

Paper 270-2011

PROC PRINT and ODS: Teaching an Old PROC New Tricks

Darylene Hecht, SAS Institute Inc., Cary, NC

ABSTRACT

PROC PRINT is one of the oldest living procedures in SAS® and is probably the most widely used procedure of all time. In SAS® 9.3, the internals of PROC PRINT were completely rewritten to update it and make it fully compatible with the Output Delivery System (ODS). This paper presents many of PROC PRINT's new tricks to help you create more complex reports for multiple destinations, including HTML, PDF, RTF, and (of course) LISTING. Topics include styles, traffic lighting, PROC DOCUMENT to store and replay the output, the new SUMLABEL and BLANKLINE options, #BYVAL and #BYVAR substitutions to customize titles, and the BYID layout.

You CAN teach an old PROC new tricks!

INTRODUCTION

From its humble beginnings as a way to provide a quick listing of your data, to today's ability to produce complex reports in a variety of output formats, PROC PRINT continues to be one of the most frequently used procedures by SAS programmers. However, until SAS 9.3, there were still areas of the Output Delivery System (ODS) that PROC PRINT was unable to take advantage of due to its older implementation. With SAS 9.3, PROC PRINT has been rewritten and is now fully compatible with ODS, thus bringing even more functionality to users.

This paper will take PROC PRINT code from its simplest form and develop it into a program that produces complex reports using styles in multiple output formats. Along the way, we'll highlight some features that are new to SAS 9.3, as well as features that have been available in previous releases, but might not be well known to users. We will also examine how to use PROC DOCUMENT to store the procedure code, along with the data, in order to replay it later using any chosen output format.

OUR EXAMPLE DATA SET

The first thing we need is data. To build our report, we will use data that represents energy expenditures in various states and regions. The following code creates the data set *energy*, with variables for region, division, state, expenditure type, and expenditure amount.

```
data energy;
  length state $2;
  input region division state $ type expenditures @@;
  datalines;
  1 1 ME 1 708 1 1 ME 2 379 1 1 NH 1 597 1 1 NH 2 301
  1 1 VT 1 353 1 1 VT 2 188 1 1 MA 1 3264 1 1 MA 2 2498
  1 1 RI 1 531 1 1 RI 2 358 1 1 CT 1 2024 1 1 CT 2 1405
  1 2 NY 1 8786 1 2 NY 2 7825 1 2 NJ 1 4115 1 2 NJ 2 3558
  1 2 PA 1 6478 1 2 PA 2 3695 4 3 MT 1 322 4 3 MT 2 232
  4 3 ID 1 392 4 3 ID 2 298 4 3 WY 1 194 4 3 WY 2 184
  4 3 CO 1 1215 4 3 CO 2 1173 4 3 NM 1 545 4 3 NM 2 578
  4 3 AZ 1 1694 4 3 AZ 2 1448 4 3 UT 1 621 4 3 UT 2 438
  4 3 NV 1 493 4 3 NV 2 378 4 4 WA 1 1680 4 4 WA 2 1122
  4 4 OR 1 1014 4 4 OR 2 756 4 4 CA 1 10643 4 4 CA 2 10114
  4 4 AK 1 349 4 4 AK 2 329 4 4 HI 1 273 4 4 HI 2 298
  ;
```

In order to display our report using BY groups, we need to sort the data by the variables that classify the data. In our example, those variables are region, division, and type. The output data set (produced by PROC SORT) that we will use for our examples is *energy_report*.

```
proc sort data=energy out=energy_report;
  by region division type;
run;
```

Last, let's create some formats that transform the numeric values of region, division, and type to descriptive text that will be used in the report.

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

```
proc format;
value regfmt 1='Northeast'
              2='South'
              3='Midwest'
              4='West';
value divfmt 1='New England'
              2='Middle Atlantic'
              3='Mountain'
              4='Pacific';
value usetype 1='Residential Customers'
               2='Business Customers';
run;
```

LISTING: AN OLDIE, BUT GOODIE

The *SAS Introductory Guide*, published in 1985, states:

“The natural first step after getting your data into a SAS data set is to print it” (Helwig 1985, p. 31).

