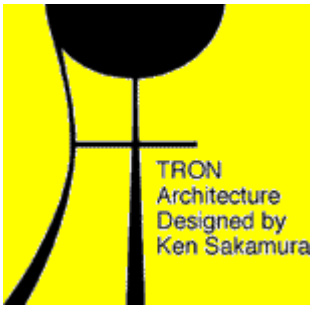


A Brief History of Character Codes

in

North America, Europe, and East Asia

Steven J. Searle

Web Master, TRON Web

Introduction to Character Codes

Character Codes: the Basis for Processing Textual Data

What makes personal computers useful to the majority of people is not that they can process numerical data--yes, a lot of people still prepare their tax statements with a handheld calculator, not a personal computer!--but that they can process textual data. Almost anyone would agree that the overwhelming majority of personal computer users employ word processors more frequently than other type of application. In fact, it would probably be difficult to find a personal computer that didn't have a word processor installed in it.

On the other hand, there are many people who are unaware of the fact that to a computer textual data are also numerical data. In modern computer systems, the individual characters of the scripts that humans use to record and transmit their languages are encoded in the form of binary numerical codes, just as are the Arabic numerals used in calculation programs (Fig. 1). This is because the circuitry of the microprocessor that lies at the heart of a modern computer system can only do two things--calculate binary arithmetic operations and perform Boolean (i.e., true or false) logic operations.

Character on the screen	Binary value used to process it	Character on the screen	Binary value used to process it
1	0110001	A	1000001
2	0110010	B	1000010
3	0110011	C	1000011
4	0110100	D	1000100
5	0110101	E	1000101

Figure 1. Binary values behind alphanumeric characters on the screen in 7-bit ASCII code

Accordingly, when a personal computer records the letter 'A' onto a floppy disk, for instance, it does not create an image of the letter 'A' with tiny magnetic dots, rather it records a binary number (made up of zeroes and ones) that represents the letter 'A' in a character code table. Ah, but doesn't the computer display 'A' on the screen when you type 'A'? Correct. However, when you strike the key for the letter 'A' on the keyboard,

the first thing that is generated is the character code for 'A'. The computer uses that as the basis for pulling the character shape of 'A' from a font file listing with the same binary number and displaying it on the computer's screen. The same thing happens when one uses a computer to print out text; the only difference is that the output is on paper rather than a computer screen.

This system of using binary character codes as "shadow representations" of the graphical characters that are output on the screen or in print allows the personal computer to flexibly and efficiently do things the mechanical typewriter could never do. Not only can a personal computer print out text like a typewriter, but it can also store, modify, sort, retrieve, and transmit text with blindingly fast speed. On top of that, with the addition of the proper software, the monolingual processing personal computer can be transformed into a multilingual processing platform. The only problem is that when languages with large character sets, such as Japanese are added, the character codes for encoding the individual characters of the script have to be longer.

English-language personal computers used in America employ a 7-bit character code called American Standard Code for Information Interchange (ASCII), which allows for a charcter set of 128 items of upper and lower case Latin letters, Arabic numerals, signs, and control characters to be used (2^7 = 128 code points). When an eighth bit is used as a "parity bit," that is a value used for checking whether or not data have been transmitted properly, then ASCII becomes an 8-bit, or one-byte (8 bits = 1 byte), character code. A true 8-bit character code allows for up to 256 items to be encoded (2^8 = 256 code points).

In the case of languages such as Japanese, which has a huge character set with tens of thousands of characters, a 16-bit, or two-byte, character code is employed. A two-byte character code allows for up to 65,536 items to be encoded (2^16 = 65,536 code points), but the standard character code used in Japanese personal computers at present, i.e., Japan Industrial Standard (JIS) X 0208-1990, lists only 6,879 characters. This is adequate for most word processing tasks in daily life, but insufficient for writing people's names, place names, historical data, and even the names of the fish that the Japanese eat at sushi shops!

In regard to character codes, it should also be noted that computers operate most efficiently when they process data in bytes. This is because their internal circuitry is usually designed with data pathways that are 8, 16, 32, or 64 bits wide. For that reason, a 10-bit or a 15-bit character code is clumsy and inefficient to handle inside a personal computer. On the other hand, if too many bytes are used for encoding characters, computers likewise process data inefficiently. For example, a three-byte character code could encode almost 17 million characters (2^24 = 16,777,216 code points), which would cover all known historical and currently used character sets throughout the world, but the majority of the world's languages only need a one-byte code for character encoding, since they are alphabetical scripts. Processing another two unneeded bytes to store an 'A' on a floppy disk, for instance, would lead to a tremendous reduction in the amount of disk space that could be used for storing data.

Early History of Character Codes

Telegraphy and the Beginning of Electronic Data Processing

Anyone who has ever seen a movie about the American West in the 19th century knows that the first widely used character code for electronically processing textual data was Morse code, which was invented for transmitting messages via telegraph lines, not word processing on computers. However, few people know that the inventor of the Morse code, American Samuel Finley Breese Morse (1791-1872), was also a renowned artist who studied painting in London, where he learned of researches into electromagnetism by British scientists. It was on a return sea voyage to the U.S. in 1832 that he conceived his telegraph system, which started the movement toward the electronically networked world in which now live, and which earned him his reputation as America's Leonardo da Vinci.

Morse did not invent the first telegraph system to be put into practical use. That honor belongs to two Britons--Sir Charles Wheatstone, a physicist and inventor, and Sir William Fothergill Cooke, an electrical engineer--who installed the first railway telegraph system in England in 1837, the same year Samuel Morse invented the first American telegraph. However, their system was not a simple one. It was based on five wires which deflected a magnetic needle onto a receiver to indicate the letters of the alphabet. In contrast to this, Morse's system was simpler. It used a single wire for transmitting the signals, and instead of a "deflecting magnetic needle" for receiving the signals, it used an "electromagnet" that attracted a small

armature when an incoming signal was received. This made it more reliable. Another interesting feature about Morse's invention was that the incoming signals could be recorded on a moving strip of paper, although this wasn't used for many years, since operators could read incoming messages from the clicking of the receiver. Morse demonstrated his system on May 24, 1844 in the first U.S. telegraph link, which ran between Baltimore, Maryland, and Washington, D.C. He sent the message "What hath God wrought!"

Morse invented the code he used to send his historic message in 1838. Like the binary system used in modern computers, it is based on combinations of two possible values--in the case of Morse code, a dot or a dash. However, unlike the character codes used in modern computers, the combinations of the two values used to represent characters in Morse code vary in length (Fig. 2). The principle Morse used was to give the most frequently used letters the shortest possible patterns, which greatly reduced the length of a message. For example, the most frequently occurring English letter, 'E', is represented by just a dot; the second most frequently occurring letter, 'T', by just a dash. Interestingly, Morse found out the frequency of letters in English not by doing a study of texts, but by counting the individual pieces of type in each section of a printer's type box. The result of his labors was a highly efficient code that with some modifications is still in use to this day, 160 years after it was invented.

