Unicode 88

Joseph D. Becker, Ph.D.

August 29, 1988

Reprinted September 10, 1998
Commemorating
Ten Years of the Unicode Standard
1988 to 1998

Unicode 88

Joseph D. Becker

August 29, 1988

A INTRODUCTION

1 Overview

1.1 Abstract

This document is a draft proposal for the design of an international/multilingual text character encoding system, tentatively called Unicode.

Unicode is intended to address the need for a workable, reliable world text encoding. Unicode could be roughly described as "wide-body ASCII" that has been stretched to 16 bits to encompass the characters of all the world's living languages. In a properly engineered design, 16 bits per character are more than sufficient for this purpose.

In the Unicode system, a simple unambiguous fixed-length character encoding is integrated into a coherent overall architecture of text processing. The design aims to be flexible enough to support many disparate (vendor-specific) implementations of text processing software.

A general scheme for character code allocations is proposed (and materials for making specific individual character code assignments are well at hand), but specific code assignments are not proposed here. Rather, it is hoped that this document may evoke interest from many organizations, which could cooperate in perfecting the design and in determining the final character code assignments.

1.2 Need for a new, world-wide ASCII

Electronic transmission and storage of the written word are based on standard numerical encoding of text characters. Currently much of the computing world relies on the character encoding for English text called 7-bit ASCII. ASCII (American Standard Code for Information Interchange) is defined by the standards ANSI X3.4-1977 and ISO 646-1973 (E). ANSI is the American National Standards Institute, Inc., and ISO is the worldwide International Organization for Standardization.

ASCII provides a common coinage for representing text content, permitting reliable exchange of English text among disparate software applications. Less obviously, sequences of ASCII characters form structural elements that interlink diverse computer systems. ASCII text files provide a widely-accepted file format among text-oriented programs (e.g. text editors, electronic mail), ASCII character streams provide one standard basis for file communication protocols, and ASCII text "filters" are capable of supporting an interesting class of text-processing applications.

The problem with ASCII is simply that the people of the world need to be able to communicate and compute in their own native languages, not just in English. Text processing systems designed for the 1990's and the 21st century must accommodate Latin-based alphabets for European languages such as French, German, and Spanish; and also major non-Latin alphabets such as Arabic, Greek, Hebrew, and Russian; and also "exotic" scripts of growing importance such as Hindi and Thai; not to mention the thousands of ideographic characters used in writing Chinese, Japanese, and Korean.

What is needed is a new international/multilingual text encoding standard that is as workable and reliable as ASCII, but that covers all the scripts of the world.

For reference, the table below ranks the world's writing systems roughly in order of commercial importance, as measured by the total GNP of countries using each system:

Rank	Writing System	Languages	% of World GNP
1	Latin	English, German, French, Spanish, Italian, Portuguese, Indonesian/Malay, ...	68
2	CJK ideographs	Chinese, Japanese, (Korean)	14
3	Cyrillic	Russian, Ukrainian, ...	14
4	Arabic	Arabic, Persian, ...	3
5	Devanāgarī family	Hindi, Bengali, Punjabi, Marathi, ...	1
6	Korean (Hangul)	Korean	1
7	Dravidian family	Telugu, Tamil, ...	ε
8	Greek	Greek	ε
9	Khmer	Thai, Lao, Khmer	ε
10	Hebrew	Hebrew	ε

1.3 Technical summary of Unicode

The power of ASCII comes from two simple properties:

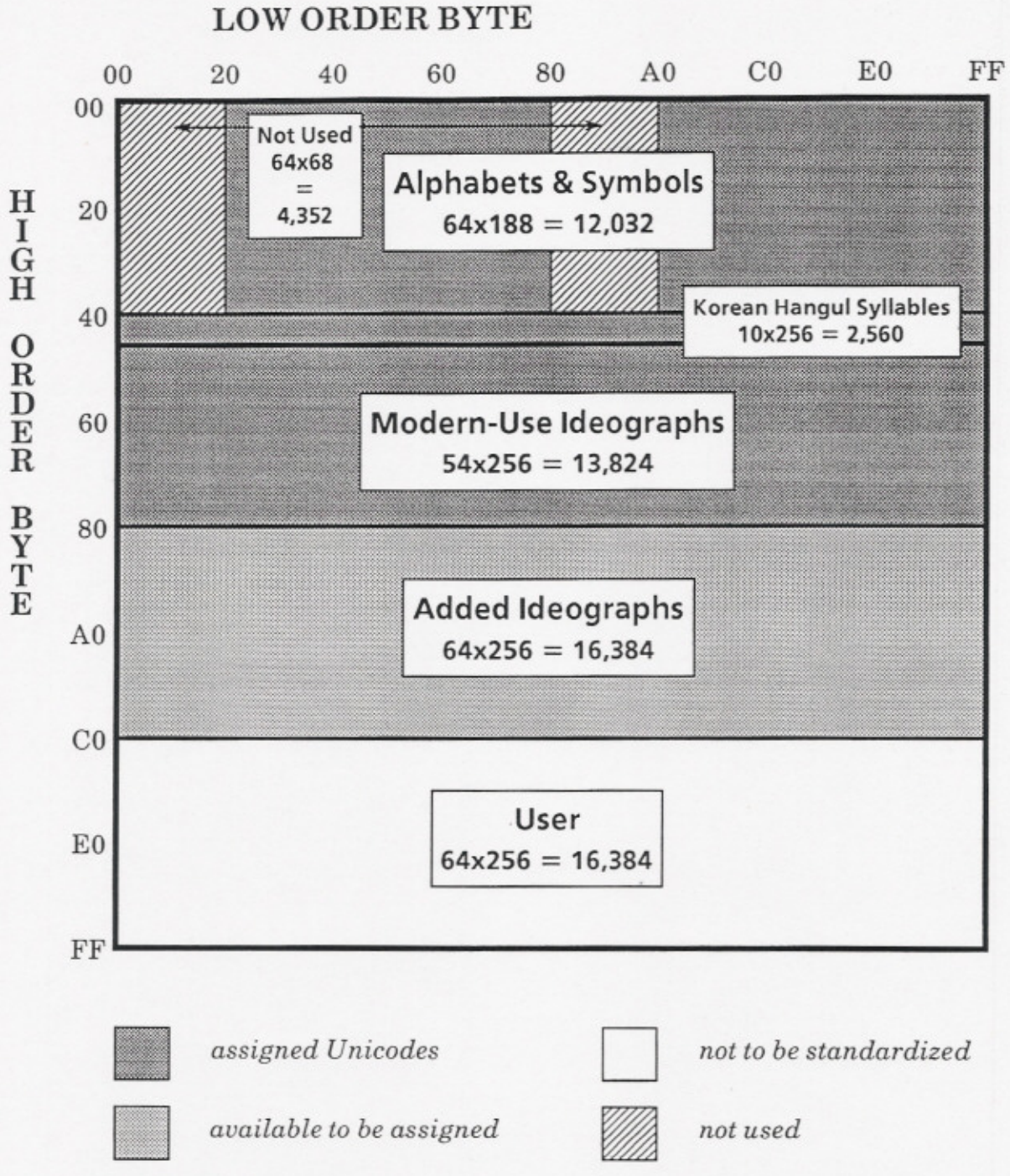
- Its *workability* in processing arises from a fixed length of character code (7 bits within an 8-bit byte)
- Its *reliability* in conveying text content arises from a fixed one-to-one correspondence with the characters of the English alphabet

Unicodes are the most straightforward multilingual generalization of ASCII codes:

- Fixed length of character code (16 bits)
- Fixed one-to-one correspondence with characters of the world's writing systems

That is, each individual Unicode code is an absolute and unambiguous assignment of a 16-bit number to a distinct character.

