Paper SS03

Using ODS RTF with Style!

Robert Walls, PPD, Bellshill, UK

ABSTRACT

In the last few years ODS functionality has virtually exploded. There is now ODS functionality which will allow a user to output to almost every possible desired destination, from PDF to XML. One of the most well used and documented of these destinations is the RTF destination, and because of this the output created by ODS RTF is probably the most malleable of all the destinations. The aim of this paper is to explain some of the techniques which can be used to manipulate the appearance of RTF output. Some of the techniques which will be covered include, in-line formatting and the escape character, inserting raw RTF code, protecting special characters, traffic lighting and using the style statement within procedures. Each of these techniques will allow the user to change the structure, appearance and format of the outputted data with ease. ODS is a very powerful programming tool, and thanks to the continual expansion of its functionalities and the introduction of capabilities akin to those above it is revolutionizing the creation of output by SAS® programmers.

INTRODUCTION

When using ODS, PROC TEMPLATE is used to create general styles which affect the layout and appearance of output, and though these Templates are able to style the individual columns in a table, doing so can be a laborious process with results which can only be applied to tables with the same structure.

As a solution SAS® incorporated the ability to alter/create styles to be expressed within the output which can be easily specified on a program to program basis. These styles can be applied using in-line formatting technique or by using procedural style statements.

Using ODS to output to RTF has become increasingly common practice within the Pharmaceutical industry, and as it is becoming more and more widespread, the demand for more and more functionality had to be addressed. This presentation is designed to take you from a starting point where you are able to create RTF output using ODS and PROC TEMPLATE, to being able to create and manipulate output in a number of useful and flexible ways. Formatting ODS output with destination code or ODS options, and the use of procedural style options will be covered within this paper, as these are the two most powerful ways of manipulating output without altering your PROC TEMPLATE styles.

Though all of the foreign, non-SAS code, used in this presentation is RTF code, many of the principals covered will be generic to many of the ODS destinations. This will become increasingly important as the FDA and other regulatory authorities are looking towards using PDF and XML output for future submissions.

DUMMY DATA

The dummy data which is going to be used in all of the examples for this paper are data sets which can be found in the SASHELP library.

FORMATTING WITH STYLE

FORMATTING TITLES AND FOOTNOTES

For people who are familiar with the functionality which is built into the titles and footnotes created while using the SAS® GRAPH procedures, they will already be familiar with the basic syntax and some of the options employed by ODS in the same way. These are the most basic way of manipulating the appearance of ODS generated output, and simply involves the addition of options to the title or footnote statement.

Formatting Titles and Footnotes[1]	
BOLD	**Bolds the title or footnote**
COLOR= (C=)	**Supplies the colour of the title or footnote**
BCOLOR= (B=)	**Supplies the background colour of the title or footnote**
FONT= (F=)	**Supplies the font for the title or footnote**
HEIGHT= (H=)	**Supplies the point size for the title and footnote**
JUSTIFY= (J=)	**Left-justifies, right-justifies, or centres the title or footnote**
LINK= (L=)	**Specifies the hyperlink to use for the title or footnote**

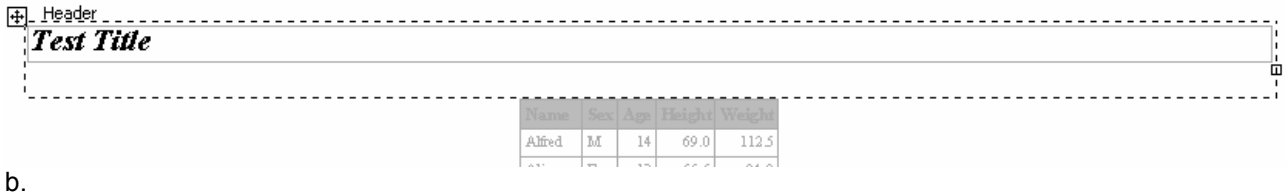
Table 1: This is a list of the options (and their abbreviations) available to titles and footnotes when being output using ODS.

