

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

Prologue

My article on character sets and character encoding--"A Brief History of Character Codes in North America, Europe, and East Asia"--has drawn a lot of attention, which has greatly surprised me. I intended this Web document to serve only as a background piece to relate how the TRON Multilingual Environment came into existence and to prepare BTRON users for the arrival of the true BTRON multilingual environment, which finally appeared in commercial form in *Cho Kanji* on November 12, 1999. Unbelievably, a large number of non-BTRON users have also viewed it, and that has led to it being selected as a Open Directory Project "Cool Site."

However, there has been some criticism of my article from proponents of the Unicode movement. One criticism is that I "incorrectly" stated that the Unicode Consortium is run by a group of American computer manufacturers. Another was that I did not mention the "surrogate pairs" mechanism through which the Unicode Consortium once again hopes to create a character set through which all of the world's written languages can be represented. Both of these questions will be addressed below, but first I would like to remind the proponents of Unicode who visit TRON Web that those of us participating in the TRON Project are not necessarily impressed by the way Unicode proponents explain the TRON Multilingual Environment on their information sites on the Web.

For example, at an IBM Corporation Web site that deals with Unicode, there is an article written by Suzanne Topping titled "The secret life of Unicode: A peek at Unicode's soft underbelly" in which the TRON Multilingual Environment is mentioned. Her view of the TRON Multilingual Environment is summed up with the statement, "TRON's claim to fame is that it doesn't unify Han characters; instead it attempts to distinguish all the glyph variants." Moreover, she doesn't seem to believe that TRON Code is limitlessly extensible when she states: "Competing character sets like TRON claim to be 'limitlessly extensible.'" It is nice to see TRON being mentioned on foreign Web sites--particularly when those Web sites sport links to TRON Web--but it would be nicer to see it being described in depth so that people can obtain a proper understanding of just how advanced the TRON Architecture is.

In fact, in the area of multilingual character processing, the TRON Project's real "claim to fame" is that it was smart enough to stay out of the character set and font creation business, which as the folks at the Unicode Consortium have no doubt found out by now is a veritable Pandora's box of vexing issues--political, cultural, and technical. In this area, all the TRON Architecture does is provide a framework into which the character sets and font collections of others are loaded. The beauty of this approach is that both legacy character sets and newly created character sets such as Unicode--yes, the *Cho Kanji 3* operating system includes the non-*kanji* parts of Unicode--can be accommodated inside TRON Code without modification. This is also why the de facto TRON character set is growing by leaps and bounds in comparison to Unicode.

Of course, there are many other areas in which the TRON Project can lay claim to fame. It is the world's first "total computer architecture," it has been in the vanguard of the open source/open architecture movement since its inception, it had its eye on ubiquitous computing more than a decade before it was "discovered" in the U.S., and it has placed importance on "real-time processing" from the start. Even in the area of personal computing, there are other goodies like TRON Application Databus (TAD), which is a collection open standards for high-level data formatting. As a result, data compatibility is guaranteed across makers' applications, and no company can walk you through an "upgrade treadmill" to enrich itself. Yes, the TRON Architecture is a marvelous technical achievement that lots of people in the U.S. and Europe still don't know that much about, but that's a story for another day. Let's get back to Unicode.

Unicode Revisited

Steven J. Searle
Web Master, TRON Web

A Little Unicode History

In my previous article on character codes, I referred to the Unicode project as a U.S. industry project, which brought a protest from a Unicode proponent. The Unicode Consortium has foreign participation, he pointed out, just take a look at the Unicode Consortium's members' page. In response, I typed out and sent him the source of my information, which was a book by computer columnist, commentator, and Silicon Valley denizen John Dvorak where the following is written [1].

Death of ASCII by 1995

Within the next year or two you can expect the worldwide adoption of the Unicode 16-bit character standard to replace ASCII and other character-exchange systems. Luckily, both 7-bit ASCII and its various 8-bit extensions, particularly Latin-1 (an ASCII superset that includes various symbols plus European letters), will be subsets of Unicode. While ASCII has 128 characters and Latin-1 has 256, Unicode will open the door to 65,536 characters.