A	▪-	N	-▪	0	-----
B	-▪▪▪	O	---	1	▪----
C	-▪-▪	P	▪--▪	2	▪▪---
D	-▪▪	Q	--▪-	3	▪▪▪--
E	▪	R	▪-▪	4	▪▪▪▪-
F	▪▪-▪	S	▪▪▪	5	▪▪▪▪▪
G	--▪	T	-	6	-▪▪▪▪
H	▪▪▪▪	U	▪▪-	7	--▪▪▪
I	▪▪	V	▪▪▪-	8	---▪▪
J	▪---	W	▪--	9	----▪
K	-▪-	X	-▪▪-	,	--▪▪-- (comma)
L	▪-▪▪	Y	-▪--	.	-▪-▪-▪ (period)
M	--	Z	--▪▪	?	▪▪--▪▪

Figure 2. Subset of International Morse Code

Morse code has evolved through several versions over the span of time. Beginning with Early Morse Code, it developed into American Morse Code, and then on into International Morse Code. In Fig. 2, the reader can see a subset of International Morse Code. Note that it has no lower case letters, and that the Arabic numerals are all represented by five-place patterns of dots and dashes. When actually transmitted, a dash is three times the length of a dot. The dots and dashes that form the patterns for the individual characters are separated by an interval equivalent to one dot, the space between the individual characters of a word are separated by an interval equivalent to three dots, and words in message are separated by an interval equivalent to six dots.

After Morse's inventions were put into practical use, other inventors contributed to the development of telegraph technology by creating hardware such as relays that would allow telegraph signals to travel farther by giving them an electrical power boost. In addition, various schemes were developed to increase the utilization of the telegraph lines. These enabled "diplexing," or the sending of two messages in the same direction at the same time; "duplexing," or the sending two signals in opposite directions at the same time; and "quadruplexing," or the sending of four messages (two in each direction) at the same time. Moreover, the reception of the incoming signals was mechanized through the introduction of tape-reader machines, which allowed traffic to speed up to 400 words per minute by 1900.

The next great leap in telegraph technology was a primitive printing telegraph, or "teleprinter," patented by Jean-Maurice-Émile Baudot (1845-1903) in France in 1874. Like Morse's telegraph, it involved the creation of a new character code, the 5-bit Baudot code, which was also the world's first binary character code for processing textual data. Messages encoded in Baudot's code were printed out on narrow two-channel transmission tapes by operators who created them using a special five-key keypad, although in later versions typewriter keyboards that automatically generated the proper five-unit sequences were employed. Another interesting feature of Baudot's teleprinter system was that it was a "multiplex" system that allowed up to six operators to share a single telegraph line using a time division system. This led to a considerable increase in the transmission capacity of a telegraph line. Baudot's system proved to be fairly successful, and it remained in widespread use in the 20th century until it was displaced by the telephone, and, of course, personal computer communications.

Binary value	Letters	Figures
00011	A	-
11001	B	?
01110	C	:
01001	D	$
00001	E	3
01101	F	!
11010	G	&
10100	H	STOP
00110	I	8
01011	J	'
01111	K	(
10010	L	)
11100	M	.
01100	N	,
11000	O	9
10110	P	0
10111	Q	1
01010	R	4
00101	S	BELL
10000	T	5
00111	U	7
11110	V	;
10011	W	2
11101	X	/
10101	Y	6
10001	Z	"
00000	n/a	n/a
01000	CR	CR
00010	LF	LF
00100	SP	SP
11111	LTRS	LTRS
11011	FIGS	FIGS

Figure 3. The Baudot Code Set

Being a 5-bit character code, Baudot code only has room for handling 32 elements (2^5 = 32 code points). This is not enough to handle both the letters of the Latin alphabet plus Arabic numerals and punctuation marks, so Baudot code employs a "locking shift scheme" to switch between two planes of 32 elements each (Fig. 3), which can be compared to the shifting and locking into place the upper case letters on a mechanical typewriter. Like the subset of International Morse Code given above, Baudot code has codes for the upper case letters of the Latin alphabet, Arabic numerals, and punctuation marks. However, in addition, it has control codes, which are also a feature of the character codes used in today's personal computers. The two binary elements of Baudot were represented as either a mark or a space on the transmission tape; they were translated into a electrical current "on" and electric current "off" during transmission. Each character of a message being sent was preceded by a start bit, and followed by a stop bit, which made it slow by today's standards.

The reason Baudot was forced to limit his character code to 5 bits--and hence leave out the lower case Latin letters--was because of hardware constraints. A more complex code--even just a 6-bit code--would have necessitated a much more complex electromechanical device to transmit it, which would have been extremely difficult to fabricate using the technology in Baudot's time. After modifying Baudot's code to 55 elements--thus allowing for three places for national variants--the CCITT (Comité Consultatif International Télégraphique et Téléphonique [Consultative Committee on International Telephone and Telegraph]) in Geneva, Switzerland, standardized it in 1932 as a 5-bit code for teleprinters. It was given the designation "International Telegraphic Alphabet No. 2." In the field of telegraphy, Baudot also left a portion of his name to posterity in the form of the "Baud rate," which refers to the number of data signalling events that occur in a second.

The 1890 U.S. Census and the Birth of a New Character Code

The end of the 19th century saw the creation of another character code, this one invented in the United States for the purpose of tabulating census data. It was created by a young American inventor named Herman Hollerith (1860-1929), who was hired by the U.S. Census Bureau in 1880 as a statistician after graduating from Columbia School of Mines in New York in 1879; and it was to have far reaching effects that lasted into the golden era of mainframe computers in the 1970s. Hollerith was none other than the creator of the

Hollerith code, a character code for encoding alphanumeric data on the "punched [or punch] card," which introduced one of the first geek expressions to the American masses--"do not fold, spindle, or mutilate," an expression that left many Americans with the impression that computers were soon to take control of their society.

Interestingly, by the time Hollerith applied it to the recording of census data in the 1880s, the punched card was very "old technology." Punched cards were originally devised around 1800 by Joseph-Marie Jacquard (1752-1834), a French inventor who employed them in a new loom for weaving patterns into fabrics. His invention, patented in 1804, later enabled the creation of the fully automatic loom, and hence the textile industry as we know it today. Moreover, Hollerith was not the first to consider Jacquard's invention for use with computers--Charles Babbage (1792-1871) in England planned to use them with his "analytical engine," a giant mechanical computer that was never successfully implemented. On top of that, Hollerith didn't consider the punched card for data input until he had experimented and failed with data encoded on strips of paper, which unfortunately had a tendency to rip during processing.

To get around the problem of data paper strips ripping, Hollerith devised what was to become the computer punched card (click here to view an image of a Hollerith punched card). In the form in which it was later standardized, it was a 186mm x 82mm stiffened card with a clipped upper left-hand corner that had with 12 rows (numbered 12, 11, 10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, top to bottom) and 80 columns (numbered 1 through 80, left to right). Each column represented a single alphanumeric character or symbol. As the cards were fed through a reader/calculator, called a "tabulating machine," pins passed through the positions where holes were punched completing an electrical circuit and subsequently registered a value. Although this sounds like a very primitive and limited system, it was very powerful and very efficient. The 1880 census in the U.S., which was done by hand by humans, took seven years to complete, making the data obtained from it almost useless. When Hollerith's "tabulating machines" were employed for the 1890 census, it took the Census Bureau just six weeks to complete the computations with more detailed data than before, and it resulted in a savings of $5 million, since the Census Bureau's personnel costs were greatly reduced.

In Hollerith's code system, a single alphanumeric character is registered across 12 rows of a punched card where there can be either no hole or a hole (i.e., '0' or '1'), which at first glance makes it a 12-bit character code. However, in contrast to a full 12-bit character set, which can have up to 4,096 elements (2^12 = 4,096 code points), Hollerith code only specifies 69 elements (upper case Latin letters, Arabic numerals, punctuation marks and symbols). Accordingly, as far as its capability to represent data is concerned, Hollerith code is little more than a 6-bit character code, or the equivalent of Baudot's 5-bit character code without shifting. So why are there so many places? One reason is that the larger number of places makes it possible to encode data with a smaller number of punches, which was done manually for many years. For example, the 10 Arabic numerals are encoded with a single punch, and 24 of the 26 letters of the Latin alphabet are encoded with just two punches, except for two, which are encoded with a single punch. Punctuation marks and the like are encoded with two or more punches, since they aren't used as often.

