Paper 105-2023

Calculate Physical Length of String in RTF file with 'Times New Roman' by SAS

William Wu, Astex Pharmaceuticals, Inc., Pleasanton, CA

Steven Li, Medtronic PLC, Mounds View, MN

ABSTRACT

While using PROC REPORT with ODS RTF for output table/listing generation, people often wish to know the 'Physical Length' of a string, instead of the number of characters in the string. The 'Physical Length' of a string is the length which is displayed on the RTF output file. The physical length of a string could be used to decide the column's minimum width without word wrapping up. This paper created a custom SAS function through PROC FCMP to calculate the physical length of either 'Times', or 'Times New Roman' font, and a concept of 'characteristic width' of character is introduced.

INTRODUCTION

SAS didn't previously provide a method for determining the physical length of a string. This paper illustrated to calculate the physical length for a string. In SAS, physical length of a string depends on the following factors:

1. Font name: There are various font names, such as 'Times', 'Times New Roman', 'Arial' and so on. Font 'Times' (or even write as 'Times Roman') and 'Times New Roman' have the same physical width. This paper only considers the physical length on both font 'Times' and 'Times New Roman'.
2. Font size: For font 'Times' and 'Times New Roman', this paper validated that the physical length of a string is proportional to the value of font size (as an integer in unit 'pt').
3. Font weight: This paper calculated physical length of a string on both font weight 'Bold' and 'Normal'. For each character, the physical length for both 'Bold' and 'Normal' must be considered separately. That's based on following facts: Among 95 character (ASCII code from 32 to 126), 48 characters are with the same physical width between 'Bold' and 'Normal', 44 character's physical width is wider on 'Bold' font (compare to 'Normal'), and 3 character's physical width is even narrower on 'Bold' font (compare to 'Normal').
4. Font style: The font style could be 'Roman' or 'Italic'. Because of a few cases on 'Italic', this paper only considered for physical length on font style as 'Roman'.
5. Combination of characters: There are variety of string which depends on the combination of the characters. But basically, the character is only 96 (include alphabet in upper case and lower case, digit from 0 to 9, and the other symbol, plus 'No-Break Space'). When these character's physical width is known (for both 'Bold' and 'Normal'), the string's physical length could be calculated by adding these widths up as a cumulative sum. That's the way we calculate the physical length of a string. In other words, the physical length of a string is regardless of the order of its characters.

All demonstrations and sample code in this paper use version 9.4 (TS1M3) of SAS on a Windows platform and 'Microsoft Word 2013'.

CREAT A STYLE FOR 'TIMES ROMAN' ON ODS RTF

In SAS statement 'ODS RTF', option 'style=<style name>' is used to specify the output style for RTF output destination. Following SAS code displayed a listing of styles in the Sashelp.Tmplmst item store:

```
proc template;
   list styles / store=sashelp.tmplmst;
run;
```

A SAS macro called %style in 'APPENDIX' of this paper could create various new styles with all fonts as

'Times', or 'Times New Roman' by using PROC TEMPLATE. In macro %style, if you set parameter 'rules' as '1' (for option rules=all), or '2' (for option rules=groups), you could get two different table formats as the following figure 1:

(1) rules=all, borderspacing=0.5pt, borderwidth=1.5pt

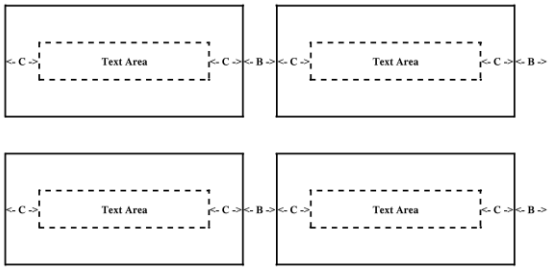
Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5

(2) rules=groups, borderspacing=0.5pt, borderwidth=1.5pt

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

Figure 1: There are 2 different table formats by choosing option 'rules'

Within the 'style table' statement in macro %style, three options as following should be set to the suitable value to get the better appearance.



- cellpadding= : Specifies the amount of white space on each of the four sides of the content in a table cell (as 'C' in the picture above).
- borderspacing= : Specifies the vertical and horizontal thickness of the spacing between cells in a table (as 'B' in the picture above).
- borderwidth= : Specifies the width of the table borders. The value of 'borderwidth=' is applied to all four borders.

The option 'borderspacing' has alias as 'cellspacing'. When the option 'borderspacing' is set as value '0', but 'borderwidth' is a positive value, ODS destination sets the value of 'borderspacing' equal as value of 'borderwidth'. See figure 2 on Table (3):

(3) rules=all, borderspacing=0pt, borderwidth=1.5pt

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

(4) rules=all, borderspacing=1.5pt, borderwidth=1.5pt

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

(5) rules=all, borderspacing=0pt, borderwidth=0pt

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

Figure 2: Table (3) with 'borderspacing=0pt and borderwidth=1.5pt' is same as Table (4) where 'borderspacing=1.5pt and borderwidth=1.5pt' on the thickness of the spacing between cells

Typically, when 'borderspacing=0pt and borderwidth=0pt', the ODS destination sets RULES=NONE and FRAME=VOID as Table (5) above.

The setting for style in macro %style could be temporarily overwritten in PROC REPORT statement by options as following coding:

```
style(column)=[rules=<all|groups> fontweight=<bold|medium> fontsize=<x>pt
borderspacing=<y>pt borderwidth=<z>pt]
```

The following is the relationship about the length unit between 'inch', 'pt', and 'twip':

1 in = 72 pt, 1 pt = 20 twip, 1 in = 1440 twip

Using PROC REPORT with ODS RTF, the width of the column is set in 'define' statement by option as:

style=[cellwidth=<W_x>pt]

But PROC REPORT/ODS RTF will set the standard column width as W_c (pt) (while the coding column width as W_x), here:

$$W_c = \text{round}(W_x, 0.05) \quad (1a)$$

Or consider the unit as 'twip', the relationship is:

$$W_c = \text{round}(W_x) \quad (1b)$$

In other words, PROC REPORT/ODS RTF set the integer column width W_c by rounding W_x to the nearest integer twip for RTF output file (because RTF file only accept integer twip as the column width). And there is no note or warning message issued into the SAS log file for rounding W_x to W_c.

