INSIDE ASCII

By R. W. Bemer

The data alphabet called ASCII (Figure 1, page 98, and Reference 1), also has two other names—International Standard 646 (the ISO Code [Reference 2]) and Alphabet No. 5 of CCITT (the International Consultative Committee for Telephone and Telegraph). It is used throughout the world, incorporated in billions of dollars of equipment.

But is it used correctly and wisely? Not always. There are misinterpretations, and gaps in definition that permit nonstandard usage. This article (in three parts) will give you the background, peculiarities, preferred practices, and new developments for ASCII. You will find a lot of information not too generally known or realized; it should help in the correct and safe usage of ASCII. For additional help, you can reference the various national and international standards given in Table 1. Some other detailed articles are listed in References 3, 4 and 5.

	ISO	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Binary-coded Character Set	646	6	X3.4-1977 $4.50	1	Z243.4	4730	1776	V.3	C6220	13052-67
Graphics for Control Characters	2047	17	X3.32-1973 $3.50	36		4730				
Character Set for Handprinting	97/3 N119		X3.45-1974 $5.75	35	Z243.34.1					
Additional Controls Character Imaging		48	BSR X3.64							
4-bit Sets	963	14		15	Z243.6	4731/1	1070			
Code Extension Techniques	2022	35	X3.41-1974 $6.00	35	Z243.35	4953				
Registration Procedures for Escape Sequences	2375									
8-bit Coded Character Set	DIS 4873	43	X3L2/77/08							
Character Set for 7 x 9 Matrix Printers		42								
Keyboard	2530	23	X4.14-1971 $3.75			4822/1	1922			
Character Sets for Programming Languages	97/5 N436	53								

Legend

ISO - International Standards Organization
ECMA - European Computer Manufacturers Association
ANSI - American National Standards Institute
FIPS - Federal Information Processing Standard
CSA - Canadian Standards Association
BS - British Standard
AS - Australian Standard
CCITT - Consultative Committee International, Telephone & Telegraph
JIS - Japanese Industrial Standard
GOST - USSR Standard

Table 1.

STICKS 4-7

ASCII, as a 7-bit code, is usually represented in 8 columns of 16 positions. The row positions are 0000 through 1111, the low-order 4 bits, 0 through 15 in decimal. The columns are 000 through 111, the next higher 3 bits, 0 through 7 in decimal. For some reason, the developers of ASCII found it convenient to refer to these eight columns as "sticks." So shall we. Each position will be represented in this article by its usual decimal representation. For example, capital A is position 4/1. Figure 2 is a representation of ASCII that is more convenient to those working in octal, rather than hexadecimal, notation.

HIGH ORDER OCTAL DIGITS →	00	02	04	06	10	12	14	16	← LOW ORDER OCTAL DIGIT
	NUL	DLE	SP	0	@	P	`	p	0
	SOH	DC1	!	1	A	Q	a	q	1
	STX	DC2	"	2	B	R	b	r	2
	ETX	DC3	#	3	C	S	c	s	3
	EOT	DC4	$	4	D	T	d	t	4
	ENQ	NAK	%	5	E	U	e	u	5
	ACK	SYN	&	6	F	V	f	v	6
	BEL	ETB	'	7	G	W	g	w	7

HIGH ORDER OCTAL DIGITS →	01	03	05	07	11	13	15	17	← LOW ORDER OCTAL DIGIT
	BS	CAN	(	8	H	X	h	x	0
	HT	EM	)	9	I	Y	i	y	1
	LF	SUB	*	:	J	Z	j	z	2
	VT	ESC	+	;	K	[	k	{	3
	FF	FS	,	<	L	\	l	¦	4
	CR	GS	-	=	M	]	m	}	5
	SO	RS	.	>	N	^	n	~	6
	SI	US	/	?	O	_	o	DEL	7

Figure 2.

The first positions of sticks 4 and 6 are respectively the "commercial at" and "accent grave." Then the upper and lower case Roman alphabets follow. This offset of one position is historical (from the United Kingdom), and of no importance as long as you remember that it is so.

Following the alphabet in both sticks 5 and 7 are three positions each that one must be very cautious about. In ASCII they are assigned as [, /¹, and] in stick 5 — {, |², and } in stick 7. But in the ISO Code and CCITT versions they are reserved for national usage. Table II gives the national use assignment for these positions. Surely you remember that the Scandinavian alphabet has 29 letters, not 26? My friend Orjar Heen in Oslo is very protective of these positions. He says "If you Americans want to sell computers and software abroad, don't use the ASCII characters for these positions in your software."

To be more precise, positions 5/11, 5/12, 5/13, 7/11, 7/12, and 7/13 (noted above) are called *primary* national usage positions. So is 4/0, where ASCII has the "commercial at." Honeywell, for example, uses the "at" in its timesharing systems for deleting the previous character upon entry. But this isn't too serious, because many nations also have the "at" in their primary sets.

	currency		1st 7 national				dia	dia	1st 7 national			dia
	2/3	2/4	4/0	5/11	5/12	5/13	5/14	6/0	7/11	7/12	7/13	7/14
Netherlands–A												
Australia	#											
Belgium–A												
W. Germany–A		$										
US												
Japan					¥							
UK	£											
Italy–A	#	ù										
Switzerland–A		ä										
France–A		$										
USSR		ю			v		^					
Netherlands–B												
Belgium–B			à		U				è		ë	
France–B	£	$	à		ç	§	^		è	ù	ë	¨
Switzerland–B			à		ç				é		ë	
Italy–B	#	$	§		ç	é	^	ù	à	ù	ë	ì
Switzerland–C					ç				à		ë	
Hungary	#	Ft	§	É	Ö	U	Á	ö	é	ú	ö	á
W. Germany–B	£	$	§	Ä	Ö	Ü	^	-	ä	ü	ö	ß
Switzerland–D				Ä	Ö	Ü			ä	ü	ö	
Sweden	#	¤	É	Ä	Ö	Å	Ü	é	ä	ü	ö	ü
Finland				Ä	Ö	Å			ä	ü	ö	
Denmark				Æ	ø	Å			æ	ø	ø	
Norway				Æ	Ø	Å			æ	ø	ø	
Spain					Ñ							

Table 2.

