Although SAS currently has numerous built-in functions for editing and reformatting character data, it does, for the most part, lack functions that deal with these data as words rather than as individual characters. This paper will present two new built-in functions to meet that need. The first function removes extra blanks from a string, while enabling the user to preserve a sentence or phrase structure. This function differs from the COMPRESS function currently provided by SAS in that the user may leave behind an optional number of blanks (defaulting to one). The second function allows the user to remove or replace all occurrences of a word in a character string.

This paper will do the following:
1) describe how to use each function,
2) compare these functions with similar character functions already provided by SAS,
3) present a concrete example using both functions, and
4) describe the algorithms for both programs.

I. Description and Use

A. COMPBL

COMPBL removes the unneeded blanks between words in a character string. All occurrences of two or more consecutive blanks are translated into a single blank. COMPBL removes blanks from a string while preserving a sentence or phrase structure. COMPBL is called as a built-in function from SAS. The call must be in the following format:

\[
\text{DEST STRING} = \text{COMPBL(SRC STRING)};
\]

where, SRC STRING = the source string being compressed.

DEST STRING = the output buffer containing the compressed string.

The following example removes extra blanks from the variable string:

\[
\begin{align*}
\text{STRING} & = \text{'HEY DIDDLE DIDDLE'}; \\
\text{STRING} & = \text{COMPBL(STRING)};
\end{align*}
\]

STRING now has the value 'HEY DIDDLE DIDDLE'.

It is important to note that COMPBL will not adjust the length of an output variable. If the length of the output is not explicitly declared then SAS will allocate the number of bytes in the input string to the output string. The user may use the LENGTH statement before calling COMPBL to set output length. For example:

\[
\text{STRING} = \text{'125 E MAIN ST'};
\]

LENGTH ADDRESS $ 10;
ADDRESS = COMPBL(STRING);

will leave ADDRESS with the value '125 E MAIN'.

B. TRANWRD

TRANWRD is a built-in function to replace or remove every occurrence of a given word (or pattern of characters) within a character string. The call must be in the following format:

\[
\text{DEST=TRANWRD(SOURCE,TARGET,REPL)};
\]

where, SOURCE = the source string being translated.
TARGET = the string being searched for in SOURCE.
REPL = the string that replaces TARGET when it is found in SOURCE.
DEST = the output buffer containing the translated SOURCE.

The following example replaces 'MRS' and 'MISS' gender titles with 'MS'.

\[
\begin{align*}
\text{LENGTH NAME1-NAME3 $ 25;} \\
\text{ARRAY NAME NAME1-NAME3;} \\
\text{NAME1='MRS JOAN SMITH'}; \\
\text{NAME2='MR JOHN JONES'}; \\
\text{NAME3='MISS ALICE COPPER'}; \\
\text{DO OVER NAMES;} \\
\text{NAME=TRANWRD(NAME, 'MRS', 'MS');} \\
\text{NAME=TRANWRD(NAME, 'MISS', 'MS');} \\
\text{END;} \\
\text{The NAMES array would now have the values: 'MS JOAN SMITH' and 'MS ALICE COPPER'}. \\
\end{align*}
\]

TRANWRD will, by default, allocate the destination string with the same length as the source string. The user may override this by explicitly defining the destination string length before calling TRANWRD.

The target string being searched for in the source string and the replacement string that replaces the target will include trailing blanks. For example, the following code would leave the variable STRING unchanged:

\[
\begin{align*}
\text{LENGTH TARGET $ 10;} \\
\text{LENGTH REPL $ 3;} \\
\text{STRING 'CATFISH'}; \\
\text{STRING = TRANWRD(STRING,TARGET,REPL)};
\end{align*}
\]

The value of TARGET is 'FISHING', a pattern not found in STRING. The user may use the TRIM function when (s)he wishes a target or replacement variable to exclude trailing blanks. If the last statement in the above example were replaced by:
STRING = TRANWRD(STRING, TRIM(TARGET), REPL);

then STRING would have the value 'CATNIP'.

TRANWRD may be used to remove a word or pattern from a character string by specifying a null replacement string. For example:

STRING = 'VERY RARE BRANDY';
STRING = TRANWRD(STRING, 'VERY', '');

would leave STRING with the value 'RARE BRANDY'.

II. Comparison to Existing Functions

SAS currently provides a wide range of functions for reformatting and editing character strings. Two of these functions (TRANSLATE and COMPRESS) are very similar to the TRANWRD and COMPBL functions described above.

A. TRANSLATE

TRANSLATE will convert every occurrence of a user supplied character to another character. TRANSLATE has the ability to scan for more than one character in a single call. When doing this, however, TRANSLATE is searching for every occurrence of a number of individual characters within a string. That is, if any letter (or character) in the target string is found in the source string, it will be replaced with the corresponding letter (or character) in the replacement string.

The TRANWRD function described above differs from TRANSLATE in that it will scan for words (or patterns of characters) and will replace those words when found with a second word (or pattern of characters).

B. COMPRESS

The COMPRESS function currently provided by SAS will remove every occurrence of any character from a string. The user may supply a blank as the character to be removed from the source string, making COMPRESS similar to COMPBL. The only difference is that COMPBL will leave a single blank between words while COMPRESS will not. COMPRESS can remove blanks from a string, but, in doing so, will destroy the sentence or phrase structure.

III. Standardizing Street Addresses: An Example

The following example converts common spellings of address components to standard equivalents. This example uses both COMPBL and TRANWRD.