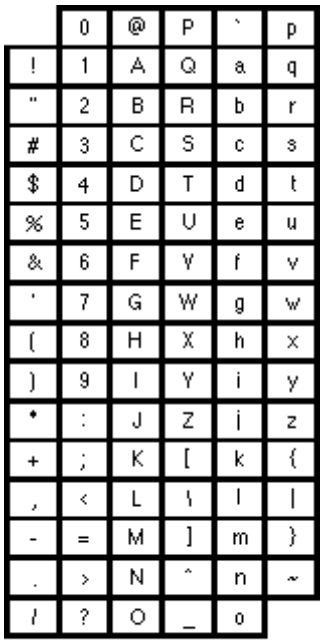
As a result of the success of his tabulating machine at the U.S. Census Bureau, Hollerith decided to commercialize it. In 1896, he set up a company to market his invention, which, not surprisingly, he called the Tabulating Machine Co. In 1911, The Tabulating Machine Co. merged with the Computing Scale Co. of America and International Time Recording Co. to become the Computing-Tabulating-Recording Co., or C-T-R for short. After penetrating markets on several continents and setting up business operations on a global scale, the Computer-Tabulating-Recording Co. changed its name in 1924 to International Business Machines Corporation. Naturally, because Hollerith's character code and the punched card became core technologies of the company that was to dominate computing into the 1970s, they were destined to become widely used for data representation until 1960s, when IBM developed yet another character code for its mainframe computers.

North American and European Computer Character Codes

ASCII, ISO 646, and EBCDIC Character Codes

As a result of the rapid development and spread of communications and data processing technologies in the United States in the first half of the 20th century, it became apparent there was a need for a standard character code for interchanging data that could handle the full character set of an English-language

typewriter. The American Standards Association (ASA, which later changed its name to the American National Standards Institute [ANSI]) began studying this problem in the late 1950s, and it eventually decided that a 7-bit code that did not require shifting in the manner of Baudot code would be sufficient. In 1963, ASA announced the American Standard Code for Information Interchange (ASCII), which originally seems to have been named the American National Standard Code for Information Interchange (ANSCII). However, ASCII as it was announced in 1963 left many positions, such as those for the lower case Latin letters, unallocated. It wasn't until 1968 that the currently used ASCII standard of 32 control characters and 96 printing characters was defined (Fig. 4; note: the "space" and "delete" positions are blank, see Fig. 5 below). Moreover, in spite of the fact that ASCII was devised to avoid shifting, it included control characters for shifting, i.e., SHIFT IN (SI) for SHIFT OUT (SO) for Baudot-style locking shift, and ESCAPE (ES) for non-locking shift. These control characters were later used to extend ASCII code into 8-bit codes with 190 printing characters.



	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(	8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[	k	{
,	<	L	\	l	|
-	=	M	]	m	}
.	>	N	^	n	~
/	?	O	_	o	

Figure 4. ASCII character set in 94-position code element

ASCII code was adopted by all U.S. computer manufacturers except IBM, which developed a proprietary character code for its mainframe computers (see below). Because U.S. computer vendors were also the largest computer vendors in the world at the time, the ASCII character code immediately became a de facto "international character code standard." Naturally, that made it necessary to adapt ASCII code to other languages that use the Latin alphabet, in particular the languages of Western Europe. This work was carried out by the International Organization for Standardization (ISO) in Geneva, Switzerland, which in 1967 issued ISO Recommendation 646. This basically called for ASCII code to be used as is except for 10 character positions to be left open for "national variants" in the manner of Baudot code mentioned above. The default characters for those 10 character positions were specified in a version of the recommendation known as the International Reference Version (IRV). ASCII code was also used as the basis for creating 7-bit character codes for languages that did not employ the Latin alphabet, such as Arabic and Greek, and in 1969 it was incorporated into the JIS character code of Japan. To date, a total of 180 character codes based on extensions of ASCII have been registered with the ISO.

Of course, the vast majority of people who come into contact with ASCII code today use it in the form of a "personal computer manufacturer's ASCII-based character set," which is usually an extended version of 7-bit ASCII designed for use throughout a region rather than in a single country. The English-language versions of the two leading personal computer operating systems, Microsoft Corp.'s MS-Windows and Apple Computer Inc.'s Macintosh, both include special characters and accent marks required for inputting Latin script-based European languages such as French, German, and Swedish without any modifications either to the operating system or word processing software. The special characters and accent marks for these languages are invoked simply by striking an option key, which unfortunately tends to slow down typing. Moreover, it should be pointed out that these 8-bit extensions of 7-bit ASCII code are not complete. For example, they do not include all the currency symbols used throughout a particular region. Furthermore, they lack certain

characters for processing historical texts, and they do not take into account some frequently used ligatures, which are two characters joined together and used as a single character.

Although the ASCII character code was adopted by manufacturers of minicomputers, workstations, and personal computers in the U.S., and later turned into an international standard by the ISO, in the field of mainframe computers, IBM--the dominant force in the world of computing in the 1960s and 1970s--decided to go its own way. This was most likely decided on the basis of concern for compatibility with past technology, but many people viewed it as primarily a marketing strategy to keep IBM customers locked into IBM hardware and software. In any event, the company created a proprietary 8-bit character code (2^8 = 256 code points) called EBCDIC [pronounced *eb-see-dick*], which stands for "Extended Binary Coded Decimal Interchange Code." It was used on the successful IBM System/360 mainframe computer series, which hit the market in April 1964. EBCDIC is an extension to 8 bits of BCDIC (Binary Coded Decimal Interchange Code), an earlier 6-bit binary code (2^6 = 64 code points) for IBM computers. (It should be noted that some sources refer to this previous 6-bit character code simply as BCD, or "Binary Coded Decimal," but BCD is also used to refer to a 4-bit character code [2^4 = 16 code points] for encoding Arabic numerals.)

Since IBM marketed its mainframe computer systems around the world, multiple versions of EBCDIC had to be defined. In fact, 57 national EBCDIC character code sets were eventually drawn up by the company. Although an International Reference Version (IRV) of EBCDIC based on ISO 646 (and hence ASCII compatible) was also created, most of the EBCDIC character sets do not contain all the characters included in ASCII, which precluded the efficient exchange of data between EBCDIC-based systems and ASCII-based systems. Moreover, detailed documentation on this large collection of character sets was not easily accessible--hence the above-mentioned view that EBCDIC was primarily a marketing ploy to trap customers in the IBM web--and consequently the international exchange of data even between IBM mainframes was a tricky task for system operators, who became overnight EBCDIC haters. In the Internet-based computing world of today, where 7-bit and 8-bit text is transmitted on the basis of ASCII and ISO codes, EBCDIC is a character code that can only serve to slow the exchange of information. For that reason, many businesses that have data in EBCDIC files are converting them into ASCII and other non-proprietary formats.

Accented Latin and Non-Latin European Character Codes

While 7-bit character codes such as ASCII and ISO 646 are sufficient for processing English-language data, specifically modern English-language data, they are not adequate for processing data written in most of the Latin-based scripts of Europe, which employ various accent marks together with the letters of the Latin script. Moreover, in Europe there are also non-Latin, native scripts, such as the Greek alphabet, in addition to scripts from nations around the periphery of Europe, such as Arabic, Cyrillic, and Hebrew, which also have occasion to be used within the borders of Europe. Accordingly, after ASCII and ISO 646 were standardized, it became necessary to go well beyond them and create a number of new character codes to handle these European data processing needs. To that end, the International Organization for Standardization first created a standard called ISO 2022, which outlines how 7-bit and 8-bit character codes are to be structured and extended. This standard was later applied to create the standard unofficially known as "Latin-1" (ISO 8859-1), which is an extension of ASCII/ISO 646 that is widely used for the interchange of information across the Internet in Western Europe today.