Also in sticks 4-7 are three diacritical marks. They are accent grave (′) in 6/0, circumflex (^) in 5/14, and tilde (~) in 7/14. These are called *secondary* national usage positions. In some countries the tilde is a straight overline.

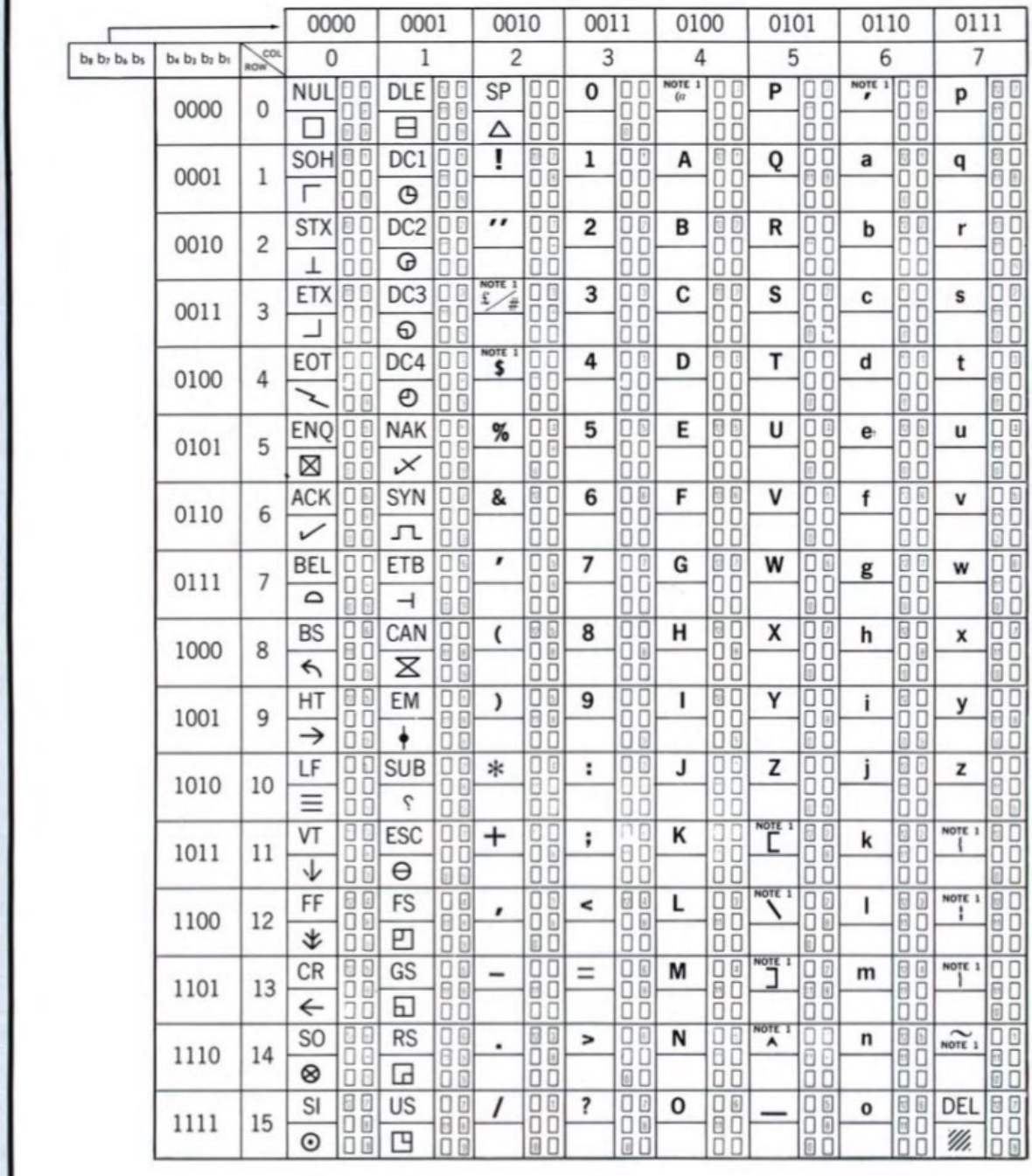
But it is the circumflex where we have a lot of confusion. Teletype first made it an "up arrow" in an earlier version of ASCII, to serve as an exponentiation symbol, primarily for BASIC. But that doesn't do very well, because the exponentiation for FORTRAN is a double asterisk! The FORTRAN version is preferable in France, certainly, because they use such words as crane, cote, cout, and so on.

A companion problem exists in position 5/15, with the underscore. The underscore is neither national nor diacritical; all countries use it just as underscore (and for typesetting it is a U.S. convention to indicate italics, but in Italy it means boldface, except when it is the last character in a line![3]). But Teletype's early version of ASCII used it as a "left arrow" — probably for an assignment symbol equivalent to := in ALGOL. The up and left arrow have been carried over from Teletype into many video terminals. Ask your terminal manufacturer to cease and desist and retrofit. It's not ASCII and will only cause trouble forever.

The last character in sticks 4-7 is the Delete, symbol DEL, in position 7/17. It was put here because the binary code is 1111111, which would be all punched holes in perforated (not always paper!) tape, and that is the only way to make sure that it cannot be misread as some other character. ASCII is a complete set; all positions are assigned to have meaning.

STICKS 2-3

These are usually called the sticks for digits and specials. Remember that they are the "digits" 0 to 9; not numbers, not numerals, not anything but digits! They are in 3/0 through 3/9 so that the low-order 4 bits are the representations for packed decimal. Originally we considered the possibility of a special 4-bit set for numerical applications (see the fifth entry in Table Ia), but it turned out that computer hardware became inexpensive enough to not deprive ourselves of the extra capabilities of the 7-bit and 8-bit sets.



b8 b7 b6 b5 →		0000	0001	0010	0011	0100	0101	0110	0111
b4 b3 b2 b1	COL / ROW	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	NOTE 1 (α)	P	NOTE 1 (ρ)	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	NOTE 1 (£ / #)	3	C	S	c	s
0100	4	EOT	DC4	NOTE 1 ($)	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	10	LF	SUB	*	:	J	Z	j	z
1011	11	VT	ESC	+	;	K	NOTE 1 ([)	k	NOTE 1 ({)
1100	12	FF	FS	,	<	L	NOTE 1 (\)	l	NOTE 1 (¦)
1101	13	CR	GS	-	=	M	NOTE 1 (])	m	NOTE 1 (})
1110	14	SO	RS	.	>	N	NOTE 1 (^)	n	NOTE 1 (~)
1111	15	SI	US	/	?	O	_	o	DEL

Note 1

These 12 positions are variable for national usage — 2 for currency, 7 primary national usage, and 3 secondary usage which are diacritical marks when preceded by BSP. The presently-known assignments are given in the table below.