These options are positioned before the text of the footnote, or even in between pieces of quoted text to effect separate pieces of text in the same title or footnote in different ways, as shown.

a.
```
**  Inserts a Bold Title, 20pts in size and left justified  **;
title j=l bold h=20pt 'Test Title';
```

b.
```
**  Inserts a Footnote in the Courier New font, right justified**;
**  and coloured red and blue  **;
footnote j=r f="Courier New" c=red 'Test ' color = blue 'Footnote';
```

a.



b.

Fig 1: Screen dumps of the titles and footnotes created using basic ODS formatting techniques.

IN-LINE FORMATTING

In order to perform In-line formatting operations, the user must first declare an escape character. This is a character (preferable not one in regular use) which tells the SAS® processing system that the following text in the quoted string is not to be output as text, but is used to format the text following the closing of the curly parenthesis. The escape character is declared in an ODS option as shown:

```
ods escapechar = "^";
```

Directly following the escape character within the quoted text of a title or footnote, the following syntax can be used, dependant on the intended purpose.

In-line Formatting Syntax[1]	
^S={*style-attributes*} *	This will modify the current paragraph or cell (very similar to the 'style={}' syntax covered later). The style attributes are the same as those in the template, so foreground, background, font_size…etc can all be changed using this option. (This can generally be used in conjunction with any of the PROC TEMPLATE attributes used to modify the output structures. A list of the more useful changeable attributes can be found in Table 4, on Page 9 of this paper)
^S={} *	A specialised case of the above; reverts the style to the style of the paragraph.
^{*function arguments*}	An open-ended mechanism for providing functions to allow a portable way to get specialised formatting effects to work across destinations. Some included functions are shown below.
- {super *text*}	Displays the word *text* as a superscript
- {sub *text*}	Displays the word *text* as a subscript
- {dagger}	Insert the "dagger" character (†)
* Note: These statements must use a capital S	

Table 2: This table shows the in-line formatting syntax, which doesn't have to be restricted to titles and footnotes, but can be applied to header text and also within the body of the data as well.

The example below shows the use of two different types of in-line formatting syntax from the above table, and the output which they generate.

```
ods escapechar = "^";

**  create a superscripted 2 at the end of the title  **;
title 'Creates a superscripted two ^{super 2} ';

**  this will create the title text in bright pink  **;
title2 j=l '^S={foreground = cxFF00CC} a test';
```

Name	Sex	Age	Height	Weight
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0
Barbara	F	13	65.3	98.0

Fig 2: Here you can see the titles created by the above code. The first title contains the superscript 2; the second has bright pink coloured text.

IN-LINE FORMATTING WITHIN DATA

In-line formatting can also be used, not only in the output of titles and footnotes, but also in the output data and in the generation of headers and labels from reporting procedures. This In-line formatting method shares all of the functionality previously described. The following code will create superscripts within the report output and the report headers.

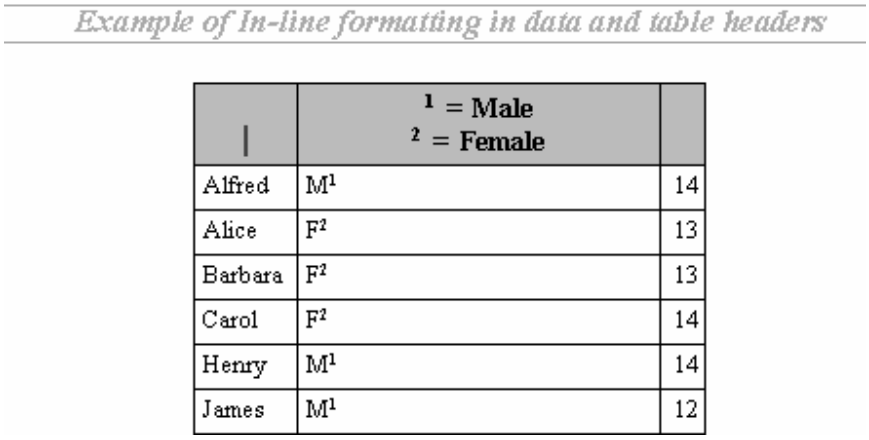
```
data work.class;
    set sashelp.class;
    if sex = 'M' then do;
        sex2 = sex || "^{super 1}";
    end;
    if sex = 'F' then do;
        sex2 = sex || "^{super 2}";
    end;
run;

ods escapechar = "^";
ods rtf file = "C:\temp.rtf";

    Title "Example of In-line formatting in data and table headers";

    proc report data = work.class nowd headline headskip split = "!";
         columns name ("^{super 1} = Male!^{super 2} = Female" sex2)
                  age;
         define name /     display "";
         define sex2 /     display "";
         define age  /     display "";
    run;

ods rtf close;
```