That’s still true 25 years later. Even if you do not physically print the data, many times you still want to display it to check accuracy, provide a reference to the data values, or look at the value ranges. In the beginning, there was the Listing, sparkling with information printed on green and white striped paper. And mainframe users saw that it was good. Even though many output formats have evolved since those early years, Listing is still considered a “quick and dirty” way to produce, well, a listing of your output. The following code shows the simplest PROC PRINT step and a partial view of the Listing output that is produced.

```
proc print data=energy_report;
run;
```

Obs	state	region	division	type	expenditures
1	ME	1	1	1	708
2	NH	1	1	1	597
3	VT	1	1	1	353
4	MA	1	1	1	3264
5	RI	1	1	1	531
6	CT	1	1	1	2024
7	ME	1	1	2	379
8	NH	1	1	2	301
9	VT	1	1	2	188
10	MA	1	1	2	2498
11	RI	1	1	2	358
12	CT	1	1	2	1405
13	NY	1	2	1	8786
14	NJ	1	2	1	4115
15	PA	1	2	1	6478
16	NY	1	2	2	7825
17	NJ	1	2	2	3558
18	PA	1	2	2	3695
19	MT	4	3	1	322
20	ID	4	3	1	392
21	WY	4	3	1	194
22	CO	4	3	1	1215

Output 1. Listing Output from Simple PROC PRINT

Listing does the job, but the output is certainly not what would typically be considered a final report format. But what a quick and easy way to see what the data looks like!

With the addition of a couple of simple ODS statements, you can quickly make the report look more polished by creating it in HTML. The following example and partial output shows the same report in HTML.

```
ods html file='my_report';
proc print data=energy_report;
run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary

Obs	state	region	division	type	expenditures
1	ME	1	1	1	708
2	NH	1	1	1	597
3	VT	1	1	1	353
4	MA	1	1	1	3264
5	RI	1	1	1	531
6	CT	1	1	1	2024
7	ME	1	1	2	379
8	NH	1	1	2	301
9	VT	1	1	2	188
10	MA	1	1	2	2498
11	RI	1	1	2	358
12	CT	1	1	2	1405
13	NY	1	2	1	8786
14	NJ	1	2	1	4115
15	PA	1	2	1	6478

Output 2. HTML Output from Simple PROC PRINT

There are many output destinations currently supported by ODS, including HTML, RTF, PDF, PostScript, and XML. Most of the output examples shown in this paper are HTML. However, any of these reports can be easily produced in any of the other output formats simply by changing the destination in the ODS statement or by adding ODS statements to create multiple output formats at the same time. For example, the following code produces the same report as a Listing, an HTML file, a PDF file, and an RTF file.

```
ods listing;
ods html file='my_report.html';
ods pdf file='my_report.pdf';
ods rtf file='my_report.rtf';
proc print data=energy_report;
run;
ods _all_ close;
```

TABLE OF CONTENTS

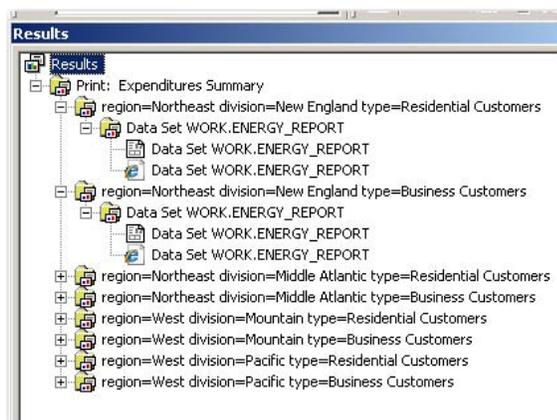
When you run SAS interactively, you can use the Results window to keep track of the output you are creating. The first difference you'll see in SAS 9.3 is in the presentation of the Results window when using PROC PRINT. In the Results window, each of the output objects is displayed by an icon indicating which output destination was used. For each BY group, a separate output object is created and is separately accessible. In previous releases, the Listing results were displayed as a single output object and you couldn't navigate to each BY group separately. The following view of the Results window in SAS 9.2 shows the Listing icon representing the entire document, while the HTML icon shows each BY group as a separate section.

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued



Display 1. SAS 9.2 Results Window

In SAS 9.3, the Listing is displayed in a consistent manner with the non-Listing destinations, and you can navigate separately to each BY group in the Listing, just as you can with the non-Listing destinations. The following view of the Results window is taken from SAS 9.3.



Display 2. SAS 9.3 Results Window

THE BLANKLINE OPTION

Now let's start sprucing up the output, creating a report that conveys the information to the reader in a format that is easier to interpret.

BLANKLINE is a new option added to PROC PRINT in SAS 9.2. It is a generalization of the DOUBLE option, which writes a blank line between each observation to make the output double-spaced. However, DOUBLE is valid only for Listing output and doesn't allow any flexibility in how often to insert the blank line. The BLANKLINE option enables you to specify how many observations you'd like to write before inserting a blank line and is valid for all destinations. BLANKLINE is specified in the PROC statement and its syntax in the simplest form is as follows:

BLANKLINE = *n*

This code writes a blank line after every *n* observations. For example, perhaps our data would be easier to read when displayed in groups of 5, as in the following example:

```
ods html file='my_report';
  proc print data=energy_report blankline=5;
    run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary

Obs	state	region	division	type	expenditures
1	ME	1	1	1	708
2	NH	1	1	1	597
3	VT	1	1	1	353
4	MA	1	1	1	3264
5	RI	1	1	1	531
6	CT	1	1	1	2024
7	ME	1	1	2	379
8	NH	1	1	2	301
9	VT	1	1	2	188
10	MA	1	1	2	2498
11	RI	1	1	2	358
12	CT	1	1	2	1405
13	NY	1	2	1	8786
14	NJ	1	2	1	4115
15	PA	1	2	1	6478
16	NY	1	2	1	7885

Output 3. Using the BLANKLINE Option

In SAS 9.3, the BLANKLINE behavior was modified slightly. In SAS 9.2, the observation count to determine when to insert a blank line was reset to 0 at the beginning of each page. With SAS 9.3, the observation count is reset at the beginning of each BY group instead. This makes the output more consistent between Listing, which has physical pages, and an output format like HTML, which does not.

TITLES, BY GROUPS, AND FORMATS

There are several standard SAS statements that add more clarity to our output. In the following example, a TITLE statement has been added to identify our report. Instead of using the BLANKLINE option to group our observations, let's use the BY statement to create groupings for easier reading. And the FORMAT statement applies the formats that we previously created, in addition to the standard SAS format DOLLAR, to specific variables.

```
ods html file='my_report';
  title "Expenditures Summary";
proc print data=energy_report;
  by region division type;
  format region regfmt. division divfmt. type usetype.
         expenditures dollar12.2;
run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

<i>Expenditures Summary</i>		
region=Northeast division=New England type=Residential Customers		
Obs	state	expenditures
1	ME	\$708.00
2	NH	\$597.00
3	VT	\$353.00
4	MA	\$3,264.00
5	RI	\$531.00
6	CT	\$2,024.00
region=Northeast division=New England type=Business Customers		
Obs	state	expenditures
7	ME	\$379.00
8	NH	\$301.00
9	VT	\$188.00
10	MA	\$2,498.00
11	RI	\$358.00
12	CT	\$1,405.00

Output 4. Using Titles, BY Groups, and Formats

THE ID AND SUM STATEMENTS

Our report is looking better, but the headings between each BY group look awkward. We can add an ID statement to help with that. Using the ID statement in conjunction with the BY statement causes PROC PRINT to use a special layout that makes the output much more readable. With the BYID layout, the heading between each BY group, which is called the BY line, is suppressed; and the value of the ID variable is listed as part of the table instead of in the BY line.

We can add even more value to the information presented by adding a SUM statement. The SUM statement totals the values of each specified numeric variable. When the SUM statement is used in conjunction with the BY statement, a total is presented for each BY group as illustrated in the following example:

```
ods html file='my_report';
  title "Expenditures Summary";
proc print data=energy_report;
  by region division type;
  id region division type;
  sum expenditures;
  format region regfmt. division divfmt. type usetype.
         expenditures dollar12.2;
run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary

region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00

region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 5. Using BYID Layout and SUM Statement

