L2/03-386 *ISO/IEC JTC1/SC2/WG2 N2677*

ISO/IEC JTC 1/SC 2/WG 2
PROPOSAL SUMMARY FORM TO ACCOMPANY SUBMISSIONS
FOR ADDITIONS TO THE REPERTOIRE OF ISO/IEC 10646[1]
Please fill all the sections A, B and C below.
(Please read Principles and Procedures Document for guidelines and details before filling this form.)
See http://www.dkuug.dk/JTC1/SC2/WG2/docs/summaryform.html for latest *Form*.
See http://www.dkuug.dk/JTC1/SC2/WG2/docs/principles.html for latest *Principles and Procedures* document.
See http://www.dkuug.dk/JTC1/SC2/WG2/docs/roadmaps.html for latest roadmaps.

A. Administrative

1. **Title:** ***Adition of six Number forms characters***
2. Requester's name: *Ricardo Cancho Niemietz*
3. Requester type (Member body/Liaison/Individual contribution): *Individual contribution*
4. Submission date: *2003-10-21*
5. Requester's reference (if applicable): ______
6. (Choose one of the following:)
This is a complete proposal: *Yes*
or, More information will be provided later: ______

B. Technical - General

1. (Choose one of the following:)
   a. This proposal is for a new script (set of characters): *No*
      Proposed name of script: ______
   . b. The proposal is for addition of character(s) to an existing block: *Yes*
      Name of the existing block: *Number Forms*
2. Number of characters in proposal: *6*
3. Proposed category (see section II, Character Categories): *B.1*
4. Proposed Level of Implementation (1, 2 or 3) (see clause 14, ISO/IEC 10646-1: 2000): *1*
   Is a rationale provided for the choice? ______
   If Yes, reference: ______
5. Is a repertoire including character names provided? *Yes*
   a. If YES, are the names in accordance with the 'character naming guidelines in Annex L of ISO/IEC 10646-1: 2000? *Yes*
   b. Are the character shapes attached in a legible form suitable for review? *Yes*
6. Who will provide the appropriate computerized font (ordered preference: True Type, or PostScript format) for publishing the standard? *Ricardo Cancho Niemietz*
   If available now, identify source(s) for the font (include address, e-mail, ftp-site, etc.) and indicate the tools used:
   Ricardo Cancho Niemietz (rcancho@tiscali.es)
7. References:
   a. Are references (to other character sets, dictionaries, descriptive texts etc.) provided? *Yes*
   b. Are published examples of use (such as samples from newspapers, magazines, or other sources) of proposed characters attached? *No*
8. Special encoding issues:
   Does the proposal address other aspects of character data processing (if applicable) such as input, presentation, sorting, searching, indexing, transliteration etc. (if yes please enclose information)?
   Specifications enclosed
9. Additional Information:
Submitters are invited to provide any additional information about Properties of the proposed Character(s) or Script that will assist in correct understanding of and correct linguistic processing of the proposed character(s) or script. Examples of such properties are: Casing information, Numeric information, Currency information, Display behaviour information such as line breaks, widths etc., Combining behaviour, Spacing behaviour, Directional behaviour, Default Collation behaviour, relevance in Mark Up contexts, Compatibility equivalence and other Unicode normalization related information. See the Unicode standard at http://www.unicode.org for such information on other scripts. Also see http://www.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html and associated Unicode Technical Reports for information needed for consideration by the Unicode Technical Committee for inclusion in the Unicode Standard.

C. Technical - Justification

1. Has this proposal for addition of character(s) been submitted before? *No*
   If YES explain
2. Has contact been made to members of the user community (for example: National Body, user groups of the script or characters, other experts, etc.)? *No*
   If YES, with whom?
   If YES, available relevant documents:
3. Information on the user community for the proposed characters (for example: size, demographics, information technology use, or publishing use) is included? *No*
   Reference:
4. The context of use for the proposed characters (type of use; common or rare) *Common*
   Reference: *Publishers of specialized books about computer software and digital hardware*
5. Are the proposed characters in current use by the user community? *Yes*
   If YES, where? Reference: *Editors thru workaround*
6. After giving due considerations to the principles in *Principles and Procedures document* (a WG 2 standing document) must the proposed characters be entirely in the BMP? *Yes*
   If YES, is a rationale provided? *Yes*
   If YES, reference: *Enclosed*
7. Should the proposed characters be kept together in a contiguous range (rather than being scattered)? *Yes*
8. Can any of the proposed characters be considered a presentation form of an existing character or character sequence? *Yes*
   If YES, is a rationale for its inclusion provided? *Yes*
   If YES, reference: *Enclosed*
9. Can any of the proposed characters be encoded using a composed character sequence of either existing characters or other proposed characters? *No*
   If YES, is a rationale for its inclusion provided?
   If YES, reference:
10. Can any of the proposed character(s) be considered to be similar (in appearance or function) to an existing character? *Yes*
    If YES, is a rationale for its inclusion provided? *Yes*
    If YES, reference: *Enclosed*
11. Does the proposal include use of combining characters and/or use of composite sequences (see clauses 4.12 and 4.14 in ISO/IEC 10646-1: 2000)? *No*
    If YES, is a rationale for such use provided?
    If YES, reference:
    Is a list of composite sequences and their corresponding glyph images (graphic symbols) provided?
    If YES, reference:
12. Does the proposal contain characters with any special properties such as control function or similar semantics? *No*
    If YES, describe in detail (include attachment if necessary)
13. Does the proposal contain any Ideographic compatibility character(s)? *No*
    If YES, is the equivalent corresponding unified ideographic character(s) identified?
    If YES, reference:

[1] Form number: N2352-F (Original 1994-10-14; Revised 1995-01, 1995-04, 1996-04, 1996-08, 1999-03, 2001-05, 2001-09)

Submitter's Responsibilities

The national body or liaison organization (or any other organization or an individual) proposing new character(s) or a new script shall provide:

1. Proposed category for the script or character(s), character name(s), and description of usage.
2. Justification for the category and name(s).
3. A representative glyph(s) image on paper:
   If the proposed glyph image is similar to a glyph image of a previously encoded ISO/IEC 10646 character, then additional justification for encoding the new character shall be provided.
   Note: Any proposal that suggests that one or more of such variant forms is actually a distinct character requiring separate encoding, should provide detailed, printed evidence that there is actual, contrastive use of the variant form(s). It is insufficient for a proposal to claim a requirement to encode as characters in the Standard, glyphic forms which happen to occur in another character encoding that did not follow the Character-Glyph Model that guides the choice of appropriate characters for encoding in ISO/IEC 10646.
   Note*:* WG 2 has resolved in Resolution M38.12 not to add any more Arabic presentation forms to the standard and suggests users to employ appropriate input methods, rendering and font technologies to meet the user requirements.
4. Mappings to accepted sources, for example, other standards, dictionaries, accessible published materials
5. Computerized/camera-ready font:
   Prior to the preparation of the final text of the next amendment or version of the standard a suitable computerized font (camera-ready font) will be needed. Camera-ready copy is mandatory for final text of any pDAMs before the next revision. Ordered preference of the fonts is True Type or PostScript format. The minimum design resolution for the font is 96 by 96 dots matrix, for presentation at or near 22 points in print size.
6. List of all the parties consulted.
7. Equivalent glyph images:
   If the submission intends using composite sequences of proposed or existing combining and non-combining characters, a list consisting of each composite sequence and its corresponding glyph image shall be provided to better understand the intended use.
8. Compatibility equivalents:
   If the submission includes compatibility ideographic characters, identify the equivalent unified CJK Ideograph character(s).
9. Any additional information that will assist in correct understanding of the different characteristics and linguistic processing of the proposed character(s) or script.

	215	216	217	218
0	▨	Ⅰ 2160	ⅰ 2170	ↀ 2180
1	▨	Ⅱ 2161	ⅱ 2171	ↁ 2181
2	▨	Ⅲ 2162	ⅲ 2172	ↂ 2182
3	⅓ 2153	Ⅳ 2163	ⅳ 2173	Ↄ 2183
4	⅔ 2154	Ⅴ 2164	ⅴ 2174	▨
5	⅕ 2155	Ⅵ 2165	ⅵ 2175	▨
6	⅖ 2156	Ⅶ 2166	ⅶ 2176	▨
7	⅗ 2157	Ⅷ 2167	ⅷ 2177	▨
8	⅘ 2158	Ⅸ 2168	ⅸ 2178	▨
9	⅙ 2159	Ⅹ 2169	ⅹ 2179	▨
A	⅚ 215A	Ⅺ 216A	ⅺ 217A	A 218A
B	⅛ 215B	Ⅻ 216B	ⅻ 217B	B 218B
C	⅜ 215C	Ⅼ 216C	ⅼ 217C	C 218C
D	⅝ 215D	Ⅽ 216D	ⅽ 217D	D 218D
E	⅞ 215E	Ⅾ 216E	ⅾ 217E	E 218E
F	⅟ 215F	Ⅿ 216F	ⅿ 217F	F 218F

Fractions

Other fraction number forms are found in the Latin-1 Supplement block.

		→ 00BC ¼ vulgar fraction one quarter
		→ 00BD ½ vulgar fraction one half
		→ 00BE ¾ vulgar fraction three quarters
2153	⅓	VULGAR FRACTION ONE THIRD ≈ 0031 1 2044 ⁄ 0033 3
2154	⅔	VULGAR FRACTION TWO THIRDS ≈ 0032 2 2044 ⁄ 0033 3
2155	⅕	VULGAR FRACTION ONE FIFTH ≈ 0031 1 2044 ⁄ 0035 5
2156	⅖	VULGAR FRACTION TWO FIFTHS ≈ 0032 2 2044 ⁄ 0035 5
2157	⅗	VULGAR FRACTION THREE FIFTHS ≈ 0033 3 2044 ⁄ 0035 5
2158	⅘	VULGAR FRACTION FOUR FIFTHS ≈ 0034 4 2044 ⁄ 0035 5
2159	⅙	VULGAR FRACTION ONE SIXTH ≈ 0031 1 2044 ⁄ 0036 6
215A	⅚	VULGAR FRACTION FIVE SIXTHS ≈ 0035 5 2044 ⁄ 0036 6
215B	⅛	VULGAR FRACTION ONE EIGHTH ≈ 0031 1 2044 ⁄ 0038 8
215C	⅜	VULGAR FRACTION THREE EIGHTHS ≈ 0033 3 2044 ⁄ 0038 8
215D	⅝	VULGAR FRACTION FIVE EIGHTHS ≈ 0035 5 2044 ⁄ 0038 8
215E	⅞	VULGAR FRACTION SEVEN EIGHTHS ≈ 0037 7 2044 ⁄ 0038 8
215F	⅟	FRACTION NUMERATOR ONE ≈ 0031 1 2044 ⁄