Example of In-line formatting in data and table headers

	[1] = Male [2] = Female	
Alfred	M[1]	14
Alice	F[2]	13
Barbara	F[2]	13
Carol	F[2]	14
Henry	M[1]	14
James	M[1]	12

Fig 3: The output from the above code showing that In-line formatting is not solely restricted to titles and footnotes, but can also be utilised within the output data, and within output header information.

INSERTING RAW DESTINATION CODE

The syntax for inserting raw destination code is similar to that for altering the styles with In-line formatting except that a capital R is used followed by the quoted raw destination text instead of the S = (it is worth noting that using the S = technique of in-line formatting can also insert raw destination code using the PRETEXT = option within the parenthesis. This will be touched on later). For example to insert the raw destination code for the automatic generation of page numbering of RTF output documents, you can use the following code:

```
ods escapechar = "^";

title  "^R'Page \field {\*\fldinst PAGE \\*MERGEFORMAT}}
         {of \field {\*\fldinst NUMPAGES \\*MERGEFORMAT} ' Text";
```

This is better programming practice than just inserting the RTF code as text, which sometimes works dependant on whether the user has declared to protect special characters or not (this will be covered in the next section). When inserting as raw destination code the RTF code will always be interpreted correctly.

When creating output, the user may not always wish to just create output to an RTF destination, but as this is RTF specific code it will not have the same effect when sent to any other destination. For example, when the above title statement is simultaneously being sent to the PDF destination, the whole string of text within the double quotes will resolve in PDF as text (with the exception of the ^R). It is, however, possible to be selective about the desired destination that the raw code should be written to. To do this the ^R should be directly followed by a forward slash (/), then the name of the codes specific destination, and then the single quoted raw destination code followed by the text.

```
ods escapechar = "^";

title "^R/RTF'Page \field {\*\fldinst PAGE \\*MERGEFORMAT}}
         {of\field {\*\fldinst NUMPAGES \\*MERGEFORMAT} ' Text";
```

This will suppress ODS from trying to write out the specified code to anywhere but the RTF destination. Alternatively you can specify ^R/PDF, ^R/HTML, etc.

PROTECTING SPECIAL CHARACTERS

Very often when using In-line formatting, you will encounter problems getting the desired effects. This is generally due to the fact that a lot of the characters being used are automatically being protected by SAS®, i.e. ODS tries to output them as text. A good example of this is the backslash (\) which we use when displaying the pathnames of programs at the bottom of an output, but it is also a very important special character in RTF code. When it is not desirable for the special character (\) to be preserved as text then the user can tell ODS not to protect the special characters and by doing this, they are made available to the destination to interpret as destination code. There is no full list of the characters that this applies to. What is usually required is a basic knowledge of the functional characters employed by the destination language.

This can be achieved through the use of the **protectspecialchars** PROC TEMPLATE output modification option. This is set to be 'on' as default, but to allow destinations to interpret code with special characters, this should be turned to 'off', which can either be done in the style template, or by modifying the style with In-line formatting techniques.

