Paper 163-26

Now There Is an Easy Way to Get to Word, Just Use PROC TEMPLATE, PROC REPORT, and ODS RTF

Bob Hull, Synteract, Inc, Encinitas, CA

ABSTRACT

Getting SAS® System output into Word® documents dynamically and easily can be a great benefit to both you and your company. A great deal of effort has gone into getting SAS output into Word documents. This is because SAS output is not in a format that can be used in reports. People prefer to be able to cut and paste into their reports. This tends to be much easier to do from Word. However getting files into Word, or another word processor, has not been easy. Anyone with a system already in place to create Word output might be hesitant to switch over immediately. However, the speed at which rtf files can be created will sway many people to try this wonderful new technique. This paper is designed to get you started using PROC TEMPLATE, PROC REPORT and ODS RTF to deliver quality RTF output quickly.

INTRODUCTION

Understanding how PROC TEMPLATE, PROC REPORT and ODS RTF are related is very important. ODS RTF statements can be used in a program to deliver RTF documents to a file directly. It is not necessary to learn PROC TEMPLATE or PROC REPORT to do this. If you surround a PROC PRINT with ODS RTF statements (syntax in next paragraph) then the SAS System will look up the default template that is already in place and create the RTF document based on that. The advantage here is that it is very quick to get RTF documents that way and all of your documents will be working off of the same template to create a uniform look. The disadvantage is that you would have very little control over what that uniform look is. PROC TEMPLATE is a procedure that allows you to take the default template and make minor or major modifications to it. Your ODS RTF statements can then reference the template you have created. PROC REPORT can make further modifications to the style of the output on a individual cell basis if desired.

DEFAULT RTF

The SAS System has created a few default templates that you could choose from. If you are just looking for something quick you could experiment with the defaults SAS offers to find the one you like best. While in the results window, click on VIEW, TEMPLATES. Then go to the directory Sashelp and click on the styles folder. That is a list of all available default styles (You can double click on one of the styles to see the code beneath it). When a new SAS session is opened, all of these styles are available to the user. No library references or ODS statements are needed to access these styles. You can get RTF output with the following code, output in Figure 1:

```
ods rtf file="c:/firstone.rtf" style=minimal;
proc print data=sashelp.class (obs=2);
run;
ods rtf close;
```

Figure 1.

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0

The styles minimal and printer are my favorite default styles.

ODS STATEMENTS

The code above would actually produce two outputs. The RTF file that would be written to your C: drive, and additional output to the output window. Now that SAS has created a variety of places the output could be routed to, you need to know how to change where your output is going. This is all done through the ODS statements. SAS has allowed users to function just as they did in previous versions of SAS. When a SAS session is opened, any output will go to the output window unless specified otherwise. Also, it will continue to go to the output window regardless of other active destinations unless SAS is told to stop doing so. The following statement will tell SAS to stop sending output to the output window:

```
ODS listing close;
```

To route the output back to the output window again, use the following statement;

```
ODS listing;
```

All of the other output destinations follow the same basic pattern.

ODS STATEMENT OPTIONS

You can have multiple output destinations open at the same time. The RTF and HTML (and others) ODS statements have various options associated with it. The focus of this paper is RTF documents and those are the statements that will be emphasized. In the example used, you have already been exposed to the FILE= and STYLE= option. The FILE= option specifies the name of the file being created and can contain macro variables for names if desired. The STYLE= option identifies the style that the output will follow. The style can affect column headers being bold, the size of titles, the background colors, whether or not grid lines show up and so on. Whether you are using your own style or a SAS default, the STYLE= option will always will always refer back to the current ODS PATH that is active. As I have not shown how to change this yet, only the SAS defaults would be available at this point. However, depending on how your template library is set up it could look like any of the following:

```
Style=printer
Style=bold.emphasis
Style=clientx.projectx.styname
```

Pay particular attention to the fact that the word "bold" is NOT a library name! These are directories within the storage area for templates that the user would have created. "Printer" is a SAS style that already exists, the other two are fictional.