Figure 1.

Position 2/0 is officially called "space." I don't and didn't like it, and would have preferred "blank." Which is why the IBM community often uses a lower case "bee" with a slash through the vertical as its symbol. From the Univac side, the space has the official symbol "delta."

Having mentioned packed decimal, where two digits go into each 8-bit group ("byte" to the American, "octet" to the French), a word of caution on the plus and minus signs — they are in stick 2, rather than stick 3 with the digits. But the low order 4 bits are distinct, and + should be used only as 1011, – only as 1101. I mention this because the nonstandard code EBCDIC permits multiple representations of + and – in packed decimal. And the ASCII representations are not even coincident with any of these, with obvious dangers!

Watch out for the "currency" positions, 2/3 and 2/4. They also have national variations. In ASCII they are customarily # and $, but there are some things to be remembered:

- # is not "number sign" for many countries, most of which use "No." or "Nr." for that purpose. And when it is "number," it must precede the digits, not follow.
- # closely resembled the "sharp sign" in music.
- # is "pound sign" only for the U.S., the only major country still not using the metric system. To the rest, it's kilograms. For now, it's best to use the abbreviation "lb." in the U.S., not the #. In any case, both must *follow* the numeral.
- To the British, a "pound" has the symbol "£", which is why that is the symbol in position 2/3 for the UK. They get very irked when # is called a "pound" sign, especially in software manuals.
- The "dollar" is peculiar to the U.S., Canada, and some others. There are also francs, marks, escudos, pesos, lire, etc., etc. Which is why the ISO code uses the universal currency symbol in position 2/4. It's a circle with outside spikes at 45, 135, 225, and 315 degrees (O), called "scarab." Table II also shows these assignments for several countries.
- ECMA has provided a separate guideline for specifying international currencies. See the "Where to Get More Information" at the end of this article.

It's a tough problem, and will get worse when we get into expanded character sets for photocomposition and such. For now, all we can do is follow the ASCII standard, which says that # is a "number sign."

Only a few more peculiarities remain for sticks 2-3. An important one is in the double quote, position 2/2, and the single quote, position 2/7. That is, you may think it is a single quote, and even use it so, but it is really an "accent acute" for vowels. It slants from top right to bottom left, to complement "accent grave" in 6/0, which slants from top left to bottom right. Some terminal makers do not realize this pairing, and will have accent grave slanting correctly, but put accent acute as a single quote in the unstylized up and down method. My Terminet is one of those that is OK.

Don't forget that to the typesetter, in contrast to typewriters, both single and double quotes have two forms — opening and closing. In fact, the typesetter gets his double quotes by using two single quotes, of either form, because the quote uses very little space in variable space typesetting. Most terminals, either video or hardcopy, use constant spacing. So double and single quotes must be distinct for that reason.

The last variation is in position 2/6, the ampersand. There are many legitimate different symbols for the ampersand. Neither ASCII nor the ISO Code prescribe any particular one. But this leads us to the next topic — how to represent the ASCII characters in handprinted form, so that they may be input to computer systems.

HANDPRINTING FOR STICKS 2-7

The classical confusion for many years was between the digit zero and the letter "oh," but there are other possibilities for confusion. American Standard X3.45 specifies the handwritten character shapes shown in Figure 3.

Figure 3.

This clears up a longstanding problem. The communications types, and the armed services, used to put a slash through the zero; somehow the IBM users got to putting the slash through the letter "oh" instead, confusing the Scandinavians greatly. Now it's neither (which helps), just a 180-degree rotation of the letter Q. The earlier German Standard DIN 66 002 prescribed the cursive loop in the upper right, as some may have learned in penmanship courses. It now permits the ANSI form as well.

UPPER AND LOWER CASE LETTERS

Many people are accustomed to using upper case only. This is a hangover from early line printers and limited sets (until the Stretch computer of IBM, characters were usually 6 bits in size). It would have been far better if they had all been lower case in those smaller sets. Putting it simply, would you buy a book to read if it were all

in upper case? Because lower case is much easier and faster to read, lower case should be the default case when one has only the one case. There is no reason why FORTRAN or BASIC processors cannot understand lower case variable names and verbs just as easily as they can understand upper case.

I always recommend getting a terminal with both cases if it is at all affordable. Second best is making sure that a single-case terminal is retrofittable later, if necessary. And if a single-case terminal, get it in lower case only, if possible. There has been much reportage in the computer trade press about eyestrain resulting from using computer terminals. Is the reason obvious?

STICKS 0, 1

These are the control characters. The most important distinction in ASCII is the split between sticks 0-1, Controls, and sticks 2-7, Graphics. We'll see this later on in the standards for Code Expansion (to 8 bits or more), and Code Extension (alternate sets, such as Cyrillic for the USSR, and Kata Kana for Japan).

Unfortunately, there is, despite the standard, much difference between the ways that various terminal devices handle these control characters. They may act differently, or they may not be operative at all. I have two very useful programs, written in the TEX language (Reference 6). One lists each symbol by name and then shows its action between parentheses. The other asks you to depress in turn all the funny keys on your terminal, and then tells you what control character(s) they generate, if any.

GRAPHICS FOR THE CONTROLS

There are standard graphical representations for the 32 controls, space, and delete. They are defined by ISO 2047, American Standard X3.32, and ECMA-17, and are shown integral to Figure 1. Some terminals are advertised as ASCII terminals, and yet generate Greek or other characters for these positions. Don't believe it! These symbols are every bit as useful as any Greek characters could be.

There are five groups in the basic control set.

