Overview

The TRANTAB procedure creates, edits, and displays customized translation tables. In addition, you can use PROC TRANTAB to view and modify translation tables that are supplied by SAS Institute. These Institute-supplied tables are stored in the SASHELP.HOST catalog. Any translation table that you create or customize is stored in your SASUSER.PROFILE catalog. Translation tables have an entry type of TRANTAB.

Translation tables are operating environment-specific SAS catalog entries that are used to translate the values of one (coded) character set to another. A translation table has two halves: table one provides a translation, such as ASCII to EBCDIC; table two provides the inverse (or reverse) translation, such as EBCDIC to ASCII. Each half of a translation table is an array of 256 two-digit positions, each of which contains a one-byte unsigned number that corresponds to a coded character.

The SAS System uses translation tables for the following purposes:

- determining the collating sequence in the SORT procedure
Concepts

Understanding Translation Tables and Character Sets

The kth element in a translation table corresponds to the kth element of an ordered character set. For example, position 00 (which is byte 1) in a translation table contains a coded value that corresponds to the first element of the ordered character set. To determine the position of a character in your operating environment’s character set, use the SAS function RANK. The following example shows how to use RANK:

```sas
data _null_
  x=rank(’a’);
  put "The position of a is " x ".";
```

The SAS log prints the following message: **The position of a is 97**.

Each position in a translation table contains a hexadecimal number that is within the range of 0 (’00’x) to 255 (’FF’x). Hexadecimal values always end with an x. You can represent one or more consecutive hexadecimal values within quotation marks followed by a single x. For example, a string of three consecutive hexadecimal values can be written as ’08090A’x. The SAS log displays each row of a translation table as 16 hexadecimal values enclosed in quotes followed by an x. The SAS log also lists reference numbers in the vertical and horizontal margins that correspond to the positions in the table. Example 1 on page 1312 shows how the SAS log displays a translation table.

Storing Translation Tables

When you use PROC TRANTAB to create a customized translation table, the procedure automatically stores the table in your SASUSER.PROFILE catalog. This enables you to use customized translation tables without affecting other users. When you specify the translation table in the SORT procedure or in a GOPTIONS statement, the software first looks in your SASUSER.PROFILE catalog to find the table. If the specified translation table is not in your SASUSER.PROFILE catalog, the software looks in the SASHELP.HOST catalog.

If you want the translation table you create to be globally accessed, have your SAS Installation Coordinator copy the table from your SASUSER.PROFILE catalog (using the CATALOG procedure) to the SASHELP.HOST catalog.
Modifying Institute-supplied Translation Tables

If an Institute-supplied translation table does not meet your needs, you can use PROC TRANTAB to edit it and create a new table. That is, you can issue the PROC TRANTAB statement that specifies the Institute-supplied table, edit the table, and then save the table using the SAVE statement. The modified translation table is saved in your SASUSER.PROFILE catalog. If you are a SAS Installation Coordinator, you can modify a translation table with PROC TRANTAB and then use the CATALOG procedure to copy the modified table from your SASUSER.PROFILE catalog to the SASHHELP.HOST catalog, as shown in the following example:

```sas
proc catalog c=sasuser.profile;
   copy out=sashelp.host entrytype=trantab;
run;
```

You can use PROC TRANTAB to modify translation tables stored in the SASHHELP.HOST catalog only if you have update (or write) access to that data library and catalog.

Using Translation Tables Outside PROC TRANTAB

Using Translation Tables in the SORT Procedure

PROC SORT uses translation tables to determine the collating sequence to be used by the sort. You can specify an alternative translation table with the SORTSEQ= option of PROC SORT. For example, if your operating environment sorts with the EBCDIC sequence by default, and you want to sort with the ASCII sequence, you can issue the following statement to specify the ASCII translation table:

```sas
proc sort sortseq=ascii;
```

You can also create a customized translation table with PROC TRANTAB and specify the new table with PROC SORT. This is useful when you want to specify sorting sequences for languages other than U.S. English. The sample program TRABASE from the SAS Sample Library builds transport-format and character-operations translation tables for a number of languages and environments. The NLSSetup Application, shipped with Release 6.11 as part of the SAS Sample Library, provides an easy way to create all the necessary translation tables, device maps, and key maps simply by selecting a country name from a listbox.