While this advancement may seem trivial and obvious, it took quite an international battle over the past few years to get Unicode accepted. It all began back in the heyday of Xerox PARC. Xerox was promoting an international character set in the early 1980s and some members of the computing community decided they could do better. So an International Standards Organization [sic] (ISO) committee was formed to develop a 32-bit standard worldwide character set to include all known characters and symbols. This organization, ISO 10646, got bogged down in politics instantly. Meanwhile, a group of multilingual system implementers at Xerox, including Joe Becker, Lee Collins (now at Taligent), Eric Mader, and Dave Opstad (both now at Apple), were already thinking about Unicode, a term coined by Becker. In what is an amazing story in itself, they began an ambitious project to unify the commonalities of the Asian character sets, ultimately persuading the Chinese, Japanese, and Koreans to take joint ownership of this effort.

Inspired by Mark Davis (then from Apple's international division, now at Taligent), they broadened the participation in Unicode development to a community of leading industry representatives, including Bill English (Sun Microsystems), Asmus Freytag (Microsoft), Mark Kernigan (Metaphor), Rick McGowan (NeXT), Isai Scheinberg (IBM), Karen Smith-Yoshimura (Research Libraries Group), Ken Whistler (a UC Berkeley linguist, now at Metaphor), and many others.

How Unicode Was Born

The ISO 10646 committee wouldn't die, but the member countries kept voting down the ISO proposals until a compromise with Unicode could be achieved. It was decided the ISO standard would be seen not as a giant 32-bit standard but as a 32-bit standard consisting of 16 separate 16-bit planes or subsections, the first of which would be Unicode. That breakthrough idea allowed Unicode to be a subset of ISO 10646 while ASCII and Latin-1 were subsets of Unicode. In other words, to most computers any change from an 8-bit character set to a 16-bit character set to a 32-bit character set will be transparent except for possible file-size growth. ISO 10646 was finally passed by the member countries in June 1992.

I don't know how ISO 10646 will ever fill its over 4 trillion [sic] character slots, but until it does, Unicode will supply all the characters needed for the languages of the modern world--28,706 characters so far.

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

Mr. Dvorak, who obviously doesn't have much of a future in commercial grade soothsaying, is a Silicon Valley insider, and thus he is able to tell us what happened in the past, how the Unicode project came into being, and even who created the Unicode name. As can be seen in the above description, the roots of the Unicode project are all American. In the beginning, there was no significant participation by East Asians, even though one of the goals of this project was to "unify" the Chinese characters used in China, Japan, and Korea.

If one considers these characters to be "part of the culture" of each country and not just mere "character codes " or "glyphs" inside a computer system, then the Unicode project started out on a very presumptuous footing. To get a feel of it from the East Asian side, try to imagine a group of Japanese publishing houses getting together to "unify" the spellings of British and American English words so they could save space when printing dictionaries. Imagine further that they had invited some British and American English language experts to participate in the project and thus lend it legitimacy. What would British and American media organizations say about such a project? Would they say, "thank you," or would they say, "who asked you to unify our cultures for us?"

Another problem with the Unicode approach that Mr. Dvorak fails to point out is that when you attempt to squeeze "all the characters needed for the languages of the modern world" into 65,536 character code points, lots of Chinese characters get left out. Which ones should be included, and which ones should be left out? That's easy, you say, the most frequently used ones should be included, and the rarely used ones should be left out. However, if one of those rarely used Chinese characters is part of your name or the name of the place in which you live, then it is a frequently used character, not a rarely used one. Even more difficult for people like Mr. Dvorak to understand is that East Asians continue to create and/or use new characters. In Japan, for example, 166 picture characters have been created for NTT DoCoMo Inc.'s i-mode handsets, and Tompa hieroglyphic characters of the Naxi minority in China are a fad among high school girls. In Japan, both character sets are used on a daily basis by ordinary people, and thus they are frequently used characters.

In the above description of Unicode's "breakthrough idea," Mr. Dvorak portrays the Unicode movement as some sort of a white knight coming to the rescue of the bumbling ISO 10646 committee, but the Japanese view of what happened is very different. I was working on the TRON Project at the time, and TRON Project Leader Ken Sakamura had his researchers studying pre-Unicode influenced ISO 10646 to figure out how to make the TRON Multilingual Environment compatible with that standard, since ISO 10646 was to be an international, and not a U.S. industry, standard. To that end, I translated every paper written on the TRON multilingual processing up to that point in time, which was released in book form by the TRON Association [2]. Here's how the Japanese side viewed the usurpation of the ISO 10646 standard committee by U.S. computer industry forces [3].