STICKS 0, 1 — Logical Communication Control (10)

This group is used for both communication and for labeling of media. It includes:

SOH (0/1) (Start of Heading) — used as the first character in the heading of an information message.

STX (0/2) (Start of Text) — terminates the heading just before the text.

ETX (0/3) (End of Text) — Last character in the text message. Unfortunately, it is generated on many terminals via Control-C, and that's just to the right of Control-X on the keyboard, which is commonly used to cancel a bad input line. And if you mis-key — ouch!

EOT (0/4) (End of Transmission) — the last character in any transmission, and usually it turns your device off!

ENQ (0/5) (Enquiry) — requests a response from a remote station, either an identification of that stations (Who are you?) or its status.

ACK (0/6) (Acknowledge) — used by a receiver to reply "yes" to a sender.

DLE (1/0) (Data Link Escape) — an Escape character, especially for communications, analogous to ESC (1/11). It signals the start of a character sequence that causes a shifting into another set of communication controls, whenever they are needed.

NAK (1/5) (Negative Acknowledge) — used by a receiver to reply "no" to a sender.

SYN (1/6) (Synchronous Idle) — needed by synchronous transmission systems to get into, or stay in, synchronization when no other such signal is available to them.

ETB (1/7) (End of Transmission Block) — indicates the end of some division of data that the transmission system must make, unrelated to any division in the format of the logical data itself.

One lists each symbol by name and then shows its action between parentheses. The other asks you to depress in turn all the funny keys on your terminal . . .

STICKS 0, 1 — Physical Communication (4)

This group is used for communications. It includes:

NUL (0/1) (Null) — the standard says that it is "used" to accomplish media fill or time fill" . . . "may be inserted into or removed from a stream of data without affecting the information content of that stream." And that's exactly what the standard also says about DELete (7/15), which it lists as a control character even though it is not in the control sticks! The only difference I can see between them is that on perforated tape you can make any character into a DELete, but none into a Null.

CAN (1/8) (Cancel) — the receiver is to disregard the data received up to that point, starting from restart point that receiver and sender have agreed upon. It is common in timesharing for Cancel (often generated by a Control-X) to work on a line-at-a-time basis, to delete an unwanted string of entry characters, and effectively put one back to the position of reentering the entire line. In this case, the agreement between sender and receiver is "back to the last CR." But there are many other ways that Cancel could be used, and for parallel as well as serial transmission.

SUB (1/10) (Substitute) — a character that says probably we would have had another character in this position if we could have figured out what it was supposed to be! There are many reasons for such confusion — perhaps parity didn't check out. But it is better to put in a SUB to keep the field lengths and such correct. Moreover, note its symbol, a mirror image (not the Spanish inverted) question mark. If this is displayable, it will tell you definitively that the system doesn't know what it is, and you can make a good guess in many cases, particularly in word text.

EM (1/9) (End of Medium) — defines the previous character as the last usable character on that medium, whether or not there is more recordable space on the medium.

STICKS 0, 1 — Device Control (11)

This group is used for control of devices such as terminals.

HT (0/9) (Horizontal Tabulation) — the standard says that is "advances" the active position to the next predetermined character position on the same line." There are two ways this can work:

1. Right at the terminal, if it has the horizontal tab capability built in. Sometimes you can set the tab positions by using the terminal only; almost always the computer can be made to set the tabs on the terminal. Then when you hit HT during entry, or HT is read from the computer output, the printing or displaying (active) position will skip to the next tab setting.
2. By a formatting program in the computer, which must be given some indication of the tab setting positions in force at any particular point in the file. The program then simulates horizontal tab movement by filling the lines with spaces as needed to achieve the alignment.

VT (0/11) (Vertical Tabulation) — the standard says that it "advances the active position to the same character position on the next predetermined line." And if you agree with somebody else, it can be to the first position in that line instead. This is a very dangerous character to use. It cannot be used directly on any terminal that I know of. Even if it could, the implementation rules are not supplied unambiguously in the ASCII standard. And for use by a formatting program, one would have to predefine the number of lines to be skipped. That's pretty tough when you are inserting and deleting lines, as every programmer knows.

LF (0/10) (Line Feed) — like vertical tab, but just to the next line, which is clean enough. If receiver and sender agree (again as in vertical tab), it can be to the first position of the next line, in which case it is called New Line (NL). Some manufacturers implement this. I personally prefer having a separate Carriage Return and Line Feed. Both codes can be generated with a single keystroke, and they often are.

FF (0/12) (Form Feed) — again like vertical tab, to the same cahracter position unless sender and receiver agree that it is to the first position in the new line, except that the tab is to a new line position that is related to a form of some size (those that fold 11 inches apart, for example). This control could run wild if your terminal or other display device is not equipped to handle it, so use it with caution in files.

CR (0/13) (Carriage Return) — moves the active position to the first position on the *same* line! Not like typewriters. They have effectively incorporated the New Line feature. But the non-advancing CR is better for terminals, even if it is misnamed. Neither video terminals nor ball and daisy wheel typewriters have carriages, so live with it.

BS (0/8) (Backspace) — Backspace is a very tricky character. On some terminals, such as video terminals, there is no key to generate Backspace for entry into the text stream or buffer. On many it can be created via Control-H. Even then, it may or may not be operative.

Backspace is meant for physical movement of the active position (which may or may not coincide with a cursor position, when such exists). Historically, it was included for hardcopy terminals and other hardcopy devices for some of these uses:

- Underscoring (underlining).
- Other forms of highlighting, such as bold. For example, the sequence A BS A BS A would strike the A three times on a hardcopy device, and make it look boldface (such a sequence can also be translated to call a boldface font in photocomposition).
- Editing indications. For example, in legislative bill drafting to indicate the deleted or changed portion:
  This is obsolete.
- Forming composite characters, e.g.:
  Ӿ ± ≠ ↾ ⌶ I } F (Hungarian forint)
- Forming accented letters, primarily for European languages. Examples:
  Ä Å Ø (Scandinavian letters following Z)
  Ñ á à ô ü

Warning: Backspace is entirely different from a cursor movement on a video terminal! When the cursor is moved to a position where a character is already entered, succeeding entry in that position usually destroys the original character and replaces it with the new entry.