THE SUMLABEL OPTION

The SUMLABEL option, introduced in SAS 9.2, is used in the PROC statement to indicate that the label of the BY variables should be used on the total lines instead of the BY variable names. The following code creates a report that shows the expenditure totals by geographic region. By adding the SUMLABEL option and a LABEL statement, we can customize the total line. In this example, we are not using the special BYID layout.

```
ods html file='my_report';
  title "Expenditures Summary";
  proc print data=energy_report sumlabel noobs;
    by region division;
    var state expenditures;
    sum expenditures;
    label region = 'Geographic Region';
    format region regfmt. division divfmt. type usetype.
           expenditures dollar12.2;
  run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary	
Geographic Region=Northeast division=New England	
state	expenditures
ME	\$708.00
NH	\$597.00
VT	\$353.00
MA	\$3,264.00
RI	\$531.00
CT	\$2,024.00
ME	\$379.00
NH	\$301.00
VT	\$188.00
MA	\$2,498.00
RI	\$358.00
CT	\$1,405.00
division	\$12,606.00
Geographic Region=Northeast division=Middle Atlantic	
state	expenditures
NY	\$8,786.00
NJ	\$4,115.00
PA	\$6,478.00
NY	\$7,825.00
NJ	\$3,558.00
PA	\$3,695.00
division	\$34,457.00
Geographic Region	\$47,063.00

Output 6. Using the SUMLABEL Option

STYLES

We can customize the appearance of the report even further using styles. Styles enable you to specify attributes such as colors, backgrounds, fonts, cell widths, cell heights, and justifications for various parts of the report. Styles affect the appearance of all destinations except Listing.

STYLE DEFINITIONS

The simplest way to specify a new style is to use one of the predefined style definitions. A *style definition* is a complete description of all the attributes to use when creating the report. *Style attributes* are the smallest building blocks of a style definition. Attributes specify the report's colors, fonts, backgrounds, cell sizes, and so on. Attributes are grouped into categories called *style elements*. A style element describes the attributes for a particular piece of the report. Examples of style elements are Header, Data, and RowHeader. Finally, the style elements are grouped together to form a complete style definition that determines the appearance of the overall report. Style attributes and style elements are discussed more fully in the following sections. The general form of a style definition is as follows:

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

```
my-style-definition
  style-element-1
    attribute(s)
  style-element-2
    attribute(s)
  .....
  style-element-n
    attribute(s)
end-my-style-definition
```

Some of the predefined style definitions provided by ODS are Analysis, Gantt, BlockPrint, and (new for 9.3) HtmlBlue. To see a complete list of the predefined style definitions, run the following code:

```
proc template;
  list styles;
run;
```

You can also create your own style definitions using PROC TEMPLATE. However, that topic is beyond the scope of this paper. For more information about creating and using style elements and definitions, see *Output Delivery System: The Basics and Beyond* (Haworth, Zender, and Burlew 2009).

There is always a default style definition for any report. To override that and use a different predefined style definition, specify it in the ODS statement. For example, the following code changes the style definition for the report to HtmlBlue:

```
ods html file='my_report' style=HtmlBlue;
  title "Expenditures Summary Using HtmlBlue Style";
proc print data=energy_report;
  by region division type;
  id region division type;
  sum expenditures;
  format region regfmt. division divfmt. type usetype.
         expenditures dollar12.2;
run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 7: Changing the Default Style Definition

STYLE SPECIFICATIONS

Another way to specify a new style is by using a style specification in the PROC, VAR, ID, and/or SUM statements. Using a style specification overrides style elements or attributes that were defined in the given style definition. There are three parts to a style specification for PROC PRINT—the location, the style element, and the style attribute. The general syntax of a style specification is as follows:

```
STYLE <(location(s))> = <style-element> <[style-attribute(s)]>;
```

Location

The location identifies the part of the report that the style affects. For PROC PRINT, there are nine defined locations. The following table shows the defined location names and the part of the report that is affected by each location.

Location Name	Area of Report Affected
TABLE	The underlying table structure
OBS	The values in the OBS and ID columns
DATA	The values in columns other than OBS and ID
OBSHEADER	The header of the OBS and ID columns
HEADER	The header of columns other than OBS and ID
BYLABEL	The BY variable name or label in the SUM line
TOTAL	The cells in the SUM line containing totals for each BY group

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Location Name	Area of Report Affected
GRANDTOTAL	The cells in the SUM line containing the grand total for the whole report
N	The N= line

Table 1. Defined Locations for PROC PRINT

Style Attributes

Style attributes describe what you want to change. Each style attribute specification has the following general form:

attribute-name = attribute-value

Some examples of attribute specifications are background=red, foreground=blue, and font_style=italic. A complete list of possible attributes is documented in the *Base SAS Procedures Guide*.

