NCITS L2/98-257
1998-07-20

Title:	**EBCDIC-Friendly UCS Transformation Format -- EF-UTF**
Source:	**V.S. UMAmaheswaran, IBM National Language Technical Centre, umavs@ca.ibm.com**
Status:	**For consideration and acceptance by NCITS-L2 and UTC for further processing as a new UTF into Unicode and ISO/IEC 10646; (part of a paper that has been submitted for presentation at IUC-13, San Jose, in September 1998).**

1 Background

UCS Transformation Format UTF-8 (defined in Amendment No. 2 to ISO/IEC 10646-1), is a transform for UCS data that preserves the subset of 128 ISO-646-IRV (ASCII) characters of UCS as single octets in the range X'00' to X'7F', with all the remaining UCS values converted to multiple-octet sequences containing only octets greater than X'7F'. This permits existing systems that have hard-coded dependency on the encoding of these characters to 'safely' process UCS characters in the UTF-8 transformed form.

There is a similar requirement to transform a UCS-encoded data to a form that is 'safe' for EBCDIC systems for the control characters and invariant characters. This document defines a transformation format for use in applications written for EBCDIC systems deriving benefits similar to what UTF-8 delivers to applications written for ASCII-based or ISO-8-based systems.

A pre-condition for any method that transforms UCS data to be processed in the EBCDIC environment, is that each EBCDIC control character must be kept as a single octet. This cannot be achieved by applying the ISO-8 to EBCDIC transform to the standard UTF-8 transformed data. Data conversions between ISO-8-bit and SBCS EBCDIC coded character sets, typically map the EBCDIC control zone into the ISO-8 control zone(s), and EBCDIC graphic character zone into the ISO-8 graphic character zone(s), and vice versa. These character-zone correspondences are respected also in mixed ISO-8-bit and mixed-byte-EBCDIC coded character sets. The standard UTF-8 converts the ISO-8 C1 zone into two-octet sequences, and hence is not usable when there is a requirement to preserve the ISO-8 C1 control characters, and the corresponding EBCDIC control characters, as single octets.

Eight-bit coded character sets based on ISO/IEC 4873 standard, or IBM's EBCDIC standard, have 65 control character positions and 191 graphic character positions (see Figure 1 on page 9). ISO/IEC 4873 defines the structure for use in ISO-8 codes such as ISO/IEC 8859-1, Latin Alphabet No. 1, and others (see Figure 2 on page 9).

The 65 control character positions are in the range X'00' to X'1F' (C0 set), at X'7F' (DELETE), and in the range X'80' to X'9F' (C1 set), for the ISO standard, and in the range X'00' to X'3F' and at X'FF' (Eight Ones) for the EBCDIC standard. A standard set of control functions are assigned to these control character positions in EBCDIC (see Figure 10 on page 17).

X'20' (SPACE), the range X'21' to X'7E' (G0 set), and the range X'A0' to X'FF' (G1 set) -- a total of 191 octets -- can be assigned graphic characters in ISO-8 single-octet codes. In the corresponding Single-Byte EBCDIC codes graphic characters may be assigned to X'40' (SPACE) and the range X'41' to X'FE' -- a total of 191 octets.

2 Criteria used for defining EF-UTF

The following criteria are used in defining the EF-UTF:

I. *Respect the invariance assumptions for characters used by file-management and other subsystems on EBCDIC platforms.*

Traditional EBCDIC-based file systems assume a core set of graphic characters for entities such as file names, attributes and others. These are SPACE, uppercase letters A to Z, numeric digits 0 to 9, '-' (hyphen), '_' (underscore), and in POSIX environments '.'(period).

When lower case letters a to z are permitted, they are often equated to their corresponding upper case letters, in entities such as file names, file attributes and other parameters passed across APIs for file management sub-systems or other similar modules.

Characters such as #, @, and $, are also allowed in file names. While the invariance of the 81 characters of the IBM Syntactic Character Set (with IBM Graphic Character Set Global Identifier - GCSGID 640) is assumed (with some known exceptions), characters such as #, @, and $, are known to be variant among existing EBCDIC coded character sets. Irrespective of whether a larger character set is permitted in file management related entities, the core set of characters are hard coded in traditional file systems and many applications -- see Figure 3 on page 10.

II. *Respect the invariance of EBCDIC control code positions.*

Code positions of X'00' to X'3F' and X'FF', are reserved exclusively for Control Characters in the IBM EBCDIC Standard -- see Figure 3 on page 10 and Figure 10 on page 17. An exception to this is the EBCDIC-presentation code page(s) primarily used in printers and printer data streams. Some products such as GDDM are known to deviate by assigning graphic characters to the EBCDIC control zone in their internal coded character sets.

III. *Respect the invariance assumptions of EBCDIC-based software.*

Most core modules in operating systems such as MVS, VM, AS/400, are hardcoded with the assumed invariance of code positions for characters in GCSGID 640 (see Figure 3 on page 10 and Figure 11 on page 18).
Following this criterion also will satisfy criterion number 1 above.

IV. *Invariance assumptions regarding the character set of ASCII:*

Operating systems such as OS/390 UNIX Services, and the C/370 and C++ run time libraries (and compiler) have internal assumptions for the ASCII character set (IBM GCSGID 103, the portable character set of POSIX) which is syntactically significant for the UNIX operating system and in POSIX environments. They have hardcoded the code position assignments from IBM coded character set with IBM Code Page Global Identifier - CPGID 1047 (the 'EBCDIC Latin-1 Open Systems' code page) as invariant. CPGID 1047 was also the preferred choice of the SHARE - ASCII-EBCDIC White Paper based on the customer usage of Left and Right Square Bracket code positions (taken from the MVS programmer's reference card showing the IBM 1403 printer positions for the square brackets, and hardcoded into several user-written applications).

Similar invariance assumptions have been made in traditional VM, MVS and AS/400 systems, and IBM data stream and object content architectures assuming other EBCDIC default CPGIDs. The significant ones among these are CPGID 500 - the Multilingual Code page and CPGID 00037 - the US EBCDIC Latin-1 code page. IBM Character Data Representation Architecture (CDRA) recommends CPGID 500 as the convergence target for all the CECP Latin-1 EBCDIC sets. CPGID 290 - the Katakana Extended code page poses an additional challenge in that the lower case letters a-z are allocated positions differing from their EBCDIC standard invariant positions. Consideration must be given to the invariance of the ASCII set of characters in these CPGIDs.

Note: There may be other EBCDIC coded character sets also needing such consideration. However, due to the prominence of OS/390 UNIX Services and the customer hardcoded applications using CPGID 1047, this proposal is based on CPGID 1047 hardcoding assumptions for the POSIX portable character set.

V. *The following properties of the standard UTF-8 are preserved:*

A. Ease of conversion from and to UCS
B. The lexicographic sorting order of UCS-4 strings
C. The entire range of 2**31 UCS-4 code positions can be encoded (though in practice only UCS-2 form -- including the S-zone of BMP -- will be sufficient)
D. Easy re-synchronization in a multiple-octet sequence (ability to find the start of a valid sequence with a minimum of scanning in either direction)
E. Stateless encoding which is robust against missing octets
F. Ability to identify the number of following octets in a sequence of a variable number of octets
G. Minimum number of octets in the sequence.

3 EF-UTF transform

The proposed EF-UTF transform consists of two parts (see Figure 4 on page 11):

1) The first part is called herein UTF-8M (modified UTF-8), (and its reverse rUTF-8M), and is described in section "First part: UTF-8M and rUTF-8M" on page 3. It is a modified form of UTF-8. This part converts between UCS-4 or UCS-2 string (called the U-string -- see section "The U-string" on page 3) and an intermediate ISO-8-compatible string (called the I8-string -- see section "The I8-string" on page 4), and

2) The second part is called herein I8-To-E (and its reverse E-To-I8) and is described in section "The Second Part: I8ToE and EToI8" on page 6. It is a single-octet to single-octet reversible conversion. This part converts between the ISO-8 compatible string (I8-string) and the EBCDIC-Friendly-UCS-transformed string, or EBCDIC-compatible string (called E-string in this document)

These parts are detailed in the following sections.

3.1 First part: UTF-8M and rUTF-8M

The proposed UTF-8M transform is modeled after the UTF-8 definition in Amendment No. 2 of ISO/IEC 10646-1 and in the Unicode standard. UTF-8M is similar to UTF-8 but preserves the C0, G0, DEL and C1 as single octets.

UTF-8M transforms the U-string, either in UCS-2 form or in UCS-4 form (see Figure 4 on page 11), into a sequence of 1 to 7 octets of the I8-string, the intermediate form. The rUTF-8M is the reverse transform. The generic term UTF-8M is used for both the forward and reverse transforms in the description below.

3.1.1 The U-string

The U-string is a string of UCS characters. The UCS character can be either in UCS-4 form or the UCS-2 form. In the UCS-4 form, it consist of 4 octets representing the value from X'00000000' to X'7FFFFFFF'. For the Basic Multi-Lingual Plane - BMP (plane 0, group 0) and the subsequent 16 planes in group 0, the range of values will be X'00000000' to X'0010FFFF'. In the UCS-2 form (including the S-zone elements, or surrogates) the values can range from X'0000' to X'FFFF'. For the purposes of this paper, byte-reversed form is considered to have been converted to non-byte-reversed form.