Roman numerals

2160	Ⅰ	ROMAN NUMERAL ONE ≈ 0049 I latin capital letter i
2161	Ⅱ	ROMAN NUMERAL TWO ≈ 0049 I 0049 I
2162	Ⅲ	ROMAN NUMERAL THREE ≈ 0049 I 0049 I 0049 I
2163	Ⅳ	ROMAN NUMERAL FOUR ≈ 0049 I 0056 V
2164	Ⅴ	ROMAN NUMERAL FIVE ≈ 0056 V latin capital letter v
2165	Ⅵ	ROMAN NUMERAL SIX ≈ 0056 V 0049 I
2166	Ⅶ	ROMAN NUMERAL SEVEN ≈ 0056 V 0049 I 0049 I
2167	Ⅷ	ROMAN NUMERAL EIGHT ≈ 0056 V 0049 I 0049 I 0049 I
2168	Ⅸ	ROMAN NUMERAL NINE ≈ 0049 I 0058 X
2169	Ⅹ	ROMAN NUMERAL TEN ≈ 0058 X latin capital letter x
216A	Ⅺ	ROMAN NUMERAL ELEVEN ≈ 0058 X 0049 I
216B	Ⅻ	ROMAN NUMERAL TWELVE ≈ 0058 X 0049 I 0049 I
216C	Ⅼ	ROMAN NUMERAL FIFTY ≈ 004C L latin capital letter l
216D	Ⅽ	ROMAN NUMERAL ONE HUNDRED ≈ 0043 C latin capital letter c
216E	Ⅾ	ROMAN NUMERAL FIVE HUNDRED ≈ 0044 D latin capital letter d
216F	Ⅿ	ROMAN NUMERAL ONE THOUSAND ≈ 004D M latin capital letter m
2170	ⅰ	SMALL ROMAN NUMERAL ONE ≈ 0069 i latin small letter i
2171	ⅱ	SMALL ROMAN NUMERAL TWO ≈ 0069 i 0069 i
2172	ⅲ	SMALL ROMAN NUMERAL THREE ≈ 0069 i 0069 i 0069 i
2173	ⅳ	SMALL ROMAN NUMERAL FOUR ≈ 0069 i 0076 v
2174	ⅴ	SMALL ROMAN NUMERAL FIVE ≈ 0076 v latin small letter v
2175	ⅵ	SMALL ROMAN NUMERAL SIX ≈ 0076 v 0069 i
2176	ⅶ	SMALL ROMAN NUMERAL SEVEN ≈ 0076 v 0069 i 0069 i
2177	ⅷ	SMALL ROMAN NUMERAL EIGHT ≈ 0076 v 0069 i 0069 i 0069 i
2178	ⅸ	SMALL ROMAN NUMERAL NINE ≈ 0069 i 0078 x
2179	ⅹ	SMALL ROMAN NUMERAL TEN ≈ 0078 x latin small letter x
217A	ⅺ	SMALL ROMAN NUMERAL ELEVEN ≈ 0078 x 0069 i
217B	ⅻ	SMALL ROMAN NUMERAL TWELVE ≈ 0078 x 0069 i 0069 i
217C	ⅼ	SMALL ROMAN NUMERAL FIFTY ≈ 006C l latin small letter l
217D	ⅽ	SMALL ROMAN NUMERAL ONE HUNDRED ≈ 0063 c latin small letter c
217E	ⅾ	SMALL ROMAN NUMERAL FIVE HUNDRED ≈ 0064 d latin small letter d
217F	ⅿ	SMALL ROMAN NUMERAL ONE THOUSAND ≈ 006D m latin small letter m
2180	ↀ	ROMAN NUMERAL ONE THOUSAND C D
2181	ↁ	ROMAN NUMERAL FIVE THOUSAND
2182	ↂ	ROMAN NUMERAL TEN THOUSAND
2183	Ↄ	ROMAN NUMERAL REVERSED ONE HUNDRED = apostrophic C • used in combination with C and I to form large numbers

Hexadecimal digits ten to fifteen

218A	A	HEXADECIMAL DIGIT TEN ≈ 0041 A latin capital letter a
218B	B	HEXADECIMAL DIGIT ELEVEN ≈ 0042 B latin capital letter b
218C	C	HEXADECIMAL DIGIT TWELVE ≈ 0043 C latin capital letter c
218D	D	HEXADECIMAL DIGIT THIRTEEN ≈ 0044 D latin capital letter d
218E	E	HEXADECIMAL DIGIT FOURTEEN ≈ 0045 E latin capital letter e
218F	F	HEXADECIMAL DIGIT FIFTEEN ≈ 0046 F latin capital letter f

Background

The hexadecimal-radix (base 16) numbering system is well-known and widely used in all fashion of computer and microprocessor programming environments and languages. Every hexadecimal digit ressembles a group of four contiguous bits, called a *nibble*, and thus is quite convenient to represent the state of any grouping of adjacent bits: 8, 16, 24, 32, 64, etc, inside a computer's memory, CPU ports and integrated circuits (IC) pins.

The hexadecimal-radix numbering system intrinsically needs sixteen different digits with positional values in a range from zero to fifteen. In a lack of historical tradition in precomputer ages, and due to limitations of available characters in the primitive encoding scripts as ITA-2 (also known –incorrectly– as *baudot code*), FIELDATA, EDBIC or ASCII, arised the solution to use the decimal digits zero to nine for the first ten hexadecimal digits (those whose positional values are the same in decimal-radix numbering) and the latin letters «a» to «f» for the last six hexadecimal digits with values ten to fifteen. The glyphs usually used for that last hexadecimal digits are the upper case «A» to «F», due to primitive pre-ASCII encoding charts have only uppercase form of latin letters—a telegraphic tradition. When a complete set of both upper and lower case basic latin letters became available, the hexadecimal numbers could be displayed with the lowercase glyphs «a» to «f» as an alternate presentation, but never appears mixed the capital and small forms together in the same printed hexadecimal number.

To contextually distinguish hexadecimal numbers in a plain text stream, specially when the hexadecimal number was formed exclusively by "decimal" digits, various kind of notation arise, generally specific of a given environment. For example, the number 33 in decimal is represented in hexadecimal (21) in the following ways,

as...	in...
`$21`	6502 assembler
`021H`	8086 assembler
`0x21`	ANSI C/C++ direct value
`\x21`	ANSI C/C++ direct character by code inside a string literal
`H&21`	Microsoft VisualBasic direct value
`%21`	URL direct character by code inside a calling parameter
`=21`	MIME quoted-printable character by code inside a text/plain MIME-part body
`#000021`	HTML RGB color code

This notations appears as-is in source code, following the specific sintax of every programming language or encoding. But in books another conventions are in use, as 21h or 21_{16}, meaning the same: "that number is 16-radix". In Unicode books, codepoints (which are hexadecimal numbers) are noted by U+0021 sintax.

About the need of new characters

Since programmers can directly write programs over a console (leaving away punched-cards), they use fixed-pitch fonts to display source code and debugging output, either on screen and paper. At the beginning this was not intentionally but by simply using the available hardware (80 column screens in classical MDA video cards and compatibles, and teletype-like dolly printers). This implicit feature can print either decimal (and other subdecimal numbering formats, as binary and octal) and hexadecimal numbers in well tabulated columns.