On the other hand, the standard column width 'W_c' has a tiny 'non-liner' shrink. The wider the string, the more shrink ratio. let 'W_a' be the actual column width after the shrink, the relationship is as following empirical formula (with enough accuracy for any case of string):

$$W_a = W_c * (1 - a * W_c^{**}b) \quad (2a)$$

$$W_c = W_a * (1 + a * W_a^{**}b) \quad (2b)$$

Here, constants a = 5.117E-14 and b = 1.492, while width 'W_c' or 'W_a' is in unit 'twip' (The evaluation of coefficient 'a' and 'b' are associated with the evaluation on 'characteristic width' in the next chapter).

When setting 'frame=hsides' in 'style table' statement in PROC TEMPLATE, and the column is in the very left side (the first column in the table, or a column wrap up to the next page to become the first column), let 'W_a' be 'actual column width', and 'W' be string's physical width, on column setting without the word/letter wrap up, the formula is as:

$$W_a >= W + 2*\text{'cellpadding'} \quad (3a)$$

When the column is not located on very left side of the page (since the second column until the last column), the formula is amended as:

$$W_a >= W + 2*\text{'cellpadding'} - \text{'borderspacing'} \quad (3b)$$

$$W_a >= W \quad (\text{if 'cellpadding'=0 and 'borderspacing'=0}) \quad (3c)$$

Both formula (3a) and (3b) get verified in macro %Column in 'APPENDIX'. And macro %Shrink in 'APPENDIX' demonstrated the relationship on 'W_x', 'W_c', and 'W_a'. The output is as Figure 3.

From Figure 3 on the first 4 lines, consecutive 139 equal sign '=' is set respectively as following coding column width (in unit 'twip'):

W_x =14257, 14265.5, 14256.46, and 14256

And PROC REPORT will round it up to the nearest integer 'twip' relatively as following standard column width (in unit 'twip'):

W_c=14257, 14257, 14256, and 14256

So, from Figure 3, 139 equal sign '=' is without wrap up for W_c =14257 (see the first line Figure 3); and with wrap up for W_c = 14256. But 278 (=139*2) equal sign '=' take width W_c =28514 (=2*14257) will be wrapped up (see the last line in Figure 3).

One single equal sign '=' on font weight 'bold' and font size 9pt, the physical width is 102.568325, so when we consider the 'non-liner' shrink as formula (2a), the calculation and conclusion are as following:

102.568325*139=14256.997175<=14257*(1-5.117E-14*14257**1.492) -- Without wrap up

102.568325*278=28513.99435>28514*(1-5.117E-14*28514**1.492) -- With wrap up

Compare line 1 to line 8 in Figure 3, that's why it's important to consider a tiny 'non-liner' shrink (it changes standard column width '$W_{c'}$ to actual column width '$W_{a'}$).

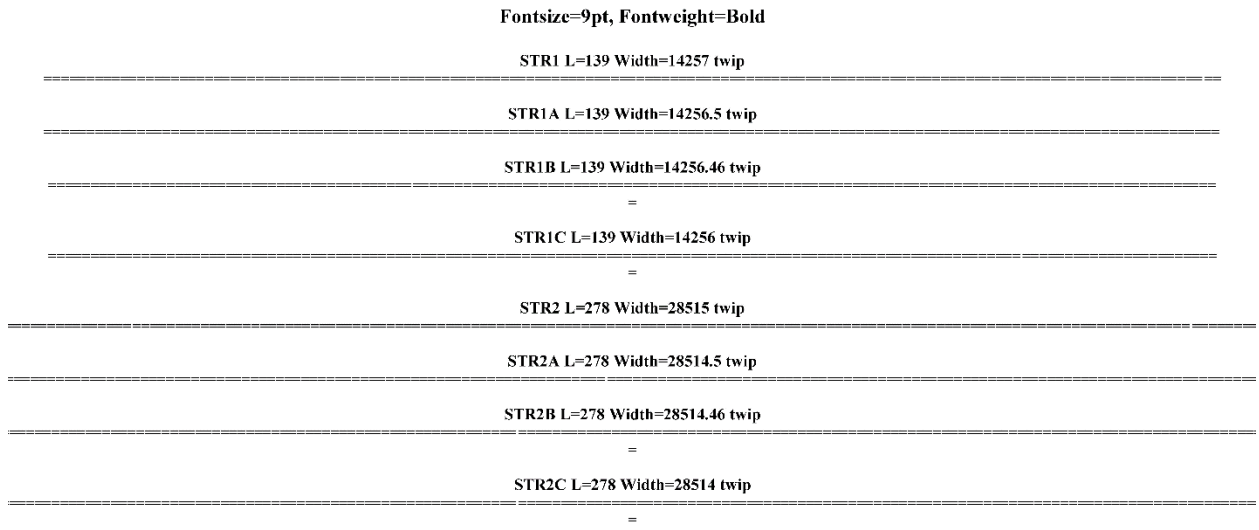


Figure 3: For Relationship on W_x W_c and W_a, the first 4 lines are string with 139 char '='. The last 4 lines are 278 (= 139*2) char '=' (displayed partially in the center of the page). The feature on 'rounding to twip' and 'non-liner shrink' showed up between line 1 and line 8.