In June 1992, the ISO discarded DIS 10646 Ver. 1 that Japan had also been promoting up to that time, and it decided on DIS Ver. 2, which made as its core Unicode that Microsoft and other American computer makers were promoting, as the future world standard character code system.

The reason DIS Ver. 1 that had been discussed over the previous five years was denied was because American computer manufacturers attaching importance to simplicity viewed from the side of the maker declared early on that they were not going to use the 8-bit multibyte code of DIS Ver. 1. In essence, if American computer makers who control almost all of the world's

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

computers use Unicode, that de facto standard and the ISO [standard] would end up standing side by side. In order to avoid that type of situation, work was carried out to integrate the 16-bit fixed length code of Unicode with DIS Ver. 1; DIS Ver. 2, which possesses a 16-bit code plane at the top that is almost the same as Unicode, was drawn up; and this was approved as ISO 10646-1.

The ISO 10646 code system originally is expressed as a code with a total of 32 bits in which the respective single octets were called group, plane, row, and cell. Using a simple calculation, the number of characters that can be accommodated here is 4.3 billion types, which is huge. However, ISO 10646-1 that was prescribed recently is called the Basic Multilingual Plane, and it is only the portion that corresponds to the 0 group 0 plane of the previously mentioned huge code. Moreover, in using only the Basic Multilingual Plane, ISO 10646-1 permits a shortened format called UCS-2 that is expressed with with two octets; in reality, UCS-2 will be used as ISO multilingual code.

ISO 10646-1 (part-1) prescribes the overall organization and the contents of the Basic Multilingual Plane; and it has come about that they will create anew and add a separate part outside of Part-1 for the contents of other planes.

However, this Basic Multilingual Plane is something that in fact adopts Unicode almost as is, and the method by which it switches into UCS-4 has not been decided. For that reason, it is all right to say that ISO 10646-1 ended up becoming essentially the same as Unicode.

Moreover, in the future even if parts increase and UCS-4 is decided, there are also lots of makers promoting Unicode with the attitude that they're undecided as to whether or not they will support them, and we can fully imagine that the Unicode situation will essentially continue in the future also.

As can be understood from the above description, the proponents of Unicode were hardly white knights coming to the rescue of the hapless members of the ISO 10646 committee bogged down in politics. Rather they were black knights trying to sabotage the original ISO standard that had been in the works for five years. Their goal was not to give their customers the most comprehensive and efficient character standard possible, rather it was to severely restrict the number of available characters to make it easier for U.S. firms to manufacture and market computers around the world. In short, Unicode was aimed at "the unification of markets," which is why its creators were oblivious for so long to the fact that they were simultaneously unifying elements of Chinese, Japanese, and Korean culture without the participation of the governments in question or their national standards bodies.

I did not attend the meetings in which ISO 10646 was slowly turned into a de facto American industrial standard. I have read that the first person to broach the subject of "unifying" Chinese characters was a Canadian with links to the Unicode project. I have also read that the people looking out for Japan's interests are from a software house that produces word processors, Justsystem Corp. Most shockingly, I have read that the unification of Chinese characters is being conducted on the basis of the Chinese characters used in China, and that the organization pushing this project forward is a private company, not representatives of the Chinese government. This may have something to do with the fact that the person in charge of the unification of Chinese characters is Ken Whistler, a Chinese linguist who believes China has more at stake than other countries of East Asia [4]. However, basic logic dictates that China should not be setting character standards for Japan, nor

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

should Japan be setting character standards for China. Each country and/or region should have the right to set its own standard, and that standard should be drawn up by a non-commercial entity.

For reference, here is a comparison of the main points of the two DIS versions of ISO 10646 mentioned above, i.e., prior to and after usurpation by the forces supporting Unicode.