Below some in-line formatted titles are created. The first PROC PRINT shows the titles where the **protectspecialchars** is set to off. This means that the destination will try to interpret any word preceded by a backslash as RTF code, which will allow the functional code in the first title to be expressed, but will stop the pathname from being output properly. In the second PROC PRINT, the special characters are protected, so though the pathname will be displayed correctly, the RTF control words will have no effect and will in fact be output as text within the title.

```
ods escapechar = '^';
ods rtf body = 'C:\PSC_temp1.rtf' style = styles.rtf;

proc print data=sashelp.class;
   title '^S={protectspecialchars = off}\ul\ql this is the first
         \line this is the second';
   title2 '^S={protectspecialchars = off}C:\Test\program.sas';
run;

proc print data=sashelp.class;
   title '^S={protectspecialchars = on}\ul\ql this is the first \line
               this is the second';
   title2 '^S={protectspecialchars = on}C:\Test\program.sas';
run;

ods rtf close;
```

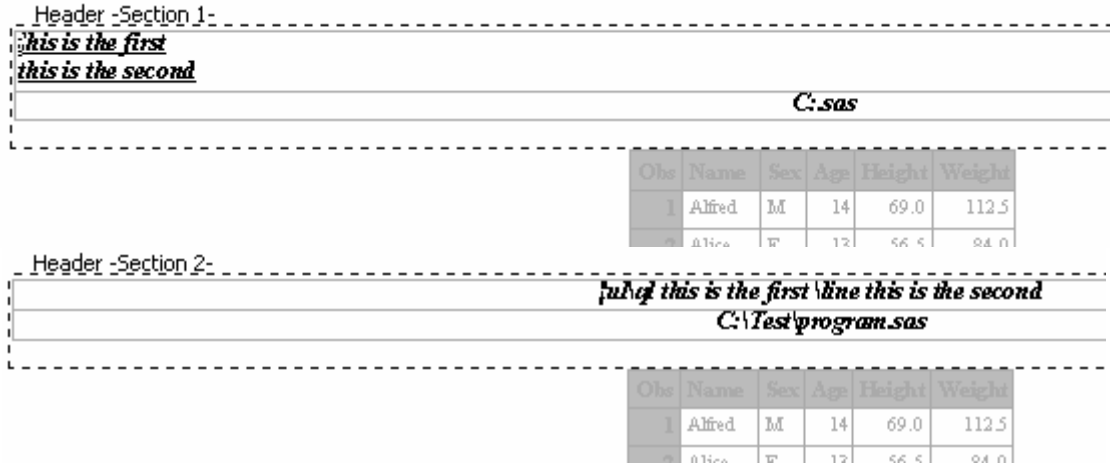


Fig 4: These screen shots show how the special RTF control character (\) is resolved or not resolved, dependant on the PROTECTSPECIALCHARS option.

From the output created using the previous code we can see that in the first PROC PRINT output title one has resolved correctly, as it has interpreted the RTF code which left aligns (\ql), underlines (\ul) and splits the text over two lines(\line). Title two however, has tried to interpret the program path name as RTF control words, due to the presence of the backslash special character, which we do not want. In the second PROC PRINT output, we can see that, although the program name remains correct in title two, the RTF control code in title one is not resolved. To get the desired results a blend of the two should be used instead, as shown below.

```
ods escapechar = '^';
ods rtf body = 'C:\PSC_temp2.rtf' style = styles.rtf;

proc print data=sashelp.class;
   title '^S={protectspecialchars = off}\ul\ql this is the first \line
         this is the second';
   title2 '^S={protectspecialchars = on}C:\Test\program.sas';
run;

ods rtf close;
```

Fig 5: This figure shows the correct way of protecting special characters, so that the two titles will resolve correctly.

Note: The PROTECTSPECIALCHARS options could have been reset within the PROC TEMPLATE style definition, but this has the effect of controlling all of the titles output. In the examples above I have shown how to control titles, or footnotes, uniquely.