MEASURE PHYSICAL LENGTH IN RTF FILE WITH 'TIMES ROMAN'

The principle of calculating the physical length of a string in 'Times' or 'Times New Roman' is based on the basic rule, which we could call it as 'principle of linear superposition':

> *By removing the leading and trailing blanks (with ASCII code 32), a string's physical length is equal to the **cumulative sum** of each character's characteristic width, multiplied by the font size (in unit pt).*

If a string is made up of 'n' characters, and each character's characteristic width are given by: 'w_1, w_2, .. , w_n' (the unit as 'twip/pt'). Let W be string's physical width (in unit 'twip'), and p be font size as an integer (in unit 'pt'), then the formula is:

$$W = p*(w_1+w_2+ .. +w_n) \quad (4)$$

In other words, by considering some of characters with different ASCII code share the same characteristic width (for example, digit 0, 1, .. , 9 have the same characteristic width). If a string is made up of 'n' characters, 'n_1' characters are with characteristic width 'w_1', 'n_2' characters are with characteristic width 'w_2', …, and 'n_k' characters are with characteristic width 'w_k', where $n= n_1+n_2+_{...}+n_k$. then the formula is:

$$W = p*(n_{1*}w_1+ n_{2*}w_2+ .. + n_k *w_k) \quad (4a)$$

For each ASCII character, the characteristic width 'w_k' is a constant, dependent on the sequence location in ASCII table, and independent of the font size. In this paper, the number of different characters to be considered is 96 (from ASCII code 32 to ASCII code 126, plus a 'No-Break Space' as ASCII code 160). Each character has 2 characteristic widths respectively for both font weight as 'Bold' and 'Normal'. So now, we have 192 (=96*2) characteristic width to be measured. Notice that the measurement is setting a standard column width 'W_c' (integer twip) on 'define' statement in PROC REPORT to see if the string is with/without word/letter wrap up – This is a coding ruler for measurement, instead of actual physical ruler. The physical length of a string 'W' is just in direct proportion to the physical length of a string displayed on computer screen by Microsoft Word, and in direct proportion to the physical length of a string printed on

paper by Microsoft Word. The value of string's physical length between value 'W' and 'printed on paper' (when using a physical ruler to measure) is very close in accuracy, the relative error range is: +/- 1% (depending on the printer).

How to measure a specific character's characteristic width? If a string has font size is p. and is composed of 'n' same characters, and each of them has characteristic width as 'w_k'. So, the formula (4a) can be written as formula (5):

$$W = p*n*w_k \quad (5)$$

While set the style with 'cellpadding'=0, and 'borderspacing'=0 from macro %style, we get: $W_a >= W$ (as formula (3c)). On the other hand, if setting standard column width as W_c (integer twip) without wrap up, but with wrap up for column width as W_c–1 (twip), we called it as a 'successful measurement', then:

$$(W_c-1) * (1-a*(W_c-1)**b)/p/n < w_k <= W_c * (1-a*W_c**b)/p/n \quad (5a)$$

On the other word, a 'successful measurement' is by watching the RTF output file to see the string is without, or with wrap up, and could calculate a range of the characteristic width 'w_k' (with lower/upper limit).

Using the macro %style in the APPENDIX, the code as following is a basic physical length measurement program. It measures 60 consecutive dot sign's physical length as 3000 twip for font size as 10 pt.

```
options mprint nodate nonumber orientation=landscape papersize=Tabloid;
ods path reset;
ods path (prepend) work.templat(update) sasuser.templat(read) sashelp.tmplmst(read);
ods escapechar='□';

data Dot60;
       length Str $60;
       retain B ' ';
       Str = repeat('.',59);
run;

%style(stylen=1, boldn=2, cellpadding=0, borderspacing=0, borderwidth=0);

%let CL = 3000;

options orientation=portrait papersize=letter center;
title "Measurement 60 consecutive dot sign's Physical Length as &CL.twip?";
footnote;
ods listing close;
ods rtf file="&path\Dot60.rtf" style=Mystyle1_2 bodytitle image_dpi=600;

proc report data=Dot60 nowd;
    column Str B Str=Str2;
    define B   /display style=[cellwidth=5pt] ' ';
    define Str /display style=[just=l cellwidth=%sysevalf(&CL/20)pt]
       "Str, W=&CL.twip %sysevalf(&CL/20)pt";
    define Str2/display style=[just=l cellwidth=%sysevalf((&CL-1)/20)pt]
       "Str2, W=%sysevalf(&CL-1)twip %sysevalf((&CL-1)/20)pt";
run;

ods rtf close;
ods listing;
```

The output is as Figure 4:

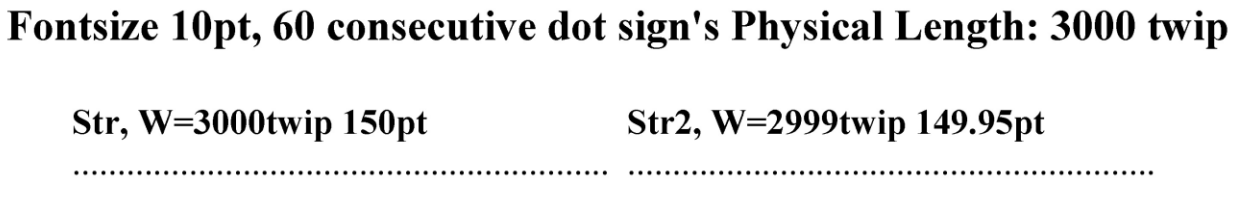


Figure 4: 60 consecutive dot signs with column width 3000 twip, no character wrapping up. But with wrap up while the width is 2999 twip

From the code as above, we get: a string with 60 dot signs is with physical length as 3000 twip. And dot sign's characteristic width 'w' is as:

2999*(1-a*2999**b)/10/60=4.99833329<w<=3000*(1-a*3000**b)/10/60=4.99999996

About the measurement to Space (ASCII code as 32) and 'No-Break Space' (ASCII code 160), an issue is: Both Spaces are all invisible, so, a third sign should be used as the starter and terminator, normally, using dot sign. Another issue (or rather say, founding) for Space (ASCII code 32) is: 'single isolated space' and '>=2 consecutive space' have totally different physical width in per character. Figure 5 as below showed the situation:

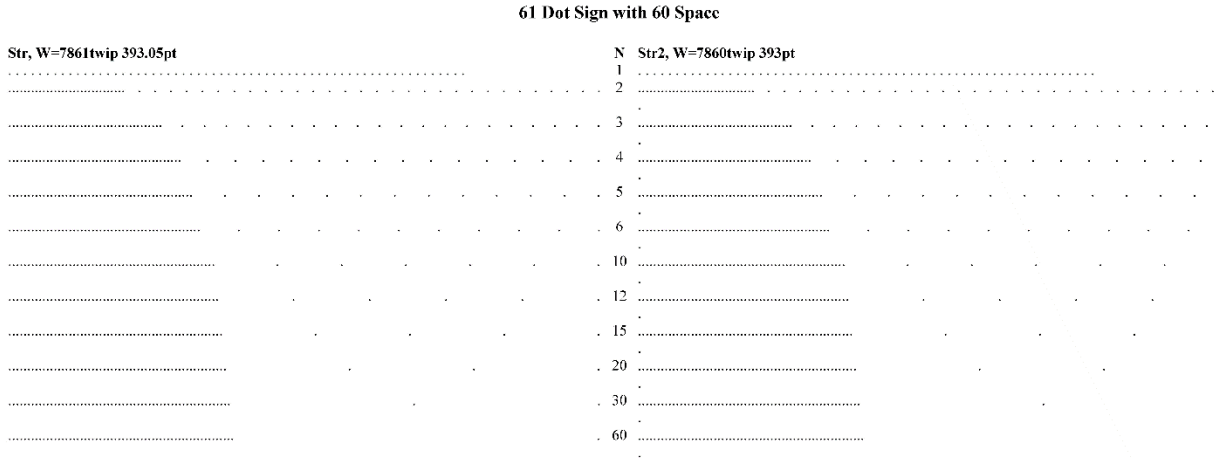


Figure 5: 61 dot signs with 60 Space will take 7861 twip as the physical length while the Space is '>=2 consecutive space' (consecutive space number: N>=2). But 61 dot signs with 60 'single isolated space' (N=1) is far shorter than them.

The table above is with 12 rows as N=1, 2, 3, 4, .. ,60 (N stands for the number of consecutive spaces). Each '>=2 consecutive space' have the same physical width. But comparing to than '>=2 consecutive space', the 'single isolated space' is in different characteristic width. That's the only case to break the 'principle of linear superposition'.

When we measured using the similar method as above, we found that some of characters are in same width. The 'Bold' font is divided into 22 width group while 'Normal' font is 23 group. Each characteristic width and the shrinkage coefficient 'a' could be solved by these inequations. Now we list the exact value of Characteristic Width into 32 groups with the absolute error level as ±1E-10 in tables (6) as Figure 6.

(6) Characteristic Width (Short as: C.W.) to Bold/Normal Font on 'Times Roman'

Group #	1	2	3	4	5	6	7	8
Bold Font				,. (No-Break Space, 1 Isolated Space)	!\il	!()-:;[]`fjt	Is	}{
Normal Font	'	|		,. (No-Break Space, 1 Isolated Space)	/:;\ijlt	!()-I[]`fr	Js	
C.W. (twip/pt)	3.6035144396	4.0039049329	4.4042954262	4.9999983552	5.5566387971	6.6601540591	7.7832005647	7.8808567826

Group #	9	10	11	12	13	14	15	16
Bold Font			(>=2 Consecutive Space)	cerz			#$*0123456789?J_agovxy	~
Normal Font	(>=2 Consecutive Space)	"		?acez	^	}{	#$*0123456789_bdghknopquvxy	
C.W. (twip/pt)	8.0175754876	8.1640598144	8.5351534424	8.8769502049	9.3847625379	9.5996062172	9.9999967105	10.400387204

Group #	17	18	19	20	21	22	23	24
Bold Font		"	Sbdhknpqu		+<=>	^	FP	BELTZ
Normal Font	~		FPS	+<=>			ELTZ	BCR
C.W. (twip/pt)	10.820308941	11.103511972	11.123043216	11.279293165	11.396480626	11.621089927	12.216792856	13.339839362

Group #	25	26	27	28	29	30	31	32
Bold Font	ACDNRUVXYw	GHKOQ	&m			@	M	%W
Normal Font	ADGHKNOQUVXYw	&m	%	M	@		W	
C.W. (twip/pt)	14.443354624	15.556635508	16.660150770	17.783197275	18.417962691	18.603509505	18.876946915	19.999993421

Figure 6: Characteristic width (as C. W.) of character for font weight as 'Bold' and 'Normal'

According to table (6) in Figure 6, and based on formula (4a), a SAS custom function 'width()' is created to calculate the physical width in font 'Times' or 'Times New Roman'. In the function, parameter 'string' is a character variable name for us to calculate the physical length of the string. The parameter 'bold' could choose 'bold=1' for font weight as 'Bold' or choose 'bold=2' for font weight as 'Normal'. And parameter 'fsize' is the font size of the string in unit 'pt'.

```
proc fcmp outlib=work.funcs.maths;
  function width(string $, bold, fsize);
    array w[32] (
3.6035144396 4.0039049329 4.4042954262 4.9999983552 5.5566387971 6.6601540591 7.7832005647 7.8808567826
8.0175754876 8.1640598144 8.5351534424 8.8769502049 9.3847625379 9.5996062172 9.9999967105 10.400387204
10.820308941 11.103511972 11.123043216 11.279293165 11.396480626 11.621089927 12.216792856 13.339839362
14.443354624 15.556635508 16.660150770 17.783197275 18.417962691 18.603509505 18.876946915 19.999993421);
    array c[2,32] $30 (' ' ' ' '|' 'A02C202E'x '''/\il' '!()-:;[]`fjt' 'Is' '}{'
        ' ' ' ' '□' 'cerz' ' ' ' ' '#$*0123456789?J_agovxy' '~'
        ' ' '"' 'Sbdhknpqu' ' ' '+<=>' '^' 'FP' 'BELTZ'
        'ACDNRUVXYw' 'GHKOQ' '&m' ' ' ' ' '@' 'M' '%W'
        ''''' '|' ' ' 'A02C202E'x '/:;\ijlt' '!()-I[]`fr' 'Js' ' '
        '□' '"' ' ' '?acez' '^' '}{' '#$*0123456789_bdghknopquvxy' ' '
        '~' ' ' 'FPS' '+<=>' ' ' ' ' 'ELTZ' 'BCR'
        'ADGHKNOQUVXYw' '&m' '%' 'M' '@' ' ' 'W' ' ');
    length string str $8800;
    str = tranwrd(tranwrd(cats(string),'  ','□□'),'□ ','□□');
    do i=1 to 32;
        if c[bold,i]>' ' then WD + countc(str,c[bold,i],'T')*w[i];
    end;
    WA = WD*fsize;
    return(ceil(WA*(1+5.117E-14*WA**1.492))/20);
  endfunc;
run; quit;

options cmplib=work.funcs;
```