In practice, most of the world's widely used scripts have been allocated code positions in the BMP. Additionally the road map document adopted by ISO/IEC JTC 1/SC 2/WG 2 and the Unicode Technical Committee shows that all the known anticipated scripts can be accommodated in supplementary planes 1 and 2 of group 0 in UCS-4. Planes 15 and 16 are reserved for private use. There is a proposal for use of plane 14 to meet the Internet protocol requirements for different types of tags.

UCS-2 is a subset of UCS-4 representing the octet pairs (called the Row/Column Element - RC Element in ISO/IEC 10646-1) of the Basic Multilingual Plane (BMP) (or plane 0 of group 0). Using the S-zone RC-elements, called the surrogates in the Unicode standard, (in the range X'D800' to X'DBFF'), an additional 16 planes (planes 1 to 16, of group 0) can be represented using the UTF-16 defined in Amendment No. 1 of ISO/IEC 10646-1 (and in Unicode). Figure 5 on page 11 (top half) illustrates how UTF-16 assembles the 10 bits from each of the S-HI and S-LO pairs into the UCS-4 form (to be padded with 11 leading 0-s).

UTF-8 as defined in Amendment No. 2 of ISO/IEC 10646 refers only to the UCS-4 form as input to the transform. Amendment No. 1 on UTF-16 states that the S-zone elements are for exclusive use by UTF-16 transform. The expectation is that the UTF-16 encoded data (using the high order and low order pairs of S-zone RC elements) will be transformed into their canonical UCS-4 form before applying the UTF-8 transform. The Unicode standard definition of UTF-16 respects this expectation.

UTF-8M defined in this proposal tolerates the U-strings that include elements from S-zone (as valid high order and low order pairs) in both the UCS-2 form and UCS-4 form. Valid pairs of S-zone elements will be converted to their UCS-4 equivalent (using UTF-16), before transforming to I8-string. However, pairs of S-zone elements are not valid as canonical UCS-4 representation of planes 1 to 16 of group 0.

3.1.2 The I8-string

The I8-string is a sequence of 1 to 7 octets.

For all I8-strings consisting of two or more octets, the number of octets in the string is indicated by the number of high order 1-bits followed by a 0-bit in the lead octet (B'110vvvvv', B'1110vvvv', B'11110vvv', B'111110vv', B'1111110v', and B'11111110', where v can be either 0 or 1), and each trailing octet always begins with the bit sequence 101 as the high-order three bits (B'101vvvvv'). In addition, an I8-string having the first octet as B'11111111' will have six trailing octets (each of the form B'101vvvvv').

When the I8-string has only one octet, its value will be between X'00' (B'00000000') and X'9F' (B'10011111').

The I8-string's octets are listed below under different categories reflecting the zones in the ISO-8 encoding structure (see the groupings shown in Figure 6 on page 12).

1) X'00' to X'9F' (B'00000000' to B'10011111') are single-octet I8-strings
2) X'A0' to X'BF' (B'10100000' to B'10111111') are trailing octets in multiple-octet I8-strings
3) X'C0' to X'DF' (B'11000000' to B'11011111') are the lead or first octet of a two-octet I8-string
   Note: Applying the 'shortest string' rule (see page 5), X'C0' to X'C4' will not be generated by the UTF-8M transform. If they appear in the I8-string, the octet sequences with them as lead bytes, will correspond to U-string values less than X'A0'.
4) X'E0' to X'EF' (B'11100000' to B'11101111') are the first octet of a three-octet I8-string
   Note: Applying the 'shortest string' rule (see page 5) , X'E0' will not be generated by the UTF-8M transform. If they appear in the I8-string, the octet sequences with them as lead bytes, will correspond to U-string values less than X'400'.
5) X'F0' to X'F7' (B'11110000' to B'11110111') are the first octet of a four-octet I8-string
6) X'F8' to X'FB' (B'11111000' to B'11111011') are the first octet of a five-octet I8-string
7) X'FC' to X'FD' (B'11111100' and B'11111101') are the first octet of a six-octet I8-string
8) X'FE' and X'FF' (B'11111110' and B'11111111') are the first octet of a seven-octet I8-string.

3.1.3 Correspondence between U-string and I8-string

The I8-strings corresponding to the different U-string value ranges are shown in Figure 7 on page 13 for the UCS-2 form and Figure 9 on page 15 for the UCS-4 form.

The U-string is obtained from the I8-string by concatenating all the v-bits together, stripping out the appropriate high order 1s and 0s of the lead and trailing octets, and filling with the appropriate number of leading 0 bits to get a two-octet or four-octet form. Note the exception for the I8-strings of 7 octets (in the correspondence tables) where there are no 0 bits in the lead octet, and the least significant 1 bit of the lead octet is kept as the most significant bit of the U-string.

The correspondence between the bits in a UCS-4 element (of the form B'0sssttttuuuuvvvvwwwwxxxxyyyyzzzz') and the bits in its corresponding UTF-8M transformed string is shown In a summary form in the table below.

	0	s	s	sttt	t	uuuu	v	vvvw	w	wwxx	x	xyyy	y	zzzz
1													0yyy	zzzz
1													100y	zzzz
2											110x	xyyy	101y	zzzz
3									1110	wwxx	101x	xyyy	101y	zzzz
4							1111	0vvw	101w	wwxx	101x	xyyy	101y	zzzz
5					1111	10uu	101v	vvvw	101w	wwxx	101x	xyyy	101y	zzzz
6			1111	110t	101t	uuuu	101v	vvvw	101w	wwxx	101x	xyyy	101y	zzzz
7	1111	111s	101s	sttt	101t	uuuu	101v	vvvw	101w	wwxx	101x	xyyy	101y	zzzz

The corresponding standard UTF-8 transformation is shown in the following table to facilitate a comparison between UTF-8 and UTF-8M.

	0		s	ss	tttt	uu	uuvv	vv	wwww	xx	xxyy	yy	zzzz
1												0yyy	zzzz
2										110x	xxyy	10yy	zzzz
3								1110	wwww	10xx	xxyy	10yy	zzzz
4						1111	0uvv	10vv	wwww	10xx	xxyy	10yy	zzzz
5				1111	10tt	10uu	uuvv	10vv	wwww	10xx	xxyy	10yy	zzzz
6		1111	110s	10ss	tttt	10uu	uuvv	10vv	wwww	10xx	xxyy	10yy	zzzz

Shortest-String Rule:

In UTF-8 (as originally defined by XPG-4, UTF-FSS), when there are multiple ways to encode a value, for example UCS value X'00000000', only the shortest encoding - X'00' in the UTF-8 form - is legal. (Note: implementations of UTF-8 can represent U-string X'0000' as multiple octet sequence such as B'11000000 10000000' (X'A0 80'), to prevent B'00000000' (X'00') from possibly ending string in some programming language libraries, when UCS-2 value X'0000' -- NUL -- was NOT meant to be a string terminator.)

This 'shortest string rule' is kept in UTF-8M definition. In the reverse direction (I8-string to U-string) the transform will be tolerant - it will recognize the longer strings and strip off the excess leading zeroes.

Of these (from Figure 7 on page 13 and Figure 9 on page 16):

1) the limit of the Basic Multilingual Plane, BMP is reached with the I8-string having the sequence of four octets:
   B'11110001 10111111 10111111 10111111' (X'F1 BF BF BF')
2) the limit of three additional supplementary planes (plane 3 of group 0) is reached with the I8-string having the sequence of four octets:
   B'11110111 10111111 10111111 10111111' (X'F7 BF BF BF'), and,

3) the limit of sixteen additional supplementary planes (the maximum UCS-4 value that can be represented using UTF-16) is reached with the I8-string having the sequence of five octets:
B'11111001 10100001 10111111 10111111 10111111' (X'F9 A1 BF BF BF')

3.1.4 UTF-16 and UTF-8M

UTF-16 defines the transformation of UCS values X'10000' to X'10FFFF' (in planes 1 to 16 of group 0 of UCS) to and from a pair of S-zone RC-elements in the BMP ('surrogates' of Unicode standard) that are reserved exclusively for use in UTF-16. UTF-16 can be defined (from the Unicode standard V2.0 publication) as follows:

C = B for non-S-zone elements
C = (HI-X'D800')*X'400' + (LO-X'DC00') + X'10000',

where,
C is the canonical value in the range X'000000' to X'10FFFF';
B is a non-S-zone BMP value in the range X'0000' to X'FFFF'
(HI, LO) pair is the UTF-16 representation of C
HI - S zone value is in the range X'D800' to X'DBFF', and,
LO - S zone value is in the range X'DC00' to X'DFFF', in the S-zone of BMP.