See Example 6 on page 1321 for an example that uses translation tables to sort data in different ways. For information on the tables available for sorting and the SORTSEQ= option, see Chapter 32, “The SORT Procedure,” on page 1015.

Using Translation Tables with the CPORT and CIMPORT Procedures

The CPORT and CIMPORT procedures use translation tables to translate characters in catalog entries that you export from one operating environment and import on another operating environment. You may specify the name of an Institute-supplied or a customized translation table in the TRANTAB statement of PROC CPORT. See “TRANTAB Statement” on page 321 in Chapter 13, “The CPORT Procedure,” on page 313 for more information.
Using Translation Tables with Remote Library Services

Remote Library Services (RLS) use translation tables to translate characters when you access remote data. SAS/CONNECT and SAS/SHARE software use translation tables to translate characters when you transfer or share files between two operating environments that use different encoding standards.

Using Translation Tables in SAS/GRAPH Software

In SAS/GRAPH software, translation tables are most commonly used on an IBM operating environment where tables are necessary because graphics commands must leave IBM operating environments in EBCDIC representation but must reach asynchronous graphics devices in ASCII representation. Specifically, SAS/GRAPH software builds the command stream for these devices internally in ASCII representation but must convert the commands to EBCDIC representation before they can be given to the communications software for transmission to the device. SAS/GRAPH software uses a translation table internally to make the initial conversion from ASCII to EBCDIC. The communications software then translates the command stream back to ASCII representation before it reaches the graphics device.

Translation tables are operating environment-specific. In most cases, you can simply use the default translation table, SASGTAB0, or one of the Institute-supplied graphics translation tables. However, if these tables are not able to do all of the translation correctly, you can create your own translation table with PROC TRANTAB. The SASGTAB0 table may fail to do the translation correctly when it encounters characters from languages other than U.S. English.

To specify an alternative translation table for SAS/GRAPH software, you can either use the TRANTAB= option in a GOPTIONS statement or modify the TRANTAB device parameter in the device entry. For example, the following GOPTIONS statement specifies the GTABTCAM graphics translation table:

```plaintext
goptions trantab=gtabtcam;
```

Translation tables used in SAS/GRAPH software perform both device-to-operating environment translation and operating environment-to-device translation. Therefore, a translation table is made up of 512 bytes, with the first 256 bytes used to perform device-to-operating environment translation (ASCII to EBCDIC on IBM mainframes) and the second 256 bytes used to perform operating environment-to-device translation (EBCDIC to ASCII on IBM mainframes). For PROC TRANTAB, the area of a translation table for device-to-operating environment translation is considered to be table one, and the area for operating environment-to-device translation is considered to be table two. See Example 1 on page 1312 for a listing of the ASCII translation table (an Institute-provided translation table), which shows both areas of the table.

On operating environments other than IBM mainframes, translation tables can be used to translate specific characters in the data stream that are created by the driver. For example, if the driver normally generates a vertical bar in the data stream, but you want another character to be generated in place of the vertical bar, you can create a translation table that translates the vertical bar to an alternate character.