SOME USEFUL RTF CONTROL WORDS

Below is a list of just a few useful RTF control words. When inserting RTF code, there should always be a space at the end of the code inserted for the code to resolve. It is however possible to put more than one piece of code together, but a space still needs to be kept, but only at the end of the RTF string (e.g. '**\i\dul\fs20** ' will give italicised, double underlined text which is font size 10).

Style	RTF Control Word[2]	Example Code
ITALICIZE	**\i**	title '**\i** italicized title';
UNDERLINE	**\ul**	title '**\ul** underline title';
DOUBLE UNDERLINE	**\dul**	title '**\dul** title';
NEW LINE	**\line**	title 'this is the first **\line** this is the second ';
BULLET	**\bullet**	title '**\bullet** bullet preceding title';
EMBOSS	**\embo**	title '**\embo** embossed title';
ENGRAVE	**\impr**	title '**\impr** engraved title';
SUBSCRIPT	**\sub**	title 'This is a subscript T**\sub** 1';
SUPERSCRIPT	**\super**	title 'This is a subscript T**\super** 2';
OUTLINE	**\outl**	title '**\outl** This is outlined';
SHADOW	**\shad**	title '**\shad** This is shadowed';
STRIKE	**\strike**	title '**\strike** This is striked';
DOUBLE STRIKE	**\strikedl**	title '**\strikedl** This is double striked';
DOTTED UNDERLINE	**\uld**	title '**\uld** dotted underline';
WAVE UNDERLINE	**\ulw**	title '**\ulw** wave underline';
THICK UNDERLINE	**\ulth**	title '**\ulth** thick underline';
FOREGROUND COLOR	**\cfN**	title '**\cf2** foreground color'; (where N = a unique RTF colour number)
FONT SIZE IN HALF POINTS	**\fsN**	title '**\fs40** fonts increased'; (where N = the desired font size times by 2, i.e. to get font size ten then N must equal 20, etc)
HIGHLIGHT	**\highlightN**	title '**\highlight2** ';
BOLD	**\b**	title '**\b** bold title';
LEFT ALIGNED	**\ql**	title '**\ql** left aligned.
RIGHT ALIGNED	**\qr**	title '**\qr** right aligned.
CENTERED	**\qc**	title '**\qc** left aligned.
DECIMAL ALIGNMENT OF NUMBERS	**\tqdec\txN**	Aligns the numeric values within a cell N number of twips (~1/20th of a character width) from the left side of the cell. (Note: the cell contents must be left aligned for this to work)

Table 3: A short list of useful RTF control words.

USING PROCEDURES WITH STYLE

As well as the formatting techniques described previously, ODS functionality has now been incorporated into the reporting procedures. This functionality allows the user to modify the template styles for specific columns, rows, headers, etc. within the output. The procedures covered within this paper are Report, Tabulate and Print.

The way that this has been done is to add style options to specific procedure statements, the affected area of the output depends on the style area option used, and also the procedure statement it is added to.

The general syntax is:

```
**  This will, generally, effect all of the procedural output area  **;
proc statement ...... style = {background = blue};

**  This will effect all of the header cells of the procedural output
    area  **;
proc statement ...... style(header) = {background = blue};

**  This will effect all of the data cells of the procedural output
    area for variable1    **;
proc statement ...........;
    var variable1    /style(column) = {foreground = green};
run;
```

Note: The style area sub-option is specified within the parenthesis after the style statement and before the equals sign.

STYLE MANIPULATING ATTRIBUTE OPTIONS

Below is a list of some of the style elements which can be altered within a procedure using the style options within procedural the statements.

