Inside Macintosh Interim Chapter Draft

Translating Between Character Sets

Written by:	Karl Young	May 25, 1987
Last Revision:	Mark Davis & Karl Young	July 8, 1987

Synopsis

This document describes a method for converting (or, more correctly, transliterating) between different character sets. It defines a standard interface for calling a procedure for the transliteration. It describes the way procedures are stored as resources and how to describe a family of such procedures.

Motivation

Two current Macintosh applications have shown need for transliterating characters between different character sets, and there are many more.

Apple File Exchange transliterates files between IBM PC compatibles, the Apple II family, and the Macintosh. Each has a different character set which, especially in the case of the PC and Apple II, can vary from language to language and country to country.

Any telecommunications program that emulates terminals must deal with the varying character sets of each terminal. Once again, character sets on terminals vary on domestic and foreign versions.
It is difficult (and inelegant) to place the knowledge of all possible character sets into the code of the application. This document proposes a character conversion architecture using transliteration definition procedures for converting between different kinds of character sets. The main goals are to provide (a) a consistent interface for definition procedures, and (b) an extendable structure that allows new character sets to be handled by installation of new definition procedures.

Character Sets

We often describe the character set in terms of a machine that uses that character set. For instance, while the Macintosh, Apple II and IBM PC all use ASCII for the basis of their characters sets, they all have different extensions to ASCII which have different repertoires, and are coded differently. Each of these different encodings forms a different character set.

Similar character sets may also be slightly different in different countries. For example, the French version of EBCDIC is different from the English version. The German variant of ASCII substitutes umlauted characters (“ä”,“ö”,“ü”) for ASCII punctuation (“[”,“/”,“]”). Each of these minor variations also form a different character set. These variant character sets are grouped as a *character set family*.

In general, characters sets may be composed of a number of subsets. For example, Xerox or ISO character sets are *shifting character sets*, allowing shifting between different character subsets by means of escape codes. On other machines, such as the Macintosh, the character set information is encoded by means of the font family number. The character conversion architecture treats different subsets as different character sets belonging to a character set family.

Character set families are distinguished by numbers, ranging from 0 to 32767. They are referenced below as being defined by:

```
Type    CharFam = Integer;
```

An initial list of CharFam values consists of the following:

```
cfMacintosh   =    0;
cfASCII       =    1;
cfIBMPC       =    2;
cfIBMPS2      =    3;
cfEBCDIC      =    4;
cfISO8859     =    5;
cfISO6937     =    6;
cfXerox       =    7;
```

Countries are also numbered by country code, as initially defined in the International Utilities Package chapter of Inside Macintosh. Further country codes are defined in MPW equate files. Note that the term really refer to regions rather than countries: a country (Canada, Belgium, Switzerland) may contain a number of regions, each with its own country code. Country codes are defined as:

```
Type    CntryCode = Integer;
```

Transliteration Procedure

Transliteration procedures can reside in application files, common resource files or in the system file. There may be many transliteration procedures (or TProc's for short) available in a particular resource file. Each TProc handles the transliteration from a particular source character set to a particular destination character set.

The call to the transliteration procedure can be described in Pascal as follows:

```
Function Translit(       params: TransPB): OSErr;


Type      TransText = record
                  trPtr:         Ptr;
                  trLen:         Longint;
                  trFont: Integer;
          end;

TransPB = record
                  verb:          Integer;       {i}
                  featureFlags:  Integer;       {i}
                  result: OSErr;         {o}
                  newCountry:    Integer;       {io}
                  srcText:       TransText;     {io,io,i}
                  dstText:       TransText;     {io,io,io}
                  reserved:      Array [0..3] of Longint;
          end;
```

result:

The function returns an error code (type OSERR) describing any error conditions that occurred during transliteration. This result is duplicated in the result field. In addition to the standard OSErr codes we define the following termination results for TProc's:

```
tDstTooShort     =      -501;   { destination buffer too short }
tSubsetSwitch    =      -502;   { source changed character subset }
tPartialChar     =      -503;   { source ends with partial character }
```

verb:

The verb can have the following values:

```
transInit        =      0;
transDone        =      1;
transLowToHigh   =      2;
transHighToLow   =      3;
```

Any other values are ignored (in current versions). The transInit and transDone verbs are called initially and finally for initialization and cleanup. Each transliteration procedure translates between two character sets. The transLowToHigh call translates from lower numbered character set to the larger number character set (e.g. Mac to IBM PC). The transHighToLow call translates in the reverse direction.

featureFlags:

The feature flags word contains a number of bits for selecting different variations of transliteration:

```
trMultiDestFont  =      1;      {allow multiple destination fonts}
trMultiDestCountry      =      2;      {allow multiple destination countries}
trNonOneToOne    =      4;      {allow one:many or many:one conversions}
trOverStrike     =      8;      {allow overstrike output, e.g. A-Backspace-`}
```

newCountry:

When a TProc is first used, this parameter should be set to -1 on input. On output, it reflects the country code that should be used for the next conversion (see below). If trMultiDestCountry is 0, this will always be the same as the current country. This parameter is used internally to insert control codes for shifting character sets.

srcText.trPtr, dstText.trPtr:

On input, these parameters are pointers to the source and destination buffers, respectively.

srcText.trLen, dstText.trLen:

As input, these parameters contain the size (in bytes) of the source and destination buffers, respectively. As output, srcText.trLen contains the number of bytes of the source string remaining that were not used; dstText.trLen contains the number of bytes remaining in the destination buffer. Note that the number of bytes converted in the output text may differ from the number of bytes used in the input text for a number of reasons:

1. The destination may not be large enough for the required conversion; note that conversions may convert many bytes to one or vice versa unless trNonOneToOne is 0.
2. The input text may contain a partial character: the last byte may be part of a multibyte character.
3. To represent the input text on the specified machine, the font may need to be changed. For example, a symbol may be available in the source that is only available in the Symbol font on the Macintosh.

srcText.trFont, dstText.trFont:

These parameters contain the source and destination font numbers, respectively. For conversion to the Macintosh, these numbers are standard Font Manager font family numbers. For other machines distinguishing character sets by fonts, the font number may be different. A value of zero should refer to the standard (or system) font, while a value of one should refer to the common application font (which may be the same as the system font). On output, the dstText.trFont may contain a different font number, indicating that the proper representation of the text requires that font or an equivalent (e.g. Symbol). If the trMultiDestFont flag is 0, the output font will never differ from the input font: fallback characters will be used instead.

reserved:

This space should be zeroed on input.

Using the Transliteration Procedure

To use a transliteration procedure, the application should get the resource, lock it down for the duration, and jump to the start of the handle contents. (In Pascal, this can be done by recasting using a procedure pointer.) The srcSubset and dstSubset should be set to the initial character subset or font number (zero if unknown). The user must unlock the resource when done (after making the transDone call).

The transliterate routine will translate characters into the destination country and font (say, Times) as long as it can. If it is forced to stop because the source has changed to a font or country that it cannot handle, then it will indicate this by setting the destination country and font, and returning the error `subsetSwitch`. If the country changes, then the application should switch to a transliteration procedure that can handle that country (see below).

TProcs that handle shifting character sets should return the lowest-numbered country code for a TProc that will handle the new text.

As a general rule, control characters are not transliterated. Exceptions are in cases where characters are composed (e.g. in ISO 6937 where *À* equals *A-Backspace-*`), or where characters are represented by shifting to another graphic character set on one machine, but are single character on another.

Transliteration Procedure Resources

Transliteration procedures are stored as resources with type 'tprc'. The resources have a header of the form (in pseudo-Pascal):

```
Record    {standard header}
branch:   Integer;       {BSR.S Code instruction}
flags:    Integer;       {flag bits}
rType:    ResType;       {'tprc'}
id:       Integer;       {resource id number}
version:  Integer;       {version number}
{tProc information}
lowCountry:      CntryCode;     {Macintosh country code}
lowCharFam:      CharFam;              {E.g. IBM PC Character set, French EBCDIC}
highCountry:     CntryCode;     {Macintosh country code}
highCharFam:     CharFam;              {E.g. Macintosh Character set}
reserved:        Array [0..3] of Longint;
End;
```

The resource ID numbers are assigned serially by Technical Support. The name of each resource should indicate the kind of transliteration that will be going on: e.g., “Mac to VT100,” “Mac to Apple IIe,” “EBCDIC to Mac,” etc. The country names can be gotten from the TProc family (see below).

The flags bits indicate which of the featureFlags the TProc will support. The low country is always the smaller number of the two countries that are translated between; the lowCharFam is the CharFam associated with the lowCountry.

One anticipated use of the resource name will be to display a list to users (say, of a terminal emulation program) so they can choose the appropriate transliteration. Remember that the resource names can be localized, so a program can’t necessarily depend on a certain name being present.

Transliteration Procedure Families

A TProc family is described by another resource, with resource type 'tprf'. It provides a way of finding the appropriate TProc for the countries and character sets involved. Note that the TProcs contain the specification information in their headers, so the TProc families (except for their names) can be reconstructed from the available TProcs.

The resource ID of this resource corresponds to the country code, and lists all of the available TProcs that transliterate to and from that country. The name of the resource also corresponds to the appropriate country, in the language of the host machine. Since this name may also be localized, programs cannot depend upon the names being constant.

The form of a TProc family resource is described as follows:

```
TPRF = RECORD
       lastEntry:     Integer;
       reserved:      Array[0..31] of Integer;
       TProcs: Array[0..lastEntry] of TPRCEntry;
END;
```

The **lastEntry** field can be used to find the number of TPRC entries described in this family. The **reserved** field is for future extensions. The **TProcs** field is an array with **lastEntry**+1 elements, each element of the form described below:

```
TPRCEntry = RECORD
          tprcID: Integer;
          curCharFam:    CharFam;
          altCountry:    CntryCode;
          altCharFam:    CharFam;
end;
```

The **tprcID** field contains the resource ID of the 'tprc' resource which will transliterate between the given countries and character sets, where the current country is derived from the family resource id. The **curCharFam** field contains the character set family number associated with the family’s country.

The **altCountry** field contains the country code for the second country. As noted earlier, this will most often be the same as the current country, but it need not be. The **altCharFam** field contains the second character set family number. Each TProc may have two entries: one in the TProc family for the lowCountry, and one for the highCountry. It will have only one entry when the low and high countries are identical.

Using Transliteration Procedure Families

How an application uses a TProc family depends upon the amount of user interaction that can be required. An application can infer the current source and destination country from the international resource 0 (use IUGetIntl). Likewise, the user might want to choose from the list of available countries (which is a list of the names of the 'tprf' resources) for both the source and destination country.

For terminal emulation, either the source or the destination will always be the Macintosh, but the source and destination countries may vary.

Given a particular source and destination machine, the application can investigate the appropriate TProc family for the appropriate TProc. In the case of Apple File Exchange, where outside parties are writing file translators, the transliteration part of their code is abstracted so the translator will work across all countries without any further localization.