In the function, code 'tranwrd(tranwrd(catt(string),' ','• • '),'• ','• • ')' replaced each space ' ' as character '• ' (ASCII code as 127) while it's belong to '>=2 consecutive space', but keep the 'single isolated space' as it was. Code "countc(str,C[bold,i],'T')" trims trailing blanks from both 'str' and 'C[bold,i]', and then counts the number of characters in 'str' that appear in substring C[bold,i]. As the final result, the physical length of the string is calculated as 'WA', but the function returns value as 'ceil(WA*(1+1.76E-13*WA**1.5))/20' - Covert 'W_a' to 'W_c' as formula (2b), rounded up to an integer (unit: twip) then, and divided by 20 again to convert the 'twip' into unit 'pt'.

In the array element c[1,4]='A02C202E'x (or, same as c[2,4]) is with 4 chars. Char 'A0'x is 'No-Break Space' with ASCII code 160, or create it by function as code 'BYTE(160)' in another way. Char '2C'x is 'Comma' sign. Char '20'x is 'Space' sign with ASCII code 32 – but it's 'single isolated space' here ('>=2 consecutive space' has been replaced by character '• ' before). Char '2E'x is 'Dot' (or called full stop) sign.

Now, let's think about the probability of miscalculation on function width(). If a calculation to a string's physical width W_B (W_B is as `'WA*(1+5.03E-14*WA**1.5)'` in function width()) with absolute error is as: $W_B \pm \delta$ (unit as twip), and value W_B is in interval (I-δ,I+δ) -- value I is any integer. It's probably a miscalculation because we don't exactly know if $W_B <= I$, or $W_B > I$. When I increased the significant digit to 12 to each 'Characteristic Width' in width(), no any miscalculation to be found for any case of string.

CODING EXAMPLE

Now, we provide a SAS code as an example to show how to decide the column's width setting to a dataset while using PROC REPORT with ODS RTF. The code is as macro %CLNMSG in 'APPENDIX'. The input dataset is 'sashelp.CLNMSG', but only the first 4 observations are used/displayed, and with a suitable column width without word wrap up. We used ODS style as 'SasDocPrinter'. This is a ODS styles supplied by SAS. Running the code as below to get the feature of the style 'SasDocPrinter':

```
proc template;
    source Styles.SasDocPrinter;
run; quit;
```

The partial of output is as following:

```
'headingFont' = ("ITC Bookman, <MTserif>, Times Roman",11pt,bold)
'docFont' = ("ITC Bookman, <MTserif>, Times Roman",10pt);
frame = HSIDES
cellpadding = 4pt
borderspacing = 0.75pt
borderwidth = 0.75pt
```

Output: The piece of SAS output for code 'source Styles.SasDocPrinter;' on PROC TEMPLETE.

Then, we know:

(A) The table header's font is 'Times Roman', the size is 11pt, and weight is 'Bold'.
(B) The table content's font is 'Times Roman', the size is 10pt, and weight is 'Normal'.
(C) The cellpadding=4pt and borderspacing=0.75pt

Note: Function width() is fit to calculated the physical length of a string if there is no option 'asis=on' in 'define' statement on PROC REPORT. For the 'define' statement with option 'asis=on', change code piece 'cats(string)' as 'catt(string)' inside of function width().

To choose a suitable column width, we should consider the physical width for both table header and table contents. So 'PROC CONTENTS' is used for getting the variable's name and label in a dataset as variable 'NAME' and 'LABEL', following code in a 'PROC SQL' calculated the header's texts on physical width:

```
width(coalescec(LABEL,NAME),1,11)
```

Function 'coalescec()' is used in case of 'LABEL' is missing, use 'NAME' in place of the header. The second argument in width() is as '1' -- for 'Bold' font in the header part. The third argument in width() is as '11' -- for font size in 11pt. Then in the next step, while a variable' name is assigned to macro variable '&&VR&i' and the header's physical length has been assigned as '&&LBL&i', following code decided the column's width:

```
max(&&LBL&i,max(width(cat(&&VR&i),2,10)))+2*4-0.75*(&i>1)
```

Code 'width(cat(&&VR&i),2,10)' get each observation's physical length, while max() get the maximum value of all 4 observations in the 'PROC SQL'. Code 'max(&&LBL&i,max(width(cat(&&VR&i),2)))' get the maximum value for both 'table header' and 'table content' part. Then code piece '+2*4-0.75*(&i>1)' is for reason of formula (3a) and (3b) in this paper.