DIS 10646 (1989)	DIS 10646 Ver. 1 vs. DIS 10646 Ver. 1.2 (1992)
<ul style="list-style-type: none">Each character is to be encoded with four octetsEach octet from the first octet is to be called in order: Group octet, Plane octet, Row octet, Cell octetEach octet is to be either in the 32-126 or 160-255 ranges to avoid control charactersThe character set of each country is to be recorded in its original form regardless of character duplicationOne-byte character sets are to be recorded on a one-byte plane, and two-byte characters sets are to be recorded on a two-byte planeWhen calling up a one- or a two-byte character set, the upper three or two octets can be abbreviated using a compaction methodThe first plane will be called the Basic Multilingual Plane (BMP), and in keeping with Proposal A of the Working Draft, CJK ideographs will be recorded in it	<p>Common Points</p> <ul style="list-style-type: none">Four octets for the overall framework (but, only UCS-4)Each octet is to be called in order: Group octet, Plane octet, Row octet, Cell octetThe first plane is to be called the Basic Multilingual Plane <p>Differences</p> <ul style="list-style-type: none">A two-octet UCS-2 is to be createdEach octet is to ignore compatibility with ISO 2022 and is to fully use 16 bits (but, the Plane octet will be within the range of 00h-7Fh)The compaction method is not to be used, and UCS-4 is to be 32-bit and UCS-2 is to be 16-bit fixed length codeBMP to be replaced with UnicodePlanes outside BMP not specified for nowUCS-2 method for switching outside the BMP will not be specified for nowNot just CJK ideographs, but all characters that have similar shapes are to be unifiedThe mechanism for specifying languages is not to be created at the character code level
<p>UCS stands for "Universal Coded Character Set." CJK stands for "Chinese, Japanese, and Korean." Unification principle mentioned at right was changed to apply only to CJK characters, and thus is called "Han Unification."</p>	

Trying to Save a Deficient Standard with Surrogates

Unicode, as can be gathered from reading the above, is a commercially oriented Eurocentric/Sinocentric standard that put the interests of computer manufacturers over the interests of end users. It is worth noting here that putting the interests of producers before those of consumers is exactly what the so-called "Japan experts" who make a living from bashing Japan in the English language media accuse Japanese companies of doing. It is further worth noting that by creating a standard that prevented the Japanese people from using all the characters that appear in books and on paper documents in their country, the U.S. companies backing Unicode were in effect creating a "non-tariff trade barrier" for themselves. It is impossible to create a digital library or to digitize public records in Japan using the Unicode Basic Multilingual Plane. That's because there are only a little more than 20,000 Chinese ideographs on this plane, and only 12,000 are applicable for Japanese.

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

For reference, here's a table of what's on the Unicode/ISO 10646-1 Basic Multilingual Plane, which corresponds to Group 00, Plane 00.

ISO 10646-1 BMP

Row(s)	Contents (scripts and characters, reserved area)
A-ZONE (19,903)	Alphabets, Symbols and Punctuation, and CJK Auxiliary
00	Basic Latin, Latin-1 Supplement (ISO/IEC 8859-1)
01	Latin Extended-A, Latin Extended-B
02	Latin Extended-B, IPA Extensions, Spacing Modifier Letters
03	Combining Diacritical Marks, Basic Greek, Greek Symbols and Coptic
04	Cyrillic
05	Armenian, Hebrew
06	Arabic
07 - 08	(Reserved for future standardization)
09	Devanagari, Bengali
0A	Gurmukhi, Gujarati
0B	Oriya, Tamil
0C	Telugu, Kannada
0D	Malayalam
0E	Thai, Lao
0F	(Reserved for future standardization)
10	Georgian
11	Hangul Jamo
12 - 1D	(Reserved for future standardization)
1E	Latin Extended Additional
1F	Greek Extended
20	General Punctuation, Superscripts and Subscripts, Currency Symbols, Combining Diacritical Mark for Symbols
21	Letterlike Symbols, Number Forms, Arrows
22	Mathematical Operators
23	Miscellaneous Technical
24	Control Pictures, Optical Character Recognition, Enclosed Alphanumerics
25	Box Drawing, Block Elements, Geometric Shapes
26	Miscellaneous Symbols
27	Dingbats
28 - 2F	(Reserved for future standardization)
30	CJK Symbols and Punctuation, Hiragana, Katakana
31	Bopomofo, Hangul Compatibility Jamo, CJK Miscellaneous
32	Enclosed CJK Letters and Months

Unicode Revisited

Steven J. Searle
Web Master, TRON Web

33	CJK Compatibility
34 - 4D	Hangul
I-ZONE (20,992)	Chinese, Japanese and Korean Ideographic Characters
4E - 9F	CJK "Unified" Ideographs
O-ZONE (14,336)	Open Zone
A0 - D7	Yi, Yi Extensions, Hangul Syllables
S-Zone (2,048)	Surrogate Zone
D8-DF	High-Half and Low-Half Zones of UTF-16
R-ZONE (8,190)	Restricted Use Zone
E0 - F8	(Private Use Area)
F9 - FA	CJK Compatibility Ideographs
FB	Alphabetic Presentation Forms, Arabic Presentation Forms-A
FC - FD	Arabic Presentation Forms-A
FE	Combining Half Marks, CJK Compatibility Forms, Small Form Variants, Arabic Presentation Forms-B
FF	Halfwidth and Fullwidth Forms, Specials