Use this attribute[1]	To do this ...
ASIS=	Specify how to handle leading spaces, trailing spaces, and line breaks.
CELLHEIGHT=	Specify the height of the cell.
CELLWIDTH=	Specify the width of the cell.
NOBREAKSPACE=	Specify how to handle space characters.
URL=	Specify a URL to link to.
VJUST=	Specify vertical justification.
BACKGROUND=	Specify the colour of the background, which is primarily the colour of the page.
CELLPADDING=	Specify the amount of white space on each of the four sides of the text in a cell.
CELLSPACING=	Specify the thickness of the spacing between cells.
FONT=	Specify a font definition.
FONT_FACE=	Specify the font face to use.
FONT_SIZE=	Specify the size of the font.
FONT_STYLE=	Specify the style of the font.
FONT_WEIGHT=	Specify the font weight.
FONT_WIDTH=	Specify the font width compared to the width of the usual design.
FOREGROUND=	Specify the colour of the foreground, which is primarily the colour of the text.
JUST=	Specify justification.
POSTIMAGE=	Specify an image to place after the table or cell.
POSTTEXT=	Specify text to place after the table or cell (can be used to insert RTF code).
PREIMAGE=	Specify an image to place before the table or cell.
PRETEXT=	Specify text to place before the table or cell (can be used to insert RTF code).
PROTECTSPECIAL CHARS=	Determine how possible destination code (i.e. RTF, HTML, etc) special characters, such as less-than signs (<), greater-than signs (>), backslash (\), ampersands (&), etc. are interpreted.

Table 4: A short list of some of the most useful style sub-options. A full list can be found in the SAS® inline documentation.

REPORTING WITH STYLE

The PROC REPORT style option is available to a number of the PROC REPORT statements.

Style Option Enabled Statements		
Statement	Affected Area	Default Location Value
REPORT Statement	Affects the whole report	REPORT
BREAK/RBREAK Statement	Affects the summary variables	SUMMARY
CALL DEFINE Statement	Affects any output referenced by the call define	CALLDEF
COMPUTE Statement	Affects computed variables	LINES
DEFINE Statement	Affects the column and header of defined variable	COLUMN and HEADER

Table 5: This table shows the PROC REPORT statements which are compatible with the style options, and also the default location values which can be altered as sub-options on the style statement to declare which part of the output you wish the style option to effect.

The syntax used to declare these style options is the same for most of the output procedures, which can take one of two major forms:

```
STYLE= {Style Attributes};

STYLE(Location Value) = {Style Attributes};
```

In the first instance the style option will take the default location value, shown in table 5, when determining the area it will affect the appearance of. The second example shows how to change the location value so that it will only effect a certain area of the output. The values that the location value can take are shown in Table 6 below.

Location Value	Part of Report Affected
CALLDEF	Cells identified by a CALL DEFINE statement
COLUMN	Column cells
HEADER|HDR	Column headers
LINES	Lines generated by LINE statements
REPORT	Report as a whole
SUMMARY	Summary lines

Table 6: This table shows the various values that the location value sub-option can take in the style statement. The statement in which the style option is used must first be affecting the area you wish to specify; otherwise the style option will have no effect.

Note: Use of the style option with all but the procedure statement requires the use of the forward slash (/) divider. This is true for both Print and Tabulate as well. Also a style option specified on the actual procedure statement will be over-ruled by any style option stated on any of the other statements affecting the same sub-option, but only for the area affected by the second statement.

```
ods rtf file = "C:\Reporting_with_style.rtf";

proc report data = sashelp.class nowd headline headskip
            style(header) = {font_face   = "Times Roman"
                             font_weight = light
                             font_style  = italic};
      columns name sex age height weight;

      define name              /     display
                                     style = {font_weight = bold
                                              font_style  = italic
                                              foreground  = blue};
      define sex         /     display;
      define age         /     display;
      define height      /     display
                               style(column) = {font_weight = bold
                                                font_style  = italic
                                                foreground  = orange};
      define weight      /     display
                               style(header) = {font_weight = bold
                                                foreground  = purple};
      compute before _page_    /     left
            style = {preimage    = 'C:\The Joker.jpg'
                     cellwidth   = 370
                     font_weight = bold
                     font_size   = 5
                     foreground  = cx993300};
            line "Class data";
      endcomp;
run;

ods rtf close;
```