SAS/GRAPH software also uses key maps and device maps to map codes generated by the keyboard to specified characters and to map character codes to codes required by the graphics output device. These maps are specific to SAS/GRAPH software and are discussed in “The GKEYMAP Procedure” in SAS/GRAPH Software: Reference.
The TRANTAB Procedure

PROC TRANTAB Statement

Tip: Supports RUN-group processing

PROC TRANTAB TABLE=table-name <NLS>;
  CLEAR <ONE|TWO|BOTH>;
  INVERSE;
  LIST <ONE|TWO|BOTH>;
  LOAD TABLE=table-name <NLS>;
  REPLACE position value-1<…value-n>;
  SAVE <TABLE=table-name> <ONE|TWO|BOTH>;
  SWAP;

To do this | Use this statement
--- | ---
Set all positions in the translation table to zero | CLEAR
Create an inverse of table one | INVERSE
Display a translation table in hexadecimal representation | LIST
Load a translation table into memory for editing | LOAD
Replace the characters in a translation table with specified values | REPLACE
Save the translation table in your SASUSER.PROFILE catalog | SAVE
Exchange table one with table two | SWAP

Note: PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or submit another DATA or PROC statement.

PROC TRANTAB Statement

Tip: If you specify an incorrect table name in the PROC TRANTAB statement, use the LOAD statement to load the correct table. You do not need to reinvoke PROC TRANTAB. New tables are not stored in the catalog until you issue the SAVE statement, so you will not have unwanted tables in your catalog.
PROC TRANTAB TABLE=table-name <NLS>

Required Arguments

TABLE=table-name

specifies the translation table to create, edit, or display. The specified table name must be a valid one-level SAS name.

Options

NLS

specifies that the table you listed in the TABLE= argument is one of five special internal translation tables provided with every copy of the SAS System. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

SASXPT

the local-to-transport format translation table (used by the CPORT procedure)

SASLCL

the transport-to-local format translation table (used by the CIMPORT procedure)

SASUCS

the lowercase-to-uppercase translation table (used by the UPCASE function)

SASLCS

the uppercase-to-lowercase translation table (used by the LOWCASE macro)

SASCCL

the character classification table (used internally), which contains flag bytes that correspond to each character position that indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to translate characters that exist in languages other than English. To make SAS use the modified NLS table, specify its name in the SAS system option TRANTAB=.

Note: When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all zeros. PROC TRANTAB does not use table 2 at all for translation in these special cases, so you do not need to be concerned about this note.
CLEAR Statement

Sets all positions in the translation table to zero. This statement is useful when you create a new table.

```
CLEAR <ONE| TWO| BOTH>
```

Options

```
ONE | TWO | BOTH

ONE
clears table one.

TWO
clears table two.

BOTH
clears both table one and table two.
```

Default: \texttt{ONE}

INVERSE Statement

Creates an inverse of table one in a translation table. That is, it creates table two.

Featured in: Example 5 on page 1319

```
INVERSE;
```

Details

INVERSE does not preserve multiple translations. Suppose table one has two (or more) different characters translated to the same value; for example, "A" and "B" are both translated to "1". For table two, INVERSE uses the last translated character for the value; that is, "1" is always translated to "B" and not "A", assuming that "A" appears before "B" in the first table.

Operating environment sort programs in the SAS System require an inverse table for proper operation.

LIST Statement

Displays a translation table in hexadecimal representation. The translation table listing appears in the SAS log.

Featured in: All examples
LIST <ONE|TWO|BOTH>,

Options

ONE | TWO | BOTH

ONE
displays table one.
TWO
displays table two.
BOTH
displays both table one and table two.
Default: ONE

LOAD Statement

Loads a translation table into memory for editing.
Tip: Use LOAD when you specify an incorrect table name in the PROC TRANTAB statement. You can specify the correct name without the need to reinvoke the procedure.
Tip: Use LOAD to edit multiple translation tables in a single PROC TRANTAB step. (Be sure to save the first table before you load another one.)
Featured in: Example 4 on page 1317

LOAD TABLE=table-name <NLS>;

Required Arguments

TABLE=table-name
specifies the name of an existing translation table to be edited. The specified table name must be a valid one-level SAS name.

