Creating Character Collating Sequences Using the TRANSLATE Function
Gary L. Katsanis, Blue Cross and Blue Shield of the Rochester Area, Rochester, NY

Sometimes you will run into situations where you want to arrange observations based on one or more character ID variables, but where you don’t want those observations arranged according to the character collating sequence. For example, you may have a combination of upper case and lower case letters in the range from a to f, and you may want to group the IDs in the following order: a, A, b, B, c, C, d, D, e, E, f, F. With the default character collating sequences used by the SAS® System under PC-DOS, observations sorted by this ID variable would be in the order: a, b, c, d, e, f, A, B, C, D, E, F.

In some cases, your best approach may be to create a new character variable based on the old variables, where sorting by the new variable would put the observations in the order you want. For simple situations, the TRANSLATE function can be used one or more times to do this. TRANSLATE takes three arguments: the variable you want to change, followed by a string consisting of the characters you want to change to, then a string of the characters you want to change from. For example:

```
NEWKEY = TRANSLATE(KEY, '#*&', '123');
```

will change "I" to ",", "2" to "*", and "3" to ",&".

To build a custom collating variable using TRANSLATE, you start by building a "from" string, which lists the values you already have in the order in which you want them to appear. For our simple example, you would use: "aAbCcCdDeEfF". Note that this is twelve characters. Next, you set up a "to" string consisting of the characters you want to change to, then a string of the characters you want to change from. For example:

```
NEWKEY = TRANSLATE(KEY, 'ABCDEFGHIJKL', 'aAbBcCdDeEfF');
```

You can use this technique to change the sorted order of a printed file as follows.

```
DATA NEWDATA ;
SET MYDATA ;
NEWKEY = TRANSLATE(KEY, "ABCDEFGHIJKLMNOPQRSTUVWXYZ", "aAbBcCdDeEfF");
RUN ;
```

```
PROC SORT DATA=NEWDATA ;
BY NEWKEY ;
RUN ;
```

```
PROC PRINT DATA=NEWDATA ;
* note that you need to use the NOTSORTED option if you want to use your original ID variable in a BY statement ;
BY KEY NOTSORTED ;
RUN ;
```

This technique can be significantly extended. You may be in a situation where you have more than one ID variable, or you may want to use a different ordering sequence for different parts of a single variable. In this example, there are two ID variables, and you want to arrange observations according to the following logic.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Column</th>
<th>Value</th>
<th>Desired Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY2</td>
<td>1</td>
<td>,</td>
<td>,I</td>
</tr>
<tr>
<td>KEY2</td>
<td>2</td>
<td>A-F</td>
<td>A,C,E,B,D,F</td>
</tr>
<tr>
<td>KEY1</td>
<td>1</td>
<td>A-F</td>
<td>F,E,D,C,B,A</td>
</tr>
</tbody>
</table>

To deal with this, you can easily build a three-column ID variable as follows.

```
DATA NEWDATA ;
SET MYDATA ;
KEY2 = TRANSLATE(KEY2, "A-F", "1,2,3");
NEWKEY = TRANSLATE(KEY1, "F,E,D,C,B,A", "aAbBcCdDeEfF");
RUN ;
```

```
PROC SORT DATA=NEWDATA ;
BY KEY NOTSORTED ;
RUN ;
```

```
PROC PRINT DATA=NEWDATA ;
* note that you need to use the NOTSORTED option if you want to use your original ID variable in a BY statement ;
BY KEY NOTSORTED ;
RUN ;
```

Note that you always use a character string right from the collating sequence as the "to" argument, and you always use the characters in the order you want them to appear as the "from" argument. The result of using this technique is a single character ID variable with no overlapping ranges.

1 Collating sequences are different for ASCII and EBCDIC. The sequences appear in the SAS Procedures Guide, under Chapter 31: The SORT Procedure. Note that any twelve characters in order from the collating sequence will work. For example, "1234567890" would work just as well. To avoid confusion, I try to avoid using symbols.
DATA NEWDATA;
  LENGTH NEWKEY $3;
  SET MYDATA;
  * deal with the first column of KEY2;
  SUBSTR(NEWKEY,1,1) = TRANSLATE(SUBSTR(KEY2,1,1),"AB","1");
  * deal with KEY1;
  SUBSTR(NEWKEY,2,1) = TRANSLATE(KEY1,"ABCDEF","FEDCBA");
  * deal with the second column of KEY2;
  SUBSTR(NEWKEY,3,1) = TRANSLATE(SUBSTR(KEY2,2,1),"ABCDEF","ACEBDF");
RUN;

In more complex situations, you may have to extend this technique by using additional tools such as the SCAN function, or a SAS format to create a new ID variable.

When used with care, this technique can be very helpful in data manipulation and data management under the SAS System.