders appear with the appropriate style, a solid line.

#### **14 Change the format for the PRICE variable in the summary row for the entire report**

A CALL DEFINE statement within the COMPUTE AFTER block changes the TAGATTR attribute for the PRICE column in the overall summary row. Additional CALL DEFINE statements could be used to change the TAGATTR for the other two analysis columns.

This example uses STYLE as the second argument to CALL DEFINE because SAS is sending a specific Excel format to the destination. For other formats, like formats defined in SAS, you can use the same concept. However, instead of STYLE as the second argument, you would use FORMAT.

#### **15 Formulas for the SALE variable in the summary row for the entire report**

The CALL DEFINE statement uses the TAGATTR attribute to apply a formula to the overall summary row. The cells that are specified in the SUM function were determined by first creating the file and seeing which rows contained the region totals. The program is then modified, adding the CALL DEFINE statement with the correct cell references.

You will likely need to run the program multiple times to determine the correct column and cell reference needed for the formula.

The first four DEFINE statements (REGION, REGIONNAME, PRODUCT, PRODUCTNAME) were not mentioned specifically above because they do not directly create any of the features of the final report. However, the variables in those DEFINE statements play a very important role. The REGION and PRODUCT variables are numeric values. They correspond to the REGIONNAME and PRODUCTNAME variables. They are used to ensure that the text values that do appear in the report are in the desired order. The SASHELP.PRICEDATA data set comes with these numeric values. You might need to create numeric variables to ensure that your categorical character values are in the desired sort order.

The placement of those variables in the COLUMN statement is also very important. The numeric REGION variable is followed by the character REGIONNAME variable. Then the numeric PRODUCT is listed, followed by the character PRODUCTNAME variables. It is important to keep the pairs together and to list the numeric variable first.

Example 8 has many features for a relatively simple report. The code becomes longer and more complex with more grouping variables or more lines of text. Using ACROSS variables also adds a layer of complexity. Planning out your report and ensuring that you have the variables needed to accomplish the output is important for these types of reports.

## **CUSTOMIZED TEXT**

In Example 8, above, it was relatively straightforward to place customized text in the summary rows. The LENGTH statement in the DATA step increased the width of the variable and the assignment statement in

the compute block provided the text. The process gets more complex if the column you want to add text to is created by a numeric variable. The ODS destination for Excel does not honor the PRETEXT= attribute that is sometimes used with other ODS destinations to add text to numeric values.

When planning your report and creating the code, think about the variables you are using. You might have a variable that is numeric but has a format that displays character strings. If you want additional text in the report for this column, you must create a character version of the variable. You can create the character version in a DATA step above the PROC REPORT code or via a COMPUTED column within the PROC REPORT code.

Example 9 demonstrates creating a character version of REGION. Notice that REGION is in the COLUMN statement. It must be in the COLUMN statement so that its value is available. It also serves as a grouping and sorting variable. The format created by the PROC FORMAT step would normally be applied to the REGION variable, but here it is used to create the CHAR\_REG variable.

**Example 9:**

```
proc format;
  value reg
    1 = 'Top Region'
    2 = 'Middle Region'
    3 = 'Bottom Region';
run;

ods excel file='example9.xlsx';
proc report data=sashelp.pricedata;
  column region char_reg cost price sale;
  define region / group noprint;
  define char_reg / computed 'Region';
  rbreak after / summarize;
  compute char_reg /char length=50;
    char_reg = put(region,reg.);
    if _break_ = '_RBREAK_' then char_reg = 'Total for All Regions';
  endcomp;
run;
ods excel close;
```

	A	B	C	D
1	<b>Region</b>	<b>Unit Cost</b>	<b>Unit Price</b>	<b>Unit Sale</b>
2	Top Region	5784	11914.68	69178
3	Middle Region	14943	32449.295	196938
4	Bottom Region	16692	35940.705	150611
5	Total for All Regions	37419	80304.68	416727
6				

**Output 9. Output from Example 9**

**GRAY LINES AND BORDERS**

You might have noticed that the default style for the ODS destination for Excel creates a table with gray lines, with the rest of the sheet space all white. The style template for the destination was created to make the table look like what is normally seen in blank Microsoft Excel spreadsheets. This section demonstrates how to remove those gray lines and how to add borders to the table.

**Microsoft Excel’s Gray Lines**

As mentioned previously, gray cell lines are not controlled by the GRIDLINES suboption. The GRIDLINES suboption only has an effect when you look at the spreadsheet in Print Preview mode or when you actually print the spreadsheet. To remove the gray lines within the spreadsheet, you must change the BORDERSTYLE style attribute to HIDDEN. PROC TEMPLATE is required to make this change.

