

Making an RTF file Out of a Text File, With SAS

David Franklin, Independent SAS Consultant, Litchfield, NH

ABSTRACT

There are many times when we want to create an RTF file out of a text file. Many solutions involve running a Microsoft Office VBA macro, and there are some solutions that read in the text file and use SAS® ODS RTF to create the RTF format file. This paper brings yet another solution (with code) to the fore where the text file is read in and the relevant RTF codes added, the advantage being that the solution is often quicker than other methods.

INTRODUCTION

A text file has been created that represents our output, but our customer wants to open the file in Microsoft Word. How would we solve this problem?

The first solution could be to open the text file in Microsoft Word, do some manual manipulation and then save the file as a Word or RTF file.

But what if the text file could be converted to an RTF file inside a single datastep using SAS? This paper introduces such an approach, where along the way we will touch on the RTF syntax and find out that it is not as bad as we think it is.

WHAT IS RTF?

RTF stands for Rich Text Format and was originally a structure that Microsoft created as a way to transfer Word documents to products like OpenOffice, WordPerfect and Lotus Symphony/WordPro. In its basic form it is no more than a text file with a number of instructions inside '{ }' which when imported into your word processor not only carries the text but also information like page size, margin size, font color and type, and whether text is bold or underlined, or both!

Below is the RTF that prints out "Hello World!" when opened inside Word (note that the text must first be saved in a text editor with the filename extension of 'rtf' before being opened in Word):

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1033
{\fonttbl{\f0\fmmodern\fprql\fcharset0 Courier New;}}
\viewkind4\uc1\pard\f0\fs20 Hello World!\par
}
```

This text does look daunting, but the text up to the point of the "Hello World!" is just the header text which contains the signal that it is an RTF file, ASCII format, code page and language information, the font that is being used, and the font size. The '\par' is in effect a carriage return, and the final '}' closes the file.

Using this solution, i.e. by reading in our text file and putting the header text on, placing a '\par' at the end of each line and closing the file, it could be said we are just about there. However, to quote Emeril Lagasse "Lets kick it up a notch!"

A MORE GENERAL SOLUTION

Now that the basics of what an RTF file is given, the next stage is to write something with a bit more information for our word processor interpreter. This information will be:

- paper size and orientation
- margin size
- font size (a small discussion here on how to set the font size)
- line height

Now lets look at a solution in the form of a macro:

```

01 %macro txt2rtf( _txtfn=                /*Text file to convert to RTF*/
02             , _rtffn=                  /*RTF file that is to be output to*/
03             , _pgwh=15840              /*Page width in twips*/
04             , _pght=12240              /*Page height in twips*/
05             , _mgns=1440               /*Margin width in twips*/
06             , _lspe=%str(\landscape)   /*If landscape, need to add this value*/
07             , _fnsz=8                  /*Font point size - integer*/
08             );
09 data _null_;
10     infile "&_txtfn" end=eof;
11     length _txt $200;
12     input;
13     file "&_rtffn";
14     if _n = 1 then do;
15         put '{\rtf1\ansi\ansicpg1252\deff0\deflang1033' /
16             '{\fonttbl{\f0\modern\fprql\fcharset0 Courier New;}}' /
17             '{\colortbl \red0\green0\blue0;}' /
18             "\paperw&_pgwh.\paperh&_pght.\margl&_mgns.\margr&_mgns." /
19             "\margt&_mgns.\margb&_mgns.&_lspe." /
20             "\viewkind4\uc1\pard\ql\fi0\li0\ri0\sb0\sa0\sl-%eval(&_fnsz*20)" /
21             "\cf0\f0\fs%eval(&_fnsz*20) " _infile_;
22     end;
23     else do;
24         if substr(_infile_,1,1)=byte(12) then do;
25             _txt='\page '||substr(_infile_,2);
26             put _txt;
27         end;
28         else if ^eof then put '\par ' _infile_;
29         else if eof then put '\par ' _infile_';
30     end;
31     run;
32 %mend;

```

There are a number of points to make about the macro.

The first point to make is the measurements – these are in 'twips', a standard unit that is used in printing. For conversion:

1 inch = 1440 twips
1 cm ≈ 567 twips

The page sizes used, as set in the `_pgwh` and `_pght` parameters, are values for Letter sized paper – for A4 sized paper the values change to 16839 and 11907 respectively when in landscape format.

As can be seen from the conversion and setting above, a margin of one inch is set around the page.

If the paper is set to landscape then the `_lspe` must be set to “\landscape” – if portrait then the value must be blank.

Point size is set using standard point sizes that you set in a word processor, the only limitation being that the value must be integer for this macro to work.

Now lets look at some actual RTF code that is being set.

Lines 15 and 16, are what we saw in the original that was created in the "Hello World!" example.

Line 17, this defines the color of the font using the RGB format, where the colors have a value from 0-255 -- the color defined is black.

Lines 18 and 19 are where the page size (paperw = paper width, paperh = paper height), margins (left, right, top, bottom), and specifying that the page should be landscape, is defined.

Lines 20 and 21 defines the last of our output definitions:

```

\viewkind4\uc1 = setup
\pard = reset paragraph definitions
\ql = text is left aligned
\fi0\li0\ri0 = first line, left/right line indent
\sb0\sa0 = spacing above/below line
\sl-%eval(&_fnsz*20) = specific line spacing, defined as font size * 20
\cf0 = Foreground color

```

```
\f0 = font color
\fs%eval(&_fnsz*2) = font size, defined as font size defined * 2
```

Lines 24 to 27 deals with the page break character to an RTF code – in this macro it looks for the CR (Carriage Return) character which is embedded in the text file and changes it to a RTF tag. Note that the page break will always occur in column 1 of the file.

The hardest part when setting up the macro is the settings to use when setting up your text output linesize and pagesize settings. As a guide, refer to the following table (margin size is set to 1 inch and Courier New font is used):

Paper Size	Font Size	PAGESIZE	LINESIZE
LETTER	8	58	134
LETTER	9	52	119
LETTER	10	46	104
A4	8	49	145
A4	9	44	129
A4	10	39	116

If other margin or font sizes or fonts are needed then attempt to get the pagesize and linesize values within your word processor by finding the maximum number of characters per line and per page it is possible to fill.

OTHER IMPROVEMENTS

Since this macro was developed, its basic structure has remained the same. One version replaced the *_pght* and *_pgwh* parameters with page sizes, e.g. values such as A4 and LETTER. Another enhancement was to allow for half point sizes, e.g. 8.5, but the gains from this were minimal and added an extra step where the calculation for the line spacing and font size had to be done before the step that actually did the conversion. The final enhancement made very recently was one where it had to handle situations where the file was an old IBM style file which contained control characters in the first column, e.g. 1 for page break and 0 for a blank line.

Despite all these enhancements, the basic macro above is often still used, and indeed carried, as part of my toolbox today.

CONCLUSION

A brief introduction to RTF and a macro were presented in the paper to show how a text file can be converted to an RTF format file without the need for Microsoft VBA or other techniques. Guidance was also given for the linesize and pagesize settings needed to get the page breaks working correctly and still fill the required margins of the output.

REFERENCES

Rich-Text Format (RTF) Specification, RTF Version 1.0 -- Microsoft Corporation.

CONTACT INFORMATION

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

Name: David Franklin
Enterprise: Independent SAS Consultant
Address: 16 Roberts Road
City, State ZIP: Litchfield, NH 03052
Work Phone: 603-275-6809
E-mail: dfranklinuk@compuserve.com
Web: <http://www.TheProgrammersCabin.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.