

# WORD-WRAPPING TEXT OUTPUT IN SAS® REPORTS

*S. David Riba, JADE Tech, Inc., Clearwater, FL*

## ABSTRACT

How do you get your SAS output that normally prints text values in a very long line to wrap in smaller lines at word breaks just like a word processor typically does?

The problem with printing long text values in SAS is that either

- a) your output gets truncated when the next field writes over it at a predefined column or
- b) you can split the values at a certain line length which may, or may not, be at a break between words

This dataset routine parses long text values into smaller output lines based on the output line length that has been defined. Instead of merely splitting the line, it defines the closest gap between words and breaks at that point.

## INTRODUCTION

There are many occasions where it would be useful to take long text strings and split them into smaller substrings, with the division occurring at word breaks. Unfortunately, while this has become a very common feature of virtually all word processors, the SAS System is incapable of performing this task without programmer intervention.

The following routine was developed for the report writer portion of an application. It simulates the word-wrap feature by splitting text at word breaks. It requires only one input parameter - the maximum length of the finished text string.

This routine can be enclosed within a macro and invoked as a macro call, or it can be used "as is" in a SAS DATA step. The routine can be included as part of a report writing DATA \_NULL\_ step, or it can be used to reformat the data that is stored in a SAS dataset.

## THE WORD-WRAP ROUTINE

For this example, the following only breaks a text string into four smaller strings (TXT1 to TXT4). Obviously, more sub-strings could have been defined, and this routine can be customized as needed.

The steps to word-wrap text strings are:

- Initialize sub-text and temporary text variables (TXT1 - TXT4 and T\_TXT)

```
length txt1 txt2 txt3 txt4 t_txt $ 80 ;
```

- Define the minimum (MIN) and maximum (MAX) number of characters per line.  
NOTE: MIN should always be 1.

```
min = 1 ;  
max = 65 ;
```

- Since MAX changes during execution, save the defined value of MAX in SAV\_MAX

```
sav_max = max ;
```

- Store length of original text string in LENTXT

```
lentxt = length(trim(text)) ;
```

- If actual text string is shorter than the maximum, store it in TXT1 and bypass further processing

```
if (lentxt lt (max+1)) then  
  txt1 = text ;
```

- If not, recursively process the text string until all of it has been stored

```
else  
  do until (max gt lentxt) ;
```

- Extract a substring from TEXT and store it in TMPTXT

```
  tmptxt = substr(text,min,max) ;
```

- Using the INDEX function, scan the string backwards until a word break (blank space) is detected. The variable RC will contain a value of 1 when a blank space has been found and the loop then stops.

The position of this word break is stored in i

```
rc = 0 ;
do i = sav_max to 1 by -1
  until (rc eq 1) ;
  rc=index(substr(t_txt,i),' ');
end ;
```

- Extract this portion of the text string and store it in the temporary variable T\_TXT.

```
t_txt = substr(t_txt,1,i) ;
```

- Assign it to the next non-empty storage string (TXT1 - TXT4)

```
if (length(trim(txt1)) le 1)
  then txt1 = t_txt ;
else if (length(trim(txt2)) le 1)
  then txt2 = t_txt ;
else if (length(trim(txt3)) le 1)
  then txt3 = t_txt ;
else
  txt4 = t_txt ;
```

- Modify the MIN and MAX variables.

MIN becomes the smaller of the next character after the word break or the current text length. MAX is incremented to the next value. If the routine has reached the end of the string, then MAX is set equal to one greater than the string length so the iterative routine will terminate.

```
min = min((min+i-1),lentxt) ;
if (max eq lentxt) then max + 1 ;
else
  max = min(sum(max,i),lentxt) ;
```

- Repeat the loop until the conditions have been satisfied

```
end ;
```

The complete routine (without explanatory text) looks like this:

```
length txt1 txt2 txt3 txt4 t_txt $ 80 ;
min = 1 ;
max = 65 ;
sav_max = max ;
lentxt = length(trim(text)) ;
if (lentxt lt (max+1)) then
  txt1 = text ;
else
  do until (max gt lentxt);
```

```
tmptxt = substr(text,min,max) ;
rc = 0 ;
do i = sav_max to 1 by -1
  until (rc eq 1) ;
  rc=index(substr(t_txt,i),' ');
end ;
t_txt = substr(t_txt,1,i) ;
if (length(trim(txt1)) le 1)
  then txt1 = t_txt ;
else if (length(trim(txt2)) le 1)
  then txt2 = t_txt ;
else if (length(trim(txt3)) le 1)
  then txt3 = t_txt ;
else
  txt4 = t_txt ;
min = min((min+i-1),lentxt) ;
if (max eq lentxt) then max + 1 ;
else
  max = min(sum(max,i),lentxt) ;
end ;
```

## CONCLUSION

With only minor modifications this routine can be adapted for most text parsing situations. The inner loop determines the point to break the text string, and the outer loop repeats the process as often as necessary to completely parse the text string.

## TRADEMARK INFORMATION

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

## AUTHOR

The author welcomes comments, suggestions, and questions by phone (813-726-6099) or by e-mail [JADETECH@MCIMAIL.COM](mailto:JADETECH@MCIMAIL.COM).

The author may be contacted at:

**S. David Riba**  
**JADE Tech, Inc.**  
 P O Box 4517  
 Clearwater, FL 33758  
 (813) 726-6099

INTERNET: [jadetechn@mcimail.com](mailto:jadetechn@mcimail.com)