In the style option on the PROC REPORT statement the sub-option has been applied to tell the option to only affect the header information of the outputted report. The attributes tell it to change the header information to be displayed in Times Roman, italic font. The second style option on the define statement for the name variable, sets the font to a blue bold italic for both the header and the column information. Looking at the output below you can see that the font face of the header has remained the same, even though we are changing options in an area which clashes with the PROC REPORT statement style option, because we are not changing the option that affects the font face. The style option on the define statement of the height variable only affects the column information, and the style options for the weight variable will only affect the header part of the output. This will again only over-rule any options from the PROC REPORT statement which they have in common, such as the font weight.

The final style options are set within the compute block of the report, and due to the statement that these options are stated in, will only affect the compute block output area, even though we are not stating a location value. This is inserting an image at the top of each page created by the report, and sets the line of text 'Class data' to be brown, bold and with an increased font size.

The results of all of these style options can be seen in Figure 6.

The hexadecimal colour value used in the foreground element of the style option used within the compute block (cx993300) is simply another way of expressing colours. Those used to the SAS GRAPH procedures will be familiar with this, or else you can use a colour picker website[4] which shows the 256 HTML safe colours and their hexadecimal values which will work on all browsers on the internet, plus every colour in between. These colours will generally work to all ODS destinations.

Class data

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

Fig 6: The output created by manipulating the PROC REPORT style options as shown in the previous code.

PRINTING WITH STYLE

The syntax used within a PROC PRINT is almost identical to that for the PROC REPORT, with the exception that there are different areas of the output affected by different statements and sub-options.

Style Option Enabled Statements	
PROC PRINT Statement	Affects the whole table output
ID Statement	Affects only the ID column
SUM Statement	Affects only the Summary row where data is contained
VAR Statement	Affects all of the variable columns

Table 7: This table shows the PROC PRINT statements which are compatible with the style options.

Location Value	Part of Report Affected
DATA	All rows containing data
HEADER (HDR)	Column headers
OBS	The observation column created when the noobs option is NOT used
OBSHEADER	The header for the observation columns
TABLE	The whole table output
TOTAL	The summary rows

Table 8: This table shows the various values that the location value sub-option can take in the style statements.

TABULATING WITH STYLE

The method to affect all of the output areas in PROC TABULATE output is slightly different to those for the other two procedures described, in that, not all of the statements which control output areas can have an associated style option. On occasions where keywords or the keylabel statement is used (e.g. ALL, n, pctn) the KEYWORD statement will have to be added to the PROC TABULATE in order to affect the label area for the keyword used. Individual class header levels create by the tabulation will also need a further statement added to affect the label regions of the classes. For these the CLASSLEV statement should be used.

```
ods rtf file = "C:\table2.rtf";

proc tabulate data=sashelp.company
              style = {foreground = yellow background = red};
      class level2      / style = {foreground = green background = blue};
      table level2;
      keylabel n = 'Total count for city';
      classlev level2   / style = {foreground = blue background = green};
      keyword n         / style = {foreground = red background = yellow};
run;

ods rtf close;
```

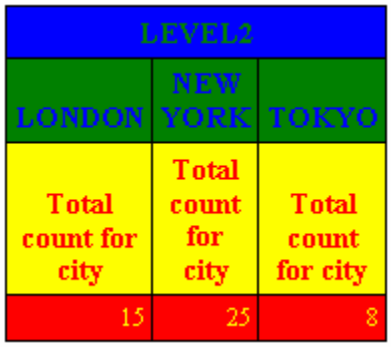


Fig 7: The output created using the style options in the PROC TABULATE above.

In the above statement you can see that the style options set on the PROC TABULATE statement affect only the data area, as all other areas have been re-specified, though the central two rows would not have had the style applied for the reasons stated above. The style options on the CLASS statement colour the top level header as green on blue. The style options from the CLASSLEV statement colour the classification levels of the table blue on green, while the KEYWORD statement's style options colour the KEYWORD labels as red on a yellow background.