Option

NLS
specifies that the table you listed in the TABLE= argument is one of five special internal translation tables provided with every copy of the SAS System. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

SASXPT
the local-to-transport format translation table
SASLCL
the transport-to-local format translation table

SASUCS
the lowercase-to-uppercase translation table

SASLCS
the uppercase-to-lowercase translation table

SASCCL
the character classification table, which contains flag bytes that correspond to each character position that indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to map characters that exist in languages other than English to programs, displays, files, or products of the SAS System.

Note: When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all zeros. PROC TRANTAB does not use table 2 at all for translation in these special cases, so you do not need to be concerned about this note.

---

**REPLACE Statement**

Replaces characters in a translation table with the values given, starting at the specified position.

**Alias:** REP

**Tip:** To save edits, you must issue the SAVE statement.

**Featured in:** Example 2 on page 1313, Example 3 on page 1315, and Example 4 on page 1317

**REPLACE** position value-1<...value-n>;

**Required Arguments**

**position**

specifies the position in a translation table where the replacement is to begin. The editable positions in a translation table begin at position decimal 0 and end at decimal 255. To specify the position, you can

- use a decimal or hexadecimal value to specify an actual location. If you specify a decimal value, for example, 20, PROC TRANTAB locates position 20 in the table, which is byte 21. If you specify a hexadecimal value, for example, '14'x, PROC TRANTAB locates the decimal position that is equivalent to the specified hexadecimal value, which in this case is position 20 (or byte 21) in the table.
- use a quoted character. PROC TRANTAB locates the quoted character in the table (that is, the quoted character's hexadecimal value) and uses that character's position as the starting position. For example, if you specify the
following REPLACE statement, the statement replaces the first occurrence of the hexadecimal value for "a" and the next two hexadecimal values with the hexadecimal equivalent of "ABC":

replace 'a' 'ABC';

This is useful when you want to locate alphabetic and numerical characters when you do not know their actual location. If the quoted character is not found, PROC TRANTAB displays an error message and ignores the statement.
To edit positions 256 through 511 (table two), follow this procedure:
1. Issue the SWAP statement.
2. Issue the appropriate REPLACE statement.
3. Issue the SWAP statement again to reposition the table.

\[
\text{value}_1 < \ldots \text{value}_n
\]

is one or more decimal, hexadecimal, or character constants that give the actual value to be put into the table, starting at position. You can also use a mixture of the types of values. That is, you can specify a decimal, a hexadecimal, and a character value in one REPLACE statement. Example 3 on page 1315 shows a mixture of all three types of values in the REPLACE statement.

---

**SAVE Statement**

Saves the translation table in your SASUSER.PROFILE catalog.

**Featured in:** Example 2 on page 1313 and Example 4 on page 1317

**Syntax**

```
SAVE <TABLE=table-name> <ONE|TWO|BOTH>;
```

**Options**

**TABLE=table-name**

specifies the table name under which the current table is to be saved. The name must be a valid one-level SAS name.

**Default:** If you omit the TABLE= option, the current table is saved under the name you specify in the PROC TRANTAB statement or the LOAD statement.

**ONE | TWO | BOTH**

**ONE**

saves table one.

**TWO**

saves table two.

**BOTH**

saves both table one and table two.

**Default:** BOTH

---

**SWAP Statement**

Exchanges table one with table two to enable you to edit positions 256 through 511.

**Tip:** After you edit the table, you must issue SWAP again to reposition the table.

**Featured in:** Example 7 on page 1323
Examples

Note: All examples were produced in the UNIX environment.