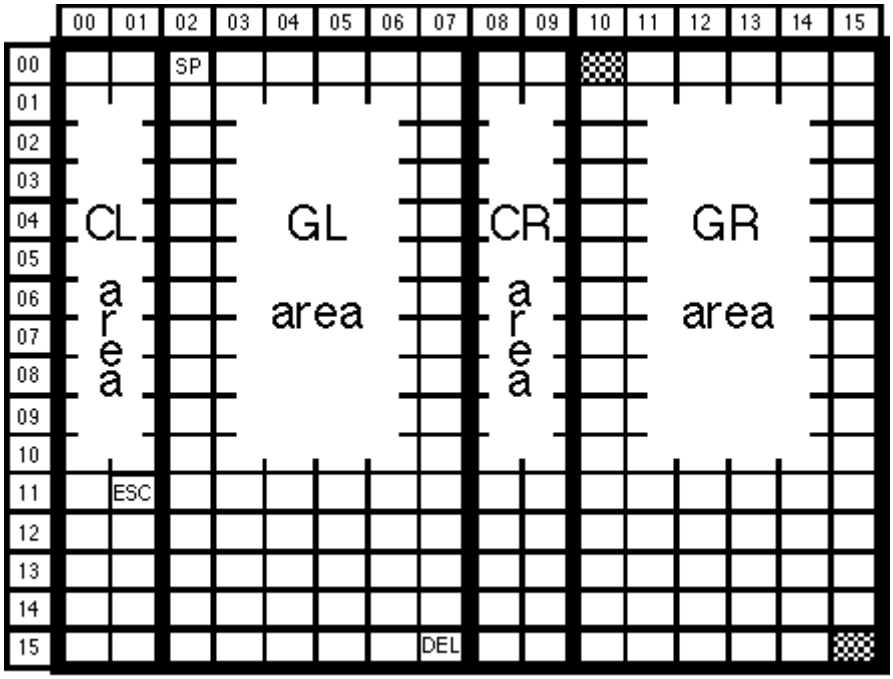


Figure 5. Layout of 8-bit code table

ISO 2022 establishes, among other things, how 7-bit and 8-bit character code tables are to be laid out (8 columns by 16 rows for 7-bit code tables, and 16 columns by 16 rows for 8-bit code tables); how their columns and rows are to be numbered (7-bit: 00~07 x 00~15; 8-bit: 00~15 x 00~15); how they are to be structured (the 7-bit code table is divided into CL and GL areas, and the 8-bit code table is divided into CL and GL, and CR and GR areas; the C areas contain control functions and the G areas contain graphic characters); and the specific locations (technically called "fixed assignments") within code tables for the ESCAPE (ESC: 01/11), SPACE (SP: 02/00), and DELETE (DEL: 07/15) control characters, which are their positions in the ASCII standard (Fig.5). However, unlike ASCII, where ESCAPE was created to serve as a kind of "typewriter carriage return," in ISO 2022 it exists to support "computer-style control functions," such as escape sequences for switching character sets. Finally, ISO 2022 also defines the terminology used in describing character sets (see Table 1), so anyone interested in becoming a specialist in character codes is encouraged to start with this standard.

Table 1 Formal Definitions of Character Set Terminology in ISO 2022

Bit combination
An ordered set of bits used for the representation of characters

Byte
A bit string that is operated upon as a unit (includes 7-bit, 8-bit, and 16-bit "bytes," octet equals 8 bits)

Coded character set; code
A set of unambiguous rules that establishes a character set and the one-to-one relationship between the characters of the set and their bit combinations

Code table
A table showing the character allocated to each bit combination in a code

Code extension
The techniques for the encoding of characters that are not included in the character set of a given code

Combining character
A member of an identified subset of a coded character set, intended for combination with the preceding or following graphic character, or with a sequence of combining characters preceded or followed by a non-combining character

Control character
A control function the coded representation of which consists of a single bit combination

Control function
An action that affects the recording, processing, transmission or interpretation of data, and that has a coded representation consisting of one or more bit combinations

Designate
To identify a set of characters that are to be represented, in some cases immediately and in others on the occurrence of a further control function, in a prescribed manner

Escape sequence
A string of bit combinations that is used for control purposes in code extension procedures; the first of these bit combinations represents the control function ESCAPE

Graphic character
A character, other than a control function, that has a visual representation normally handwritten, printed or displayed, and that has a code representation consisting of one or more bit combinations

Invoke
To cause a designated set of characters to be represented by one or more bit combinations of a coded character set

Repertoire
A specified set of characters that are each represented by one or more bit combinations of a coded character set

Once ISO 2022 was in place as an international standard, it allowed all sorts of other character codes to be wrapped around ISO 646. Its inventors compared it to a typewriter on which the typehead could be changed--the interchangeable typeheads being the 7-bit and 8-bit character code sets built on top of ISO 2022. The most important collection of these interchangeable "typeheads" is defined in ISO 8859, which is a massive, multipart 8-bit character code set aimed squarely at data processing needs of Western and Eastern Europe. The standard is divided as follows:

ISO 8859
Part 1: *Latin alphabet No. 1* (Revised 1998)
Character sets of Western European languages (this is the famous "Latin-1")
Part 2: *Latin alphabet No. 2*
Character sets of Eastern European languages (Slavic, Albanian, Hungarian, Romanian)
Part 3: *Latin alphabet No. 3*
Character sets of Southern European languages (Maltese) plus Esperanto
Part 4: *Latin alphabet No. 4* (1998)
Northern European languages
Part 5: *Latin/Cyrillic alphabet*
Part 6: *Latin/Arabic alphabet*
Part 7: *Latin/Greek alphabet*
Part 8: *Latin/Hebrew alphabet*
Part 9: *Latin alphabet No. 5*
Latin character set used for modern Turkish
Part 10: *Latin alphabet No. 6* (1998)
Icelandic, Nordic, and Baltic character sets
Part 13 (DIS) *Latin alphabet No. 7*
Part 14 (DIS) *Latin alphabet No. 8* (Celtic)

Of course, everybody knows that in the real world companies do not voluntarily follow standards, particularly when they were drawn up in foreign countries, and that is exactly what happened with ISO 8859. ISO 8859 could not come into widespread use until personal computers appeared on the scene. However,

since personal computers were an American invention, the American computer industry adapted ISO standards to suit its needs. The result was ASCII-based personal computer operating systems with OS-specific extensions for European languages that are not completely compatible with ISO 8859. Want to see the results? Look under the Document Encoding menu item of your Web browser (in Netscape Corporation's Netscape Navigator, it is under the Options menu). There you will discover special character sets listed for Microsoft Corporation's MS-Windows and Apple Computer Inc.'s Macintosh operating systems. Fortunately, the inventors of the Internet created data transfer protocols that allow for textual data written in any number of different character sets--even character sets defined by a single individual (look way down at the bottom of the Document Encoding menu)--to be transferred across it. Thus these character set variations are merely a troublesome nuisance, and not an insurmountable barrier, to the international transfer of data.