Figure 5 on page 11 shows the UTF-16 transform from the (HI, LO) pair to the UCS-4 canonical form and to UTF-8M octet sequence. For comparison, the resultant standard UTF-8 form is also shown. The 'v' bits shown in Figures 7, 8 and 9 are shown as 'p', 'q', 'r', 's', 't', 'u' and 'w' to better illustrate the correspondences between the different forms (following the description of UTF-16 in the Unicode Standard Version 2.0).

In UTF-8M, valid pairs of S-zone elements will be converted to their UCS-4 equivalent (using UTF-16), before converting to I8-string octets. If the U-string consists of invalid pairs with one or both elements of the pair from the S-zone, the values from the S-zone are treated as single values and are transformed as shown (in Figure 7 on page 13) for the range X'4000' to X'FFFF'. When the U-string is in the UCS-2 form, UTF-8M always converts I8-string sequences in the ranges X'F2 A0 A0 A0' to X'F7 BF BF BF' and X'F8 A8 A0 A0 A0' to X'F9 A1 BF BF BF' (corresponding to the U-string values in the ranges X'010000' to X'03FFFF' -- planes 1 to 3, and X'04000' to X'10FFFF' -- planes 4 to 16) to and from valid S-zone (HI, LO) pairs. This makes UTF-8M analogous to combining UTF-8 and UTF-16.

3.2 The Second Part: I8ToE and EToI8

The second part of EF-UTF (see Figure 4 on page 11) consists of using a single-octet to single-octet conversion between the octets of the ISO-8 compatible I8-string and the octets of the EBCDIC-compatible E-string (defined below).

3.2.1 The E-string

The E-string, like the I8-string, is a multiple-octet transformed representation of the U-string. The selected I8-string to/from E-string conversion table has a unique one to one mapping between the input octets and output octets, and is symmetrical. While the graphic character preservation principle is used for the octets X'00' to X'9F' of the I8-string, the principle of octet preservation is applied for the range X'A0' to X'FF.'

3.2.2 I8ToE Octet Pairing

The I8ToE octet-pairing chosen:

1. preserves the single octet representation for all the EBCDIC controls, mapping the I8-string octets in the range X'00' to X'1F', X'7F', and X'80' to X'9F', to E-string octets in the range X'00' to X'3F' and X'FF'. Figure 10 on page 17 shows the default pairings for control

characters used in the industry between several EBCDIC code pages and ISO-8 code pages, including the conversion between ISO 8859-1 (CPGID 819) and CPGID 1047.

2. preserves the single octet representation for the set of 95 (including SPACE) graphic characters of the ISO-646 IRV (IBM GCSGID 103, the ASCII character set), at their allocated positions in the target EBCDIC code page 1047. Figure 11 on page 18 shows the mapping between G0 set of ISO 8859-1 and some EBCDIC CPGIDs including CPGID 1047.
3. preserves the leading octets and the trailing octets from the I8-string as their corresponding single octets in the E-string, and,
4. maintains the symmetry between the forward and reverse pairings.

It is important to note that, besides the octets of C0 set, C1 set, and DEL, only the octet values (code points) that correspond to the G0 set of ISO 8859-1 (and not the entire Latin-1 repertoire) are relevant to be preserved as single octets in the E-string. Octets of the I8-string are converted to/from octets of the E-string using the octet conversion tables shown in Figure 12 on page 19.

Figure 13 on page 20 shows the octet distribution of E-string among single octet stings -- shown subdivided among control characters, invariant and variant graphic characters of ISO 646-IRV, and the leading or trailing octets of multiple-octet E-strings. To facilitate checking whether the E-string sequence is a multiple-octet sequence, or whether one of its octets is a leading octet or trailing octet, a shadow vector can be constructed from Figure 13 on page 20. Figure 14 on page 21 shows such a table containing values from 0 to 9 indicating the different E-string octet types.

4 Special nature of UCS values X'FFFE' and X'FFFF'

X'FFFE' and X'FFFF' are not used for character allocation in any plane of UCS. X'FFFE' is used as a Signature. X'FFFF' is used to represent a numeric value that is guaranteed not to be a character, for uses such as the final value at the end of an index. UTF-8 also avoids use of X'FF' and X'FE' as octets in its sequences. In UTF-8M, however, X'FE' and X'FF' are used. The following paragraphs expand on which combinations of X'FF' and X'FE' may occur in an I8-string or an E-string.

- ***X'FFFE' and X'FFFF' in the I8-string:***

  The X'FE' and X'FF' are lead octets of seven-octet I8-strings. They will be surrounded (in a properly formed UTF-8M transformed string) by a value less than X'C0'. Neither X'FFFF' nor X'FFFE' sequences are valid in a properly formed I8-string sequence. The I8-E octet pairings are: X'FE' to X'4A', and X'FF' to X'E1'

- ***X'FFFE' and X'FFFF' in The E-string:***

  The values X'FE' and X'FF' are generated in an E-string by converting I8-string using X'BF' to X'FE' and X'9F' to X'FF' (from Figure 12 on page 19).

  X'BF' is the last element of the set of trailing octets possible in a multiple-octet I8-string and must be preceded by a lead octet and zero or more trailing octets (all within the range X'A0' to X'FF'). An X'9F' cannot precede it in a properly formed I8-string, and hence the sequence X'FFFE' should not appear in an E-string.

  The X'9F' is assigned to the control character - Application Program Command (APC) - in ISO-8 C1. According to ISO/IEC 6429, APC is followed by a parameter string using bit combinations from 0/8 to 0/13 (X'08' to X'0D') and 2/0 to 7/14 (X'20' to X'7E' and terminated by the control function String Terminator (ST) (coded at X'9C' in C1). So the sequence X'FFFF' (equivalent of two APC controls without intervening parameters or STs) also should not appear in an E-string.

5 Normalization

Dealing with a variable number of octets may not be possible or desirable in some processing situations (even though proper handling of UCS text strings will require ability to correctly deal with combining sequences). Normalization into a form with a fixed number of bits is needed for such cases. It would be always desirable to revert to the original UCS-2 (16-bit form) or UCS-4 (32-bit form) as a normalization to fixed-width data. However, this would be possible only if processing is possible with native UCS encoding. If transparency to EBCDIC invariance and controls is needed also in the normalized form, then UCS cannot be directly used for normalization. It can be seen from Figure 7 on page 13 that the last code position in the BMP -- X'FFFF' -- of UCS, requires a four-octet sequence in the I8-string and in the corresponding E-string. A 32-bit integer can be used for normalization of up to four-octet sequences.

The maximum value of UCS-4 that a four octet sequence of I8-string can represent is:

B'11110111 10111111 10111111 10111111' (X'3FFFF')

corresponding to end of plane 3 in group 0 of UCS-4. Using UTF-16 to represent planes 1 to 16 of UCS-4, the S-zone RC-elements in the BMP can be used. By treating the S-zone elements as any other BMP value, up to plane 16 can be encoded using the UCS-2 form, and hence can be contained within the 32-bit normalized form of E-string. Care has to be taken to correctly process the corresponding E-string octet sequences corresponding to the S-zone pairs, similar to dealing with combination sequences. When it is desirable to convert valid pairs of S-zone elements into corresponding canonical form and then apply UTF-8M, only up to plane 3 can be contained within the 32-bit normalized value. For all values beyond group 0, plane 3, of UCS, the UTF-8M will generate sequences of more than 4 octets The normalization for these cases will need 64-bits (assuming nothing between 32 and 64 bits is practical).

6 Bibliography

ISO/IEC 10646-1: 1993(E): Information Processing - Universal Coded Character Set (UCS):Part 1, Basic Multilingual Plane
Amendment 1 to ISO/IEC 10646-1: Transformation Format for 16 Planes of Group 00 (UTF-16); 1996

Amendment 2 to ISO/IEC 10646-1: Transformation Format 8 (UTF-8)

ISO/IEC 646: Information Processing - 7-Bit Coded Character Set for Information Interchange

ISO/IEC 2022: Information Processing - 7-Bit and 8-Bit Coded Character Sets - Code Extension Techniques

ISO/IEC 4873: Information Processing - 8 Bit Code for Information Interchange -Structure and Rules for implementation

ISO/IEC 6429: Information Processing - 7-Bit and 8-Bit Coded Character Sets -Control Functions for Coded Character Sets

ISO/IEC 8859-xx: Information Processing - 8-Bit Single-Byte Coded Graphic Character Sets

ISO/IEC-IR: International Register of Coded Character Sets to be Used with Escape Sequences - Registration Authority: ITSCJ, Japan

The Unicode Standard Version 2.0: The Unicode Consortium ISBN 0-201-48345-9, Addison Wesley Developers Press, July 1996.

SHARE Report SSD No. 366: ASCII and EBCDIC Character Set and Code Issues in Systems Application Architecture, The ASCII/EBCDIC Character Set Task Force. Edited by Edwin Hart, The Johns Hopkins University, Applied Physics Laboratory, Laurel, Maryland, USA Published by Share Inc., 111 East Wacker Drive, Chicago, Illinois, USA 60601; June 1989

CDRA: IBM - Character Data Representation Architecture - Reference and Registry, SC09-2190-00, December 1996.