Example 1: Viewing a Translation Table

Procedure features:
LIST statement

This example uses PROC TRANTAB to display the Institute-supplied ASCII translation table.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;

list both;
```

SAS Log
Example 2: Creating a Translation Table

Procedures features:
- LIST statement
- REPLACE statement
- SAVE statement

This example uses PROC TRANTAB to create a customized translation table.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=newtable;
```
The REPLACE statement places the values into the table starting at position 0. You can use hexadecimal strings of any length in the REPLACE statement. This example uses strings of length 16 to match the way translation tables appear in the SAS log.

```sas
replace 0
'00010203a309e57ff9ecc40b0c0d0e0f'
x
'1011213a5e008e71819c6c51c1d1e1f'
x
'c7fce9e2e40a171beaebe8efee050607'
x
'c9e616f4f6f2fb04ff6dca2b6a7501a'
x
'20e1edf3fafa1d1aababfa22e3c282b7c'
x
'265facdbca1abb5f5f21242a293bac'
x
'2d2f5fa6a6a62b2ba6a62c255f3e3f'
x
'a62b2b2b2b2b2b2d2d603a2340273d22'
x
'2b6162636465666768692da6a62b2b'
x
'2d6a6b6c6d6e6f7071722da6d2d2d2d'
x
'd2e7737475767778787a2d2b2b2b2b2b'
x
'2b2b2b5f5fa65f5f5f5f6f6f5f5f5f5f5f'
x
'7b4a4b4c4e4f5051525f5f5f5f5f5f5f5f'
x
'7d4a4b4c4d4e4f5051525f5f5f5f5f5f5f'
x
'5c83535455555758595a5f5f755f5f5b0'
x
'30313233343536373839b75f6eb25f5f'
x
;
```

The SAVE statement saves the table under the name specified in the PROC TRANTAB statement. By default, the table is saved in your SASUSER.PROFILE catalog.

```sas
save;
```

The LIST statement specifies that both table one and table two be displayed.

```sas
list both;
```

**SAS Log**

Table 2 is empty; that is, it consists entirely of 0s. To create table 2, you can use the INVERSE statement. (See Example 5 on page 1319.) To edit table 2, you can use the SWAP statement with the REPLACE statement. (See Example 7 on page 1323.)
The TRANTAB Procedure
Example 3: Editing by Specifying a Decimal Value for Starting Position

procedure features:
LIST statement
REPLACE statement
SAVE statement

Example 3: Editing by Specifying a Decimal Value for Starting Position

This example edits the translation table created in Example 2 on page 1313. The decimal value specified in the REPLACE statement marks the starting position for the changes to the table.

The vertical arrow in both SAS logs marks the point at which the changes begin.
Program

options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=newtable;

The first LIST statement displays the original NEWTABLE translation table.
list one;

SAS Log

The Original NEWTABLE Translation Table

<table>
<thead>
<tr>
<th>Table</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWTABLE table 1:</td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>'00010203A309E57FF9ECC40B0C0D0E0F'x</td>
</tr>
<tr>
<td>10</td>
<td>'10111213A5E008E718199C6C51C1D1E1F'x</td>
</tr>
<tr>
<td>20</td>
<td>'C7FCE9E2E40A171BEAEBEEFEE050607'x</td>
</tr>
<tr>
<td>30</td>
<td>'C9E616F4F6F2FB04FFDF6DCA2B6A7501A'x</td>
</tr>
<tr>
<td>40</td>
<td>'20E1EDF3FAF1D1AABAFA22E3C282B7C'x</td>
</tr>
<tr>
<td>50</td>
<td>'265FACBDBCA1ABB5F5F21242A293BAC'x</td>
</tr>
<tr>
<td>80</td>
<td>'2B61262636466666666666666666666'x</td>
</tr>
<tr>
<td>90</td>
<td>'2D6A6B6C6D6E6F7071722DA62D2D2D'x</td>
</tr>
<tr>
<td>B0</td>
<td>'2B2B2B5F5FA65F5FDF5F65F5F5F5F5F'x</td>
</tr>
<tr>
<td>C0</td>
<td>'7B41243444546478495F5F5F5F5F'x</td>
</tr>
<tr>
<td>D0</td>
<td>'7D4A4B4C4D4E4F50515255F5F5F5F5F5F'x</td>
</tr>
<tr>
<td>E0</td>
<td>'5C535455565758595A5F5F755F5F5B0'x</td>
</tr>
<tr>
<td>F0</td>
<td>'30313233343536373839B75F6EB25F5F'x</td>
</tr>
</tbody>
</table>