DATA ONE;
/* THIS STEP READS IN A FILE OF */
/* ADDRESSES */
INPUT ADDRESS $ 1-35;
CARDS;
.
Addresses to Standardize
.
PROC PRINT;
TITLE UNSTANDARDIZED ADDRESSES;
DATA TWO; SET ONE;
/* DECLARE AND INITIALIZE TABLES OF */
/* NON STANDARD AND STANDARD ADDRESS */
/* ABBREVIATIONS */
LENGTH ABBR1-ABBR19 $ 15;
LENGTH STABBR1-STABBR19 $ 2;
ARRAY NONSTAND(X) ABBR1-ABBR19;
ARRAY STANDARD(X) STABBR1-STABBR19;
RETAIN ABBR1-ABBR19 STABBR1-STABBR19 X;
/* ON FIRST PASS INITIALIZE ARRAYS */
/* OF NON STANDARD AND STANDARD */
/* ABBREVIATIONS */
IF _N_ = 1 THEN
DO OVER NONSTAND;
INPUT NONSTAND $ 1-15
STANDARD $ 16-17;
FILE CONVTBL NOTES;
PUT @50 NONSTAND @70 STANDARD;
END;
/* REMOVE EXTRA BLANKS FROM THE */
/* ADDRESS FIELD */
IF ADDRESS = COMPBL(ADDRESS),
/* REPLACE NONSTANDARD ABBREVIATIONS */
/* WITH THEIR STANDARD EQUIVALENTS */
DO OVER NONSTAND;
ADDRESS = TRANWRD(ADDRESS,
TRIM(NONSTAND) ,
STANDARD) ;
END;
RETURN;
NEWPAGE;
PUT @54 'CONVERSION TABLE' II
@50 'BEFORE' @70 'AFTER' II;
RETURN;
CARDS;
.
Conversion Table
.
PROC PRINT;
TITLE STANDARDIZED ADDRESSES;
END OF DATA QED

The output from the above program is listed below:

CONVERSION TABLE

<table>
<thead>
<tr>
<th>BEFORE</th>
<th>AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO BOX</td>
<td>BX</td>
</tr>
<tr>
<td>P O BOX</td>
<td>BX</td>
</tr>
<tr>
<td>POST BOX</td>
<td>BX</td>
</tr>
<tr>
<td>BOX</td>
<td>BX</td>
</tr>
<tr>
<td>APT</td>
<td>AP</td>
</tr>
<tr>
<td>APARTMENT</td>
<td>AP</td>
</tr>
<tr>
<td>AVENUE</td>
<td>AV</td>
</tr>
<tr>
<td>AVENUE</td>
<td>AV</td>
</tr>
<tr>
<td>AVEN</td>
<td>AV</td>
</tr>
</tbody>
</table>
CONVERSION TABLE (continued)

BEFORE   AFTER
AVN       AV
AVE       AV
EAST      E
WEST      W
NORTH     N
SOUTH     S

UNSTANDARDIZED ADDRESSES

QBS   ADDRESS
1  BOX 742
2  F O BOX 25622
3  734 EAST GRAND AVE
4  521 BARTON ST APARTMENT
5  BOX 2797
6  708 EAST COMMERCE
7  521 N 18TH
8  RT3 BOX 162
9  2715 W MARKUM
10 3308 FRANKLIN AVNUE
11 119 N WARD
12 11310 HARRY HILL AVENUE
13 1513 NW 25TH ST
14 POST OFFICE BOX 2713
15 PO BOX 4

STANDARDIZED ADDRESSES

QBS   ADDRESS
1  EX 742
2  EX 25622
3  734 E GRAND AV
4  521 BARTON ST AP 25
5  EX 2797
6  708 E COMMERCE
7  521 N 18TH
8  RT3 BX 162
9  2715 W MARKUM
10 3308 FRANKLIN AV
11 119 N WARD
12 11310 HARRY HILL AV
13 1513 NW 25TH ST
14  EX 2713
15  EX 4

IV. Algorithms

COMPBL and TRANWRD are written in IBM 360/370 Assembler. The code has been donated to the SAS Institute and should be a part of the next release of SAS.

A. COMPBL

The following is the algorithm for COMPBL:

1. Save calling program registers using OS conventions.
2. Obtain the address and length of the source string.
3. Store the address of the beginning and the end of the source string and of the beginning of the destination string.
4. Scan the source string one character at a time looking for blanks.
   a. If a blank is found then check the next character in the source string.
      1) If the next character is also blank do not move this character.
      2) If the next character is not blank then move this character and add one to the address of the destination string.
   b. If this character is not blank then move this character and add one to the address of the destination string.
5. After the source string is scanned, subtract the address of the end of the destination string from the address of the beginning of the destination string. Store the difference in the high order byte of the address pointed to by the second parameter.
6. Restore registers and return control to SAS using OS conventions.

B. TRANWRD

The following is the algorithm for TRANWRD:

1. Save calling programs registers using OS conventions.
2. Read four words from the parameter list. The words represent:
   a. Address of address of source
   b. Address of address of target
   c. Address of address of replacement
   d. Address of address of destination
3. Store the length of the source, target, and replacement strings.
4. Store the address of the end of the source string minus the length of the target string as the address of the last byte to scan.
5. Compare the length of the source string and the target string.
   a. If source length is less than target length then copy the source to destination and return.
   b. If source length is greater than or equal to target then continue.
6. Scan the source string for the target string, moving through the source string one character at a time.
   a. If target string is found then:
      1) Move replacement into destination.
      2) Increase address of source.
by the length of the target string.
3) Increase address of destination by the length of the replacement string.
4) Continue scanning source.
b. If target string is not found, then:
   1) Move one byte of source into destination.
   2) Increase address of source and destination by one.
   3) Continue scanning source.

7. When the last byte to be scanned is reached (address of end of source minus length of target) then move remaining unscanned characters into destination.
8. Store the length of the destination string.
9. Restore registers using OS conventions and return control to SAS.