YOUR OWN TEMPLATE

If all you desire to do is create basic RTF files from your SAS output, you should be applying the information already given. Adding your own touches to the RTF files you create and being able to modify them to meet the needs of others will require additional techniques. That is where PROC TEMPLATE comes in. PROC TEMPLATE is what you will use to create new styles. You don't have to create a style from top to bottom, you can create a minor adjustment to an existing SAS style if that is all you wanted to do.

STARTING A TEMPLATE

When you are creating your own template, I recommend that you start one in your own directory structure separate from the SAS defaults. When you first start this way you may find that you have not included all necessary aspects of the template, but this has two distinct advantages. First, you will learn what each element of the template is doing as you add them in. Second, you will know exactly how everything is being done

because there won't be any defaults that it is looking back on. However, I mentioned earlier how you could find the source code for the SAS defaults and that could be used as a beginning template. Then you would take pieces out to see what changes. Either way, to start a template of your own you must first choose an area where you will put it. If you are working on a PC you might choose an area on the C: drive to start with. No matter what area you have chosen, establish a libname to that area and then put the following statement:

```
ods path mylib.rtfplate (write);
```

Where "mylib" is your library name and "rtfplate" is any name you choose for the template. This establishes which area the template will be created in. The "WRITE" inside of the parenthesis will allow you to create the file. "UPDATE" and "READ" could be used in place of "WRITE". When you are creating the template, you will need to use "WRITE". However, if you are going to use the template and not make any modifications to it, you will want to use "READ". This is because if "WRITE" is used it will delete the template assuming that you will be recreating it. The ODS PATH statement is done in open code. In this case, "rtfplate" would be the name of the file created that would hold the styles you have yet to create.

TEMPLATE BASICS

Anytime you are creating a template you must begin with:

```
Proc template;
```

Next you should establish what style you are creating using a DEFINE statement. By inserting the following code:

```
Define style lookgood;
```

You have told the template procedure to create the style LOOKGOOD inside of the active template. In this case, the active template would come from the ODS PATH statement already submitted and the style LOOKGOOD would be stored in the template RTFPLATE which is in the directory MYLIB. Note that the path statement that is outside of the PROC TEMPLATE, is what was used to determine where LOOKGOOD would go. There was no need to reference the library. When naming the style you are creating you can create a directory structure inside of your template. By using the following code:

```
Define style folder.lookgrt;
```

you have created a style underneath FOLDER. Notice that the syntax **does not** reference the library name that was set up. FOLDER is a user created name that will be placed inside of the active template. There can be as many dots as you need to organize your styles (This was a hurdle for my understanding, I hope this information helps others). Every DEFINE statement must have a corresponding END statement. So all together the code to begin a new template style might look something like this:

```
Libname mylib "c:/temp";
ods path mylib.rtfplate (write);
Proc template;
  Define style lookgood;
  End;
Run;
```

STYLES AND MORE STYLES

Now you have started your own template that can modify the way all of your output looks. When using an ODS RTF statement, simply reference your newly created style LOOKGOOD in the STYLE= option. As long as your ODS PATH points to the correct location you will see your output take on the specifications you desire. For instance, if you desire the style LOOKGOOD to have Courier font titles when available, you can specify that using the following statement between the DEFINE and END statement:

```
Style systemtitle "Controls the system title" /
  Font_face="Courier";
```

Notice that a new kind of style is being introduced. The style SYSTEMTITLE is a definition of how to handle titles when using the style LOOKGOOD. To avoid confusion with the word "style", I will continue to use the word "style" to refer to any user created Style Definition that can be used in the STYLE= option of the ODS RTF statement. I will refer to styles within styles as "Style Elements". This will be consistent with the SAS web site. That is where the emphasis will be placed now, on the Style Elements.