7 Figures

Figure 1 ***Graphic and Control Zones in EBCDIC Encoding***

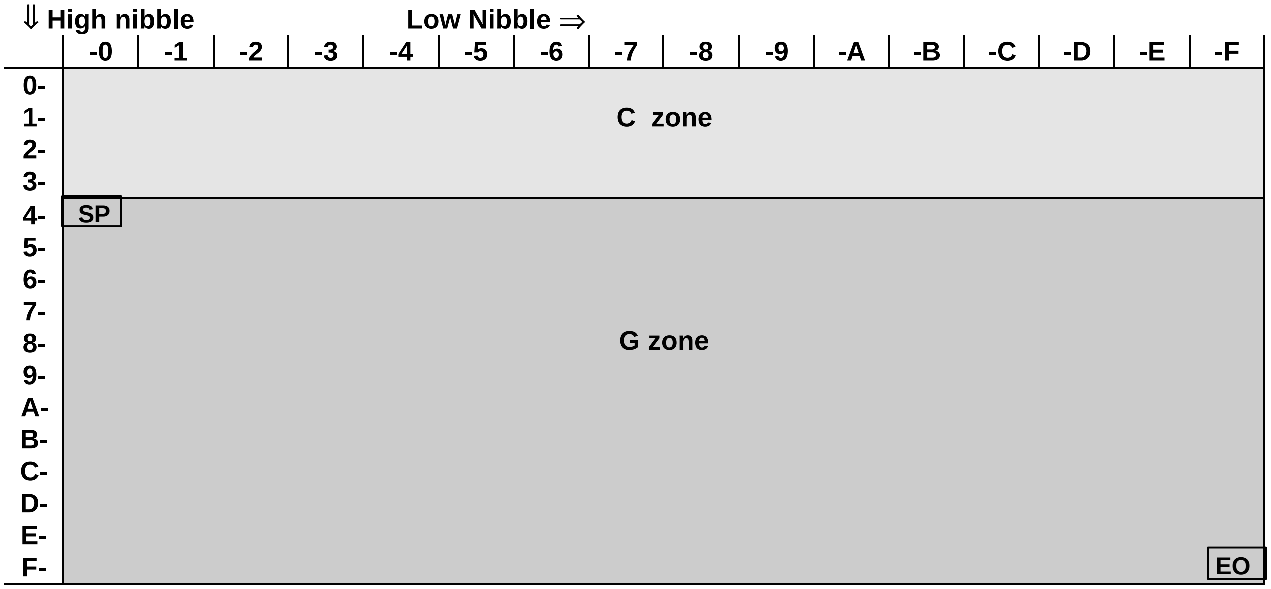

Figure 2 ***Graphic and Control Zones in ISO-8 Encoding***

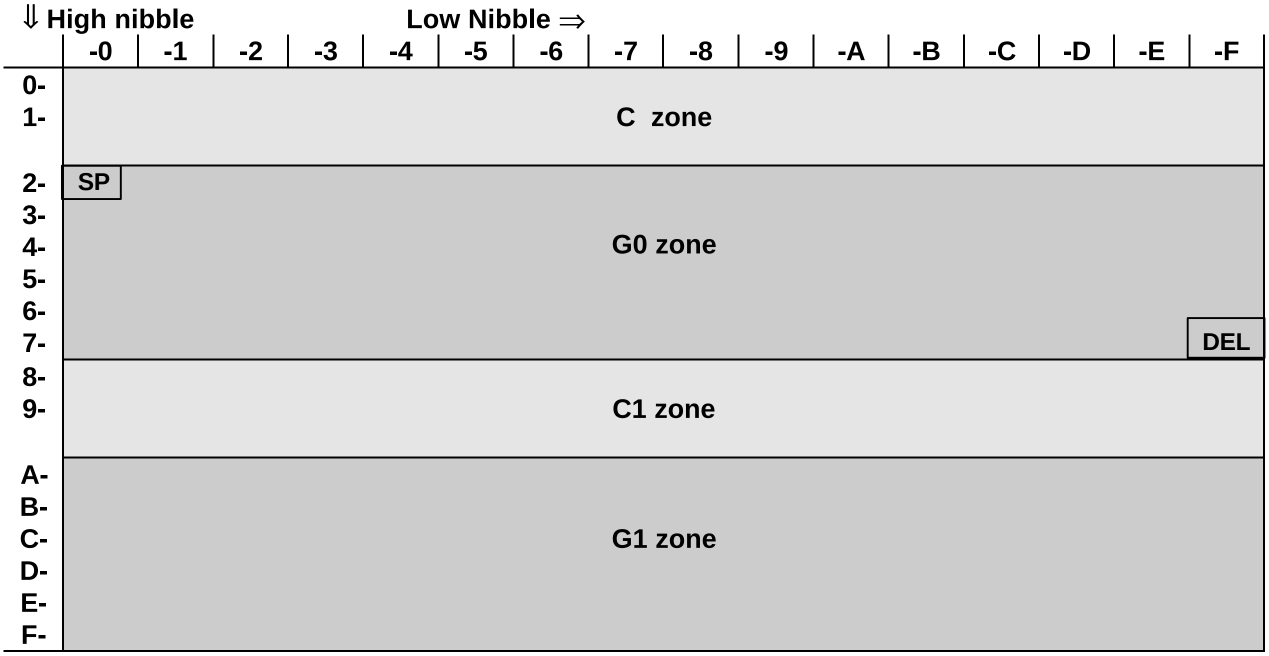

Figure 3 *Distribution of EBCDIC Invariants, Variants and Controls*

Legend:
cc = control character (see Figure 10 on page 17);
ii = invariant - part of IBM syntactic character set, which is a subset of ISO/IEC 646 (IRV) (ASCII) and is invariant among most primary EBCDIC code page definitions (see Figure 11 page 18);
vv = variant - part of ISO/IEC 646 (IRV) (ASCII) but varies among EBCDIC code page definitions (see Figure 11 on page 18);
... = characters outside ASCII set, and are variant.
The letters a-z, A-Z and digits 0-9 are shown in their invariant positions. All letters, digits and octets marked as cc, ii and vv are single octets in the E-string.

⇓ High nibble **Low Nibble ⇒**

	-0	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**	**-8**	**-9**	**-A**	**-B**	**-C**	**-D**	**-E**	**-F**
0-	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
1-	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
2-	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
3-	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
4-	II	...	...	...	...	...	...	...	...	...	...	II	II	II	II	vv
5-	II	...	...	...	...	...	...	...	...	...	vv	vv	II	II	II	vv
6-	II	II	...	...	...	...	...	...	...	...	...	II	II	II	II	II
7-	...	...	...	...	...	...	...	...	...	vv	II	vv	vv	II	II	II
8-	...	a	b	c	d	e	f	g	h	i	...	...	...	...	...	...
9-	...	j	k	l	m	n	o	p	q	r	...	...	...	...	...	...
A-	...	vv	s	t	u	v	w	x	y	z	...	...	...	vv	...	...
B-	...	...	...	...	...	...	...	...	...	...	...	...	...	vv	...	...
C-	vv	A	B	C	D	E	F	G	H	II	...	...	...	...	...	...
D-	vv	J	K	L	M	N	O	P	Q	R	...	...	...	...	...	...
E-	vv	...	S	T	U	V	W	X	Y	Z	...	...	...	...	...	...
F-	0	1	2	3	4	5	6	7	8	9	...	...	...	...	...	cc