I personally haven't seen any video terminals with a true backspace. A former president of Infoton told me it could be done as an engineering special for about $5,000 one-time cost.

Warning: There are three ways to create underscored text for hardcopy terminals:

1. The characters, that many backspaces, and that many underscores (or vice versa).
2. A character, BS, underscore, the next character, etc. This is called the canonical form, and is used quite commonly.
3. Underscore, BS, character, underscore, etc.

I have noticed a lot of difficulty moving back and forth between hardcopy (at my home) and video (in my office) terminals. One tends to underscore on the hardcopy terminal and forget that half of the pairs are going to be wiped out by the cursor on the video terminal. In the first two methods above, it's the text that gets wiped out, and it's hard to read on the fly. So if you plan to display a file on a video terminal, find another highlighting method, or use the third underscoring convention. Even that may give problems if done by embedding an underscoring command in the file you pass to a formatting program; most such programs put the underscore last instead of first.

BEL (0/7) (Bell) — sounds an audible signal to get the user's attention. Some terminals are not so equipped, but they should be. It's good human engineering. But please give me an adjustable volume control!

And then there are the four device controls for unspecified purposes, DC1, DC2, DC3, and DC4 — in positions 1/1 through 1/4. Different manufacturers treat these like a wild card in poker — they make them anything that they want. Doesn't lead to much compatibility, so beware.

STICKS 0, 1 — Field Separators (4)

This group is used for formatting and string processing. These are the separators in positions 1/12 to 1/15. I got the idea originally from the Word Mark in the IBM 1401, which used an extra bit in the low-order character in a field as a delimiter. ASCII uses special and separate characters to indicate a hierarchical structure. Originally I put in eight such characters, but only these four remain:

FS(FileSeparator — 1/12)
GS(GroupSeparator — 1/13)
RS (Record Separator — 1/14)
US(UnitSeparator — 1/15)

FS is most inclusive, US the least inclusive. And we can consider the blank/space as the next lower order separator from these. Suppose we had a line of text like this:

(text1)US(text2)US(text3)RS(text4)US(text5)GS(text6)

On many terminals these delimiting control characters would not print, so we would see only a continuous stream. On others they might show as spaces. A TEX command to break the line at the record separator would be:

scan:line:*rs

The variable *left would contain "(text1) . . . (text3)". The variable *right would contain "(text4) . . . (text6)".

STICKS 0,1 — Changing Sets (3)

This group is used for moving to and from alternate graphic and control sets. This includes ESCape (1/11), Shift Out (0/14), and Shift In (0/15).

These basic control characters have permitted design of a quite marvelous structure for extension and expansion. It allows us to code and classify most of the world's graphic symbols for computer storage, interchange, and display. This big area will form most of Part III of this article.

IN THE NEXT INSTALLMENT

The ASCII Collating Sequence

ASCII and Programming Languages

ASCII and Media

Keyboards

ASCII and Display/Printing

Code Extension — Alternate Controls

Code Extension — Alternate Graphics

ASCII and Non-Latin Alphabets

Code Expansion — 8-bit ASCII

WHERE TO GET MORE INFORMATION

There are four sets of Information Processing Standards that may be of concern to you:

- ISO. Sold only through ANSI (American National Standards Institute), which has the franchise. That makes the prices high — much higher than in other countries.
- ANSI. These are American National Standards developed via the X3 and X4 committees, mostly. Prices still pretty high.
- ECMA (European Computer Manufacturers Association), 114 Rue du Rhone, 1204 Geneva, Switzerland). Free, and they have a lot more advanced standards than ISO and ANSI. But a modest donation would not be unwelcome.
- Your friendly U.S. Government, in the person of the Department of Commerce, National Bureau of Standards, Institute for Computer Sciences and Technology, in Gaithersburg, MD 20760. If by any chance you are employed by the U.S. Government, you get FIPS PUBS (Federal Information Processing Standards Publications) for cheap. Otherwise, see ANSI. (Refer to Tables 1a, 1b, and 1c). In many cases they are essentially reprints of the ANSI standards, for a fraction of the cost.

If you can't wait for the standards to be approved and published, catch them in progress. Ask CBEMA, the sponsor of ANSI X3, to put you on an observer list for the committee in your area of interest. The address is:

Robert Brown, Director of Standards
Computer & Business Equipment Manufacturers Association
1828 L Street NW
Washington, D.C. 20036
(202) 466-2288 Telex 89 29042

REFERENCES

1. ANS X3.4-1977, available from the American National Standards Institute, 1430 Broadway, New York, NY 10018.
2. ISO 646, avialable from ANSI (Reference 1).
3. R.W. Bemer, "ASCII — the data alphabet that will endure," in *Management of data elements in information processing*, National Bureau of Standards, 1975 October, 17-22.
4. R.W. Bemer, "A view of the history of the ISO character code," Honeywell Computer J. 6, No. 4, 1972, 274-282.
5. E.H. Clamons, "Character codes: who needs them?", Honeywell Computer J. 5, No. 3, 1971, 143-146.
6. The TEX Subsystem of the Timesharing System, Series 60 Level 66, Honeywell Information Systems, 200 Smith Street, Waltham, MA 02154, Order DF72.

ACKNOWLEDGEMENTS

Thanks go to co-workers at Honeywell Information Systems: Eric Clamons for much background, insight, and experience gained from working for a long time as chairman of X3J2 — the committee charged with the development of ASCII. And to Pat Skelly, ACM representative on ANSI X3, for collecting all the various national and international standards documentation upon which many of the figures were based.

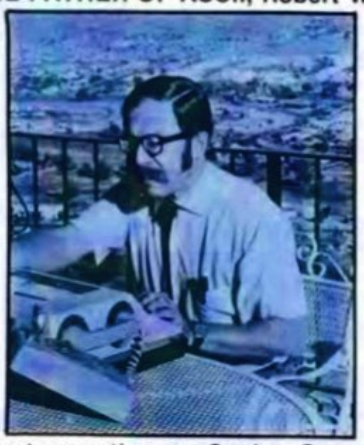
FOOTNOTES

[1]For those curious about the reverse slash, it came from ALGOL 58. The reference language specified Λ and V as the symbols for AND and OR respectively. I put the reverse slash in so these could be made as 2-character groups — and

[2]You will still see many terminals where this vertical bar is broken in the middle. This resulted from a hassle with the PL/I people, who wanted to stylize the exclamation point (2/1) as a vertical bar for OR in that language. And of course that would make the graphics the same. The compromise (at horrendous cost in people time) was to break the real vertical bar in ASCII. But it turned out that the PL/I people didn't really need it, or else it gained no momentum, so the real vertical bar is back to normal in ASCII-1977. Let's fix those terminals.