(7) Set Column Width by function width()

MSGID	MNEMONIC	LINENO	LEVEL	TEXT	PBUTTONS
14	IN_SEL_NOT_AVAIL	1	N	The selection %1$ is not available.	SASHELP.FSP.OK.SLIST
24	IN_NOT_AVAILABLE	1	N	This option is not yet available.	SASHELP.FSP.OK.SLIST
32	IO_DICT_CANNOT_OPEN_STUDY	1	E	Unable to open the study. There are problems with the dictionary.	SASHELP.FSP.OK.SLIST
82	IO_NO_ACCESS	1	E	%lYou do not have authorization to access %1$.	SASHELP.FSP.OK.SLIST

(8) Set Column Width by function width() and Minus 1 twip Again

MSGID	MNEMONIC	LINENO	LEVEL	TEXT	PBUTTONS
14	IN_SEL_NOT_AVAIL	1	N	The selection %1$ is not available.	SASHELP.FSP.OK.SLIST
24	IN_NOT_AVAILABLE	1	N	This option is not yet available.	SASHELP.FSP.OK.SLIST
32	IO_DICT_CANNOT_OPEN_STUDY	1	E	Unable to open the study. There are problems with the dictionary.	SASHELP.FSP.OK.SLIST
82	IO_NO_ACCESS	1	E	%lYou do not have authorization to access %1$.	SASHELP.FSP.OK.SLIST

Figure 7: Table (7) used function 'width()' and the formula (1) and (2) for column setting in PROC REPORT. Table (8) is minus only 1 twip to each column and get wrap up either on table contents, or on table header.

CONCLUSION

This paper presents some basic methods and rules to solve the physical length calculation for font 'Times Roman' on ODS RTF destination. And the same methods and rules could be used for the physical length calculation to the other fonts (such as font 'Arial'). In the same way, we could create another SAS custom function to calculate the physical length for ODS PDF destination for font 'Times Roman'.

ACKNOWLEDGMENTS

The author would like to thank the SAS user Lydia Kung and Jerry Lu for carefully reviewing the draft of this paper and providing their comments.

REFERENCES

Carey G. Smoak, (2004), "Creating Word Tables using PROC REPORT and ODS RTF", Proceeding of PharmaSUG 2004, Paper TT02. Available at

https://www.lexjansen.com/pharmasug/2004/TechnicalTechniques/TT02.pdf

Matthews, Carol. and Kalchenko, Elena. 2013. "Pretty Please?! Making RTF Output "Pretty" with SAS" Proceedings of the PharmaSUG 2013 Conference, Paper IB08. Available at

https://www.pharmasug.org/proceedings/2013/IB/PharmaSUG-2013-IB08.pdf

CONTACT INFORMATION

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

William Wu

Astex Pharmaceuticals, Inc.

Phone: (650)350-8604

Email: willywu2001@hotmail.com

Steven Li

Medtronic PLC., Mounds View, MN

Email: steven_li99@hotmail.com

APPENDIX

This 'APPENDIX' provided the code for custom function 'width()', and code in 6 macros. They are:

1. Macro %style generates various styles for using by the other macro on ODS RTF destination.
2. Macro %Class generated 8 different table styles into one RTF file as 'Class.rtf'. It's showed as Picture 1 and Picture 2 in the paper.
3. Macro %CellSpace verified the column's width setting based on the physical width of the texts, 'cellpadding', 'borderspacing' and so on. Four RTF files are generated as 'CellSpace.rtf'
4. Macro %Physical verified 95 different character's physical length on both 'Bold' and 'Normal' font. The SAS custom function width() is generated based on the data from this macro. Two RTF files are generated as 'Physical_<Bold|Normal>.rtf'
5. Macro %Shrink demonstrated the non-liner' shrink to the width setting (on 'style=[cellwidth=< W_x>pt]'). On 'Bold' font in size as 9pt, 139 equal sign '=' could be hold on the column width as W_x W_x=14257 twip without wrap up. But the double letters (278 equal sign '=') with double column width as W_x=28514 twip is with wrap up. The RTF files are generated as 'Shrink.rtf'
6. Macro %CLNMSG is an example code to show how to code for deciding the length setting for each column on PROC REPORT. The output file is 'CLNMSG.rtf'.
7. Macro %Round set width in 5 columns for file 'Round.rtf' as 1199.5, 1199.75, 1200, 1200.25, 1200.46 twip. But each column's actual width is rounded to1200 twip by 'ODS RTF'. People could verify it by open file 'Round.rtf' with 'Notepad' -- Find 'cellx1200', 'cellx2400', .. in the file.

```
%let path = <Your path for ODS RTF destination>;

%macro style(stylen=, fontsize=10, rules=2, boldn=2, margin=45
           , cellpadding=3, borderspacing=0.5, borderwidth=1);
%local x FT;
%let FT = Title Title Strong Emphasis FixedEmphasis FixedStrong FixedHeading BatchFixed
Fixed HeadingEmphasis Heading Doc;

proc template;
  define style Mystyle&stylen._&boldn /store=work.templat;
    parent=styles.Printer;
    replace fonts /%do x=1 %to 12; "%scan(&FT,&x)Font%scan(2,&x)"=
      ("Times Roman",%scan(&fontsize %eval(10+2*(&x=2)-(&x=9)),2-(&x=12))pt
        %if %index(2367110,&x)>0 | &boldn&x=112 %then, Bold;)
    %end;;
    replace color_list /'link'=blue 'bgH'=white 'fg'=black 'bg'=white;
    style body from body /leftmargin=&margin.pt rightmargin=&margin.pt topmargin=&margin.pt
        bottommargin=&margin.pt;
    style SystemFooter from SystemFooter /just=left protectspecialchars=on font=Fonts('FixedFont');
    style table from table /frame=hsides rules=%scan(all groups,&rules)
        cellpadding=&cellpadding.pt borderspacing=&borderspacing.pt borderwidth=&borderwidth.pt;
  end;
run;

%mend style;

proc fcmp outlib=work.funcs.maths;
  function width(string $, bold, fsize);
    array w[32] (
```

```
3.6035144396 4.0039049329 4.4042954262 4.9999983552 5.5566387971 6.6601540591 7.7832005647 7.8808567826
8.0175754876 8.1640598144 8.5351534424 8.8769502049 9.3847625379 9.5996062172 9.9999967105 10.400387204
10.820308941 11.103511972 11.123043216 11.279293165 11.396480626 11.621089927 12.216792856 13.339839362
14.443354624 15.556635508 16.660150770 17.783197275 18.417962691 18.603509505 18.876946915 19.999993421);
    array c[2,32] $30 (' ' ' ' '|' 'A02C202E'x ''''/\il' '!()-:;[]`fjt' 'Is' '}{'
        ' ' ' ' '□' 'cerz' ' ' ' ' '#$*0123456789?J_agovxy' '~'
        ' ' '"' 'Sbdhknpqu' ' ' '+<=>' '^' 'FP' 'BELTZ'
        'ACDNRUVXYw' 'GHKOQ' '&m' ' ' ' ' '@' 'M' '%W'
        ''''' '|' ' ' 'A02C202E'x '/:;\ijlt' '!()-I[]`fr' 'Js' ' '
        '□' '"' ' ' '?acez' '^' '}{' '#$*0123456789_bdghknopquvxy' ' '
        '~' ' ' 'FPS' '+<=>' ' ' ' ' 'ELTZ' 'BCR'
        'ADGHKNOQUVXYw' '&m' '%' 'M' '@' ' ' 'W' ' ');
    length string str $8800;
    str = tranwrd(tranwrd(cats(string),'  ','□□'),'□ ','□□');
    do i=1 to 32;
        if c[bold,i]>' ' then WD + countc(str,c[bold,i],'T')*w[i];
    end;
    WA = WD*fsize;
    return(ceil(WA*(1+5.117E-14*WA**1.492))/20);
  endfunc;
run; quit;

options cmplib=work.funcs mprint nodate nonumber nobyline varlenchk=nowarn noquotelenmax missing=' ';
ods path reset;
ods path (prepend) work.templat(update) sasuser.templat(read) sashelp.tmplmst(read);
ods escapechar='□';

%macro Class;

%local h i j B C D VAR;

%let VAR = Name Sex Age Height Weight;

%style(stylen=1, rules=1, margin=36, cellpadding=2, borderspacing=0.5, borderwidth=1.5);

options orientation=portrait papersize=Letter;
ods listing close;

%do h=1 %to 2;

ods rtf file="&path\Class_&h..rtf" style=Mystyle1_2 startpage=no bodytitle
    columns=%eval(1+&h) image_dpi=600;

%do i=1 %to 1+&h;

%let B = %scan(all groups,1+(&h&i=12));
%let C = %scan(0.5 0 1.5,1+(&h=2)*(1+(&i=2)),' ');
%let D = %scan(1.5 0,1+(&h&i=23),' ');

title h=10pt "(%eval(&i+2*(&h-1))) rules=&B, borderspacing=&C.pt, borderwidth=&D.pt";

proc report data=sashelp.Class(obs=5 keep=&VAR) nowd
            style=[rules=&B borderspacing=&C.pt borderwidth=&D.pt];
    column &VAR;
  %do j=1 %to 5;
    define %scan(&VAR,&j) /display style=[just=%scan(l c c d d,&j)
                                          cellwidth=%scan(36 19 21 33 37,&j)pt];
  %end;
run;

%end;

ods rtf close;

%end;

ods listing;

%mend Class;


%macro CellSpace;

%local j k L m;

data CellSpace;
    length Str1 $40;
    do n=1 to 2;
        Str1 = repeat(char('go',n),39);
        output;
    end;
```

```
run;

%style(stylen=6, rules=1, margin=18, cellpadding=3, borderspacing=1, borderwidth=1);

options orientation=portrait papersize=Tabloid;
ods listing close;
ods rtf file="&path\CellSpace.rtf" style=Mystyle6_2 startpage=no nobodytitle;

%do j=1 %to 2;
%do k=1 %to 2;
%do m=0 %to 1;

proc report data=CellSpace nowd style=[borderspacing=&j.pt borderwidth=&k.pt];
