WORD PROCESSING USING SAS

Ross Z. Merlin, Pinkerton Computer Consultants, Inc.

Abstract
SAS provides many features that aid in the production of quality reports. There are times, however, when graphs, charts, tables, even "custom-tailored" reports cannot properly convey some aspect of a report; explanations of methods used and statements of conclusions, for example. One aspect of report generation that SAS does not provide is Word Processing. This paper describes a program that provides such a capability. Some of the features are: 

- Margin justification (lines of text expand to go completely to the margin), paragraph indentation, centering, automatic double-space after end of sentence or colon, "as is" (reproduce text as input without justification), page titles, page numbers, and others.
- The desired features may be specified through a TSO CLIST, thereby permitting one program to satisfy a variety of needs without modification (the program is also suitable for batch execution). Unlike some Word Processors, all processing is accomplished in a single pass over the data. The program can be modified to process text from other Word Processing systems.

The Need For A SAS WP Program
Text can be written to SAS output pages by using INPUT and PUT _INFILE_ statements, but there are several disadvantages to this approach. You must insert spaces to expand the text to the right margin; the page-width cannot be greater than the input record's length; text to be centered, such as section headings, must be centered manually. Once the text has been entered (into a dataset or punched on cards), even a single change usually means that every subsequent line will have to be modified to re-establish the correct spacing. Certainly the computer is better suited to this task than are we hunt-and-peck typists.

Program Considerations
The program has a few guidelines for how to process text. For the purposes of this paper, a "word" is any string of characters delimited by a space of a record boundary. The output lines normally extend from the left margin completely to the right margin, with one space between words, except that there must be two spaces following the end of a sentence (period, question mark, or exclamation point) or a colon. If there are extra spaces at the right end of the line (but the next word is too big to fit in the remaining spaces), then they are distributed between words as evenly as possible. Extra spaces in the raw text are ignored. Extra words from a line of raw text flow down to the next line. The last line of a paragraph is not justified, and indented lines do not start at the left margin.

Most sentences end with a period — unfortunately, so do abbreviations. When a period is encountered at the end of a word, the program must determine if it indicates end of sentence. To do this, the program contains an array with several common abbreviations. Each word ending with a period is compared to those in the array; if there is no match, then the period indicates a sentence so it will be followed by two spaces. Both the size of the array and its contents are easily modified.

Other guidelines vary with each job, such as page size, line size, page titles, and page numbers.

Execution of the Program
The program consists of a single SAS DATA step, with the variable "guidelines" brought into the program via %INCLUDE. The raw text resides in an OS dataset or member, or as in-stream data. The processed output may be written to an OS dataset or member, or directly to a printer. Execution may be in either batch or interactive mode; when processed interactively, a CLIST facilitates the generation of the SAS statements for the variable "guidelines", which may be MACROS, OPTIONS, and TITLE statements. The CLIST may also specify certain options on the SAS command.

Text Commands
By embedding certain characters in the text, you can inform the program that special processing is required (the command characters are not treated as words, hence they do not appear in the output). For example, $I indicates that the next word starts a new paragraph and is to be indented; $A indicates that the rest of the line is to be printed "as is" (without "word" processing -- this is particularly useful for columnar or tabular data); $C indicates that the rest of the line is to be printed without "word" processing, but centered on the line (useful for section headings). Other commands include $B -- blank line; $P -- page eject; and $T -- treat the next word as text, even if it is a "command" (this has very limited use). These particular command characters were chosen because the "$" reduces the likelihood that the
Command will naturally occur within the text, and the single letter is easy to associate with the intended function. Only a minor change to the program is required to use a different command set; this would be useful if you already have text with embedded commands intended for a different WP "language". Note: other minor changes to the program may be required, depending on the logic of the other WP processor.

Obviously, these six commands cannot provide every word processing feature, but they are adequate for most WP requirements, particularly those related to data processing. One of the design objectives for this program was to provide maximum flexibility with a minimum of "details", such as commands, parameters, and restrictions.

Conclusion

Surprisingly, SAS is well-suited to the task of Word Processing. The SAS language has many features that simplify the recognition and manipulation of words. The techniques used by this program for word processing have a variety of related applications. For example: COBOL programs could be checked for departures from FIPS or ANSI standards, or COBOL program libraries could be unloaded (using PROC SOURCE) and examined to produce a cross-reference of program names versus COPYLIB members.

For printing text within SAS jobs (such as between pages of procedure output), this program has an advantage over other WP products: it runs as a DATA step within the SAS job, allowing all processing to be done in one step. Whether used for reports or for administrative typing (such as letters, memoranda, or documentation), the program is easy to use and has excellent execution speed.

Figure A shows raw text and commands; Figure B shows the text after being processed by the program.

$C FIGURE A
$B $I This is sample text to illustrate the SAS Word Processor. In figure A, the lines look quite ragged. There might be just one space after a period. There might be only one word to a line. There might be several spaces between words. No problem, SAS can handle it. $I As mentioned in the paper, the period at the end of Mr. or Ms. etc. is not handled the same as a period for end of sentence. $B
$T Here is "as is" text: $B
$A year total
$A 1981 $ 195
$A 1982 $ 221

$B $I Many of the features are not demonstrated in this simple sample. As you can see, the raw text may be very sloppy. This is typical after a few changes have been made. Note: after this text was listed for Figure A, the "A" in the first line was changed to "B". $I As you may have guessed, the entire paper was prepared by the SAS WP program. $B

FIGURE B

This is sample text to illustrate the SAS Word Processor. In Figure A, the lines look quite ragged. There might be just one space after a period. There might be only one word to a line. There might be several spaces between words. No problem, SAS can handle it. As mentioned in the paper, the period at the end of Mr. or Ms. etc. is not handled the same as a period for end of sentence.

Here is "as is" text:

<table>
<thead>
<tr>
<th>year</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>$ 195</td>
</tr>
<tr>
<td>1982</td>
<td>$ 221</td>
</tr>
</tbody>
</table>

Many of the features are not demonstrated in this simple sample. As you can see, the raw text may be very sloppy. This is typical after a few changes have been made. Note: after this text was listed for Figure A, the "A" in the first line was changed to "B". As you may have guessed, the entire paper was prepared by the SAS WP program.