Later, when high resolution raster devices became available, as EGA/VGA/SVGA video cards and 9- and 24-pin dot matrix/ink jetting/laser printers, graphical user interfaces (GUI) tends to be WYSIWYG ("What-You-See-Is-What-You-Get") capable. Among other things, this means the main use of proportional fonts to display both plain and rich text, but programmers still prefer using fixed-pitch fonts for source code and debugging output, just like memory dump. In some way, it emulates older console devices.

This is also the common workaround when fragments of sample source code are printed in documentation, as shown in the following figures.

Closing a window

You can close a window with the **close** method. You cannot close a frame without closing the entire parent window.
Each of the following statements closes the current window:

```
window.close()
self.close()
// Do not use the following statement in an event handler
close()
```

The following statement closes a window called *msgWindow:*

```
msgWindow.close()
```

Taken from "JavaScript Guide for JavaScript 1.1" Copyright ©1995-1996 Netscape Communications Corporation.

Creating a Foundation Project

You'll need to create a suitable Visual C++ project for this tour. With no other workspace or project loaded in the Visual C++ integrated development environment (IDE), select **New** from the File menu. Select the **Files** tab. Select **C++ Source File**, and enter **EC_Demo.cpp** as the file name. You may enter a location of your choice. Click **OK** to create the file. Type the following source code into the empty source file:

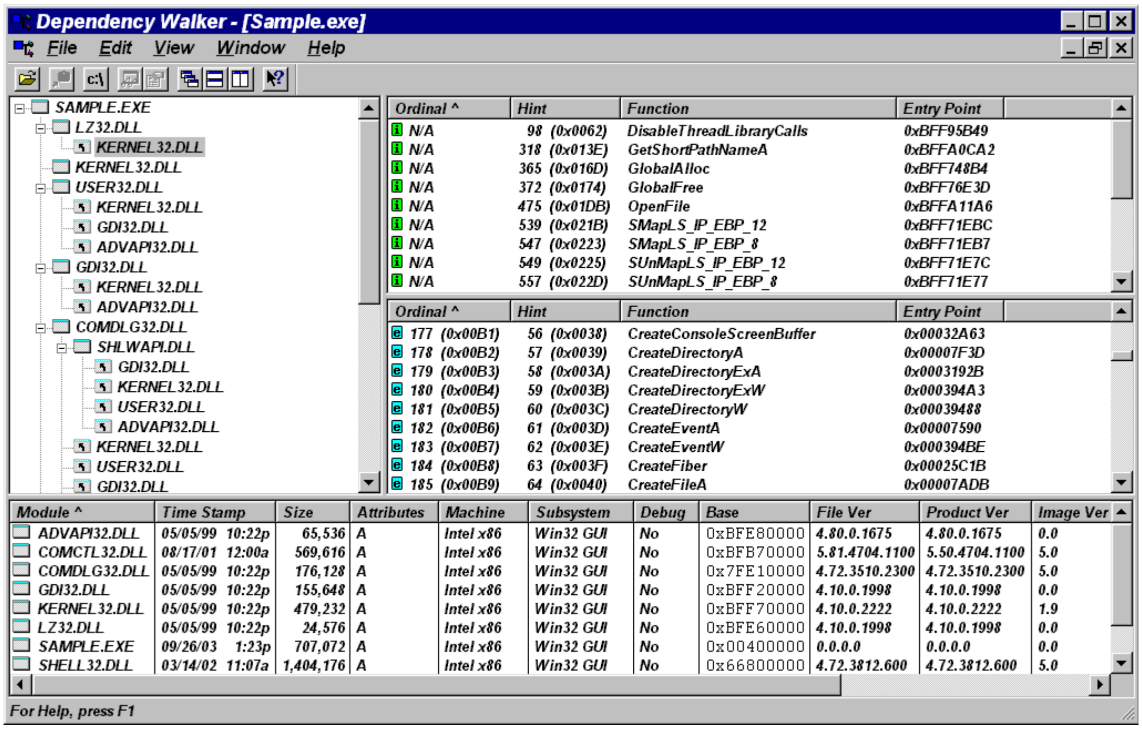
```
#include <stdio.h>

int main()
{
   printf("Introducing Edit and Continue!\n");
   return 0;
}
```

Taken from "Juny 1999 release of the MSDN Library" Copyright ©1999 Microsoft Corporation.

But out of showing source code samples, text books generally prints hexadecimal numbers using the «A» to «F» latin capital letters, which are proportional in most of fonts, mixed with «0» to «9» digits, which generally have figure-space pitch. Then, when displaying hexadecimal numbers of the same number of digits, they have different physical widths. This fact usually ruins the tabulated hexadecimal columns.

As said, the usual workaround is also to use a monospace or fixed-pitch font to print hexadecimal numbers in tables. This is tolerable in technical programming and debugging environments, in which the primary need is to monitorize memory binary data, but when printed, the tabulated hexadecimal values looks clearly in a different font family than the rest of the table or page.

The next figure shows a display output that mixes well-formed and ill-formed tabulated hexadecimal values.



```
SAMPLE.EXE
  LZ32.DLL
    KERNEL32.DLL
  KERNEL32.DLL
  USER32.DLL
    KERNEL32.DLL
    GDI32.DLL
    ADVAPI32.DLL
  GDI32.DLL
    KERNEL32.DLL
    ADVAPI32.DLL
  COMDLG32.DLL
    SHLWAPI.DLL
      GDI32.DLL
      KERNEL32.DLL
      USER32.DLL
      ADVAPI32.DLL
    KERNEL32.DLL
    USER32.DLL
    GDI32.DLL
```

Ordinal ^	Hint	Function	Entry Point
N/A	98 (0x0062)	DisableThreadLibraryCalls	0xBFF95B49
N/A	318 (0x013E)	GetShortPathNameA	0xBFFA0CA2
N/A	365 (0x016D)	GlobalAlloc	0xBFF748B4
N/A	372 (0x0174)	GlobalFree	0xBFF76E3D
N/A	475 (0x01DB)	OpenFile	0xBFFA11A6
N/A	539 (0x021B)	SMapLS_IP_EBP_12	0xBFF71EBC
N/A	547 (0x0223)	SMapLS_IP_EBP_8	0xBFF71EB7
N/A	549 (0x0225)	SUnMapLS_IP_EBP_12	0xBFF71E7C
N/A	557 (0x022D)	SUnMapLS_IP_EBP_8	0xBFF71E77