STYLE ELEMENTS

Style Elements take on a basic pattern. Now with one more option to the Style Element, let's examine the syntax of a basic Style Element:

```
Style systemtitle "Controls the system title" /
   Font_weight=bold
   Font_face="Courier, Arial";
```

Inside of the quotes before the forward slash is a comment that describes the style. The options afterward seem to be fairly clear. FONT_WEIGHT will cause the titles to be bolded and FONT_FACE will first look for courier and then arial. If the person viewing the file does not have either of these fonts then the browser will choose another default font. This Style Element will only affect the text of the titles. You could use the Style Element TITLESANDFOOTERS to change both the titles and the footers at the same time. Anything that is used as an option in both of these Style Elements would be taken from SYSTEMTITLE.

SOME KEY RTF STYLE ELEMENTS

There is an enormous amount of Style Elements available that allows you to modify your output to meet many different requirements. It is not within the scope of this paper to document all of the Style Elements that are available and how to use them (I couldn't use them all if I had to). However, there are some key Style Elements that have helped me to create great output that I would like to show here. The following Style Element will affect column headers as shown in Figure 2:

```
style header  /
  font_weight=bold
  background=white
  font_size=1;
```

Figure 2.

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0

You will find that the options associated with Style Elements can be used across many other Style Elements. Whether you are modifying the titles, footnotes, column headers or the data, the same font options and syntax can be used. SYSTEMFOOTER and TITLESANDFOOTERS are Style Elements affecting the footnotes and both titles and footnotes respectively. Another important Style Element is Table. Example usage, display shown in Figure 3:

```
style Table "Controls overall table style." /
  cellpadding=3
  rules=groups
  frame=void;
```

Figure 3.

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0

This Style Element will affect the main body of the output. CELLPADDING will place extra space in each cell. RULES=GROUPS will place a line at the top and bottom of the output only. You could have rules for every row and column if you wanted. FRAME=VOID will eliminate a box around the output. The list of available Style Elements and options within them goes on and on. When you are looking for more options

and details on this, the documentation is on the SAS web site at: “http://www.sas.com/rnd/base/early-access/odsdoc2/sashtml/tw5195/index.htm” Follow the template procedure link and then go to the Define Style link to find other Style Elements out there.

PARENT STYLES

One way to take advantage of the default RTF output SAS is already able to produce and still get the modifications you need is to use parent styles. If you like everything about the printer style, you could use it as the parent style and then make the necessary modifications to it. The code might look something like this:

```
Ods path mylib.rtfplate (write) sashelp.tmplmst(read);
Proc template;
   Define style lookbest;
    Parent=styles.printer;
    Style systemfooter from titlesandfooters/
         Just=left;
   End;
Run;
```

A couple of new things were introduced in here. Notice that more than one template store was placed in the ODS PATH statement. By placing a read reference to the SASHELP template, you can read from any of the styles that SAS has installed with the system software. You are creating the style LOOKBEST which is identical to the style PRINTER, except for the footnote that is left justified in LOOKBEST. Notice also that the style element SYSTEMFOOTER will have all of the identical attributes of the style element TITLESANDFOOTERS, except those that are specifically changed. In this case, the justification is all that is changed.

SPECIFIC MODIFICATIONS

While you are working in PROC TEMPLATE you are creating a general feel of what the viewer will see. If you need to emphasize specific columns or titles, that needs to be done inside of the program that produces the output. To emphasize a specific column in a PROC REPORT, you would add some extra information onto the end of your DEFINE statement and the report would work fine wherever your ODS destination is. For instance, the following code is valid SAS that will produce a column that is emphasized over the others.

```
Define prot /display “Protocol”  style(column)=
{foreground=  yellow background=red};
```

This is yet another use of the word “style”. This is unrelated to the styles and style elements I have been referring to so far. This, for lack of a better word, will be referred to as a “keyword style” from here out. That is because this is a specific phrase that when used in PROC REPORT is recognized as a request to modify that specific column. If for a single report you desired to modify all columns, the keyword STYLE statement could be used in the PROC REPORT statement, just like the DATA= option. Then all of the columns would take on that appearance. Also note that titles and footnotes can be modified on an individual basis within each program. A single title or footnote could be left justified while all others are centered. For example:

```
Title2 just=left “on the left side”;
```

This would be a left justified title, regardless of the style used.