The REPLACE statement starts at position decimal 10, which is byte 11 in the original table, and performs a byte-to-byte replacement with the given values.
replace 10
20 10 200 'x' 'ux' '092040'x;

The SAVE statement saves the changes made to the NEWTABLE translation table.
save;
The second LIST statement displays the edited NEWTABLE translation table.

```
list one;
```

**SAS Log**

The Edited NEWTABLE Translation Table

```
NOTE: Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.
NEWTABLE table 1:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'00010203A309E57FF9EC140AC8787578'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'09204013A5E008E71819C6C51C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'C7FCE92E40A171BEAE9E8EE0F050600'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'9B6516F46F2B04FDF6DA2B6A7501A'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'20E1EDF3F1FD1AABBAFA22E3C2B87C'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'265FACDBDCA1ABBB5F21242293BAC'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'2B6162636465666768692D2BA6A62B2B'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'2D6A6B6C6D6E6F7071722DA62B2D2D2D'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'2D7E737475767778787A2D2B2B2B2B2'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'2B2B2B5F5A5F5F5F5F5F65F5B55F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'7B414234444445464748495F55F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'7D4A4B4C4D4E4F501525F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'5C835355555555555555555555555555'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'303132333435363738393B75F6EB25F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

At position 10 (which is byte 11), a vertical arrow denotes the starting point for the changes to the translation table.

- At byte 11, decimal 20 (which is hexadecimal 14) replaces hexadecimal C4.
- At byte 12, decimal 10 (which is hexadecimal 0A) replaces hexadecimal OB.
- At byte 13, decimal 200 (which is hexadecimal C8) replaces hexadecimal 0C.
- At byte 14, character 'x' (which is hexadecimal 78) replaces hexadecimal 0D.
- At bytes 15 and 16, characters 'ux' (which are hexadecimal 75 and 78, respectively) replace hexadecimal OE and OF.
- At bytes 17, 18, and 19, hexadecimal 092040 replaces hexadecimal 101112.

---

**Example 4: Editing by Using a Quoted Character for Starting Position**

**Procedure features:**
- LIST statement
- LOAD statement
- REPLACE statement
SAVE statement

This example creates a new translation table by editing the Institute-supplied ASCII translation table. The first occurrence of the hexadecimal equivalent of the quoted character specified in the REPLACE statement is the starting position for the changes to the table. This differs from Example 3 on page 1315 in that you do not need to know the exact position at which to start the changes to the table. PROC TRANTAB finds the correct position for you.

The edited table is saved under a new name. Horizontal arrows in both SAS logs denote the edited rows in the translation table.

Program

options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;

The LIST statement displays the translation table in the SAS log.

list one;

SAS Log

```
NOTE: Table specified is ASCII.
ASCII table 1:
0123456789ABCDEF
00 '00102030405060708090A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788889A8B8C8D8E8F'x
90 '9091929394959697989999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDBEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACCCCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADDCCCCDDEDF'x
E0 'E0E1E2E3E4E5E6E7E8E9EAEBCDEEEFF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFFE'x
```

The REPLACE statement finds the first occurrence of the hexadecimal "a" (which is 61) and replaces it and the next 25 hexadecimal values with the hexadecimal values for uppercase "A" through "Z."
Example 5: Creating the Inverse of a Table

Procedure features:

**Example 5: Creating the Inverse of a Table**

The **TRANTAB** Procedure △  

Example 5: Creating the Inverse of a Table 1319

replace ‘a’ ‘ABCDEFGHIJKLMNOPQRSTUVWXYZ’;

The **SAVE** statement saves the changes made to the ASCII translation table under the new table name **UPPER**.

```sql
save table=upper;
```

The **LOAD** statement loads the edited translation table **UPPER**. The **LIST** statement displays the translation table **UPPER** in the SAS log.