Since there are vastly more than $2^8 = 256$ characters in the world, the 8-bit byte has become a useless commodity in the context of modern international/multilingual



Unicode Codespace Allocation Overview

character encoding. Stated otherwise, the evolution from ASCII to Unicode means precisely the expansion of character codes from an 8-bit to a 16-bit basis. In Unicode, the 8-bit byte plays no role of any kind.

The name "Unicode" is intended to suggest a unique, unified, universal encoding. A sequence of Unicodes (e.g. text file, string, or character stream) is called "Unitext".

ASCII text

01110100	t
01101000	h
01101001	i
01110011	s
00100000	
01101001	i
01110011	s
00100000	
01110100	t
01100101	e
01111000	x
01110100	t

Unitext

0000000001110100	t
0000000001101000	h
0000000001101001	i
0000000001110011	s
0000000000100000	
0010011101110001	я
0010011101011011	й
0010011101101000	ц
0000000000100000	
0100101000011011	中
0100101000001010	國
0100101010010101	話

The Unicode design includes major principles that support the pure 16-bit encoding:

- *characters vs. glyphs:* A clear and all-important distinction is made between *characters*, which are abstract text content-bearing entities, and *glyphs*, which are visible graphic forms. This model permits the resolution of many problems regarding variant forms, ligatures, and so on.
- *CJK ideograph unification:* The clear model of characters and glyphs permits unification of tens of thousands of equivalent ideographs that are currently given separate codes in China, Japan, and Korea.
- *public vs. private:* The design provides for the distinction between common-use encodings which are public, and other encodings which are kept private so as to enable vendor-specific implementations, vertical-market applications, and so on.
- *plain vs. fancy text:* A simple but crucial distinction is made between *plain text*, which is a pure sequence of Unicodes, and *fancy text*, which is any text structure that bears additional information beyond pure character content.
- *process-based design:* The design is founded on the fact a text encoding exists solely to support the various processes that act upon text. Thus processes such as rendering, filtering, and so on participate in the design.

Unicode may find its initial utility as a standard international/multilingual *interchange encoding*, but it is also designed to serve as the basis for efficient *internal text representation* (a.k.a. *process encoding*) in any text environment where more than 256 different characters are required.

1.4 Structure of this document

The many aspects of text character encoding are highly interrelated, and indeed each topic is best conceived in terms of a conception of all the others. Lacking hypertext or the ability to discuss all topics at once, the document is arranged as follows:

- Part A is an overview of the major concepts of Unicode.
- Part B is a detailed presentation of Unicode's architectural underpinnings.
- Part C applies the Unicode approach to major specific problems of character encoding.
- Part D contains reference lists of particular details, including suggested Unicode allocations and assignments.

Although the document is laid out from the general to the particular, solutions to the problems of character encoding actually evolve from the particular to the general. For example, the definitions in Part B are made only because they were found necessary to handle the particular problems described in Part C. Thus, the document may make an equal amount of sense if read backward.

2 The 16-bit Approach

The idea of expanding the basis for character encoding from 8 to 16 bits is so sensible, indeed so obvious, that the mind initially recoils from it. There must be a catch to it, otherwise why didn't we think of this long ago?

The major catch is simply that the 16-bit approach requires перестройка *(perestroika)*, i.e. restructuring our old ways of thinking. Rather than struggling to salvage obsolete 8-bit encodings via horrendous "extension" contrivances, we need to recognize that the current absence of a standard international/multilingual encoding is a unique opportunity to rethink and revitalize the design concepts behind text encoding.

However, there do exist specific concerns that initially appear to be the "catch" to a 16-bit encoding. To some extent these concerns are overrated, and to some extent they are legitimate but inevitable. This section outlines how the Unicode 16-bit approach either provides for these concerns or trades them off against the greater good. A much longer list of detailed design issues is addressed in Section D.

2.1 Sufficiency of 16 bits

Are 16 bits, providing at most 65,536 distinct codes, sufficient to encode all characters of all the world's scripts? Since the definition of a "character" is itself part of the design of a text encoding scheme, the question is meaningless unless it is restated as: Is it possible to engineer a reasonable definition of "character" such that all the world's scripts contain fewer than 65,536 of them?