Source: Web article by Olle Järnefors, "A Short overview of ISO/IEC 10646 and Unicode" with revisions from various other sources to bring it up to date with latest version of the BMP

Somewhere along the way, the elementary fact--which should have been obvious from the beginning-- that a vastly greater number character code points would be needed hit the creators of Unicode. At this point in Unicode's history, the large U.S. computer manufacturers backing this undertaking should have pulled the plug on it and moved onto something better. Information on alternatives, such as the highly efficient TRON Multilingual Environment, was already available in English. Therefore, there was no excuse for trying to "improve" Unicode. However, the backers of Unicode did not have the plug pulled on their project, and thus Unicode's creators devised "improvements" that would enable Unicode to process a grand total of 1,114,112 code points. Here's how they nonchalantly describe their decision to create character code points off the Basic Multilingual Plane on their Web site.

Unicode was originally designed as a pure 16-bit encoding, aimed at representing all modern scripts. (Ancient scripts were to be represented with private-use characters.) Over time, and especially after the addition of over 14,500 composite characters for compatibility with legacy sets, it became clear that 16-bits were not sufficient for the user community. Out of this arose UTF-16 [5].

So what did Unicode's creators do to increase the number of character code points when they were limited to only 65,536 on first plane of the ISO 10646 standard? If one subtracts the total number of character code points on the 16-bit Basic Multilingual Plane, i.e., 65,536, from the grand total of 1,114,112 code points, one is left with 1,048,576 (2^{20}) code points. These character code points were created by multiplying one half of the 2,048 "surrogate character codes" on the Basic

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

Multilingual Plane against the other half, i.e., $1,024 \times 1,024 = 1,048,576$. Since the codes are stacked on top of each other on a table, the former half are referred to as "high-surrogates" and the latter "low-surrogates," and the resulting character code points are referred to as "surrogate pairs." For any techies reading this, here's the official Unicode explanation of what happens.

UTF-16 allows access to 63K characters as single Unicode 16-bit units. It can access an additional 1M characters by a mechanism known as surrogate pairs. Two ranges of Unicode code values are reserved for the high (first) and low (second) values of these pairs. Highs are from 0xD800 to 0xDBFF, and lows from 0xDC00 to 0xDFFF. In Unicode 3.0, there are no assigned surrogate pairs. Since the most common characters have already been encoded in the first 64K values, the characters requiring surrogate pairs will be relatively rare (see below).

Since I was criticized for not mentioning surrogates in my previous article on character sets and character encoding--in fact, I hadn't heard of them until after I wrote my article, because surrogate pairs hadn't been implemented on a commercial operating system [6]--I contacted a TRON engineer to help me find out when they first appeared in Unicode standards. When I went to his place of work, there was a huge stack of books on a table--the massive output of the Unicode Consortium. According to our investigation, surrogates first appeared in Unicode 2.0, which was released in 1996. The latest version of Unicode at the time of this writing is version 3.1, but according to what I have been able to ascertain, no operating system manufacturer has shipped a Unicode-based operating system that can actually display characters outside of the Basic Multilingual Plane, so for practical purposes Unicode still seems to be a 16-bit character code at the time of this writing.

Now the non-specialist reading this is probably saying to himself/herself that the above-mentioned surrogate mechanism has solved the problem of an insufficient number of character code points in Unicode, so it should be clear sailing from here on for Unicode. However, there is one very huge problem that has been created as a result of this surrogate pairs mechanism, which coincidentally seems to violate one of the basic tenets of programming, Occam's Razor: "never multiply entities unnecessarily." Since each new surrogate pair character code point is created by multiplying a two-byte code by a two-byte code, the result is a four-byte code, i.e., it's 32 bits long, which requires twice as much disk space to store as the 16-bit characters codes on the Unicode Basic Multilingual Plane. Accordingly, the new and improved Unicode has essentially become an inefficient 32-bit character encoding system, since 94 percent of the grand total of 1,114,112 character code points (1,048,576) are encoded with 32-bit encodings [7].