Chinese, Japanese, and Korean (CJK) Character Codes

Telegraph Codes

In East Asia, as in the U.S. and Europe, the first character codes for the national scripts of China, Japan, and Korea were developed for telegraph use. Some might ask why these were created in the first place--after all, Romanization systems existed, so Chinese, Japanese, and Korean language data could have easily been sent via Morse code. However, Romanization is not adequate for sending place names and personal names written with Chinese characters, since there are so many homonyms. It was necessary, therefore, to systematically number thousands of Chinese characters. Moreover, there were other considerations. Because the telegraph had come into being in an era when there were few public schools in East Asia, many of the recipients of a Romanized telegraph message would have been unable to read it. In addition, there was the cost of sending messages via the telegraph, which was very expensive at the time. Thus if the use of a Romanization system led to the lengthening of a message, the cost of sending that message would rise accordingly. In fact, at the beginning of the 20th century the cost of sending telegraph messages was so high, that even in the U.S. and Europe special company- and/or industry-specific telegraph codes were created to shorten several words into one word in order to save money on telegraph costs. (See here for more information.) Consequently, the characters of the scripts of the languages of East Asia came to be numbered and set down in character code sets for the first time in the age of the telegraph. However, the most important character codes for East Asian languages were to be those developed for use on computer systems in 1970s, 1980s, and 1990s, since they would eventually be used by the masses.

Japanese Computer Character Codes

The Europeans and the ISO were not the last people to wrap national character sets around ISO 646. On the other side of the world in Japan, the Japanese electronics industry was in the process of building itself into a world power, and one of the things it was focussing on was the development of computer technologies capable of processing the Japanese language. To that end, the Japan Industrial Standards Committee (JISC) was given the task of drawing up character sets for Japanese-language information interchange. However, processing Japanese on a computer is a considerably more difficult task than processing English and other European languages on a computer. While 7-bit or 8-bit character sets are adequate for English and other European languages, a 16-bit character set is mandatory for Japanese. This is because four different scripts are normally used in the processing of Japanese--the two *kana* syllabaries (*hiragana* and *katakana*), several thousand *kanji* (Chinese characters), plus the Latin alphabet. Nevertheless, in the 1970s, an 8-bit JIS character code was created for processing Japanese in "rudimentary form." Called JIS C 6220 (it was renamed JIS X 0201-1976 in 1987), its left half consisted of a Japanese national variant of ISO 646 (a yen symbol replaced the backslash, and an overline replaced the tilde), and its right half consisted of Japanese punctuation marks and the *katakana* syllabary. This allowed the transfer of data between early Japanese computer systems. JIS X 0201-1976 consists of the following:

JIS X 0201-1976
numerals (10)
Latin alphabet (52)
symbols (32)
non-printing characters (34)
katakana (63 half-width characters)

Of course, since Japan's first character code for data processing did not even include the *hiragana* syllabary, anyone who wanted to make easy money could have bet that JISC would soon create a Japanese equivalent of ISO 2022 for switching between 8-bit character sets. They did, and they called it JIS X 0202, which was published both in Japanese and English. Then, JISC began creating 94 x 94 tables for the many *kanji* used for writing the Japanese language. In JIS standards, *kanji* are divided into "levels," although there are also what are referred to as "JIS auxiliary or supplementary *kanji* " that have been defined. The first of these levels appeared in JIS C 6226-1978, or "Old JIS." Old JIS was redefined to include of the *Jooyo kanji* [*kanji* for daily use] list drawn up by the Japanese government in 1981, which led to the establishment of JIS 6226-1983, or the "New JIS" standard. New JIS was subsequently renamed JIS X 0208-1983 in 1987; and then in 1990 level 2 *kanji* were added to level 1 *kanji* to create JIS X 0208-1990, which is the standard in widest used today. In addition, an auxiliary *kanji* standard, JIS X 0212-1990, was also created in 1990, although only one personal computer operating system (the BTRON-specification operating system, see below) supports it at present. A breakdown of the characters included in these two 1990 standards is as follows:

JIS X 0208-1990
punctuation, symbols (93, 53)
ISO 646 alphanumerics (10 numerals, 52 characters)
hiragana (83)
katakana (86)
Greek alphabet (48)
Cyrillic (Russian) alphabet (66)
line drawing elements (32)
kanji level 1 (2,965 characters, ordered by Chinese style reading)
kanji level 2 (3,390 characters, order by Chinese character radical)
miscellaneous *kanji* (6 characters)
JIS X 0212-1990
diacritics and symbols (21)
Greek characters (21 letters with diacritics)
Eastern European characters (26)
alphabetic characters (198)
kanji (5,801 characters)

As can be seen from the above, JIS X 0208-1990 is in fact a "multiscript character set." In its original form, JIS is supposed to use escape sequences--which can be fairly complex--to switch between the 7-bit characters of ISO 646 and the 16-bit *kanji* characters. However, since these required extra processing time and ate up disk storage space, JIS in its original form was deemed to be inappropriate for use on small computer systems, such as early 16-bit personal computers. To get around this, Microsoft Corporation invented an "encoding method" for the JIS character set called Shift-JIS, which eliminates the escape sequences, and thus the need to switch between character sets. It does this by "specifying value ranges" for 8-bit (one-byte, or alphabetic characters) and the first byte of 16-bit (two-byte, or *kanji*) characters. Depending on the value it meets in character data, the computer determines whether it has encountered a one-byte alphabetic character or the first byte of a two-byte *kanji* character, and then displays characters accordingly. There are two problems with this scheme though. The first is that by establishing ranges in which only certain characters can fall, the possible coding space for characters is reduced. Thus Shift-JIS-based personal computers can only use JIS X 0208-1990 character set; they cannot utilize JIS X 0212-1990 supplemental *kanji* as part of their fixed character set. The second is that without special software a Shift-JIS computer can only handle character data encoded with 8 bits or 16 bits. However, on a communications medium such as the Internet, character data encoded with only 7 bits can also exist, which can lead to character corruption after the downloading of data.

Because Shift-JIS is far from being the ideal encoding method for the JIS character set, particularly when it is used in a heterogenous environment such as the Internet, other encoding methods were developed for encoding JIS. One of these is ISO-2022-JP (a subset of 7-bit ISO 2022), which is widely used for sending e-mail and network news messages on the Internet in 7-bit environments. As can being surmised from its name, it is based on the ISO 2022 standard mentioned above, and thus it uses the escape sequences JIS-Shift avoids, to switch between the ASCII and *kanji* character sets. Although ISO-2022-JP is well liked by server operators, they report that there are some problems when it is used in HyperText Markup Language (HTML), the programming language used for creating hypermedia pages on the World Wide Web. The other encoding

method for JIS is the method called Extended UNIX Code-JP (EUC-JP), which as the name indicates is the encoding method used on the UNIX workstations that are the backbone of the Internet. EUC works in 8-bit environments and is basically a "wrap around" scheme in which the character codes of East Asian languages (Chinese, Japanese, and Korean) are wrapped around the local version of the ASCII character set. EUC does this by "adding values" to the characters of each East Asian language character set. In the case of EUC-JP, for example, values are added to the characters of JIS, which distinguishes them from ASCII character codes. The advantages of this system is that any two of four character sets (the local ASCII and the national character set) can be handled without the use of escape sequences, and even the JIS X 0212-1990 *kanji* can be used. The disadvantages are that it requires a powerful processor and a lot of disk space to implement. However, since those resources are available in the world of UNIX workstations, the disadvantages were not deemed to outweigh the advantages.

While JISC was developing its character sets and encoding methods--in fact, it continues to do so; JISC has opened JIS levels 3 and 4 to public review--China, Korea, and Taiwan were watching what was going on as East Asia's most technically advanced nation as it computerized its national language. It should comes as no surprise then to learn that the national character sets defined for use in China, Korea, and Taiwan--which all include Chinese characters--share some of the features of JIS character code sets. For example, like JIS, they are wrapped around the local variant of ISO 646 that includes the local currency sign instead of the backslash, they are built up on 94 x 94 tables, they include characters from other scripts (Cyrillic, Greek, Japanese syllabaries, etc.), they have line drawing elements, and so on. However, the scripts of China, Korea, and Taiwan also have their own peculiarities. Let's take a look at what characters are included in the present standard character sets in those countries below.