Figure 4 The two parts of EF-UTF Transform

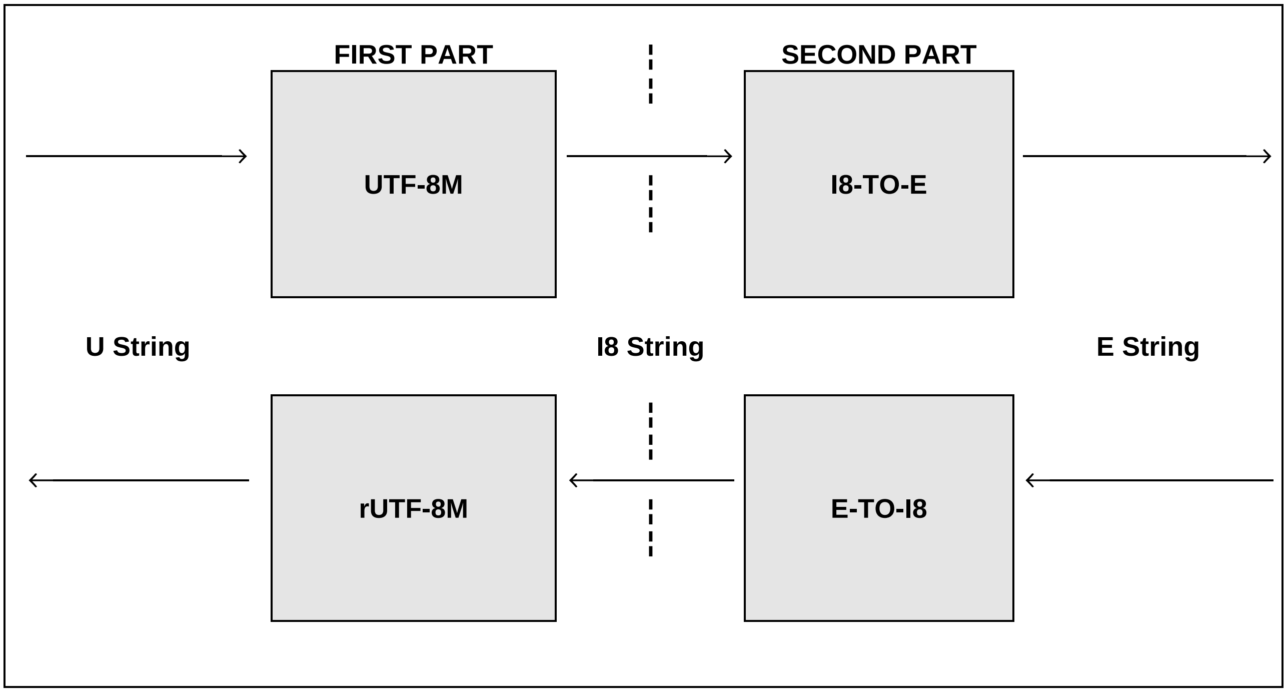

Figure 5 Transforming S-zone pairs in U-string to I8-string octet sequence

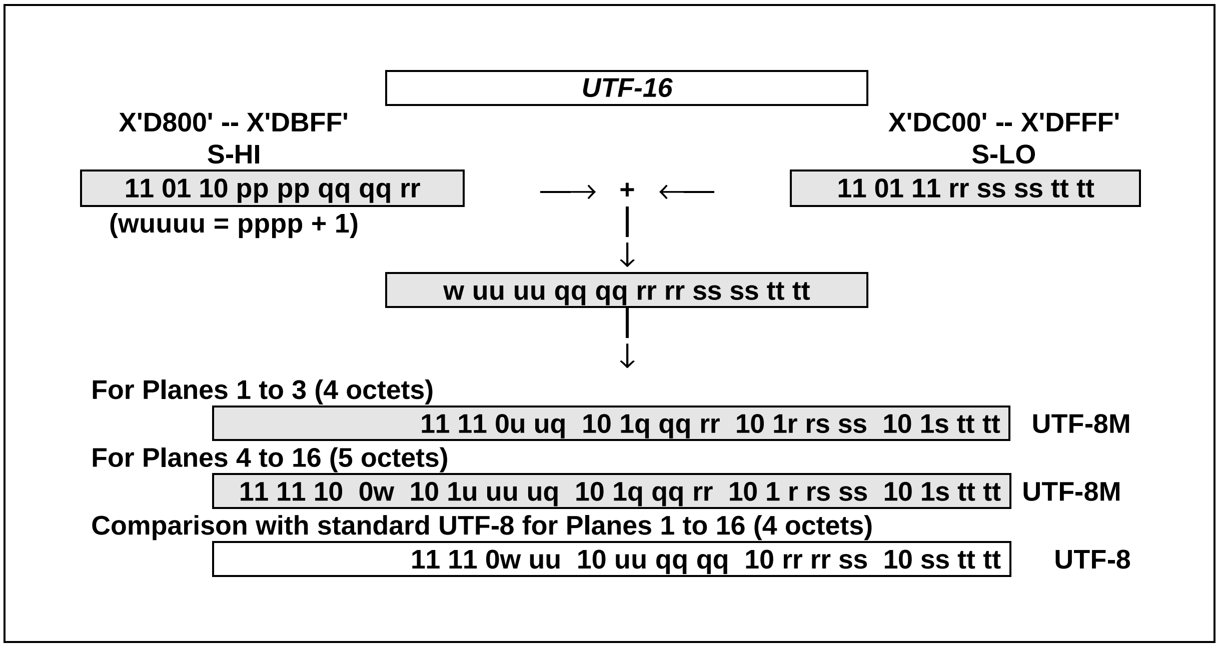
UTF-16
X'D800' -- X'DBFF'
S-HI
11 01 10 pp pp qq qq rr
(wuuuu = pppp + 1)
+
X'DC00' -- X'DFFF'
S-LO
11 01 11 rr ss ss tt tt
w uu uu qq qq rr rr ss ss tt tt
For Planes 1 to 3 (4 octets)
11 11 0u uq 10 1q qq rr 10 1r rs ss 10 1s tt tt
UTF-8M
For Planes 4 to 16 (5 octets)
11 11 10 0w 10 1u uu uq 10 1q qq rr 10 1 r rs ss 10 1s tt tt
UTF-8M
Comparison with standard UTF-8 for Planes 1 to 16 (4 octets)
11 11 0w uu 10 uu qq qq 10 rr rr ss 10 ss tt tt
UTF-8

Figure 6 Distribution of I8-string octets from UTF-8M in an ISO-8 structure

⇓ High nibble Low Nibble ⇒

<table>
<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>
<tr><td>0-</td><td colspan="16" rowspan="2">C zone</td></tr>
<tr><td>1-</td></tr>
<tr><td>2-</td><td>SP</td><td colspan="15" rowspan="5">G0 zone</td></tr>
<tr><td>3-</td><td rowspan="4"></td></tr>
<tr><td>4-</td></tr>
<tr><td>5-</td></tr>
<tr><td>6-</td></tr>
<tr><td>7-</td><td colspan="15"></td><td>DEL</td></tr>
<tr><td>8-</td><td colspan="16" rowspan="2">C1 zone</td></tr>
<tr><td>9-</td></tr>
<tr><td>A-</td><td colspan="16" rowspan="2">32 Trailing Octets</td></tr>
<tr><td>B-</td></tr>
<tr><td>C-</td><td colspan="16" rowspan="2">32 Lead Octets of 2-Octet Sequence</td></tr>
<tr><td>D-</td></tr>
<tr><td>E-</td><td colspan="16">16 Lead Octets of 3-Octet Sequence</td></tr>
<tr><td>F-</td><td colspan="8">8 Lead Octets of 4-Octet Sequence</td><td colspan="4">4 Lead Octets of 5-Octet Sequence</td><td colspan="2">2 Lead Octets of 6-Octet Sequence</td><td colspan="2">2 Lead Octets of 7-Octet Sequence</td></tr>
<tr><td></td><td>-0</td><td>-1</td><td>-2</td><td>-3</td><td>-4</td><td>-5</td><td>-6</td><td>-7</td><td>-8</td><td>-9</td><td>-A</td><td>-B</td><td>-C</td><td>-D</td><td>-E</td><td>-F</td></tr>
</table>

⇑ High nibble Low Nibble ⇒

Figure 7 Correspondence between U-string (UCS-2 form) and I8-string in UTF-8M

From (Hex)	To (Hex)	No. of Octets	No. of bits (v)	Octet Sequence	
				bits (v=0 or 1)	Hex
⇒ UTF-8M ⇒					
U-string (UCS-2 form, including the S-zone)		I8-string			
⇐ rUTF-8M ⇐					
00	1F (C0 zone)	1	8 (5)	00000000 <= 000vvvvv <= 00011111	00 <= hh <= 1F
20	7F (G0 zone + DEL)	1	8 (7)	00100000 <= 0vvvvvvv <= 01111111	20 <= hh <= 7F
80	9F (C1 zone)	1	8 (5)	10000000 <= 100vvvvv <= 10011111	80 <= hh <= 9F
A0	3FF	2	10 (10)	11000101 10100000 <= 110vvvvv 101vvvvv <= 11011111 10111111	C5 A0 <= hh hh <= DF BF
400	3FFF	3	14 (14)	11100001 10100000 10100000 <= 1110vvvv 101vvvvv 101vvvvv <= 11101111 10111111 10111111	E1 A0 A0 <= hh hh hh <= EF BF BF
Note: See section "UTF-16 and UTF-8M" on page 6 on transforming valid pairs of HI and LO S-zone RC-elements. A breakdown of the S-zone range of octets is shown in Figure 7. Single or malformed pairs are treated as single values and are transformed as shown next for the range X'4000' to X'FFFF'.					
4000	FFFF (BMP limit)	4	18 (16)	11110000 10110000 10100000 10100000 <= 1111000v 101vvvvv 101vvvvv 101vvvvv <= 11110001 10111111 10111111 10111111	F0 B0 A0 A0 <= hh hh hh hh <= F1 BF BF BF

LEGEND for entries above and for Figures 8 and 9 below:

The following describes how to read the content of the correspondence tables in Figures 7 to 9 below:

The information in the U-string to I8-string correspondence tables is arranged in six columns. The first two columns are the From and To values for the U-string in hex. The last four columns show the following information about the I8-string:

- the number of octets in the I-8 string
- the number of bits from the U-string that are contained in the I8-string -- indicated by 'v'. It is of the form M (n) -- where M is the maximum number of bits that can be carried in the number of octets assigned to I8-string, of which 'n' bits are varied to represent the 'From' to 'To' range in the first two columns.
- the octet sequence for the I8-string is shown in 'bits' form and 'hex' form in the next two columns. Each of these columns has three values shown in the form (First Value <= range <= Last Value)
  - The first value shows the sequence corresponding to the 'From' value for the U-string.
  - The second value shows the intermediate values; the 'bits' column showing the bits from the U-string distributed among the I8-string octets.
  - The third value shows the sequence corresponding to the 'To' value for the U-string.