```sql
load table=upper;
list one;
```

**SAS Log**

The **UPPER** Translation Table
The horizontal arrows in the SAS log denote the rows in which the changes are made.

```
NOTE: Table **UPPER** being loaded.
UPPER table 1:
  0 1 2 3 4 5 6 7 8 9 A B C D E F
  00 '000102030405060708090A0B0C0D0E0F'x
  10 '101112131415161718191A1B1C1D1E1F'x
  20 '202122232425262728292A2B2C2D2E2F'x
  30 '303132333435363738393A3B3C3D3E3F'x
  40 '404142434445464748494A4B4C4D4E4F'x
  50 '505152535455565758595A5B5C5D5E5F'x
  60 '606162636465666768696A6B6C6D6E6F'x
  70 '707172737475767778797A7B7C7D7E7F'x
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACA
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDCEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CABCBCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DBDCDDDEF'x
E0 'E0E1E2E3E4E5E6E7E8E9EAEBCDEEEEF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFF3F'x
```

Example 5: Creating the Inverse of a Table

Procedure features:

**INVERSE** statement
LIST statement
SAVE statement

This example creates the inverse of the translation table that was created in Example 4 on page 1317. The new translation table created in this example is the operating environment-to-device translation for use in data communications.

Program

options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;

The INVERSE statement creates table two by inverting the original table one (called UPPER). The SAVE statement saves the translation tables. The LIST BOTH statement displays both the original translation table and its inverse.

inverse;
save;
list both;

SAS Log

The UPPER Translation Table and Its Inverse
The SAS log lists all of the duplicate values that it encounters as it creates the inverse of table one. To conserve space, most of these messages are deleted in this example.
Example 6: Using Different Translation Tables for Sorting

Procedure features:

The INVERSE statement lists in the SAS log all of the multiple translations that it encounters as it inverts the translation table. In Example 4 on page 1317, all the lowercase letters were converted to uppercase in the translation table UPPER, which means that there are two sets of uppercase letters in UPPER. When INVERSE cannot make a translation, PROC TRANTAB fills the value with 00. Note that the inverse of the translation table UPPER has numerous 00 values.
PROC SORT statement option:
   SORTSEQ=

Other features:
   PRINT procedure

This example shows how to specify a different translation table to sort data in an order that is different from the default sort order. Characters that are written in a language other than U.S. English may require a sort order that is different from the default order.

Note: You can use the TRABASE program in the SAS Sample Library to create translation tables for several different languages.

Program

   options nodate pageno=1 linesize=80 pagesize=60;

   data testsort;
      input Values $10.;
      datalines;
      Always
      always
      Forever
      forever
      Later
      later
      Yesterday
      yesterday
   ;

PROC SORT sorts the data using the default translation table, which sorts all lowercase words first, then all uppercase words. PROC PRINT prints the sorted data set.

   proc sort;
      by values;
   run;
   proc print noobs;
      title 'Default Sort Sequence';
   run;

SAS Output

Output from Sorting Values with Default Translation Table
The default sort sequence sorts all the capitalized words in alphabetical order before it sorts any lowercase words.
The TRANTAB Procedure

Example 7: Editing Table One and Table Two

The SORTSEQ= option specifies that PROC SORT sort the data according to the customized translation table UPPER, which treats lowercase and uppercase letters alike. This is useful for sorting without regard for case. PROC PRINT prints the sorted data set.

```
proc sort sortseq=upper;
  by values;
run;
proc print noobs;
  title 'Customized Sort Sequence';
run;
```

**SAS Output**

Output from Sorting Values with Customized Translation Table

The customized sort sequence sorts all the words in alphabetical order, without regard for the case of the letters.