Chinese Computer Character Codes

China's national script stands out among those of the countries in the East Asia in that it uses the greatest number of "simplified Chinese characters," and for that reason, the character set of the People's Republic of China is called "Simplified Chinese." These simplified characters were originally supposed to be a step along the way to an easy-to-learn Romanized writing system for the national language aimed at increasing national literacy, but that goal has been abandoned due to problems the vast number of homonyms in the Chinese language would cause. The result is that the Chinese character set used in China is very different from the one used in Taiwan. The current standard in China is GB 2312-80 (GB is an abbreviation of *Guo-jia Biao-zhun*, or "National Standard"). The characters it includes are as follows:

GB 2312-80
symbols (94)
numerals (72)
ISO 646-CN (94 full-width characters)
hiragana (83)
katakana (86)
Greek alphabet (48)
Cyrillic (Russian) alphabet (66)
pinyin and *bopomofo* characters (26, 37)
line-drawing elements (76)
hanzi level 1 (3,755, ordered by *pinyin* reading)
hanzi level 2 (3,008, ordered by Chinese character radical, then stroke)

Simplified Chinese uses the following encoding methods: 7-bit ISO 2022, ISO-2022-CN (e-mail message encoding), EUC-CN, and HZ (HZ-GB-2312). HZ is an encoding method used for e-mail that was created by a Chinese researcher at Stanford University. It is similar to 7-bit ISO 2022, the main difference being that it uses "shift sequences of two printable characters" rather than an escape sequence to switch from one-byte (alphabetic) characters to two-byte (*hanzi*) characters.

Korean Computer Character Codes

Korea is the only country in East Asia whose national script is an alphabet, which is called *hangul*. Although Chinese characters are mixed together with *hangul* in writing, the trend in Korea is toward moving away from the use of Chinese characters, except for those used in place names and personal names. The *hangul*

script only has 24 letters (10 vowels and 14 consonants), so it would seem that processing it on a computer would be a relatively easy matter. However, in writing, the letters of *hangul* are built into "syllabic blocks" in which characters on stacked on top of each other--and this has led to heated debates about how the characters of the simple Korean writing system should be processed inside a computer system. The current standard for data processing and information interchange, KS C 5601-1992 (KS stands for "Korean Standard"), registers the most frequently used syllables in prearranged syllabic blocks. The KS C 5601-1992 standard is made up as follows:

KS C 5601-1992
symbols (94)
abbreviations and symbols (69)
ISO 646-KR (94 full-width characters)
hangul elements (94)
Roman numerals and letters of the Greek alphabet (68)
line-drawing elements (68)
abbreviations (79)
phonetic symbols, circled characters, and fractions (91)
phonetic symbols, parenthesized characters, subscripts, and superscripts (94)
hiragana (83)
katakana (86)
Cyrillic (Russian) alphabet (66)
hangul (2,350 pre-combined syllables)
hanja (4,888 Chinese characters ordered by hangul reading, then radical)

Korean is encoded with the following methods: 7-bit ISO 2022, ISO-2022-KR (e-mail message encoding), EUC-KR, and Johab. Johab, or "two-byte combination code," is a system in which all possible *hangul* syllabic blocks (11,172 of them) are encoded, whether they actually occur in the Korean language or not. Microsoft created a "Unified Hangul Code" (UHC), also called "Extended Wansung," for the Korean version its Windows 95 operating system (Win95K), which maintains compatibility with KS C 5601-1992 while adding support for the full Johab encoding system (8,822 precombined syllables that are not in KS C 5601-1992). In addition, Microsoft Corporation is planning to support this system on its Korean Windows NT operating system.

Taiwanese Computer Character Codes

Although so-called Mandarin Chinese is spoken in both Taiwan and China, the writing system of Taiwan is different from that of the mainland in that it is based on Chinese characters as they were traditionally written for centuries, and thus Taiwan's character set is called "Traditional Chinese." The Chinese characters of the Traditional Chinese character set, incidentally, are very similar to the the Chinese characters used in Hong Kong, and Singapore. To date, two character sets have been developed for processing Traditional Chinese on computers. These are the Big-5 (so called because it was drawn up by "five large computer makers") and CNS 11643-1992 (CNS stands for "Chinese National Standard") character sets, which are somewhat compatible. The main difference is that CNS 11643-1992 contains considerably more Chinese characters than the Big-5 character set. The characters in these standards are as follows:

Big-5
symbols (157)
symbols (157)
symbols (94)
hanzi level 1 (5,401 Chinese characters ordered by number of strokes, then radical)
hanzi level 2 (7,652 Chinese characters ordered by number of strokes, then radical)
(Note: 2 Chinese characters in *hanzi* level 2 are duplicates, so there are only 7,650 unique *hanzi*)
CNS 11643-1992
symbols and *hanzi* :
- 438 symbols
- 213 classical radicals
- 33 graphic representations of control characters
- *hanzi* 1 (5,401 Chinese characters)

hanzi 2 (7,650 Chinese characters)
hanzi 3 (6,148 Chinese characters)
hanzi 4 (7,298 Chinese characters)
hanzi 5 (8,603 Chinese characters)
hanzi 6 (6,388 Chinese characters)
hanzi 7 (6,539 Chinese characters)

Traditional Chinese is encoded with the following methods: 7-bit ISO 2022, ISO-2022-CN (e-mail message encoding), EUC-TW, and Big-5. Big-5 is the encoding system used on personal computers based on Microsoft Corporation's MS-DOS and MS-Windows operating systems, and Apple Computer Inc.'s Macintosh operating system.

It should be pointed out that all of the character sets mentioned in this section bear a date in their designations. This indicates that the character sets have been updated, usually through extension, a number of times. When these updates are done in a logical and consistent manner, it benefits the users of the computer systems that incorporate these character sets. However, if the updates involve the shifting of character positions, then the interchange of data between systems using different versions of the same character set can lead to the loss of data. This has been the case with some of the character sets developed by JISC.

Multilingual Character Sets and Processing Schemes

Early Multilingual Systems

So far character codes for data processing and information interchange have been discussed only in terms of the needs on individual nations, but in the real world there are many cases in which it is necessary to process multiple languages in a single document. For example, the publishing industry needs to use multiple national scripts in foreign language dictionaries and textbooks, and now that we have entered the Age of the Internet, it is more likely than ever that people will come into contact with foreign language data written in the character set of another nation. For such reasons, efficient multilingual character sets and multilingual data processing schemes are essential for life in the 21st century, but such character codes and processing systems have not been the subject of research and development for very long. The earliest multilingual character sets and processing schemes appeared in the first half of the 1980s, a time when most personal computers were 8-bit and 16-bit machines. The most noteworthy of these were Xerox Corporation's Star Workstation, which had a multilingual word processor called ViewPoint, and IBM Corporation's 5550 office computers. Both of these systems could process multiple Asian languages, in addition to multiple languages that use the Latin script, but they were never came into wide use because of their high cost. The Xerox multilingual character code, on the other hand, took on a following among computer specialists and linguists in the U.S. This eventually led to the Unicode movement that created a multilingual character set and encoding system now vying for worldwide acceptance (see below).