[3]The Italians also have a different solution to hyphenation and right justification. It ignores the syllable structure and simply demands that if, when you get to the last position in the line, the current word is not yet completed, that last character shall be underscored, and the word continued without fuss on the next line. I rather like it.

THE FATHER OF ASCII, Robert W. Bemer



Robert "Bob" Bemer received his A.B. in Mathematics from Albion College in 1940, and a Certificate in Aeronautical Engineering from Curtiss-Wright Tech a year later.

His vast work experience includes employment with the major leaders of the aircraft and electronics industries—most recently, as Senior Consulting Engineer with Honeywell Information Systems.

Highlights of his many accomplishments include: The discovery of polynomial telescoping (1954); creation of the PRINT 1 programming system for IBM (1956); development of FORTRANSIT; development of COMTRAN, one of the three major inputs to COBOL; development of XTRAN, predecessor to ALGOL (1958); was a major influence in the choice of the 8-bit character in IBM System 360 (1960); an influence in building the 1108 and 6000 systems; and editor of Honeywell Computer Journal.

He has an impressive list of over 71 publications to his credit.□

INSIDE

PART 2 OF

The abstract aspects of ASCII were treated in Part I. Now we come to some aspects of usage and implementation. Certainly one major use area is the ordering of files.

THE ASCII COLLATING SEQUENCE

To put items in some ordering, the entire precedence relationship for that ordering must be defined. Higher or lower, precedes or follows, or whatever. For single characters, this ordering relationship is called the "collating sequence".

The ASCII standard used to say that the collating sequence for both graphics and control characters is defined simply by their binary representations. Later it added a warning that this collating sequence "cannot be used in many specific applications that define their own sequence". What an understatement!

The 1977 version hedges and speaks all around the problem without making it clear. It's not all that difficult. Suppose you have two files, and you want to know how they differ and/or how they are the same. For this purpose, the implied collating sequence (straight binary comparison) is just fine. The two files will be in the same order, and can be matched.

Whether that straight binary ordering can be used for any other purpose is doubtful. It won't work for signed numbers.

Ordering Numerals

Take these four values: 22, 13, minus 6, and minus 31. If the sign is placed before the digits, ordering by the ASCII collating sequence yields:

+13
+22
−06
−31

This is obviously worthless. It's because ordering is decided left to right, and the minus sign has a binary value 2 higher than the plus sign. Or if the sign were to follow the numeric values we would get:

06−
13+
22+
31−

because the complete decision is made in the leading digit. Again, a worthless sequence.

The way to achieve a proper ascending sequence is to separate the values into two groups, ordering those with plus signs in ascending sequence, and those with minus signs in descending sequence. Then put the plus group following the minus group. And vice versa for a total descending sequence. Notice that this works regardless of whether the sign precedes or follows the digits.

Ordering Alphabetic Fields

Alphabetic ordering is even more complex, particularly in handling both upper and lower case. Again the implied ASCII collating sequence can go wrong. People who have not studied the collating problem for data containing both upper and lower case are inclined to jump to wrong conclusions. I did myself, for the IBM Stretch computer in 1958, assigning the ascending binary sequence as AaBbCc. Using this for a telephone directory would give us the lefthand column. The straight binary sequence of ASCII would yield the righthand column, just slightly different:

De Carlo	De Carlo
De La Rue	De La Rue
De Long	De Long
DeLair	DeLaRue
DeLancey	DeLair
DeLaRue	DeLancey
Delancey	Delancey
de Carlo	de Carlo
de la Rue	de la Rue
deLancey	deLancey

Either version will get a lot of anguished subscribers!

In the simplest case, two alphabetic items must be compared with the case ignored. Only if they are *then* equal is case called into consideration to break the tie, and it is also applied successively left-to-right!

In short, the upper and lower case versions of a letter do not both get full graphic significance. Typing either "Y" or "y" will indicate a "yes" reply, but "N" will not. Because the case distinction is minor, comparisons must first be made on major distinctions, with the minor distinctions used only as tie-breakers. Accenting of let-

ASCII

3 PARTS

By R. W. Bemer

ters must also be considered minor, if accomplished via backspace, but this leads us into rules controlled by foreign governments, and won't be considered here.

Real life is more complicated than this. The ordering and sequencing of characters and words cannot always be accomplished by simple binary comparison of codes. There are constructions such as O'Reilly, l'Informatique (as data processing is called in French), and Smith-Jones — to say nothing of the Juniors, IIIs, Esq., FBCS (which I am), and so on.

Making an ASCII comparison, with the case as a minor, gives us:

De Carlo
de Carlo
De La Rue
de la Rue
De Long
DeLair
DeLancey
Delancey
deLancey
DeLaRue

Because we at first ignored case here, De Carlo and de Carlo have identical bit patterns. Tie-breaking is done by appending the binary pattern representing case, "0" for upper, "1" for lower. Specifically, 01001111 for De Carlo, 11001111 for de Carlo.

	D	E		C	A	R	L	O	
De Carlo	44	45	20	43	41	52	4C	4F	(4F)
de Carlo	44	45	20	43	41	52	4C	4F	(CF)

But even this method will not put "DeLaRue" and "De La Rue" in the same cluster. And surely this is desirable and even mandatory. It will require some special handling for spaces. The New York Telephone Company's document on this problem runs to several pages! They'd probably give you a copy upon request. You might need to know those rules before trying one of the toughest acts in data processing — putting last name first, or vice versa.