The answer to this is Yes. (Of course, the converse need not be true, i.e. it is certainly possible, albeit uninteresting, to come up with *unreasonable* definitions of "character" such that there are more than 65,536 of them.) There are two main concepts in Unicode's approach to this fundamental question:

- The proper definition of character
- The distinction of "modern-use" characters from "obsolete/rare" ones

Proper definition of "character": Unicode does not confuse the notion of character with that of glyph. There are far more glyphs than characters because of the existence of variant forms, rendering forms, and fragment glyphs that can be used to compose graphic forms dynamically. Also, Unicode avoids tens of thousands of character replications by consolidating together the ideographic characters used in writing Chinese, Japanese, and Korean.

Distinction of "modern-use" characters: Unicode gives higher priority to ensuring utility for the future than to preserving past antiquities. Unicode aims in the first instance at the characters published in modern text (e.g. in the union of all newspapers and magazines printed in the world in 1988), whose number is undoubtedly far below $2^{14} = 16{,}384$. Beyond those modern-use characters, all others may be defined to be obsolete or rare; these are better candidates for private-use registration than for congesting the public list of generally-useful Unicodes.

In other words, given that the limitation to 65,536 character codes genuinely does satisfy all the world's modern communication needs with a safety factor of about four, then one can decide up-front that preserving a pure 16-bit architecture has a higher design priority than publicly encoding every extinct or obscure character form. Then the sufficiency of 16 bits for the writing technology of the future becomes a matter of our active intention, rather than passive victimization by writing systems of the past.

2.2 Relation of Unicode to ASCII and other existing codings

Given two sequences of bits ("bit patterns") that supposedly represent the same series of text characters in two different encoding systems, either:

- the sequences are bit-for-bit identical, or
- they are not identical, in which case they require explicit software conversion.

Clearly, almost every possible pair of text encoding schemes require explicit software interconversion; that is, rarely is one encoding truly a pure "extension" of another. Once the inevitability of explicit conversion processes is recognized, the proper design goals for "compatibility" of a new encoding scheme with existing encodings become:

- to minimize the complexity of conversion processes
- to minimize the number of conversion processes

ASCII text

01110100	t
01100101	e
01111000	x
01110100	t

Unitext

0000000001110100	t
0000000001100101	e
0000000001111000	x
0000000001110100	t

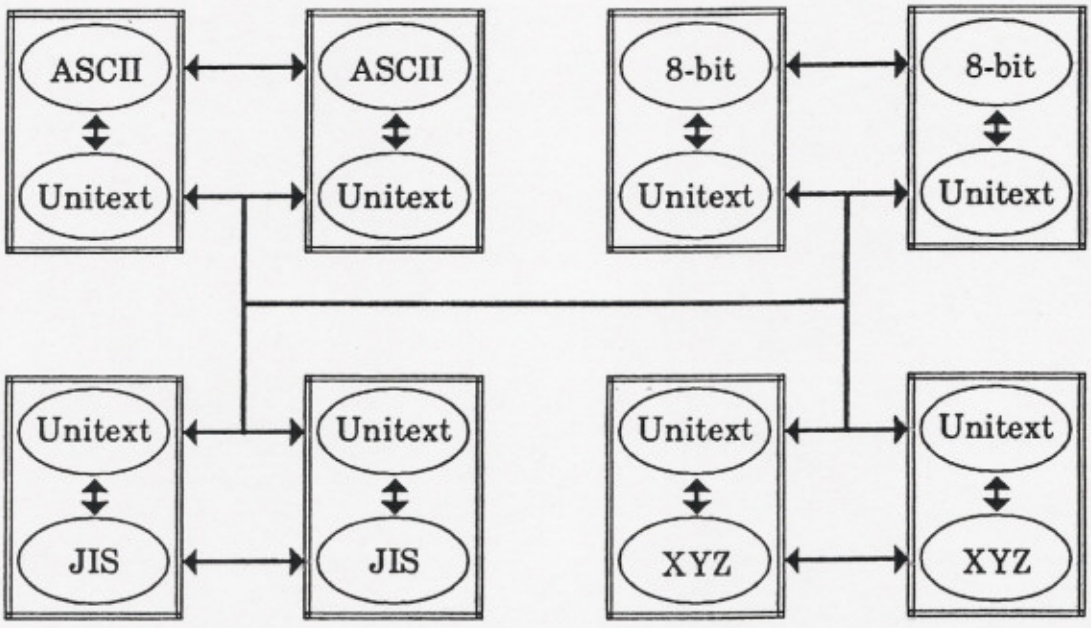
As an example of minimizing conversion complexity, interchange between ASCII text and Unitext is performed by a simple loop of the following operations:

- To convert a 7-bit ASCII character to a Unicode:
  - Preface it with the 9 bits 000000000.
- To convert a Unicode to a 7-bit ASCII character:
  - If the first 9 bits are 000000000, remove them.
  - Otherwise, assign it to a junk code, e.g. SUB (hex 1A).

Insofar as the above algorithms are quite trivial, ASCII text and Unitext may be said to be *conversion-compatible*. One of the design priorities in making the particular Unicode assignment of character codes is to preserve "conversion-compatibility", i.e. the simplicity of these conversion algorithms. This approach is consistent with the Unicode philosophy that processes should be explicitly taken into account as part of the encoding system, rather than being implicitly taken for granted.

Trivial interconversion with existing standards is easily attained for most alphabetic scripts. Unfortunately, straightforward conversion mappings are not possible when it comes to the CJK ideographic characters. Interconversion of these scripts is mainly a matter of indexing through a large table ... which after all is a trivial algorithm once the table is provided.

The goal of minimizing the number of conversion processes is attained simply by using Unitext as an interchange code among disparate encoding systems. Each system could be taught to speak Unitext as an interlingua, while optionally retaining its own "native language" for internal use and local communications. Such a world might be visualized as in the figure below (the rectangles represent systems, the ovals represent the text encodings they support). Evidently having each system implement only 1 conversion process to/from Unicode is vastly more efficient than implementing a number N of conversions that grows as new local encoding schemes are invented.



Since *any* new international/multilingual text encoding will inevitably require explicit conversion to/from existing encodings, this fact might as well be viewed as an opportunity. Within the bounds of "conversion-compatibility", it releases new designs from the need for strict conformity with designs of the past. With luck, the future of computing and electronic communications will be longer than the past. A text encoding design with hopes of serving the 1990's, and perhaps the 21st century, should be engineered primarily to best serve the future, not the past.

2.3 Twofold expansion of ASCII English text

Nothing comes for free, and the price of Unicode's fixed-length 16-bit character code design is the twofold expansion of ASCII (or other 8-bit-based) text storage, as seen in the figure on the previous page. This initially repugnant consequence becomes a great deal more attractive once the alternative is considered.

The only alternative to fixed-length encoding is a variable-length scheme using some sort of flags to signal the length and interpretation of subsequent information units. Such schemes require flag-parsing overhead effort to be expended for every basic text operation, such as get next character, get previous character, truncate text, etc. Any number of variable-length encoding schemes are possible (this fact itself being a major drawback); several that have been implemented are described in a later section.

By contrast, a fixed-length encoding is flat-out *simple*, with all of the blessings attendant upon that virtue. The format is unambiguous, unique, and not susceptible to debate or revision. It is a logical consequence of the fundamental notion of character stream. Since it requires no flag parsing overhead, it makes all text operations easier to program, more reliable, and (mainly) *faster*. It also greatly facilitates the process of unambiguously interpreting text received from other systems, and the deciphering of text that is found embedded within some unknown or extinct data structure.