I can already hear the reader saying the Unicode creators have simply replaced one "self-created non-tariff barrier" with another "self-created non-tariff barrier." Correct. This is why the U.S. government constantly has to put pressure on Japan to "open its markets," which is trade negotiator doublespeak for the forced purchase of American-made products no matter how ill suited they are for the Japanese market. Japanese databases that have to employ a large number of surrogate pairs in their data, such as a university library or the telephone company, are going to have to pay more to store Unicode data than data created with a more efficient character encoding alternative. In addition, consumers also can be hurt by using surrogate pairs. In Japan, NTT DoCoMo Inc.'s i-mode users are currently charged on the basis of how many bytes of data they send or receive, so if four-byte character codes are employed they will be paying more money to send and receive certain types of data (e.g., popular Tompa heiroglyphic characters).

There is, of course, the additional problem of how the surrogate pair character code points will be incorporated into ISO 10646. Originally, the Unicode Consortium was only supposed to supply the first

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

plane, the Basic Multilingual Plane, of the ISO 10646 standard. Will the supporters of Unicode be able to force more changes on the ISO 10646 committee? Will they demand the right to create the second and/or subsequent character planes for ISO 10646 to build a newer and even more improved version of Unicode? From the Japanese perspective, it doesn't really matter what the Unicode proponents and the ISO 10646 committee do. Two independently created unabridged *kanji* character sets, Konjaku Mojikyo and GT Shotai Font, have already been implemented on personal computers in Japan, and people are currently using them to build databases [8]. A third, eKanji, has been created in conjunction with Unicode. They will all be used in parallel until one obtains more support than the others and emerges as the clear winner. Yes, we are going to have an unabridged *kanji* character set war in Japan, and Unicode is invited to join in--if it ever gets finished.

Unicode Deficiencies Vindicate TRON Approach

One of the amazing things about the TRON alternative to Unicode, the TRON Multilingual Environment, is that it hasn't changed since it was introduced to the world in 1987 at the Third TRON Project Symposium. It is based on two basic concepts: (1) language data are divided into four layers: Language, Group, Script, and Font; and (2) language specifier codes, which specify each of those four layers, are used to switch between 16-bit character planes. By extending the language specifier codes, it is possible to increase the number of planes that can be handled indefinitely. At present, 31 planes with 48,400 character usable code points have been defined for TRON Code, which means that a BTRON computer can access up to 1,500,400 character code points. This may seem like an incredible number of characters, but in fact the current implementation of the BTRON3-specification operating system, *Cho Kanji 3*, has 171,500 characters, and yet it does not include the Konjaku Mojikyo and eKanji character sets, which could easily add 140,000 characters to this total.

For those of us working on the TRON Project, the above-mentioned Unicode deficiencies stand as our vindication. Decades ago, some of the TRON Project's critics tried to claim that Japan didn't have the engineering knowhow to produce a modern real-time operating system, never mind an advanced computer architecture that could be serve as a basis for computerizing human society in the 21st century. Since the U.S. mass media controlled the dissemination of information to the people of the world at that time--we now have the Internet to get opposing views out to the masses--the mantra that Japan is only good at hardware took root. Moreover, there were all those books about a Japanese plot to take over the world--as if Japan's current political culture produces leaders on a par with the megalomaniac villains in James Bond novels! It was all anti-Japanese propaganda from beginning to end, and the TRON Project got swept up in it. For the objective reader visiting this page, there are three things we can learn from the Unicode fiasco described above.

First, and probably most important, the American, bottom-up, market driven approach--of which Unicode is but one example--is inferior to the TRON Project's top-down, macro design approach. Not only does bottom-up, market driven computer system design produce incompatible systems that mainly enrich producers who use political clout to have them marketed worldwide, it also leads to the implementation of technically deficient systems of limited longevity that hurt the interests of end users. The reason TRON Code is superior is that it was designed from the start as a character encoding system that all the people of the world could use--even the handicapped. TRON Code has included character code points for Braille characters from the beginning, because even the handicapped are part of world society who have "rights" when computer systems are designed. TRON Code also has an advantage in that the TRON Project is not in the character set creation business. It merely provides a framework into which others' character sets are loaded.