```
Customized Sort Sequence

Values
Always
Always
Forever
Later
forever
forever
Yesterday
later
yesterday
```

**Example 7: Editing Table One and Table Two**

Procedure features:
- LIST statement
- REPLACE statement
This example shows how to edit both areas of a translation table. To edit positions 256 through 511 (table two), you must

1. Issue the SWAP statement to have table two change places with table one.
2. Issue an appropriate REPLACE statement to make changes to table two.
3. Issue the SWAP statement again to reposition the table.

Arrows in the SAS logs mark the rows and columns that are changed.

**Program**

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

The LIST statement displays the original UPPER translation table.

```sas
list both;
```

**SAS Log**

The Original UPPER Translation Table
NOTE: Table specified is UPPER.
UPPER table 1:

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'000102030405060708090A0B0C0D0E0F'x</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'101112131415161718191A1B1C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'202122232425262728292A2B2C2D2E2F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'303132333435363738393A3B3C3D3E3F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'404142434445464748494A4B4C4D4E4F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'505152535455565758595A5B5C5D5E5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'606162636465666768696A6B6C6D6E6F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>'707172737475767778797A7B7C7D7E7F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'808182838485868788898A8B8C8D8E8F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'909192939495969798999A9B9C9D9E9F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'B0B1B2B3B4B5B6B7B8B9BABBBCBDDEBF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'C0C1C2C3C4C5C6C7C8C9CAACCCDCECF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'D0D1D2D3D4D5D6D7D8D9DADEDDDF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'F0F1F2F3F4F5F6F7F8F9FAFBFCFDDEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The REPLACE statement starts at position 1 and replaces the current value of 01 with '0A'.

```
replace 1 '0A'x;
```

The first SWAP statement positions table two so that it can be edited. The second REPLACE statement makes the same change on table two that was made on table one.

```
swap;
replace 1 '0A'x;
```

The second SWAP statement restores table one and table two to their original positions. The SAVE statement saves both areas of the translation table by default. The LIST statement displays both areas of the table.
swap;
save;
list both;

---

### The Edited UPPER Translation Table

In byte 2, in both areas of the translation table, hexadecimal value '0A' replaces hexadecimal 01. Arrows denote the rows and columns of the table in which this change is made.

#### NOTE: Table specified is UPPER.

**UPPER table 1:**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'000A2030405060708090A0B0C0D0E0F'x</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'011112131415161718191A1B1C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'202122232425262728292A2B2C2D2E2F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'303132333435363738393A3B3C3D3E3F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'404142434445464748494A4B4C4D4E4F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'505152535455565758595A5B5C5D5E5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'604142434445464748494A4B4C4D4E4F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>'705152535455565758595A5B5C5D5E5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'80B8B8B8B8B8B8B8B8B8B8B8B8B8B8'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'909192939495969798999A9B9C9D9E9F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDBEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'C0C1C2C3C4C5C6C7C8C9CA6BC7DDCFBC'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'D0D1D2D3D4D5D6D7D8D9DADBDDCDDDF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'E0E1E2E3E4E5E6E7E8E9EAEBCDEDEEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'F0F1F2F3F4F5F6F7F8F9FABFBCFDFEFE'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**UPPER table 2:**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'000A2030405060708090A0B0C0D0E0F'x</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'011112131415161718191A1B1C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'202122232425262728292A2B2C2D2E2F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'303132333435363738393A3B3C3D3E3F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'404142434445464748494A4B4C4D4E4F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'505152535455565758595A5B5C5D5E5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'604142434445464748494A4B4C4D4E4F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>'705152535455565758595A5B5C5D5E5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'80B8B8B8B8B8B8B8B8B8B8B8B8B8B8'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'909192939495969798999A9B9C9D9E9F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDBEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'C0C1C2C3C4C5C6C7C8C9CA6BC7DDCFBC'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'D0D1D2D3D4D5D6D7D8D9DADBDDCDDDF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'E0E1E2E3E4E5E6E7E8E9EAEBCDEDEEF'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'F0F1F2F3F4F5F6F7F8F9FABFBCFDFEFE'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>