Unquestionably the twofold expansion of ASCII text will engender increased storage space expense as Unitext is adopted. However, it may be argued that this expense will not prove intolerable. With regard to English text storage, systems may be divided into three categories:

- Software: Most system or application level software should contain little or no inherent English text. Indeed, the prevailing requirement is for program-internal text to be internationalized into message files that can be made multilingual .... precisely the purpose for which Unicode is designed.

- Compressor clients: A few text-system clients create and store vast quantities of English text, and therefore make use of explicit compression/expansion processes. For these systems, Unicode will have no impact at all, since Unicode English text compresses to precisely the same size as ASCII English text.

- Acceptor clients: Nearly all text-system clients create and store quantities of English text small enough that *it is not worth their while to use the currently available techniques for compressing ASCII English text by a factor of 2 or more.* These clients unquestioningly accept the "wasteful" storage of ASCII in order to receive the benefits of its simplicity in processing. There is no reason to alter this behavior when it comes to Unicode, given that the cost of storage media is still rapidly declining. It turns out that in designing a text encoding to serve for the 1990's and beyond, the expense of storage space may be the least important factor that could be brought into consideration.

Historically, computer and communication systems originally implemented 5-bit Baudot character encodings, but it was later discovered that these did not encompass lower-case letters. Then 7-bit ASCII/ISO encodings were implemented, but it was later discovered that these did not encompass European languages beyond English. Then 8-bit extended ISO encodings were implemented, but it was later discovered that these did not encompass Japanese. Then 14-bit JIS and derivative encodings were implemented, but it was later discovered that these did not encompass Chinese.

The bottom line is that the world of computing has now become a fully international and multilingual one, in which 5-bit, 7-bit, 8-bit, and 14-bit text architectures are all extinct. The modern length of a computer word is 32 bits, and the ultimate length of a character code is 16 bits. All we have to do is recognize what is already true.

3 The Unicode Proposal

3.1 Background of the Unicode proposal

Unicode has evolved from a dozen years of practical experience in implementing multilingual computer systems, beginning at Xerox Palo Alto Research Center. This effort has included product or prototype implementation of the most important Latin-script languages (including Hausa, Hungarian, Polish, Turkish, Vietnamese, and many others), plus non-Latin-script languages including Amharic, Arabic, Armenian, Bulgarian, Chinese, Georgian, Greek, Hebrew, Hindi, Japanese, Korean, Persian, Russian, Ukrainian. This work involved the creation of over 100,000 ideographic character images in various sizes and styles for Chinese, Japanese, and Korean, plus tables cross-referencing the many "standard" encodings of these characters.

In 1978, the initial proposal for a set of "Universal Signs" was made by Bob Belleville at Xerox PARC. Many persons contributed ideas to the development of a new encoding design. Beginning in 1980, these efforts evolved into the Xerox Character Code Standard (XCCS) by the present author, a multilingual encoding which has been maintained by Xerox as an internal corporate standard since 1982, through the efforts of Ed Smura, Ron Pellar, and others.

Unicode arose as the result of eight years of working experience with XCCS. Its fundamental differences from XCCS were proposed by Peter Fenwick and Dave Opstad (pure 16-bit codes), and by Lee Collins (ideographic character unification). Unicode retains the many features of XCCS whose utility have been proved over the years in an international line of communicating multilingual system products.

3.2 Status of the Unicode proposal

This document is currently a conceptual exploratory draft only. It in no way represents the policy of Xerox Corporation, which currently uses the Xerox Character Code Standard in all of its systems products.

Many aspects of Unicode remain to be perfected, and the design itself calls for an ongoing organization devoted to its maintenance, particularly in determining the public registration of new characters.

If the idea of Unicode as a potential new ASCII does have validity, it should be of interest to many companies, standards bodies, and other organizations. The hope is that this document may form the nucleus of a cooperative effort to finish the development of Unicode in a form satisfactory to all who have an interest in it. If this effort were to be successful, it might naturally lead to the formation of an appropriate Unicode maintenance organization.

Meanwhile, readers' comments for improving Unicode design or its presentation in this draft are avidly solicited.