TRON Character Code and the TRON Multilingual Environment

It may come as a surprise to some, but Japan, which is often criticized for having a closed and inward looking society, also proposed a multilingual character set and processing scheme in the first half of the 1980s. These were proposed as a part of the TRON Project, which was launched in 1984. The goal of the TRON Project is to create a "total, open computer architecture" based on the current von Neumann standard for the computerization of human society in the 21st century. Naturally, part of computerizing human society involves putting masses of human knowledge on line in a way that it can accessed by all, and that requires the creation of a well designed character set and an efficient processing scheme for handling multiple languages on personal computers and workstations. To that end, an open-ended TRON character code and the TRON Multilingual Environment have been developed and implemented as part of the BTRON-specification operating system, which has been commercialized by Personal Media Corporation. The latest version of the BTRON-specification operating system from Personal Media, B-right/V, handles multiple East Asian character sets as a standard feature, and the next version, which has been tentatively named "3B/V," will be able to utilize an unabridged *kanji* character set of roughly 65,000 characters called GT Mincho, which is currently being drawn up at the University of Tokyo.

The TRON character code and the TRON Multilingual Environment were described in English for the first time in the proceedings of the Third TRON Project Symposium in 1987. There are several features that make the TRON approach to multilingual processing unique. One is that the TRON character set is "limitlessly extensible," and thus it is capable of including all scripts that have ever been used, and even new scripts that have yet to be invented. This is done through escape sequences, which are used to switch between 8-bit and 16-bit character sets on very large character planes. Another is that it employs "language specifier codes," which are necessary so that the correct sorting algorithms, for example, can be applied to data in a multilingual environment. Still others are that it does not provide for "user defined characters," which can cause problems during data transmission (graphic elements embedded in text strings are used instead); and it separates text data for storage from text data for display, since two letters at the storage level, for example, can be merged into one character at the display level in the form of a "ligature." Other interesting features are: a character database, which is particularly necessary for properly employing the huge GT Mincho character set; and a multilingual writing system, which allows multilingual data to be sent across the World Wide Web and displayed with reasonable quality without complex pagemaking algorithms at the other end.

The BTRON3-specification B-right/V operating system, which does not fully implement the TRON Multilingual Environment, currently supports the following character sets:

BTRON3 Character Sets (Partial)
ISO 8859-1
GB 2312-80
JIS X 0208-1990
JIS X 0212-1990
KS C 5601-1992
Six-dot braille
Eight-dot braille

Although the TRON character code and TRON Multilingual Environment are superior to anything that has been proposed to date for multilingual data processing and information interchange, there are many who would contend that it has little chance of becoming a standard, since certain major U.S. software houses that produce operating systems are opposed to it. However, this opinion ignores the fact that there is a "real need" in Japanese society for a personal computer operating system that can handle all the *kanji* that have been created for writing the Japanese language to date. Who needs it? For starters, banks, government offices, schools, and any other organization that has to record the names of Japanese people in their original form on lists or in databases. In addition, Japanese libraries need it to computerize their card catalogues and to put on line sections of their collections for which copyrights have expired. Finally, once page layout software is developed to run on the BTRON-specification operating system, the Japanese publishing industry needs it. So far from fading away because it has not been given the blessing of certain American software houses that want to control operating system software far into the future, chances are that TRON character code and the TRON Multilingual Environment will be around for a long time.

Unicode and ISO 10646

As mentioned above, U.S. computer firms began work in the first half of the 1980s on multilingual character sets and multilingual character encoding systems, and Xerox Corporation and IBM Corporation successfully implemented computer systems based on their research results. The Xerox researchers then proselytized their work to other U.S. software firms, and they were eventually successful in launching a U.S. industry project called Unification Code, or Unicode, the goal of which was to unify all of the worlds character sets into a single large character set. Unicode was also to be simple for computers to process, and to that end it had two important design goals: (1) avoiding the use of escape sequences to switch planes of characters, and (2) limiting the character space to 16 bits, or a maximum of 65,536 characters, and giving each character a fixed-length code (16 bits, or two bytes). However, the International Organization for Standardization (ISO) in Geneva, Switzerland, the creator of those pesky escape sequences in the widely used ISO 2022 standard, also wanted to create a multilingual character code and encoding system. Unlike Unicode, however, it aimed at creating a 32-bit Universal Coded Character Set (UCS), which would use escape sequences to switch between large planes of characters that together would have enough space for as many as 4,294,967,296 characters--in other words, a character code that was for all practical purposes unlimited in nature **[1]**.

This ISO multilingual standard, which went by the name of ISO/IEC DIS 10646 Version 1, was supported by Japanese and European researchers. However, ISO/IEC DIS 10646 Version 1 was not supported by the American computer firms, which were doing parallel research on Unicode, and who had even gone so far as to create a Unicode Consortium to conduct that research. They deemed Unicode to be superior to ISO/IEC DIS 10646 Ver. 1, since it was simpler. For that reason, they counter proposed that Unicode be made the "Basic Multilingual Plane" [2] of the ISO's multilingual standard. Of course, since they were also the developers of the world's leading operating systems, they were in a position to create a parallel, de facto alternative multilingual character code and encoding system to any multilingual character set and encoding system that the ISO might develop. Accordingly, they were successful in persuading the ISO that ISO/IEC DIS 10646 Version 1 should be dropped. It was, and a Unicode-based multilingual scheme called ISO/IEC 10646 Version 2 came into being. In essence, Unicode had swallowed the ISO standard, which is now called ISO/IEC 10646-1: 1993. It consists of the following:

ISO/IEC 10646-1: 1993
ISO 646
ISO 8859-1
Eastern European accented characters
International Phonetic Alphabet (IPA)
Greek (including accented characters, "monotoniko" and "polytoniko")
Cyrillic, Georgian and Armenian
Hebrew
Arabic characters (all four forms: initial, medial, final and stand-alone)
Indian subcontinent character sets (including Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada and Malayalam)
Thai and Lao
Chinese/Japanese/Korean (CJK) ideographic characters (including *hangul*, *katakana*, *hiragana*, and *bopomofo*)
Mathematical operators and special character forms
Box and line drawing characters
Geometric shapes and Dingbats
Special OCR characters used on cheques
Encircled characters and numbers

The reader might be asking him/herself how could all the world's characters be squeezed into 65,536 character points when it was mentioned above that a character set of about 65,000 characters is being developed at the University of Tokyo just for Japanese? The answer is that this was to be achieved through the "unification" of similar characters. No, all the 'A's in alphabetic scripts were not to be unified into one character point. What was to be unified were the thousands of Chinese characters that make up the scripts of East Asian languages. In fact, the Unicode Consortium has set up a Chinese/Japanese/Korean Joint Research Group (CJK-JRG) that is busily carrying out the Unicode Consortium's main goal, "Han Unification," even though the vast majority of the people living in the Han cultural sphere are probably not aware of what they are doing. To date, they have assigned approximately 20,000 Chinese characters to code points, and there are another 30,000 code points left to be filled. Although Unicode has been criticized as being little more than an exercise in cultural imperialism on the part of western computer manufacturers, the biggest problems with Unicode are in fact technical ones, which can be listed as follows:

1) There is no information to identify the language being used, which affects sorting, etc.
2) User defined characters, which cause problems in data transmission, are allowed
3) There is no room for further expansion using two-byte (16-bit) codes
4) There is an excess of similar characters
5) Some characters are created through composition, which destroys the fixed-length scheme
6) Conversion to/from Unicode is not simple for users of Chinese, Japanese, and Korean

Even though one of the driving forces behind Unicode is the powerful Microsoft Corporation, which has implemented Unicode on its Windows NT server operating system, Unicode is meeting fierce opposition in East Asia. This is not just because the people of East Asia do not want American computer vendors deciding what characters they can and cannot use on their computer systems. That, of course, quite understandably exists. More importantly, it is because "Unicode does not answer the needs of the on-line East Asian societies

of the future." For example, Unicode is inadequate for creating on-line digital libraries. Such libraries need unabridged character sets that include every character that has ever been used. Unicode does not provide that. Unicode-based systems also cannot be used for writing the personal and place names of many people and places in Japan, so they cannot be used for computerizing government offices in Japan. Because of its limited nature, Unicode likewise cannot be used for computerizing print shops and publishing offices in East Asia. So in the final analysis, Unicode is really of more benefit to computers themselves and their manufacturers, rather than to computer users in East Asia. However, considering its design goals, it is hardly surprising that it turned out to be such. (For more information on Unicode, click here)**[3]**.