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

Second, the open source/open architecture movement--which includes TRON, GNU/Linux, and FreeBSD--can actually create better standards than commercially oriented interests, and this is in spite of the fact that the various groups developing open source software and/or open architecture computer systems have considerably less money to spend on research and development. When it comes to standards, Unicode is not the first time the U.S. computer industry dropped the ball on character codes and encoding. When the U.S. computer industry created the ASCII character set back in the 1960s, it totally ignored the data processing needs of countries Western Europe, even though they use the same alphabet as the Americans. The developers of ASCII seem to have been focusing solely on a solution for the U.S. market at the time, and thus the ISO had to step in to develop a character set for Europeans. And then, of course, there was the Y2K problem, which was the result of trying save a little money because memory was expensive in early computer systems. The Y2K fiasco ended up costing a lot more money to fix than it ever saved.

Third, we learn here again the maxim about organizations that "once a bad idea takes hold, it is almost impossible to kill it off." The Unicode movement has been around so long now that a priesthood has come into being to propagate it throughout the world. Followers of Unicode object to anyone who voices complaints about it. Perhaps such people will write to me to tell me modern high-speed microprocessors have no problem handling the Unicode surrogates, but I live in Japan where low-power microprocessors predominate inside handheld devices where every calculation reduces precious battery power. Perhaps they will try to tell me that four-byte encodings are no big deal, since high-capacity hard disks are available at low cost. But what about fixed capacity CD-ROMs and low-capacity memory sticks that are scheduled for use in next generation cell-phones and even electronic books with minimal hardware resources? For every point, there is a counterpoint. It would be much better for the Unicode proponents to spend their time finishing their multilingual system, while the TRON Project finishes the TRON Multilingual Environment. Then, let the people of the world decide which one they want to use.

[1] John Dvorak. *Dvorak Predicts: An Insider's Look at the Computer Industry*, McGraw-Hill 1994, pp. 142-3.

[2] *Collected Papers on BTRON Multilingual Processing with an Appended BTRON1 Introductory Operation Manual*, TRON Association 1992.

[3] The University of Tokyo Research Group on the Construction of a Humanities Multilingual Text Processing System. *Jimbun-kei takokugo tekisuto puroseshigu shisutemu-no koochiku-ni mukete* [Toward the construction of a humanities multilingual text processing system], 1995, Unpublished Pamphlet, p. 16.

[4] Ken Whistler's comments in response to criticism of a Web article by Mr. Norman Goudry, "Why Unicode Won't Work on the Internet: Linguistic, Political, and Technical Limitations," on the deficiencies of Unicode can be read here: <http://slashdot.org/features/01/06/06/0132203.shtml>. In his comments, he states, "The effort is led by China, which has the greatest stakeholding in Han characters, of course, but Japan, Korea, Taiwan and the others are full participants, and their character requirements have not been neglected." Thus we learn belatedly of Unicode's Sinocentricity.

Unicode Revisited

Steven J. Searle

Web Master, TRON Web

[5] In Unicode circles, UTF stands for "Unicode Transformation Format." Other Web sites, however, give "Universal Transformation Format" and "UCS Transformation Format." What UTFs are, and there are a lot of them, are methods for converting Unicode raw 16-bit data into binary encodings that will pass through communication networks without corruption. UTF-7 and UTF-8 are backward compatible with ASCII and ISO-8859-1, respectively, and UTF-16 is the default. Along with UTF-8, UTF-16 supports multibyte encodings. In addition, there are UTF-16LE (Little Endian), UTF-16BE (Big Endian), UTF-32, UTF-32LE, and UTF-32BE.

[6] Implementations of Unicode that support surrogates are apparently due in the middle of 2001. An answer in a FAQ at the Unicode Consortium Web states, "Since only private use characters are encoded as surrogates now, there is no market pressure for implementation yet. It will probably be around the middle of 2001 before surrogates are fully supported by a variety of platforms." Accordingly, my article, which was written at the end of 1998, was correct when it claimed that Unicode is a 16-bit code. Surrogate pairs were only on the drawing board at that time.

[7] It should be pointed out that the total number of usable character code points in Unicode is only 1,112,064, since the surrogate character code points are only used for combining (i.e., $65,536 - 2,048 + 1,048,576 = 1,112,064$ usable character code points).

[8] For people who do not know East Asian languages, it should be pointed out that having enough character code points to record all the Chinese ideographs used in Japan, for example, is not enough. A powerful character search utility is also needed to quickly find the character you are looking for or to obtain information about a character you do not know. Both the Konjaku Mojikyo and GT Shotai Font unabridged *kanji* characters sets have such a search utility, which is why they are quickly gaining acceptance among Japanese personal computer users.