Figure 8 *Correspondence for S-zone elements X'4000' to X'FFFF' in UTF-8M*

From (Hex)	To (Hex)	No. of Octets	No. of bits (v)	Octet Sequence	
				bits (v=0 or 1)	**Hex**
⇒ **UTF-8M** ⇒					
U-string (UCS-2 form, including the S-zone)		**I8-string**			
⇐ **rUTF-8M** ⇐					
See LEGEND for Figure 7 above on reading the contents of this table. Values shown in From-To columns will appear as 16-bit entities (or as Row and Column octet sequences in interchange) in a UCS-2 string.					
Valid Pairs of (HI, LO) S-zone RC-elements - used to transform planes 1 to 3 in UTF-16, are shown next.					
HI=D800, LO=DC00 (=10000)	**HI=D8BF, LO=DFFF (=3FFFF)**	**4**	**18 (18)**	**11110010 10100000 10100000 10100000 <= 11110vvv 101vvvvv 101vvvvv 101vvvvv <= 11110111 10111111 10111111 10111111**	**F2 A0 A0 A0 <= hh hh hh hh <= F7 BF BF BF**
Valid Pairs of (HI, LO) S-zone RC-elements - used in UTF-16 to transform planes 4 to 16, are shown next.					
HI=D8C0, LO=DC00 (=40000)	**HI=DBFF, LO=DFFF (=10FFFF)**	**5**	**22 (21)**	**11111000 10101000 10100000 10100000 10100000 <= 1111100v 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111001 10100001 10111111 10111111 10111111**	**F8 A8 A0 A0 A0 <= F8 hh hh hh hh <= F9 A1 BF BF BF**
HI - S-zone RC-elements that are used for planes 1 to 3 in UTF-16, but are not part of valid (HI, LO) pairs are shown next.					
D800	**D8BF S-zone HI (for first 3 planes)**	**4**	**18 (16)**	**11110001 10110110 10100000 10100000 <= 1111000v 101vvvvv 101vvvvv 101vvvvv <= 11110001 10110110 10100101 10111111**	**F1 B6 A0 A0 <= hh hh hh hh <= F1 B6 A5 BF**
HI - S-zone RC elements that are used for planes 4 to 16 in UTF-16, but are not part of valid (HI, LO) pairs are shown next.					
D8C0	**DBFF S-zone HI (for 4 to 16 planes)**	**4**	**18 (16)**	**11110001 10110110 10100110 10100000 <= 1111000v 101vvvvv 101vvvvv 101vvvvv <= 11110001 10110110 10111111 10111111**	**F1 B6 A6 A0 <= hh hh hh hh <= F1 B6 BF BF**
LO - S-zone RC elements that are not part of a valid (HI, LO) pair are shown next.					
DC00	**DFFF S-zone LO (for 1 to 16 planes)**	**4**	**18 (16)**	**11110001 10110111 10100000 10100000 <= 1111000v 101vvvvv 101vvvvv 101vvvvv <= 11110001 10110111 10111111 10111111**	**F1 B7 A0 A0 <= hh hh hh hh <= F1 B7 BF BF**

Figure 9 Correspondence between U-string (UCS-4 form) and I8-string in UTF-8M

From (Hex)	To (Hex)	No. of Octets	No. of bits (v)	Octet Sequence	
				bits (v=0 or 1)	Hex
⇒ UTF-8M ⇒					
U-string (UCS-2 form, including the S-zone)		I8-string			
⇐ rUTF-8M ⇐					
See LEGEND for Figure 7 above on reading the contents of this table. Values shown in From-To columns will appear as 16-bit entities (or as Row and Column octet sequences in interchange) in a UCS-2 string.					
00	1F (C0 zone)	1	8 (5)	00000000 <= 000vvvvv <= 00011111	00 <= hh <= 1F
20	7F (G0 zone + DEL)	1	8 (7)	00100000 <= 0vvvvvvv <= 01111111	20 <= hh <= 7F
80	9F (C1 zone)	1	8 (5)	10000000 <= 100vvvvv <= 10011111	80 <= hh <= 9F
A0	3FF	2	10 (10)	11000101 10100000 <= 110vvvvv 101vvvvv <= 11011111 10111111	C5 A0 <= hh hh <= DF BF
400	3FFF	3	14 (14)	11100001 10100000 10100000 <= 1110vvvv 101vvvvv 101vvvvv <= 11101111 10111111 10111111	E1 A0 A0 <= hh hh hh <= EF BF BF
Note: See section "UTF-16 and UTF-8M" on page 6 on transforming valid pairs of HI and LO S-zone RC-elements. Single or malformed pairs are treated as single values and are transformed as shown next for the range X'4000' to X'FFFF'.					
4000	FFFF (BMP limit)	4	18 (16)	11110000 10110000 10100000 10100000 <= 1111000v 101vvvvv 101vvvvv 101vvvvv <= 11110001 10111111 10111111 10111111	F0 B0 A0 A0 <= hh hh hh hh <= F1 BF BF BF
10000	3FFFF (Planes 1 to 3)	4	18 (18)	11110010 10100000 10100000 10100000 <= 11110vvv 101vvvvv 101vvvvv 101vvvvv <= 11110111 10111111 10111111 10111111	F2 A0 A0 A0 <= hh hh hh hh <= F7 BF BF BF
40000	10FFFF (Planes 4 to 16)	5	22 (21)	11111000 10101000 10100000 10100000 10100000 <= 1111100v 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111001 10100001 10111111 10111111 10111111	F8 A8 A0 A0 A0 <= F8 hh hh hh hh <= F9 A1 BF BF BF

From (Hex)	To (Hex)	No. of Octets	No. of bits (v)	Octet Sequence	
				bits (v=0 or 1)	Hex
⇒ UTF-8M ⇒					
U-string (UCS-2 form, including the S-zone)		I8-string			
⇐ rUTF-8M ⇐					
See LEGEND for Figure 7 above on reading the contents of this table. Values shown in From-To columns will appear as 16-bit entities (or as Row and Column octet sequences in interchange) in a UCS-2 string.					
110000	3FFFFF	5	22 (22)	11111001 10100010 10100000 10100000 10100000 <= 111110vv 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111011 10111111 10111111 10111111 10111111	F9 A2 A0 A0 A0 <= hh hh hh hh hh <= FB BF BF BF BF
400000	3FFFFFF	6	26 (26)	11111100 10100100 10100000 10100000 10100000 10100000 <= 1111110v 101vvvvv 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111101 10111111 10111111 10111111 10111111 10111111	FC A4 A0 A0 A0 A0 <= hh hh hh hh hh hh <= FD BF BF BF BF BF
4000000	3FFFFFFF	7	30 (30)	11111110 10100010 10100000 10100000 10100000 10100000 10100000 <= 11111110 101vvvvv 101vvvvv 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111110 10111111 10111111 10111111 10111111 10111111 10111111	FE A2 A0 A0 A0 A0 A0 <= FE hh hh hh hh hh hh <= FE BF BF BF BF BF BF
40000000	7FFFFFFF	7 (Special Lead Byte)	31 (31)	11111111 10100000 10100000 10100000 10100000 10100000 10100000 <= 11111111 101vvvvv 101vvvvv 101vvvvv 101vvvvv 101vvvvv 101vvvvv <= 11111111 10111111 10111111 10111111 10111111 10111111 10111111	FF A0 A0 A0 A0 A0 A0 <= hh hh hh hh hh hh hh <= FF BF BF BF BF BF BF

Figure 10   ISO-8 Controls (C0, C1, DEL) to/from EBCDIC controls (incl. EO)