If we wanted to change the background color of the data cells in our report to blue and the background of the OBS cells to red, we could use the following style specification in the PROC statement:

```
proc print data=energy_report
  style (data) = [background=blue]
  style (obs) = [background=red foreground=white];
```

Expenditures Summary Using Styles on PROC Statment				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			Ri	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			Ri	\$368.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 8: Specifying Style Attributes

Style Elements

As stated previously, a style element consists of a grouping of style attributes. The PRINT procedure assigns a default style element to each location, providing each location with a default set of attributes. For example, the style element Header is a predefined style element that contains the set of attributes that seem appropriate for a header in a report. As you see in the table below, that style element is then assigned to be the default style element for several

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

locations in the report. As you review the following table of locations and default style elements, do not be confused by the same name being used as both a location and a style element. They refer to two different things, even though they have the same name.

Location Name	Default Style Element
TABLE	Table
OBS	RowHeader
DATA	Data, for all but ID columns; RowHeader, for ID columns
OBSHEADER	Header
HEADER	Header
BYLABEL	Header
TOTAL	Header
GRANDTOTAL	Header
N	NoteContent

Table 2. Default Style Elements for Each Location in PROC PRINT

For example, let's say you'd like for the cells containing the totals in the SUM lines to have the same appearance as the rest of the data cells. You could use the following style specification in the PROC statement to specify that the Data style element should be used in place of the default element Header for those locations. Note that the labels in the SUM lines are not affected. That style is controlled by the BYLABEL location.

```
proc print data=energy_report style (total) = Data
                    style (grandtotal) = Data;
```

Expenditures Summary Using Styles on PROC Statment				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 9: Changing the Default Style Element

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

OVERRIDING AND INHERITING STYLE SPECIFICATIONS

As stated previously, style specifications can be specified in the PROC, VAR, ID, and SUM statements. So far, we have explored the effect only when they are specified in the PROC statement. The difference when they are specified in the other statements is how global the effect is. Styles that are specified in the PROC statement are global to the entire procedure. Styles specified in the VAR, ID, and SUM statements are local to the statement they are specified in. Styles specified in the ID statement affect the ID columns only. Styles specified in the SUM statement are applied to the cells containing sums. Styles specified in the VAR statement are applied to all columns of variables specified in the VAR statement.

The hierarchy in which style specifications are combined to form the final definition starts at the top level with the default style definition. Style elements or attributes specified in the PROC statement are then used to override those same elements or attributes defined in the default. Similarly, style elements or attributes specified in the VAR, ID, or SUM statements override those same elements or attributes defined in the PROC statement. In general, anything that is not overridden by a subsequent style specification is inherited from the parent's specification.

Styles Specified in the PROC Statement

Overriding and inheriting styles can get very confusing because there are so many combinations available. Although this variety gives you a great deal of flexibility in creating your report, it can also be a bit overwhelming. The best approach is to experiment with various combinations until you get a feel for how it all works. Let's start by looking at overriding the default style elements and attributes in the PROC statement.

In the PROC statement, the default style element for a location can be completely overridden and replaced by a new style element for the location. For example, you could use the Header style element for all of the data cells instead of the default style element of Data. The following statement would override the style used for the data cells, but inherit the styles of all the other locations from the default style definition.

```
proc print data=energy_report style(data)=header;
```

Or you might choose to change only certain attributes of the style element and inherit the rest. For example, you could change the background of the data cells to blue, but leave the rest of the default attributes of the Data style element unchanged.

```
proc print data=energy_report style (data) = [background=blue];
```

Or you could combine the two to override the default style element of the location, and then further override specific attributes of that element while inheriting the rest. For example, the following statement uses the Header style element for the data location, but also changes the foreground to be blue. The rest of the attributes are inherited from the Header style element.

```
proc print data=energy_report style(data)=header [foreground=blue];
```

Styles Specified in the VAR, ID, and SUM Statements

In the same way, styles specified in the VAR, ID, and SUM statements override or inherit from the PROC statement. When a new style element is specified for a location, it overrides the previously active one. If new attributes are specified, those attributes are overridden and the rest are inherited from the active style element.