TRAFFIC LIGHTING

Traffic lighting is the term given to a technique, whereby the appearance of a cell is dictated by the value within that specific cell. There are two ways to do this. The first is to use formats, and can be accessed using any of the output procedures, and the second is to use the compute and call define statements in PROC REPORT.

The first method, using formats, is relatively straight forward. Within the style option, the attribute that you wish to change depending on the data values is set to equal a format. This format looks like a standard format, with the exception that the outputted format values are the values that you wish the attribute to take for that instance of the data value.

```
proc format ;
      value bground
                  low-14 = 'red'
                  other = 'green';
      value fsize
                  low-65 = '8pt'
                  other = '20pt';
      value fweight
                  low-100 = 'medium'
                  other = 'bold';
      value $url
                  'M' = 'C:\MALE.doc'
                  'F' = 'C:\FEMALE.doc';
run;

ods rtf file = "C:\traffic lighting.rtf";

proc report data = sashelp.class nowd headline headskip;
      columns name sex age height weight;
      define name       /     group;
      define sex        /     display
                              style = {url = $url.};
      define age        /     display
                              style = {background = bground.};
      define height     /     display
                              style = {font_size = fsize.};
      define weight     /     display
                              style = {font_weight = fweight.};
      run;

ods rtf close;
```

Here you can see that, for the sex variable, we are setting up a hyperlink to different documents depending on whether the person is male or female. The background is set to change either red or green, dependant on the age, the font size changes dependant on the height, and the font weight is changed depending on the weight of an individual.

The expressions of these style formats can be seen in Figure 8.

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**

Fig 8: Formatted traffic lighting output.

The second type of traffic lighting is achieved within a compute block and a call define within a PROC REPORT. In the example below, the appearance of the variable age is changed dependant on whether the person is over fourteen or not. The first call define statement is stating that if a person is fourteen or under then the background for that column (and therefore, only that cell) is set to red, whilst for those over fourteen the background of the entire row is set to green.

```
ods rtf file = "C:\traffic lighting1.rtf";

proc report data = sashelp.class nowd headline headskip;
    columns name sex age height weight;
    define name           /     group;
    define sex            /     display;
    define age            /     display;
    define height   /     display;
    define weight   /     display;
    compute age;
        if _C3_ <= 14 then
            call define(_col_, "style", "style = {background = cxFF0000}");
        else if _C3_ ^<= 14 then
            call define(_row_, "style", "style = {background = cx00FF00}");
    endcomp;
run;

ods rtf 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

Fig 9: Traffic lighting[3] output using a call define statement in the compute block of a PROC REPORT.

CONCLUSION

ODS RTF and PROC TEMPLATE on their own are very powerful tools which can be used to great effect to get a desired appearance for an output. They can only take you part of the way there. In order to be able to produce more complicated RTF output, eventually you are going to have to use some of the techniques described in this paper. Using RTF code to manipulate your output can also, not only be used to make your output look great, but it can be used to save programming time. For example, when aligning decimals, what could have potentially taken several data and proc steps, can now be done just by inserting one piece of RTF code.

The abilities of ODS RTF are only constrained by the abilities of what can be done in an RTF document. This means that, if you know the right way to do it, your output can be made to look any way that the end user may desire.

REFERENCES

1. SAS® ONLINE DOC - http://v8doc.sas.com/sashtml/

2. MSDN Library - http://msdn.microsoft.com/library/default.asp

3. Haworth, Lauren E., The Output Delivery System: The Basics, Cary, NC: SAS Institute Inc., 2001.

4. http://www.pagetutor.com/pagetutor/makapage/picker/

CONTACT INFORMATION

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

Robert Walls
PPD
Avondale House
Phoenix Crescent
Strathclyde Business Park
Bellshill
North Lanarkshire
ML4 3NJ
+44 (0) 1698 572 632
robert.walls@eurpoe.ppdi.com
www.ppdi.com