    column Str1 Str1=Str2 Str1=Str3;
  %do L=1 %to 3; %let WT = %sysevalf(40*100+20*(2*3-(&L>1)*&j)-&m);
    define Str&L /display style=[just=l cellwidth=%sysevalf(&WT/20)pt]
    "borderspacing=&j.pt borderwidth=&k.pt, m=&m, W=&WT.twip, %sysevalf(&WT/20)pt";
  %end;
run;

%end;
%end;
%end;

ods rtf close;
ods listing;

%mend CellSpace;


%macro Physical;

%local i j k m NUM u;

%let NUM = 2 3 4 5 8 10 15 20 30 50 75 100;
%let u   = %sysfunc(countw(&NUM));

options orientation=landscape papersize=Tabloid;
ods listing close;

%do i=1 %to 2;

%style(stylen=4, fontsize=10, rules=2, boldn=&i, margin=1, cellpadding=0, borderspacing=0, borderwidth=0);

ods rtf file="&path\Physical_%scan(Bold Normal,&i).rtf" style=Mystyle4_&i startpage=no bodytitle image_dpi=600;

%do j=1 %to 21+&i;

data Physical&i._&j;
    array C[2,23] $30 _temporary_('|' 'A02C202E'x '''/\il' '!()-:;[]`fjt' 'Is'
    '}{' '□' 'cerz' '#$*0123456789?J_agovxy' '~' '""
    'Sbdhknpqu' '+<=>' '^' 'FP' 'BELTZ' 'ACDNRUVXYw'
    'GHKOQ' '&m' '@' 'M' '%W' ' '
    '''' '|' 'A02C202E'x '/:;\ijlt' '!()-I[]`fr'
    'Js' '□' '"' '?acez' '^' '}{'
    '#$*0123456789_bdghknopquvxy' '~' 'FPS' '+<=>' 'ELTZ' 'BCR'
    'ADGHKNOQUVXYw' '&m' '%' 'M' '@' 'W');
    length char b $1;
    array M[&u] _temporary_(&NUM);
    array S[&u] $102 s1-s&u;
    retain b ' ';
    do h=1 to length(C[&i,&j]);
        char = char(C[&i,&j],h);
        do i=1 to &u;
            if &j=7 then S[i] = '.'||repeat(' ',M[i]-1)||'.';
            else if char=' ' then do;
                S[i] = repeat('.',mod(M[i]-1,2));
                if i>1 then S[i] = catt(S[i],repeat(char||'.',floor((M[i]-1)/2)-1));
            end;
            else if char='A0'x then do;
                if M[i]=2 then S[i] = '..';
                else S[i] = '.'||repeat(char,M[i]-3)||'.';
            end;
            else S[i] = repeat(char,M[i]-1);
            if h=1 then do;
                call symputx(cat("L&i._&j._",i),width(S[i],&i,10),'L');
                call symputx(cat('M',i),M[i],'L');
            end;
        end;
        output;
    end;
```

```
run;

%do m=0 %to 1;

title "%scan(Bold Normal,&i), Group #&j, m=&m";

proc report data=Physical&i._&j nowd;
    column %do k=1 %to &u; b s&k %end;;
    define b   /display style=[just=c cellwidth=5pt] ' ';
  %do k=1 %to &u; %let Len&i._&j._&k = %sysevalf(&&L&i._&j._&k-&m/20);
    define s&k /display style=[just=l cellwidth=&&Len&i._&j._&k..pt]
    "s&&M&k %scan(W=&&Len&i._&j._&k,1+%sysevalf(&&Len&i._&j._&k<800),|)";
  %end;
run;

%end; %*do m=0 %to 1;
%end; %*do j=1 %to 21+&i;

ods rtf close;

%end; %*do i=1 %to 2;

ods listing;

%mend Physical;

%macro Shrink;

%local i j LEN COL;

%let LEN = 14257;

data Shrink;
    length STR1 STR2 $278;
    STR1 = repeat('=',139-1);
    STR2 = repeat('=',139*2-1);
run;

options orientation=landscape center papersize=(22in 17in);

%style(stylen=7, fontsize=9, rules=2, boldn=1, margin=1, cellpadding=0, borderspacing=0, borderwidth=0);

ods listing close;
ods rtf file="&path\Shrink.rtf" style=Mystyle7_1 startpage=no bodytitle image_dpi=600;

title "Fontsize=9pt, Fontweight=Bold";

proc report data=Shrink nowd;
    column %do i=1 %to 2; STR&i STR&i=STR&i.A STR&i=STR&i.B STR&i=STR&i.C %end;;
  %do i=1 %to 2; %do j=1 %to 4; %let COL = %sysevalf(&LEN*&i+(&i=2)-%scan(0 0.5 0.54 1,&j,' '));
    define STR&i%scan(A B C,&j-5) /display style=[just=c cellwidth=%sysevalf(&COL/20)pt]
          "STR&i%scan(A B C,&j-5) L=%eval(139*&i) Width=&COL twip" %if &i&j>11 %then page;;
  %end; %end;
run;

ods rtf close;
ods listing;

%mend Shrink;

%macro CLNMSG;

%local i m Vn;

proc template;
    source Styles.SasDocPrinter;
run; quit;

data CLNMSG;
    set sashelp.CLNMSG(obs=4);
run;

proc contents data=CLNMSG out=Vars(keep=VARNUM NAME LABEL) noprint;
run;

options orientation=landscape papersize=Letter;
ods listing close;

ods rtf file="&path\CLNMSG.rtf" style=SasDocPrinter startpage=no bodytitle image_dpi=600;

proc sql noprint;
```

```
    select distinct VARNUM, NAME, NAME as NAME2, width(coalescec(LABEL,NAME),1,11)
      into :VN1-:VN99, :VAR separated by ' ', :VR1-:VR99, :LBL1-:LBL99
      from Vars;
%let Vn = &sqlobs;

    select %do i=1 %to &Vn; %if &i>1 %then,;
           max(&&LBL&i,max(width(cat(&&VR&i),2,10)))+2*4-0.75*(&i>1) %end;
      into %do i=1 %to &Vn; %if &i>1 %then,; :WD&i trimmed %end;
      from CLNMSG;
quit;

%do m=0 %to 1;

title "(%eval(7+&m)) Set Column Width by function width() %scan(and Minus 1 twip Again,2-&m,|)";

proc report data=CLNMSG nowd;
    column &VAR;
  %do i=1 %to &Vn;
    define &&VR&i/display style=[just=%scan(l c,1+%sysevalf(&&WD&i<54))
                                 cellwidth=%sysevalf(&&WD&i-&m*0.05)pt];
  %end;
run;

%end;

ods rtf close;
ods listing;

%mend CLNMSG;

%macro Round;

%local i;

data Round;
    array CL[5] COL1-COL5;
    do i=1 to 5;
        CL[i] = 1200+choosen(i,-0.5,-0.25,0,0.25,0.46);
        call symputx(cat('W',i),CL[i]/20,'L');
    end;
run;

options orientation=portrait papersize=Letter;
title "By Rounding - Each Column's Width: 1200 twip";

%style(stylen=7, fontsize=10, rules=2, boldn=2, margin=45, cellpadding=0, borderspacing=0, borderwidth=0);

ods listing close;
ods rtf file="&path\Round.rtf" style=Mystyle7_2 startpage=no bodytitle image_dpi=600;

proc report data=Round nowd;
    column COL1-COL5;
  %do i=1 %to 5;
    define COL&i /display style=[just=c cellwidth=&&W&i..pt];
  %end;
run;

ods rtf close;
ods listing;

%mend Round;

%Class;
%CellSpace;
%Physical;
%Shrink;
%CLNMSG;
%Round;

proc fcmp outlib=work.funcs.maths;
    deletefunc width;
run;
options cmplib=();
```