In the following example, new font_style and background attributes are specified for the Header location in the PROC statement. This overrides those two attributes from the default style definition and inherits the rest. Then in the ID statement, the foreground attribute is changed to red and the background is changed to white. The result is that the ID columns inherit the italic font_style from the PROC statement, override the background to be white, and then override the foreground attribute to be red. Notice that the foreground specification is actually overriding the default style definition, since the attribute wasn't changed in the PROC statement. The other columns use the italic font with the yellow background as specified in the PROC statement.

```
ods html file='my_report';
title "Expenditures Summary";
proc print data=energy_report
  style (header) = [font_style=italic background=yellow];
  by region division type;
  id region division type /style (header) = [foreground=red
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

```

                                background=white];
sum expenditures;
format region regfmt. division divfmt. type usetype.
       expenditures dollar12.2;
run;
ods html close;

```

Expenditures Summary Combining Style Specifications				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 10: Overriding and Inheriting Styles

USING STYLES WITH FORMATS

Traffic lighting is a term used to describe a class of reports in which the appearance of a cell depends on the actual data value in it. This technique can be used to visually highlight fields that have values of importance to the reader. A good example of traffic lighting is a report that indicates when a value gets too high or too low. In PROC PRINT, traffic lighting can be accomplished by creating a format that associates style attributes to the data values and then using that format in your style specification.

Let's first look at how to create the format. Using our example of energy expenditures, let's say that expenditures greater than \$1000 need to be flagged for management to review. Expenditures less than \$500 should be flagged for commendation. And the rest of the values are in an expected range and therefore need no attention. We can create the following format to associate different background and foreground colors with each of these conditions:

```

proc format;
value backcolor   low - 499 = 'light blue'
                   500 - 1000 = 'white'
                   1001 - high = 'light red';
run;

proc format;
value forecolor   low - 499 = 'white'
                   500 - 1000 = 'black'

```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

```

                                1001 - high = 'yellow';
run;

```

Now we use these formats as part of our style specification in the VAR statement to associate the different colors with the value of the expenditures variable.

```

ods html file='my_report';
title "Expenditures Summary Using Traffic Lighting";
proc print data=energy_report;
  var state;
  var expenditures / style(data) = [background=backcolor.
                                   foreground=forecolor.
                                   font_weight=bold];

  by region division type;
  id region division type;
  sum expenditures;
  format region regfmt. division divfmt. type usetype.
         expenditures dollar12.2;
run;
ods html close;

```

Expenditures Summary Using Traffic Lighting				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 11: Using Styles with Formats

As you can see, the usage of styles can be as simple or as complex as you like. The best way to create the final report that you want is to explore and experiment with the various combinations of style definitions, elements, and attributes that are available.

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

CUSTOMIZING TITLES

There are several SAS statements and options that interact with PROC PRINT to enable you to easily customize titles on your report. The NOBYLINE system option used in conjunction with the #BYVAR and #BYVAL arguments in the TITLE statement can greatly enhance the readability of your report.

Specifying the NOBYLINE option suppresses the automatic printing of BY lines in the output. In our previous examples using the BYID layout for PROC PRINT, the NOBYLINE option was implicitly set for the user in order to create the special layout without BY lines. However, when NOBYLINE is explicitly set by the user, PROC PRINT inserts a page break between each BY group, along with suppressing the BY lines. This causes the text in the TITLE statements to appear at the top of each BY group, since that is the top of each page.

Repeating the same title before every BY group doesn't really add anything to the report. But adding the #BYVAR and/or #BYVAL arguments to the TITLE statement can create a unique title at the top of each BY group, inserting the BY variable name or the value of the BY variable as part of the title.

#BYVAR

The #BYVAR argument in the TITLE statement substitutes the BY variable name, or variable label if one was specified, into the title text and displays the name or label in the title. You specify which BY variable to use for the substitution by either using the syntax of #BYVAR*n*, where *n* represents the variable's order in the BY statement, or #BYVAR(*variable-name*).

#BYVAL

The #BYVAL argument substitutes the current data value of the specified BY variable into the title text and displays the value in the title. As with #BYVAR, you specify which BY variable to use for the substitution by either using the syntax of #BYVAL*n*, where *n* represents the variable's order in the BY statement, or #BYVAL(*variable-name*).

The following example uses the #BYVAR and #BYVAL arguments, showing each form of syntax, to customize our TITLE statements. Don't forget to include the NOBYLINE option. Remember that it is a global option, so it remains in effect until explicitly changed to BYLINE. To give the title a more formal look, I also added a LABEL statement to create a label for the region variable that capitalizes the first letter.