Using Controls in Ordering

There is one more aspect of ASCII useful to the order-

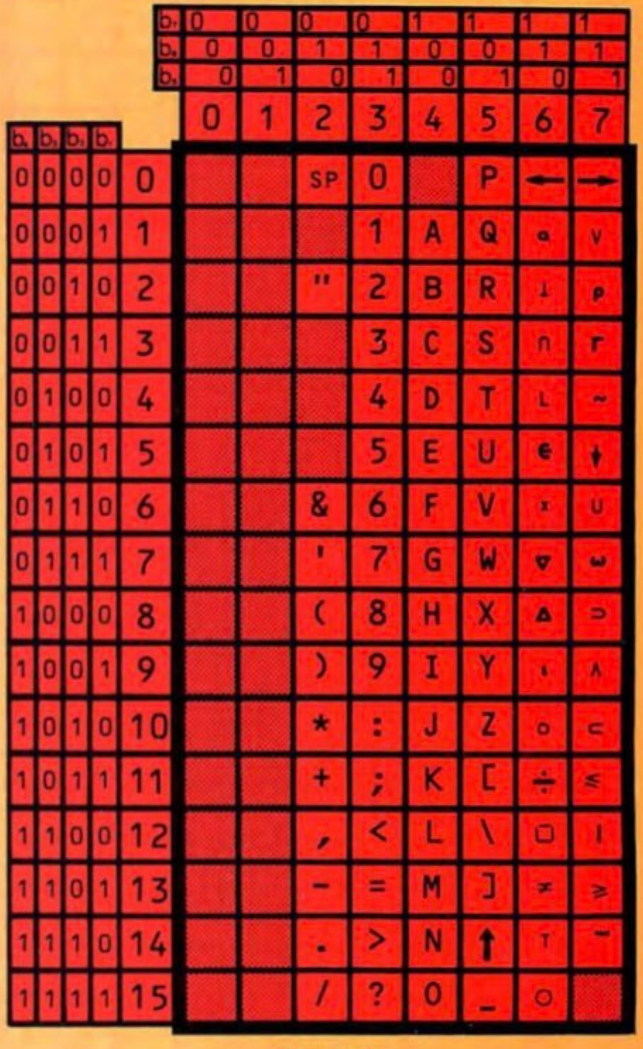
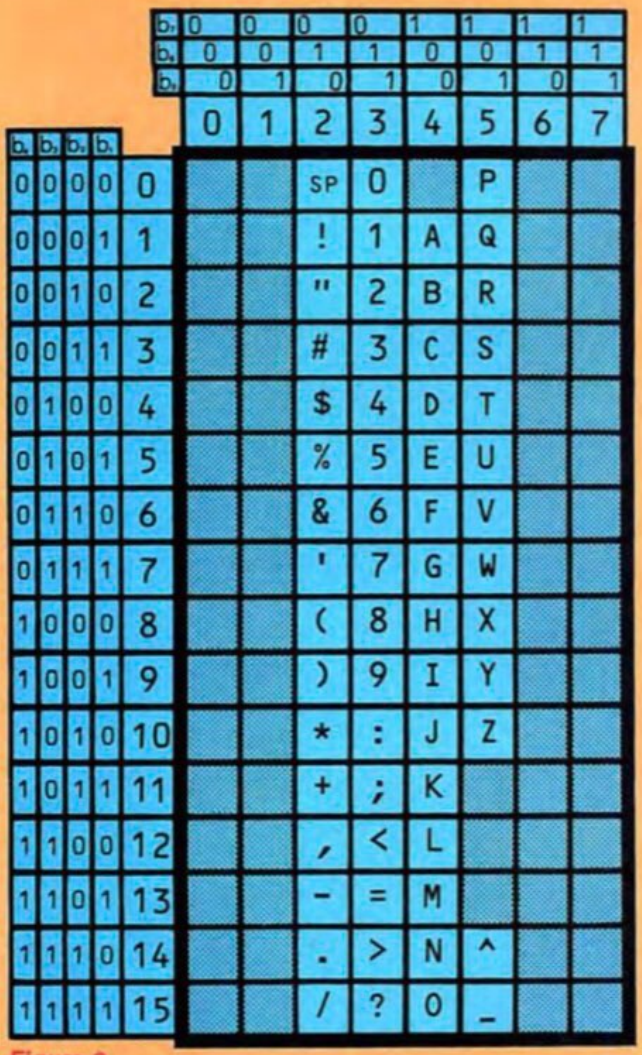
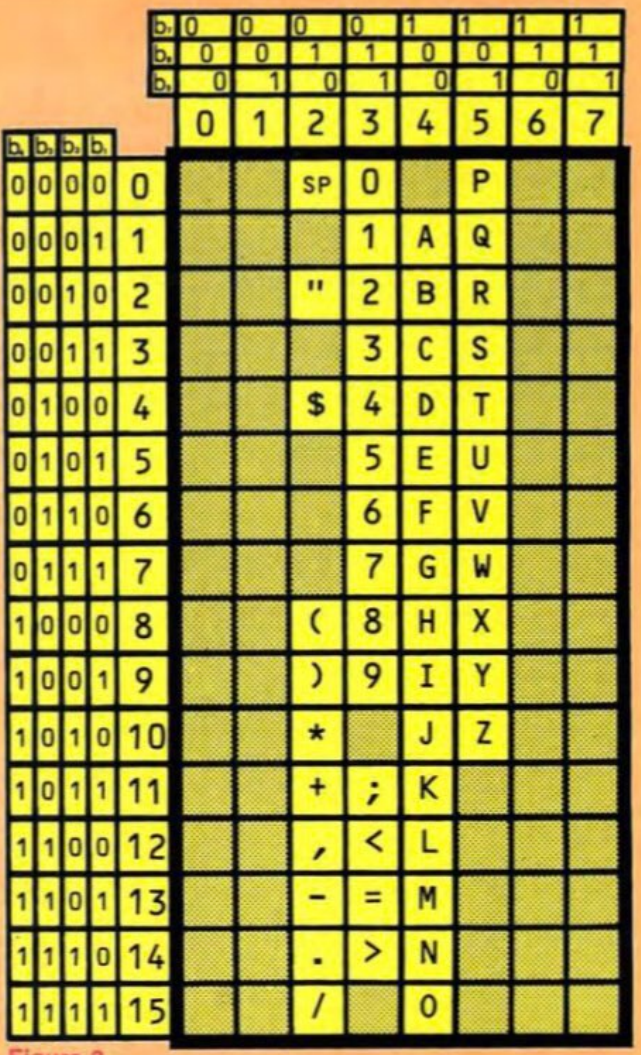