Ordinal ^	Hint	Function	Entry Point
177 (0x00B1)	56 (0x0038)	CreateConsoleScreenBuffer	0x00032A63
178 (0x00B2)	57 (0x0039)	CreateDirectoryA	0x00007F3D
179 (0x00B3)	58 (0x003A)	CreateDirectoryExA	0x0003192B
180 (0x00B4)	59 (0x003B)	CreateDirectoryExW	0x000394A3
181 (0x00B5)	60 (0x003C)	CreateDirectoryW	0x00039488
182 (0x00B6)	61 (0x003D)	CreateEventA	0x00007590
183 (0x00B7)	62 (0x003E)	CreateEventW	0x000394BE
184 (0x00B8)	63 (0x003F)	CreateFiber	0x00025C1B
185 (0x00B9)	64 (0x0040)	CreateFileA	0x00007ADB

Module ^	Time Stamp	Size	Attributes	Machine	Subsystem	Debug	Base	File Ver	Product Ver	Image Ver
ADVAPI32.DLL	05/05/99 10:22p	65,536	A	Intel x86	Win32 GUI	No	0xBFE80000	4.80.0.1675	4.80.0.1675	0.0
COMCTL32.DLL	08/17/01 12:00a	569,616	A	Intel x86	Win32 GUI	No	0xBFB70000	5.81.4704.1100	5.50.4704.1100	5.0
COMDLG32.DLL	05/05/99 10:22p	176,128	A	Intel x86	Win32 GUI	No	0x7FE10000	4.72.3510.2300	4.72.3510.2300	5.0
GDI32.DLL	05/05/99 10:22p	155,648	A	Intel x86	Win32 GUI	No	0xBFF20000	4.10.0.1998	4.10.0.1998	0.0
KERNEL32.DLL	05/05/99 10:22p	479,232	A	Intel x86	Win32 GUI	No	0xBFF70000	4.10.0.2222	4.10.0.2222	1.9
LZ32.DLL	05/05/99 10:22p	24,576	A	Intel x86	Win32 GUI	No	0xBFE60000	4.10.0.1998	4.10.0.1998	0.0
SAMPLE.EXE	09/26/03 1:23p	707,072	A	Intel x86	Win32 GUI	No	0x00400000	0.0.0.0	0.0.0.0	0.0
SHELL32.DLL	03/14/02 11:07a	1,404,176	A	Intel x86	Win32 GUI	No	0x66800000	4.72.3812.600	4.72.3812.600	5.0

Note that at the right side, in the upper and middle lists, the columns labeled “Entry Point” displays 32 bits hexadecimal values with a proportional font, and hexadecimal digits are not well aligned. But in the lower list, in the column labeled “Base”, the 32 bit hexadecimal values are displayed with a monospaced font and all the hexadecimal digits are well vertically aligned, although they do not share the same font family than the rest of data.

Another example of ill-tabulated hexadecimal colums are taken directly from the Unicode book:

When referring to code points in the Unicode Standard, the usual practice is to refer to them by their numeric value expressed in hexadecimal, with a “U+” prefix. (See *Section 0.3, Notational Convention*s.) Encoded characters can also be referred to by their code points only, but to prevent ambiguity, the official Unicode name of the character is often also added; this clearly identifies the abstract character that is encoded. For example:

U+0061 LATIN SMALL LETTER A
U+10330 GOTHIC LETTER AHSA
U+201DF CJK UNIFIED IDEOGRAPH-201DF

Such citations refer only to the encoded character per se, associating the code point (as an integral value) with the abstract character that is encoded.

Taken from “The Unicode Standard, Version 4.0, Chapter 2, General Structure” Copyright © 1991–2003 by Unicode, Inc.

Note that items “U+10330” and “U+201DF” has the same number of hexadecimal digits (five), but different widths: the «F» do not fall exactly below the «0».

In the last example, this fact has a relative relevance, but in other material with larger descripting tables this could became a serious one. This could be avoided by introducing a set of «hexadecimal digits ten to fifteen» characters, which have similar glyphs that latin letters «A» to «F» have, but share the same figure space width that decimal digits. These are the basics of this proposal.

Semantics and compatibility

- It is proposed to include this hexadecimal digits in the «number forms» script rather than others (like «letterlike symbols» script or any mathematical blocks) due to the intrinsic numeric value of every hexadecimal digit. These values must be reflected in the Unicode Character Database.

- The proposed code points U+218A..U+218F were selected due they terminate in the same hexadecimal values that the proposed characters have, and could be used as nmenonics.

- The letter-like apparience of the hexadecimal digits ten to fifteen should be considered mere coincidence, and thus no case pairs should be assigned in the Unicode Standard.

- As compatibility forms, this hexadecimal digits could to default to the latin capital letters «A» to «F». This is for display purposes only because they can show the same shapes of the hexadecimal digits ten to fifteen, and inform of the value of the hexadecimal numbers to the users, although tabulation surely becomes visually corrupted.

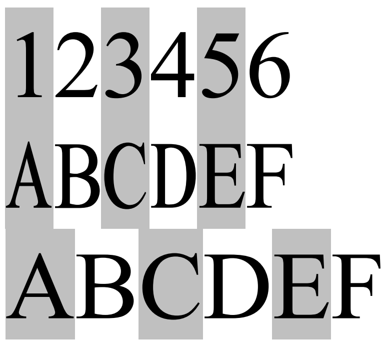
- In any case it is not recommended to use greek capital letters or cyrillic capital letter as compatibility forms. Latin capital letters «A» to «F» are the unique guaranteed to be present in any ASCII-based font.