```
ods html file='my_report';
options nobyline;
proc print data=energy_report;
  title1 "Expenditures Summary by #byvar1";
  title2 'This table shows #byval3 expenditures';
  title3 'in the #byval(division) division.';
  label region ='Region';
  by region division type;
  id region division type;
  sum expenditures;
  format region regfmt. division divfmt. type usetype.
         expenditures dollar12.2;
run;
ods html close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary by Region
This table shows Residential Customers expenditures
in the Middle Atlantic division.

region	division	type	state	expenditures
Northeast	Middle Atlantic	Residential Customers	NY	\$8,786.00
			NJ	\$4,115.00
			PA	\$6,478.00
Northeast	Middle Atlantic	Residential Customers		\$19,379.00

Expenditures Summary by Region
This table shows Business Customers expenditures
in the Middle Atlantic division.

region	division	type	state	expenditures
Northeast	Middle Atlantic	Business Customers	NY	\$7,825.00
			NJ	\$3,558.00
			PA	\$3,695.00
Northeast	Middle Atlantic	Business Customers		\$15,078.00
Northeast	Middle Atlantic			\$34,457.00
Northeast				\$47,063.00

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary by Region				
This table shows Residential Customers expenditures in the New England division.				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00

Expenditures Summary by Region				
This table shows Business Customers expenditures in the New England division.				
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 12: Customizing Titles

THE DOCUMENT DESTINATION AND PROCEDURE

The DOCUMENT output destination creates a special type of SAS file that contains the results of your analyses in a generic form before it has been formatted for any specific output format. You can think of this file as the intermediate output created by the executing SAS procedure, before reaching its final output destination (for example, a PDF report). You can then use this intermediate file, called an ODS document, as input to the DOCUMENT procedure to view and manage the various pieces of your output, make changes to titles and footnotes, replay the document to create output in a specific format, and much more. New in SAS 9.3, PROC PRINT fully supports the DOCUMENT destination. In this paper, we touch on only a few of the most basic topics related to the DOCUMENT destination. For more detailed information, please see the *SAS Output Delivery System: User's Guide*.

CREATING AN ODS DOCUMENT

Just like the other output destinations, the DOCUMENT destination is specified in the ODS statement. The difference is that the file created is a member in a SAS library instead of being a final result file. The following example creates an ODS document in the TEST library with the name my_document. Remember that the output produced is an ODS document that contains all the information needed to create the final report, but not the final report itself.

```
ods document name=test.my_document(write);
title "Expenditures Summary";
proc print data=energy_report;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

```

    by region division type;
    id region division type;
    sum expenditures;
    format region regfmt. division divfmt. type usetype.
           expenditures dollar12.2;
run;
ods document close;

```

MANAGING AN ODS DOCUMENT

Using the DOCUMENT procedure, you can now list the contents of the ODS document, as well as rearrange, duplicate, or remove output from it. The following code creates a listing of the contents of the document. The report lists each output object within the document and its internal name, which is used in other statements to identify each output piece. Note the use of the QUIT statement to terminate the DOCUMENT procedure, since DOCUMENT allows multiple RUN statements before terminating the procedure.