ISO/IEC 6429 NAME		Hex	Hex		EBCDIC Name
NULL	NUL	**00**	**00**	NUL	NULL
START OF HEADER	SOH	**01**	**01**	SOH	START OF HEADING
START OF TEXT	STX	**02**	**02**	STX	START OF TEXT
END OF TEXT	ETX	**03**	**03**	ETX	END OF TEXT
END OF TRANSMISSION	EOT	**04**	**37**	EOT	END OF TRANSMISSION
ENQUIRY	ENQ	**05**	**2D**	ENQ	ENQUIRY
ACKNOWLEDGE	ACK	**06**	**2E**	ACK	ACKNOWLEDGE
BELL	BEL	**07**	**2F**	BEL	BELL
BACKSPACE	BS	**08**	**16**	BS	BACKSPACE
CHARACTER TABULATION	HT	**09**	**05**	HT	HORIZONTAL TABULATION
LINE FEED	LF	**0A**	**25**	LF	LINE FEED
LINE TABULATION	VT	**0B**	**0B**	VT	VERTICAL TABULATION
FORM FEED	FF	**0C**	**0C**	FF	FORM FEED
CARRIER RETURN	CR	**0D**	**0D**	CR	CARRIAGE RETURN
SHIFT-OUT	SO	**0E**	**0E**	SO	SHIFT OUT
LOCKING-SHIFT ONE	LS1	**0E**	**0E**	SO	SHIFT OUT
SHIFT-IN	SI	**0F**	**0F**	SI	SHIFT IN
LOCKING-SHIFT ZERO	LS0	**0F**	**0F**	SI	SHIFT IN
DATA LINK ESCAPE	DLE	**10**	**10**	DLE	DATA LINK ESCAPE
DEVICE CONTROL ONE	DC1	**11**	**11**	DC1	DEVICE CONTROL ONE
DEVICE CONTROL TWO	DC2	**12**	**12**	DC2	DEVICE CONTROL TWO
DEVICE CONTROL THREE	DC3	**13**	**13**	DC3	DEVICE CONTROL THREE
DEVICE CONTROL FOUR	DC4	**14**	**3C**	DC4	DEVICE CONTROL FOUR
NEGATIVE ACKNOWLEDGE	NAK	**15**	**3D**	NAK	NEGATIVE ACKNOWLEDGE
SYNCHRONOUS IDLE	SYN	**16**	**32**	SYN	SYNCHRONOUS IDLE
END OF TRANSMISSION BLOCK	ETB	**17**	**26**	ETB	END OF TRANSMISSION BLOCK
CANCEL	CAN	**18**	**18**	CAN	CANCEL
END OF MEDIA	EM	**19**	**19**	EM	END OF MEDIUM
SUBSTITUTE	SUB	**1A**	**3F**	SUB	SUBSTITUTE
ESCAPE CHARACTER	ESC	**1B**	**27**	ESC	ESCAPE
INFORMATION SEPARATOR FOUR	IS4	**1C**	**1C**	IFS	INFORMATION FILE SEPARATOR
INFORMATION SEPARATOR THREE	IS3	**1D**	**1D**	IGS	INFORMATION GROUP SEPARATOR
INFORMATION SEPARATOR TWO	IS2	**1E**	**1E**	IRS	INFORMATION RECORD SEPARATOR
INFORMATION SEPARATOR ONE	IS1	**1F**	**1F**	IUS/ITB	INFORMATION UNIT SEPARATOR
DELETE	DEL	**7F**	**07**	DEL	DELETE
RESERVED	xxx	**80**	**20**	DS	DIGIT SELECT
RESERVED	xxx	**81**	**21**	SOS	START OF SIGNIFICANCE
BREAK PERMITTED HERE	BPH	**82**	**22**	FS	FIELD SEPARATOR
NO BREAK HERE	NBH	**83**	**23**	WUS	WORD UNDERSCORE
INDEX	IND	**84**	**24**	BYP/INP	BYPASS OR INHIBIT PRESENTATION
NEXT LINE	NEL	**85**	**15**	NL	NEW LINE
START OF SELECTED AREA	SSA	**86**	**06**	RNL	REQUIRED NEW LINE
END OF SELECTED AREA	ESA	**87**	**17**	POC	PROGRAM OPERATOR COMMUNICATION
CHARACTER TABULATION SET	HTS	**88**	**28**	SA	SET ATTRIBUTE
CHARACTER TABULATION WITH JUSTIFICATION	HTJ	**89**	**29**	SFE	START FIELD EXTENDED
LINE TABULATION SET	VTS	**8A**	**2A**	SM/SW	SET MODE OR SWITCH
PARTIAL LINE DOWN	PLD	**8B**	**2B**	CSP	CONTROL SEQUENCE PREFIX
PARTIAL LINE UP	PLU	**8C**	**2C**	MFA	MODIFY FIELD ATTRIBUTE
REVERSE LINE FEED (OR REVERSE INDEX)	RI	**8D**	**09**	SPS	SUPERSCRIPT
SINGLE SHIFT TWO	SS2	**8E**	**0A**	RPT	REPEAT
SINGLE SHIFT THREE	SS3	**8F**	**1B**	CU1	CUSTOMER USE ONE
DEVICE CONTROL STRING	DCS	**90**	**30**	xxx	RESERVED
PRIVATE USE ONE	PU1	**91**	**31**	xxx	RESERVED
PRIVATE USE TWO	PU2	**92**	**1A**	UBS	UNIT BACK SPACE
SET TRANSMIT STATE	STS	**93**	**33**	IR	INDEX RETURN
CANCEL CHARACTER	CCH	**94**	**34**	PP	PRESENTATION POSITION
MESSAGE WAITING	MW	**95**	**35**	TRN	TRANSPARENT
START OF GUARDED AREA	SPA	**96**	**36**	NBS	NUMERIC BACKSPACE
END OF GUARDED AREA	EPA	**97**	**08**	GE	GRAPHIC ESCAPE

ISO/IEC 6429 NAME		Hex	Hex		EBCDIC Name
START OF STRING	SOS	**98**	**38**	SBS	SUBSCRIPT
RESERVED	xxx	**99**	**39**	IT	INDENT TABULATION
SINGLE CHARACTER INTRODUCER	SCI	**9A**	**3A**	RFF	REVERSE FORM FEED
CONTROL SEQUENCE INTRODUCER	CSI	**9B**	**3B**	CU3	CUSTOMER USE THREE
STRING TERMINATOR	ST	**9C**	**04**	SEL	SELECT
OPERATING SYSTEM COMMAND	OSC	**9D**	**14**	RES/ ENP	RESTORE / ENABLE PRESENTATION
PRIVACY MESSAGE	PM	**9E**	**3E**	xxx	RESERVED
APPLICATION PROGRAM COMMAND	APC	**9F**	**FF**	EO	EIGHT ONES

Figure 11 Character correspondences for G0 set (X'20'--X'7E')

Hex	Glyph	GCGID	UCS Name	Hex	Var	Hex	=/#	Hex	=/#	Hex	=/#
ISO/IEC 8859-1 (CPGID 819)				**1047**		**500**		**37**		**290**	
Note: The I / V in the Var(iant)column indicates if the character is part of GCSGID 640 (I, Invariant) or not (V, Variant); '=' sign in the '=/#' column indicates the code point under the Hex column equals the code point under the Hex column for CPGID 1047, and a '#' sign indicates inequality. GCGID is the IBM Graphic Character Global Identifier assigned to the character in an IBM Registry (published in IBM CDRA). Digits 0 to 9, letters 'a' to 'z' , and 'A' to 'Z' are NOT included in the table below. The =/# column for CPGID 290 would have been marked with '#' for the set of letters 'a' to 'z' (lowercase only), if these characters were included in this table. See Figure 3 on page 10 or Figure 13 on page 20 for their EBCDIC code positions.											
20	SPACE	SP01	SPACE	**40**	I	40	=	40	=	40	=
21	!	SP02	EXCLAMATION MARK	**5A**	**V**	4F	**#**	5A	=	5A	=
22	"	SP04	QUOTATION MARK	**7F**	I	7F	=	7F	=	7F	=
23	#	SM01	NUMBER SIGN	**7B**	**V**	7B	=	7B	=	7B	=
24	$	SC03	DOLLAR SIGN	**5B**	**V**	5B	=	5B	=	E0	**#**
25	%	SM02	PERCENT SIGN	**6C**	I	6C	=	6C	=	6C	=
26	&	SM03	AMPERSAND	**50**	I	50	=	50	=	50	=
27	'	SP05	APOSTROPHE	**7D**	I	7D	=	7D	=	7D	=
28	(	SP06	LEFT PARENTHESIS	**4D**	I	4D	=	4D	=	4D	=
29	)	SP07	RIGHT PARENTHESIS	**5D**	I	5D	=	5D	=	5D	=
2A	*	SM04	ASTERISK	**5C**	I	5C	=	5C	=	5C	=
2B	+	SA01	PLUS SIGN	**4E**	I	4E	=	4E	=	4E	=
2C	,	SP08	COMMA	**6B**	I	6B	=	6B	=	6B	=
2D	-	SP10	HYPHEN-MINUS	**60**	I	60	=	60	=	60	=
2E	.	SP11	FULL STOP	**4B**	I	4B	=	4B	=	4B	=
2F	/	SP12	SOLIDUS	**61**	I	61	=	61	=	61	=
3A	:	SP13	COLON	**7A**	I	7A	=	7A	=	7A	=
3B	;	SP14	SEMICOLON	**5E**	I	5E	=	5E	=	5E	=
3C	<	SA03	LESS-THAN SIGN	**4C**	I	4C	=	4C	=	4C	=
3D	=	SA04	EQUALS SIGN	**7E**	I	7E	=	7E	=	7E	=
3E	>	SA05	GREATER-THAN SIGN	**6E**	I	6E	=	6E	=	6E	=
3F	?	SP15	QUESTION MARK	**6F**	I	6F	=	6F	=	6F	=
40	@	SM05	COMMERCIAL AT	**7C**	**V**	7C	=	7C	=	7C	=
5B	[	SM06	LEFT SQUARE BRACKET	**AD**	**V**	4A	**#**	BA	**#**	70	**#**
5C	\	SM07	REVERSE SOLIDUS	**E0**	**V**	E0	=	E0	=	B2	**#**
5D	]	SM08	RIGHT SQUARE BRACKET	**BD**	**V**	5A	**#**	BB	**#**	80	**#**
5E	^	SD15	CIRCUMFLEX ACCENT	**5F**	**V**	5F	=	B0	**#**	B0	**#**
5F	_	SP09	LOW LINE	**6D**	I	6D	=	6D	=	6D	=
60	`	SD13	GRAVE ACCENT	**79**	**V**	79	=	79	=	79	=
7B	{	SM11	LEFT CURLY BRACKET	**C0**	**V**	C0	=	C0	=	C0	=
7C	|	SM13	VERTICAL LINE	**4F**	**V**	BB	**#**	4F	=	4F	=
7D	}	SM14	RIGHT CURLY BRACKET	**D0**	**V**	D0	=	D0	=	D0	=
7E	~	SD19	TILDE	**A1**	**V**	A1	=	A1	=	A0	**#**