- Note that there is no proposal here to include hexadecimal digits zero to nine; they are supposed to be unified with ASCII decimal digits, as in this sense the hypothaetical octal digits zero to seven and binary digits zero and one are.

- The non-contiguous ranges to both ASCII decimal digits and proposed hexadecimal digits it is not an issue: the range of the ASCII latin capital letters are non-contiguous with digits, too.

Typographical remarks

- When present in a font, it is mandatory that proposed characters HEXADECIMAL DIGIT TEN to HEXADECIMAL DIGIT FIFTEEN have the same figure-space pitch than decimal digits DIGIT ZERO to DIGIT NINE of the source font.



Digit figures

Hexadecimal digit figures

Proportional latin capital letters «A» to «F»

- The glyphs for this hexadecimal digits should not to be the same than that mapped to LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER F when the font is proportionally spaced (the usual case), but they should have their own specific glyphs.

  When the font is monospaced, proposed code points HEXADECIMAL DIGIT TEN to HEXADECIMAL DIGIT FIFTEEN may share the LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER F glyphs of the font, by appropiate mapping.

  When decimal digits on the font are proportional (i. e., the width of the DIGIT ONE is less than the DIGIT FOUR, for example), the font should lack this proposed characters. Such family fonts will be consider decorative fonts, not suitable for technical documentation.

- Hexadecimal digits should share the "personality" of the source font (same serifed or sans-serifed look, same weight, escapement and orientation, etc.), and they should be affected in the same way that decimal digits are when editing stylistic changes are made in a rich text document: sizing, stretching, colouring, bolding, stroking, outlining, etc.

- No switching must ocurr between capital and small size, i. e., there is only one case for this digits, not two. It is not mandatory than hexadecimal digits ten to fifteen resembles the latin capital letters «A» to «F». It could be a font designers' choice to select the latin small letters «a» to «f» apparience, although capital forms are preferred, or perhaps will be useful to standarize a variant selection of the two options, being the capital form the default and small form the variation.

- Care should be taken in desinging the HEXADECIMAL DIGIT ELEVEN («B») and HEXADECIMAL DIGIT THIRTEEN («D») glyphs, due to their similarity with decimal digits eigth «8» and zero «0», respectively. Generally this is not a trouble in serifed Times-like fonts, but in sans-serifed Helvetica-like fonts could lead to confussion, specially at small points sizes or low resolution devices. It is suggested that even in sans-serifed fonts, the HEXADECIMAL DIGIT ELEVEN («B») and HEXADECIMAL DIGIT THIRTEEN («D») glyphs renders with discrete serifs too.

Inputting and editing proposed methods (for advanced desktop publishing software only)

- For keyboard interfaces, some kind of toggle-command should be implemented to switch between write "hexadecimal numbers A-F" and "latin letters A-F", in the same fashion as the bold, italics, underlined, etc. classical toggles. While in "hex" mode, the latin capital letters «A» to «F» stroked on the keyboard (and even the latin small letters «a» to «f») would be stored with U+218A..U+218F proposed internal codes in memory.

- For selected text, some kind of toggle-command should be implemented to switch between "to hexadecimal digits A-F ↔ to latin letters A-F", in the same fashion as the "to uppercase ↔ to lowercase" classical toggle. This toggle will affect only to the capital letters «A» to «F» presents in the selected text, which becomes hexadecimal digits ten to fifteen, and these back to latin capital letters when the toggle is executed again.

- When inserting fullwidth letter forms in a far east language text that use fullwidth forms, hexadecimal digits should render to the fullwidth forms FULLWIDTH LATIN CAPITAL LETTER A (U+FF21) to FULLWIDTH LATIN CAPITAL LETTER F (U+FF26), because all fullwidth latin letters and digits have the same width, and the main goal with hexadecimal numbers is acomplished that way. Thus, FULLWIDTH LATIN CAPITAL LETTER A to FULLWIDTH LATIN CAPITAL LETTER F characters could be considered the "presentation forms" of the hexadecimal digits ten to fifteen when writting far east languages fullwidth text.

Sample font provided

This PDF document has embebbed glyphs of a TrueType font made by me called “Hexadecimal Serif”, based on popular “Times New Roman” regular font that came with standard installation of Microsoft Windows 9*x*, copyrighted by The Monotype Corporation plc. The sample include only the original digits zero to nine plus the latin capital letters «A» to «F» made in hexadecimal digits fashion (i.e., they have figure space pitch), the last also mapped as the proposed hexadecimal digits ten to fifteen U+218A..U+218F.

The glyphs are full camera-ready (in fact, they are just pure vectors), but the font lack on hinting information—kerning is not needed in this case.

No copyright notice is supplied with this sample font; the goal is that at appropiate time the owner (Monotype) implements their own glyphs, full with hinting info.

The sample Truetype font file will be send as separate file if requested.

Issues of the proposal

The main question is if this new proposed characters should be mere presentation forms or if they should be pure number forms. Both are also cases of disunification from the latin capital letters «A» to «F», a proccess that is discouraged by the Unicode Consortium.

It is true that, from an compositing point of view, proposed hexadecimal digits ten to fifteen are merely a kind of alphabetic presentation forms of the latin capital letters «A» to «F», which shares the same general visual apparience that the corresponding source font, but having a figure space pitch rather than proportional pitch. If this was the case, it implies to consider the hexadecimal digits ten to fifteen as alphabetic characters. We read in *The Unicode Standard, Version 4.0, Chapter 4 "Character propierties":*

> **4.9 Letters, Alphabetic, and Ideographic**
>
> The concept of letters is used in many contexts. Computer language standards often characterize identifiers as consisting of letters, syllables, ideographs, and digits, but do not specify exactly what a "letter," "syllable," "ideograph," or "digit" is, leaving the definitions implicitly either to a character encoding standard or to a locale specification. The large scope of the Unicode Standard means that it includes many writing systems for which these distinctions are not as self-evident as they may once have been for systems designed to work primarily for Western European languages and Japanese. In particular, while the Unicode Standard includes various "alphabets" and "syllabaries," it also includes writing systems that fall somewhere in between. As a result, no attempt is made to draw a sharp property distinction between letters and syllables.
>
> ***Alphabetic.*** The alphabetic property is an informative property of the primary units of alphabets and/or syllabaries, whether combining or noncombining. Included in this group would be composite characters that are canonical equivalents to a combining character sequence of an alphabetic base character plus one or more combining characters; letter digraphs; contextual variants of alphabetic characters; ligatures of alphabetic characters; contextual variants of ligatures; modifier letters; letterlike symbols that are compatibility equivalents of single alphabetic letters; and miscellaneous letter elements. Notably, U+00AA FEMENINE ORDINAL INDICATOR and U+00BA MASCULINE ORDINAL INDICATOR are simply abbreviatory forms involving a Latin letter and should be considered alphabetic rather than nonalphabetic symbols.