SURPRISES

As you begin to produce RTF output using PROC REPORT, there might be a few things that you find surprising. The wonderful world of SAS is such that all ideas are constantly being improved on. I would expect that some of the solutions I present here could be improved, but nevertheless they work. For instance, the BREAK command used in PROC REPORT has no affect on RTF output. I have found that using a COMPUTE statement instead and inserting a blank line often does the trick. Indentations can be a problem. If blank spaces are used to distinguish one category from another, the RTF output will ignore the spaces and left justify all text equally. SAS is aware of this and this will be handled in a new release of SAS (this paper is based on V8.1). In the mean time, a hyphen followed by spaces is the best I have come up with. You might be testing your output to the output window and then be surprised to see that your column width has changed dramatically in the RTF file. This is because the WIDTH= command in the PROC REPORT has no effect on the RTF file. SAS will determine the best length for you unless you specify in the define statement a specific cellwidth. It would look like this:

```
Define var / display “Header” style= {cellwidth=100};
```

Then you can pad each individual column to your exact specifications. The way GROUP and ORDER variables function is slightly different when obtaining RTF output. When going to your output window the GROUP variables will print on the beginning of each page. With RTF output a new page may begin with blank values in the columns for GROUP variables. Putting a macro around the PROC REPORT to call the PROC REPORT once for each page is a simple coding work around. For larger reports this may have different issues. I have written a macro that will call the PROC REPORT as many times as it needs to based on the number of observations you wish to display on a page. Then each page begins with the appropriate display of GROUP and ORDER variables.

TAKING ADVANTAGE OF RTF

One of the main advantages of rtf output is that it can mark up text to be bold, italic, underlined, superscript, subscript, different sizes, etc. While I have already explained some ways to get bold text and specific fonts, many of these advantages have not been explained yet. If for instance, you desired a few words in a title to be bold and the rest to be plain text you would do the following:

```
Title “{The} {\b BEST} {Class is}”;
```

Since an RTF file is merely a text document, by including RTF language in the title you can put RTF statements straight into the title and the word processor will perform their appropriate RTF function. This opens up a world of possibilities. If you create a very small document and save it as RTF, you can open it in a text editor and view the RTF code used to create your original document. Then you can place RTF statements mixed throughout your code when you need them. Another example of this follows showing superscript with the output shown in Figure 4:

```
Title2 "{Mr Smith’s 8 }{\super th}{Grade class}";
```

Figure 4.

The **BEST** Class is
Mr Smith's 8th Grade class

Obs	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0

Page numbering has also been a hot topic with SAS output. How to get the total number of pages on each page of the output has generated much discussion. With RTF, this can be done very easily. Using the technique I described earlier, I have found that the following title will give a total number of pages right justified:

```
title1 j=r "{\field{\*\fldinst { NUMPAGES  \\* MERGEFORMAT
}}{\fldrslt {\lang1024\langfe1024\noproof 1}}}" ;
```

It may be ugly here, but a word processor will interpret it as the total number of pages in a document.

CONCLUSION

Producing output in RTF is a distinct advantage for those who can do it. Many companies have put forth great effort to have systems in place that take SAS system output and put it into Word files. SAS has now developed a way that can produce quality RTF output quickly. Those who have been behind in this area can be caught up very quickly. This paper is enough to get you started exploring the possibilities of SAS system RTF output. Unfortunately, this paper had to leave many related

topics untouched for the reader to explore. Those who do explore and develop code in this area will be rewarded with great word documents!

ACKNOWLEDGMENTS

Thanks to Jesus Christ for giving me all that I have. Thanks to the programmers at Synteract for helping to develop and refine these ideas.

CONTACT INFORMATION

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

Bob Hull
Synteract, Inc.
187 Calle Magdalena
Encinitas, CA 92024
Work Phone: 760 634 2133
Fax: 760 634 2170
Email: rhull@synteract.com
Web:

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 registered trademarks or trademarks of their respective companies.