### Example 10:

```
ods path(prepend) work.templat(update);
proc template;
define style styles.nogray;
  parent=styles.excel;
  class header, data, dataemphasis, linecontent /
    borderstyle=hidden;
end;
run;

ods excel file='example10.xlsx' style=styles.nogray;
proc report data=sashelp.class;
run;
ods excel close;
```

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

### Output 10. Results for Example 10

Some style templates might not produce the gray lines. If you want to include them, modify the style you are using with the CLASS statement from the code below.

```
proc template;
define style styles.gray;
  parent=styles.ocean;
  class header, data, dataemphasis, linecontent /
    borderstyle=solid;
end;
run;
```

### Borders Inside the Table

When creating an Excel report by hand, you can add borders around specific cells by right-clicking them and choosing **Format Cells-Borders**. You can also add borders when creating a report with SAS. Currently, the FRAME= and RULES= attributes often used with PROC REPORT style overrides in other destinations are not honored in the destination for Excel. PROC TEMPLATE code can generate a new style that inserts the borders. Example 11 shows code to apply borders to tables generated by PROC REPORT.

### Example 11:

```
ods path(prepend) work.templat(update);
proc template;
define style styles.eborders;
  parent=styles.excel;
  class data /
    borderstyle=solid
    borderwidth=1px
    bordercolor=black;
  class header /
    borderstyle=solid
    borderwidth=1px
    bordercolor=black;
end;
run;

ods excel file='example11.xlsx' style=styles.eborders;
proc report data=sashelp.class;
run;
ods excel close;
```

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

### Output 11. Results for Example 11

#### COLOR IN THE HEADERS

Excel reports often have text in the rows above the column headers/labels. They provide dates, company information, or simply more informative text. PROC REPORT can generate these headers using spanning headers, via the COLUMN statement. Often, the cells containing the spanning headers need a background color. Inline formatting commands are used to control the attributes of a spanning header. However, Microsoft Excel does not honor the BACKGROUND= attribute inside of a spanning header.

To change the background color of text above column headers/labels, you must convert the spanning headers to variables that can be defined as ACROSS. The first PROC REPORT step in Example 12 demonstrates including spanning headers. The second portion of Example 12 uses a DATA step to create

three new variables that contain the spanning header text from the first step. The second PROC REPORT step then defines the new variables as ACROSS and assigns an appropriate background color.

**Example 12:**

```
ods excel file='example12.xlsx' options(sheet_interval='none');
proc report data=sashelp.pricedata;
  column region ('A' cost) ('B' price) ('C' sale);
  define region / group;
  define cost / style(header)=[background=yellow];
  define price / style(header)=[background=lightblue];
  define sale / style(header)=[background=lightred];
run;

data pricedata;
  set sashelp.pricedata;
  acr1='A';
  acr2='B';
  acr3='C';
run;
proc report data=pricedata;
  column region acr1,cost acr2,price acr3,sale;
  define region / group;
  define cost / style(header)=[background=yellow];
  define price / style(header)=[background=lightblue];
  define sale / style(header)=[background=lightred];

  define acr1 / across ' ' style(header)=[background=yellow];
  define acr2 / across ' ' style(header)=[background=lightblue];
  define acr3 / across ' ' style(header)=[background=lightred];
run;
ods excel close;
```

	A	B	C	D
1		A	B	C
2	Region ID	Unit Cost	Unit Price	Unit Sale
3	1	5784	11914.68	69178
4	2	14943	32449.295	196938
5	3	16692	35940.705	150611
6				
7		A	B	C
8	Region ID	Unit Cost	Unit Price	Unit Sale
9	1	5784	11914.68	69178
10	2	14943	32449.295	196938
11	3	16692	35940.705	150611
12				

**Output 12. Results for Example 12**

**CONCLUSION**

The ODS destination for Excel provides an amazing opportunity to save time by creating Microsoft Excel reports directly from SAS. You no longer have to spend hours manually creating your weekly, monthly, or quarterly report.

The combination of PROC REPORT and the ODS destination for Excel has the capability to create tabular output that includes lines of text, colors, and customized formatting. The resulting workbook can have multiple tables per worksheet and multiple worksheets. You can even include formulas so that it is obvious how a value was calculated.

The time you spend learning this destination and PROC REPORT will be invaluable. You will eliminate hours of wasted time that could be used on more important tasks than formatting an Excel workbook!

## REFERENCES

SAS Institute Inc. 2016. *SAS 9.4 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc. Available at

<http://support.sas.com/documentation/cdl/en/odsug/67921/HTML/default/viewer.htm#titlepage.htm>.

Eslinger, Jane. 2016. *The SAS Programmer's PROC REPORT Handbook Basic to Advanced Reporting Techniques*. Cary, NC: SAS Press.

## ACKNOWLEDGMENTS

The author is immensely grateful to Marty French for her inspiration, patience, and challenging questions, and to Cynthia Zender and Chevell Parker, both of whom contributed to this paper, and to Vivian McGee, who edited the paper.

## RECOMMENDED READING

Parker, Chevell. 2016. "A Ringside Seat: The ODS Excel Destination versus the ODS ExcelXP Tagset." *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available at

<http://support.sas.com/resources/papers/proceedings16/SAS5642-2016.pdf>.

Eslinger, Jane. 2015. "The REPORT Procedure: A Primer for the Compute Block." *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at

<http://support.sas.com/resources/papers/proceedings15/SAS1642-2015.pdf>.

## CONTACT INFORMATION

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

Jane Eslinger  
SAS Institute Inc.  
Cary, NC  
Email: [support@sas.com](mailto:support@sas.com)  
Web: [support.sas.com](http://support.sas.com)

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

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