We see that under this definition of "alphabetic" the proposed characters hexadecimal digits ten to fifteen do not fit well, due they are not part of any alphabet nor syllabary *per se*, despite its shape of latin letters. In other words, the hexadecimal digits have not any latin letter semantics, and thus, they cannot be considered alphabetics in any way. Consecuently, they cannot be considered a mere presentation forms of any true alphabetic latin letter character.

In the same sense, hexadecimal digits ten to fifteen cannot be considered letterlike symbols nor mathematical symbols:

- hexadecimal digits ten to fifteen are not symbols anyway; they do not represent any abstract idea individually but positional digits in a well defined numbering system, and
- hexadecimal digits ten to fifteen are not used in any mathematical nor physical ecuation; furthermore, they are not operators anyway.

Finally, we can consider the hexadecimal digits ten to fifteen as a new Number Forms sub-block. We read in *The Unicode Standard, Version 4.0, Chapter 4 "Character propierties":*

> **4.6 Numeric Value—Normative**
>
> *Numeric value* is a normative property of characters that represent *number*s. This group includes characters such as fractions, subscripts, superscripts, Roman numerals, currency numerators, encircled numbers, and script-specific digits. In many traditional numbering systems, letters are used with a numeric value. Examples include Greek and Hebrew letters as well as Latin letters used in outlines (II.A.1.b). These special cases are not included here as numbers.

> *Decimal digits* form a large subcategory of numbers consisting of those digits that can be used to form decimal-radix numbers. They include script-specific digits, not characters such as Roman numerals (<1, 5> = 15 = fifteen, but <I, V> = IV = four), subscripts, or superscripts. Numbers other than decimal digits can be used in numerical expressions, but it is up to the user to determine the specialized uses.
>
> The Unicode Standard assigns distinct codes to the forms of digits that are specific to a given script. Examples are the digits used with the Arabic script, Chinese numbers, or those of the Indic languages. For naming conventions relevant to Arabic digits, see the introduction to *Section 8.2, Arabi*c.
>
> The Unicode Character Database gives the numeric values of Unicode characters that can represent numbers.

We see now that the proposed hexadecimal digits ten to fifteen fits well under this definition of numeric characters, because they have a well defined semantic of numbers, but it is true that they are not decimal digits:

- they have precisally known numeric values: A=10, B=11, C=12, D=13, E=14 and F=15, and the Unicode Character Database can reflect that;
- the Roman numerals exists as a precedent, with its own associated numeric values and with the added pseudorule to be mandatory serifed (for example, the text "Along 18th and 19th centuries..." should be rendered in spanish as "Durante los siglos XVIII y XIX...", not *"Durante los siglos XVIII y XIX..." when using a sans serifed font);
- they can not to be considered any traditional numbering system like ancient greek and hebrew, in which letters have numeric values, nor having the outline use of the latin letters. The hexadecimal digits ten to fifteen are true numeric characters in their own right.

As they have no alphabetic semantics, they are not affected by casing: there is not "upper" nor "lower" hexadecimal digits ten to fifteen, in the same way that there are not "upper" nor "lower" decimal digits zero to nine. The option to present hexadecimal numbers with the small case of the latin letters «a» to «f» will be considered a font's designers choice (although discouraged, because then the hexadecimal numbers can not be capitalized, which can be confuse), or perhaps to be a variant selection of the "basic" capital form of the hexadecimal digits ten to fifteen.

On the other hand, there is the question of the disunification of these characters from its ASCII counterparts. In a strict sense, this is a strong disunification, due that in up-to-date lack of specialized hexadecimal digits ten to fifteen, the common practice is to use the ASCII latin capital letters «A» to «F». Then, the main costs of the disunification would be:

- the cost of accidental confusion and mis-identification: the glyphs for the latin capital letters «A» to «F» are almost the same except for the figure space pitch, and
- keyboards from which it is easy to type the existing and now-disunified characters are practically all in the world.

But the proposed characters are not intended to be used while writting program code (as source code is ever plain text), but to be used in all kind of documentation, on paper or on screen output, where hexadecimal values often occurr—just like Unicode books. Then, the proposed hexadecimal digits ten to fifteen acts as a kind of specialized presentation forms of the latin capital letters «A» to «F» for purposes of desktop publishing of rich text.

The main goal is that allocating this proposed characters, the general purpose proportional, book-like and Unicode compatible fonts have an explicit placeholder to this figure spaced "variations" of the latin letters «A» to «F», with its own standard codepoints. In this sense, they suffer the same disunification costs than the Roman numerals «I», «V», «X», «L», «C», «D» and «M» (and some less perhaps, because Roman numerals include the small forms «i», «v», «x», «l», «c», «d» and «m» too, while the proposed hexadecimal digits ten to fifteen does not).

Following the cost/benefits analysis criteria found in the *Principles and Procedures for Allocation of New Characters and Scripts and handling of Defect Reports on Character Names (ISO/IEC JTC 1/SC 2/WG 2 N2352R)*, every question is listed below with a proposed answer.