```

proc document name=test.my_document;
  list / levels = all;
run;
quit;

```

Listing of: \Test.My_document\		
Order by: Insertion		
Number of levels: All		
Obs	Path	Type
1	\Print#1	Dir
2	\Print#1\ByGroup1#1	Dir
3	\Print#1\ByGroup1#1\Print#1	Report
4	\Print#1\ByGroup2#1	Dir
5	\Print#1\ByGroup2#1\Print#1	Report
6	\Print#1\ByGroup3#1	Dir
7	\Print#1\ByGroup3#1\Print#1	Report
8	\Print#1\ByGroup4#1	Dir
9	\Print#1\ByGroup4#1\Print#1	Report
10	\Print#1\ByGroup5#1	Dir
11	\Print#1\ByGroup5#1\Print#1	Report
12	\Print#1\ByGroup6#1	Dir
13	\Print#1\ByGroup6#1\Print#1	Report
14	\Print#1\ByGroup7#1	Dir
15	\Print#1\ByGroup7#1\Print#1	Report
16	\Print#1\ByGroup8#1	Dir
17	\Print#1\ByGroup8#1\Print#1	Report

Output 13: Contents of ODS Document

Using other PROC DOCUMENT statements, you can now change various aspects of the output, such as titles, footnotes, and page breaks, by referencing the internal name.

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

REPLAYING AN ODS DOCUMENT

One of the most important uses of PROC DOCUMENT is to replay a document. When the document was created, all the information was stored in a generic way. Now, you can create a final report in any output format you choose. For example, we might now wish to create the HTML version of this report using the following code. The HTML output produced from PROC DOCUMENT replay looks exactly the same as if we had executed the PROC PRINT step directly to HTML.

```
ods html file = 'my_report.html';
proc document name=test.my_document;
  replay;
run;
quit;
ods html close;
```

Expenditures Summary				
region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00
region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 14: HTML Replay of ODS Document

MODIFYING AN ODS DOCUMENT

After the document is created, you can modify many parts of the report such as titles, footnotes, and labels, as well as rearrange the order of the output. Using the internal names we listed previously, the following simple example shows how to add a subtitle to the report using the OBSTITLE statement and then produce the report in PDF. The DIR statement sets a current path so that it doesn't have to be explicitly specified in subsequent statements.

```
ods pdf file = 'my_report.pdf';
proc document name=test.my_document;
  dir \Print#1;
  obstitle1 ByGroup1#1\Print#1 'Expenditures for July, 2010';
  obstitle2 ByGroup1#1\Print#1 'Created for SAS Institute';
  /* obstitle statements for other BY groups would go here */
  replay;
run;
quit;
ods pdf close;
```

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Expenditures Summary

Expenditures for July, 2010
Created for SAS Institute

region	division	type	state	expenditures
Northeast	New England	Residential Customers	ME	\$708.00
			NH	\$597.00
			VT	\$353.00
			MA	\$3,264.00
			RI	\$531.00
			CT	\$2,024.00
Northeast	New England	Residential Customers		\$7,477.00

region	division	type	state	expenditures
Northeast	New England	Business Customers	ME	\$379.00
			NH	\$301.00
			VT	\$188.00
			MA	\$2,498.00
			RI	\$358.00
			CT	\$1,405.00
Northeast	New England	Business Customers		\$5,129.00
Northeast	New England			\$12,606.00

Output 15: Modifying an ODS Document

There are many other features available in PROC DOCUMENT that enable you to customize your output. There is also a Documents window to interactively manipulate the document and customize the output, which is sometimes easier than using PROC DOCUMENT in a batch environment. The best news is that with SAS 9.3, PROC PRINT fully supports all of the features of the DOCUMENT destination and procedure.

CONCLUSION

We've explored PROC PRINT from its simplest form to more complex programs to create and customize reports. Hopefully, you found some new tricks and techniques that will prove useful in your own environment. If your past use of PROC PRINT has been only to create quick listings of your data, I hope you'll consider broadening your view of PROC PRINT to include using other output destinations, styles, and the many options available to create complex reports easily.

REFERENCES

- Haworth, Lauren E. 2001. *Output Delivery System: The Basics*. Cary, NC: SAS Institute Inc.
- Haworth, Lauren E., Cynthia L. Zender, and Michele M. Burlew. 2009. *Output Delivery System: The Basics and Beyond*. Cary, NC: SAS Institute Inc.
- Helwig, Jane T. ed. 1985. *SAS[®] Introductory Guide*. 3rd ed. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. 2009. *Base SAS[®] 9.2 Procedures Guide*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. 2009. *SAS[®] 9.2 Output Delivery System: User's Guide*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

PROC PRINT and ODS: Teaching an Old PROC New Tricks, continued

Darylene Hecht
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-7736
Fax: (919) 677-4444
E-mail: darylene.hecht@sas.com
Web: www.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.