Figure 12 I8-string to/from E-string octet conversion tables

From I8-string to E-string

⇓ High nibble Low Nibble ⇒

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	00	01	02	03	37	2D	2E	2F	16	05	25	0B	0C	0D	0E	0F
1-	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
2-	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3-	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4-	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5-	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	AD	E0	BD	5F	6D
6-	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7-	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	A1	07
8-	20	21	22	23	24	15	06	17	28	29	2A	2B	2C	09	0A	1B
9-	30	31	1A	33	34	35	36	08	38	39	3A	3B	04	14	3E	FF
A-	80	8C	8D	8E	8F	90	9C	9D	9E	9F	A0	AC	AE	AF	BC	BE
B-	BF	CC	CD	CE	CF	DC	DD	DE	DF	EC	ED	EE	EF	FC	FD	FE
C-	42	43	44	45	46	47	48	49	52	53	54	55	56	57	58	59
D-	62	63	64	65	66	67	68	69	71	72	73	74	75	76	77	78
E-	8A	9A	AA	BA	CA	DA	EA	FA	8B	9B	AB	BB	CB	DB	EB	FB
F-	B2	B3	B4	B5	B6	B7	B8	B9	6A	70	B0	B1	41	51	4A	E1

From E-string to I8-string

⇓ High nibble Low Nibble ⇒

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	00	01	02	03	9C	09	86	7F	97	8D	8E	0B	0C	0D	0E	0F
1-	10	11	12	13	9D	85	08	87	18	19	92	8F	1C	1D	1E	1F
2-	80	81	82	83	84	0A	17	1B	88	89	8A	8B	8C	05	06	07
3-	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
4-	20	FC	C0	C1	C2	C3	C4	C5	C6	C7	FE	2E	3C	28	2B	7C
5-	26	FD	C8	C9	CA	CB	CC	CD	CE	CF	21	24	2A	29	3B	5E
6-	2D	2F	D0	D1	D2	D3	D4	D5	D6	D7	F8	2C	25	5F	3E	3F
7-	F9	D8	D9	DA	DB	DC	DD	DE	DF	60	3A	23	40	27	3D	22
8-	A0	61	62	63	64	65	66	67	68	69	E0	E8	A1	A2	A3	A4
9-	A5	6A	6B	6C	6D	6E	6F	70	71	72	E1	E9	A6	A7	A8	A9
A-	AA	7E	73	74	75	76	77	78	79	7A	E2	EA	AB	5B	AC	AD
B-	FA	FB	F0	F1	F2	F3	F4	F5	F6	F7	E3	EB	AE	5D	AF	B0
C-	7B	41	42	43	44	45	46	47	48	49	E4	EC	B1	B2	B3	B4
D-	7D	4A	4B	4C	4D	4E	4F	50	51	52	E5	ED	B5	B6	B7	B8
E-	5C	FF	53	54	55	56	57	58	59	5A	E6	EE	B9	BA	BB	BC
F-	30	31	32	33	34	35	36	37	38	39	E7	EF	BD	BE	BF	9F

Figure 13 Categories of octets in E-string converted from I8-string

LEGEND:

mmm MMM	Control characters (immm = ISO 646 MMM = EBCDIC mnemonic)	▨	Variant characters from IRV of ISO 646
░	Invariant characters from IRV of ISO 646	□	tt =- trailing octet; 22, 33, 44, 55, 66, 77 = First of 2-octet, first of 3-octet, .. , or first of 7-octet sequence

⇓ High nibble Low Nibble ⇒

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	nul NUL	soh SOH	stx STX	etx ETX	st SEL	ht HT	ssa RNL	del DEL	epa GE	ri SPS	ss2 RPT	vt VT	ff FF	cr CR	so/ ls1 SO	si/ ls0 SI
1-	dle DLE	dc1 DC1	dc2 DC2	dc3 DC3	osc RES/ ENP	nel NL	bs BS	esa POC	can CAN	em EM	pu2 UBS	ss3 CU1	is4 IFS	is3 IGS	is2 IRS	is1 IUS/ ITB
2-	xxx DS	xxx SOS	bph FS	nbh WUS	ind BYP/ INP	lf LF	etb ETB	esc ESC	hts SA	htj SFE	vts SM/ SW	pld CSP	plu MFA	enq ENQ	ack ACK	bel BEL
3-	dcs XXX	pu1 XXX	syn SYN	sts IR	cch PP	mw TRN	spa NBS	eot EOT	sos SBS	xxx IT	sci RFF	csi CU3	dc4 DC4	nak NAK	pm XXX	sub SUB
4-	SP	66	22	22	22	22	22	22	22	22	77	.	<	(	+	|
5-	&	66	22	22	22	22	22	22	22	22	!	$	*	)	;	^
6-	-	/	22	22	22	22	22	22	22	22	55	,	%	_	>	?
7-	55	22	22	22	22	22	22	22	22	`	:	#	@	'	=	"
8-	tt	a	b	c	d	e	f	g	h	i	33	33	tt	tt	tt	tt
9-	tt	j	k	l	m	n	o	p	q	r	33	33	tt	tt	tt	tt
A-	tt	~	s	t	u	v	w	x	y	z	33	33	tt	[	tt	tt
B-	55	55	44	44	44	44	44	44	44	44	33	33	tt	]	tt	tt
C-	{	A	B	C	D	E	F	G	H	I	33	33	tt	tt	tt	tt
D-	}	J	K	L	M	N	O	P	Q	R	33	33	tt	tt	tt	tt
E-	\	77	S	T	U	V	W	X	Y	Z	33	33	tt	tt	tt	tt
F-	0	1	2	3	4	5	6	7	8	9	33	33	tt	tt	tt	apc EO

Figure 14 *Shadow flags associated with E-string octets*

LEGEND:
0 = Single octet Control character **1 = Single octet Graphic character**
2 = Lead octet of a 2-octet string **3 = Lead octet of a 3-octet string**
4 = Lead octet of a 4-octet string **5 = Lead octet of a 5-octet string**
6 = Lead octet of a 6-octet string **7 = Lead octet of a 7-octet string**
9 = A trailing octet of a multi-octet string

⇓ High nibble **Low Nibble ⇒**

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4-	1	6	2	2	2	2	2	2	2	2	7	1	1	1	1	1
5-	1	6	2	2	2	2	2	2	2	2	1	1	1	1	1	1
6-	1	1	2	2	2	2	2	2	2	2	5	1	1	1	1	1
7-	5	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
8-	9	1	1	1	1	1	1	1	1	1	3	3	9	9	9	9
9-	9	1	1	1	1	1	1	1	1	1	3	3	9	9	9	9
A-	9	1	1	1	1	1	1	1	1	1	3	3	9	1	9	9
B-	5	5	4	4	4	4	4	4	4	4	3	3	9	1	9	9
C-	1	1	1	1	1	1	1	1	1	1	3	3	9	9	9	9
D-	1	1	1	1	1	1	1	1	1	1	3	3	9	9	9	9
E-	1	7	1	1	1	1	1	1	1	1	3	3	9	9	9	9
F-	1	1	1	1	1	1	1	1	1	1	3	3	9	9	9	0

ANNEX - Intellectual Property Related

Transcript of Letter
regarding Disclosure of IBM Technology - EF-UTF
(Hard copy available on request from V.S. Umamaheswaran, umavs@ca.ibm.com)
Transcribed on 1998-07-11

======================================

International Business Machines Corporation

IBM LOGO
Route 100
Somers, NY 10589

June 2, 1998

The Chair, Unicode Technical Committee

Subject: Disclosure of IBM Technology - EBCDIC-Friendly UCS Transformation Format (EF-UTF)

The attached document entitled "EBCDIC-Friendly UCS Transformation Format (EF-UTF)" contains IBM technology that has been filed for application for Canadian Patent. However, IBM believes that the technology could be beneficial to the EBCDIC community at large; allowing the community to derive the enormous benefits provided by UCS (ISO/IEC 10646 and Unicode).

This letter is to inform you that IBM is pleased to make the attached documentation, and the associated technology that has been filed for patent, freely available to anyone concerned towards making the transformation format as part of the UCS standards.

Sincerely

SIGNED

Elizabeth G. Nichols
Director of National Language Support
and Information Development

EGN:ghs
Attachment

======================================

=========== END OF DOCUMENT ===========