UNIX and Mule

The people who created the Internet, which is in the process of becoming the main thoroughfare for the international exchange of multilingual data, are none other than the programmers and engineers who spend day after day working at their UNIX-based workstations. Their world is very different from the the world of personal computing. They run their high-performance workstations using operating systems that are either partially or completely open. They have access to lots of free software applications and development tools, and people who write programs often make them available to the other members of the UNIX community free of charge. Thus unlike the world of personal computing, where technical standards are created by powerful market leaders whose main goal is to create and maintain technological hegemony, the world of UNIX is based on openness, cooperation, and tolerance. In such a world, it should not be surprising that there is little support for Unicode, which in addition to being controversial is still incomplete. However, it might be surprising to some that the UNIX world's answer to the problem of handling multiple languages is, like the TRON Multilingual Environment mentioned above, a product of Japan.

The UNIX approach to multilingual processing is simple, although it requires powerful computing resources to realize it. UNIX accepts as is whatever character sets have been drawn up by the countries of the world, and then runs them through a multilingual-capable editor with a unified and extensible mechanism for handling them. This editor is called Mule (MULtilingual Enhancement to GNU Emacs), and it was developed at the Japanese government's Electrotechincal Laboratory in the science city of Tuskuba to the northeast of Tokyo by a research group under the direction of Kenichi Handa. Mule, which was released as freeware in 1993, is an outgrowth of a previous enhancement of the freeware GNU Emacs editor known as Nemacs, or Nihongo [Japanese] Emacs. As in the case of the TRON Multilingual Environment and the multilingual character encoding system originally proposed by the ISO, Mule uses escape sequences to switch between languages. The present implementation can handle European languages, Greek, Russian, East Asian languages, Thai, Vietnamese, Hebrew, Arabic, and others. Support for Indian subcontinent languages is presently being developed, and the entire system is being integrated into the original GNU Emacs editor for distribution worldwide. (For more inforamtion on Mule. click here.)

Some people would say that since Mule was developed by a software professional for the UNIX community, it has little chance of coming into widespread use. But there are several factors that could lead to Mule breaking out of the traditional UNIX world and gaining wider acceptance. These are: (1) Mule allows people to use existing character sets and encoding methods, so it is not controversial; (2) personal computers are now more powerful than early UNIX workstations, so the computing power is there to support Mule; and (3) the UNIX-like freeware operating system Linux is beginning to gain widespread acceptance--particularly in Europe--so people can easily get their hands on a platform for running it. The only question is, how many people will be willing to switch over to a UNIX-like operating system from current personal computer operating systems? On the other hand, using a longer term perspective, one has to wonder how long Mule will be around. Mule is clearly a stopgap measure that answers a "present need" for multilingual computing at a time when there are conflicting standards being proposed for multilingual information interchange in the 21st century. However, until one of those emerges as a clear leader, Mule will will probably play an important role for people who require a multilingual computing environment and are willing to switch operating systems to do it.

Future of Multilingual Character Sets and Encoding

As anyone with even a casual knowledge of computing can easily predict at this point, the future of computing is going to revolve around networking on a worldwide scale. The Internet of today is but this vast worldwide network in its embryonic state. However, networking on a worldwide scale cannot take place

efficiently until many more high-speed trunk lines are laid, high-speed local lines are connected to factories, offices, and residences, and new standards are created for the exchange of data. Among these new standards that have to be created, one of the most important will be a multilingual character code set and encoding system for processing data written in all of the world's languages. To date, two new standards have been proposed to meet this need. One is the TRON character code and TRON Multilingual Environment, and the other is Unicode. In the interim, the multilingual Mule editor that runs of UNIX and UNIX-like operating systems offers a compromise.

So how are things going to turn out in the future? That's hard to predict, but unless the American computer vendors with or without the help of the U.S. government try to force Unicode on the rest of the world in some sort of "market opening campaign," the most likely outcome is that there will be a battle between competing standards that will eventually be decided on the basis of what computer users, not computer vendors, prefer. The commercially available BTRON-based operating system currently runs on the same hardware as Unicode-based Windows NT, so all users will have to do in Japan is switch from one side of a hard disk partition to the other to select the system of their choice. Moreover, the Internet communication protocols, as noted above, allow for multiple character sets and encoding methods to be used, so this battle between competing multilingual character code sets and encoding systems will have little effect on data networks.

And thus one can predict with certainty that for the first time in a long time, personal computer users are really going to have a choice!

---

[1] Unlike earlier character sets in which code tables were arranged into tiny 16 x 16 or 94 x 94 grids, both Unicode and the ISO's UCS use code tables based on large 256 x 256 grids. The main difference between Unicode and the ISO's UCS is that the former uses a single such grid while the latter uses thousands of them.

[2] It should be noted that the "Basic Multilingual Plane" of ISO/IEC DIS 10646 was originally defined as follows: "Basic Multilingual Plane: The plane containing widely applicable non-ideographic characters, i.e., Group 32 Plane 032." In other words, it was never intend for the character sets of Chinese, Japanese, or Korean.

[3] After this article was written, the Unicode Consortium implemented a mechanism that allows Unicode to process in excess of 1 million characters. For a description of this mechanism plus TRON Web's appraisal of it, please see the article "Unicode Revisited," which was also written by the author of this article.

Bibliography

Lunde, Ken. *Understanding Japanese Information Processing*. Sebastopol: O'Reilly & Associates, 1993. (Some of the information in this work is available on the Internet; click here.)

Lunde, Ken. *CJKV Information Processing: Chinese, Japanese, Korean & Vietnamese Computing*. Sebastopol: O'Reilly & Associates, 1998.

Sakamura, Ken. "Multi-language Character Sets Handling in TAD." TRON Project 1987 Open-Architecture Systems: Proceedings of the Third TRON Project Symposium. Tokyo: Springer-Verlag, 1987, pp. 97-111. (The contents of this paper are available on TRON Web, click here.)

Sakamura, Ken. "Multilingual Computing as a Global Communications Infrastructure." *Proceedings of the 12th TRON Project International Symposium*. Los Alamitos: IEEE Computer Society Press, 1995, pp. 2-14.

Web References

Information on 8-bit character sets is available from the Comité Européen de Normalisation (CEN), or European Committee for Standardization, at the following URLs:

8-Bit Character Sets: Index to Standards

http://www.stri.is/TC304/guide/gindex.htm

8-Bit Character Sets: Concepts and Terminology

http://www.stri.is/TC304/guide/gconcept.htm

8-Bit Character Sets: Historical Background

http://www.stri.is/TC304/guide/ghist.htm#ascii

There is a list of character set standards at the following European Commission Open Information Interchange (OII) URL; all major character set standards are listed and summarized here. However, there is no information on TRON Project or the GT Mincho font.

http://www2.echo.lu/oii/en/chars.html

Last updated: August 6, 2004