I. Costs

1. **Is there a glyphic distinction?** Yes in proportional fonts, although certainly slighty: the proposed hexadecimal digits ten to fifteen have all a figure space pitch, the same that the decimal digits in the same base font, while latin capital letters «A» to «F» are clearly proportional. In the case of monospaced fonts, there is no difference in principle, unless *ad hoc* differences would be introduced by font designers to distinguish more clearly latin capital letters «A» to «F» from hexadecimal digits ten to fifteen. In proportional fonts, the proposed characters must be added to the glyph collection of that fonts; in monospaced fonts, they can internally map to the basic latin capital letters «A» to «F». Decorative fonts, that is, those not suitable for technical documentation, should explicitaly lack this new characters, due to generally they have even proportional decimal digits zero to nine, that is, not figure spaced ones.

2. **Is there a behaviour difference?** Yes in proportional fonts, because hexadecimal numbers written with these new proposed characters perfectly aligns vertically to form true hexadecimal numeric columns in tables. This difference do not affect to monospaced fonts, of course. Also, the hexadecimal digits ten to fifteen are not affected by case changes (to upper, to lower) of selected text that normally affects to latin letters «A» to «F» and «a» to «f». The first target is achieved by the own glyph metrics of the new characters (all figure spaced pitch); the second, by asigning no alphabetic property nor case mapping at all to these new characters in the Unicode Character Database.

3. **Is the use of the new character restricted to a new context (for example, use with a novel script)?** No. The hexadecimal numbering system is widely used.

4. **Is the use of the existing, ambiguous character instead of the proposed new character common, prevalent or established practice?** Yes, in lack of previous specialized and separate characters for the hexadecimal digits ten to fifteen. But it is not the intended goal to substitute this common practice but to add a new, best suited resource for future publications.

5. **Does the character exist in ASCII (ISO 646 IRV)?** They are the U+0041 “A” LATIN CAPITAL LETTER A to U+0046 “F” LATIN CAPITAL LETTER F, which are the most commonly used to write hexadecimal numbers, and alternatively the U+0061 “a” LATIN SMALL LETTER A to U+0066 “f” LATIN SMALL LETTER F. In fact, the latin capital letters «A» to «F» should be the compatibility characters of these new proposed characters.

II. Benefits

1. **Appearance: does disunification help to allow multilingual monofont text in an environment where this is commonly needed?** Yes, in publications of specialized books of computers programming and digital hardware. **In what way?** It allows to write true “pure” hexadecimal numbers with the same font of the main running text, instead of use of two distinct fonts, both a proportional and a monospaced depending of the context—running text or tabulated.

2. **Layout: does disunification solve common layout differences (this would mostly be true for punctuation)?** Yes, as noted before, doing the case of columns of hexadecimal numbers in tabulated material perfectly consistent with the rest of the hexadecimal and non-hexadecimal numbers of the documents.

3. **Searching/sorting: Is there a common case where disunification allows better support for these?** No. The proposed hexadecimal digits ten to fifteen can collate as regular latin letters «A» to «F» (although in a more strict sense they should sort just after the decimal digit nine, and before than the latin letter «A»—or the first letter in every alphabet available).

4. **Mapping to another standard: Is there a widely used standard that disunifies the characters in question?** No known standard actually defines these characters. **Are the characters in question the only ones that prevent cross mapping?** Actually there are not any cross mapping needs.

III. Alternatives

1. **Can the desired effect be achieved by changes to the display layer?** No, while using a single proportional font.

2. **Can the desired effect be achieved by changes to protocols?** There are no current protocols for acomplish that task.

3. **Can the desired effect be achieved by processing algorithms?** No.

Conclussion

The asserted main cost is the adding of the new six glyphs to a collection of widely used fonts for books (those which are Times-like, Helvetica-like and Courier-like, principally). Font vendors should update its main products. However, rich text documents made with this new characters generally can have embebbed font subsets (EPS, PDF), so it is not necessary to update every font installation—a prepress service bureau can generate correct plates without having the explicit font files installed. The worst case is the same that for the Roman numerals: if a given font lack to have these, the text must to switch to its compatibility forms: the latin capital letters «I», «V», «X», «L», «C», «D» and «M», in the case of the Roman numerals, and the latin capital letters «A» to «F» in the case of the hexadecimal digits ten to fifteen.

Another main cost is the adding of a new interface for inputting these new characters. In general desktop publishing software, this could be acomplished thru some kind of specialized plug-in extension—just like that to write mathematical formulae or musical scores—or by customization thru user's scripting macros. De facto standard word processors usually have an "insert character" input feature to arbitrary choose any Unicode character, and they do not explicitly needs new inputs methods (but every implementation will be welcome).

The only one asserted benefit is to raise hexadecimal digits ten to fifteen to the same numeric formal and typographical range than decimal digits zero to nine have: to have all figure space pitch and case preventing with the same fonts than the running text in technical specialized documents.

About the author

Ricardo Cancho Niemietz was born in Madrid, Spain, in may 9 of 1969. He begans to program with the Sinclair ZX Spectrum BASIC when he was fifteen years old. Few years after, in 1988, he joined the greatest spanish company of entertainment home software, ERBE Software, thru its seal Topo Soft.

Between 1989 and 1991 he left the "Z80 videogames-age" behind and evolved to professional digital image processing, desktop publishing and PostScript printing PC software programming in both 8086 assembler and C. Since 1993 and up to 2002 he was one the software partners of the Agencia EFE, the main spanish international press agency. Along this years digital photo capturing, retouching, transmitting and receiving software, news client and word processor applications in both latin and arabic scripts, and many communication I/O techniques and protocols developed and implemented for the central and subsidiaries information-switching CPU systems of that agency, written in C/C++ under a variety of Microsoft platforms (from DOS to Windows 3.*x*/9*x*/NT/2000), are his more remarkable works. Also, he work closely with AT&T/Lucent Technologies factory in Tres Cantos, Madrid, in some production improvement software for their clean-room, and even a new video digital format driver called SuperAVI® was integrally developed by him under MS Windows 3.*x* for the main spanish CD factory, Iberofon-Ibermemory (but this had no continuity in newer Win32 environments).

Also, he is a children books writer and illustrator. Now, he is trying to publish his work.

He still lives in Madrid, and his current electronic address is <rcancho@tiscali.es>. His mail address is:

José de Cadalso 75, bajo A
28044 Madrid – Spain