Figure 1.



b7	0	0	0	0	1	1	1	1
b6	0	0	1	1	0	0	1	1
b5	0	1	0	1	0	1	0	1
b4 b3 b2 b1	0	1	2	3	4	5	6	7
0 0 0 0 0			SP	0		P		
0 0 0 1 1			!	1	A	Q		
0 0 1 0 2			"	2	B	R		
0 0 1 1 3			#	3	C	S		
0 1 0 0 4			$	4	D	T		
0 1 0 1 5			%	5	E	U		
0 1 1 0 6			&	6	F	V		
0 1 1 1 7			'	7	G	W		
1 0 0 0 8			(	8	H	X		
1 0 0 1 9			)	9	I	Y		
1 0 1 0 10			*	:	J	Z		
1 0 1 1 11			+	;	K			
1 1 0 0 12			,	<	L			
1 1 0 1 13			-	=	M			
1 1 1 0 14			.	>	N	^		
1 1 1 1 15			/	?	O	_		

Figure 2.



b7	0	0	0	0	1	1	1	1
b6	0	0	1	1	0	0	1	1
b5	0	1	0	1	0	1	0	1
b4 b3 b2 b1	0	1	2	3	4	5	6	7
0 0 0 0 0			SP	0		P		
0 0 0 1 1				1	A	Q		
0 0 1 0 2			"	2	B	R		
0 0 1 1 3				3	C	S		
0 1 0 0 4			$	4	D	T		
0 1 0 1 5				5	E	U		
0 1 1 0 6				6	F	V		
0 1 1 1 7				7	G	W		
1 0 0 0 8			(	8	H	X		
1 0 0 1 9			)	9	I	Y		
1 0 1 0 10			*		J	Z		
1 0 1 1 11			+	;	K			
1 1 0 0 12			,	<	L			
1 1 0 1 13			-	=	M			
1 1 1 0 14			.	>	N			
1 1 1 1 15			/		O			

Figure 3.

ing problem. In the days of punch cards, before computers, one often used several card files related by a key. A sorter (with pockets for the cards to drop into) might be used to select the cards for all redheaded females between 18 and 24 years of age. But these cards would have only the employee number and such characteristics on them. To get the name, address, and telephone number one might have to go to a second (related) deck of cards. So the first deck (the subset of interest) would be placed in the first hopper of a collator, and the deck with all names and phone numbers in the second hopper. Then a card would be fed from the first hopper, followed by successive cards from the second hopper, until a match was found on employee number. Obviously both decks had to be in the same ordering for this to work, and thus the term "collating sequence".

In effect, we were sticking the cards of the first deck upright just in front of the corresponding cards of the second. To do this with ASCII requires that we have characters that collate lower than the lowest graphic, the space (2/1). We do have them. The best to use are NUL, FS, GS, RS, and US. Put one of these after each search key, then put the two files together and order them as adjoined. Now those records having a search key with one of our five control characters appended will precede the corresponding record having an ASCII graphic following the key.

Note that the four information separators (FS, GS, RS, US) are designed to collate just behind Space, in that order. This contiguity means that they can be used as a hierarchy of spaces of different class.

Other Collating Features

ASCII was designed when there was substantial investment in files already ordered on a Topsy-class IBM sequence, where the basic punctuation was low to the alphabet, but the digits were high to it. How then to accommodate this and still provide a 4-bit subset? My morning shower provided a solution (it still does!).

The 4-bit subset is formed of the first 10 graphics of stick 3 (the digit graphics) and the last 6 of stick 2. This jog was shown shaded in the early forms of ASCII, but has all but disappeared from memory now. It enables stick 3 (with the digits and new special graphics) to be ordered high to all the others via passive logic, thus overcoming opposition to the adoption of ASCII.

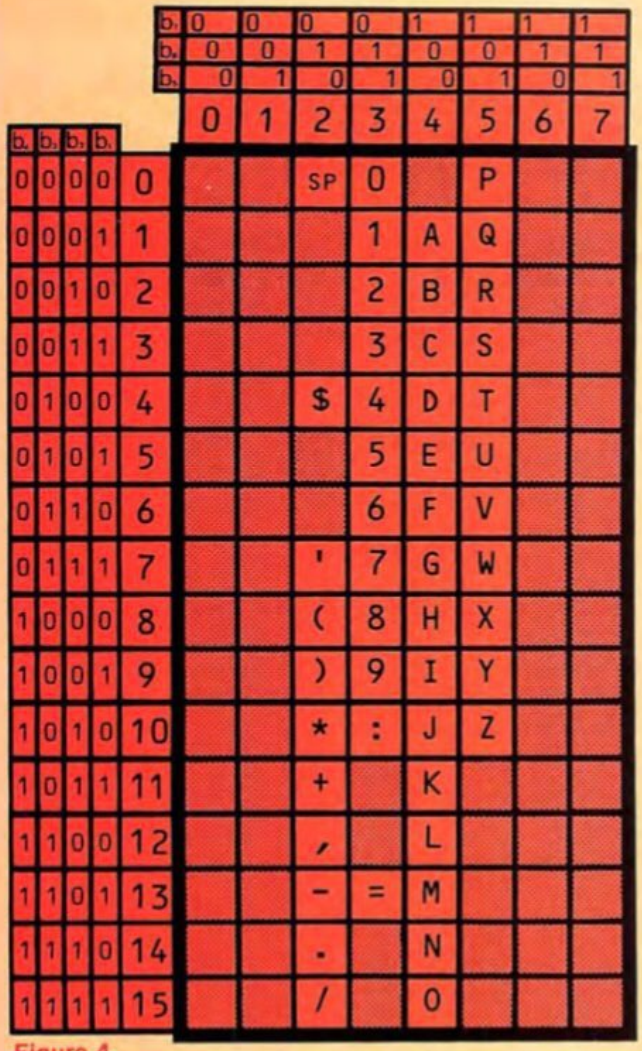
ASCII AND PROGRAMMING LANGUAGES

Standard ECMA-53 (1978 Jan), "Representation of Source Programs for Program Interchange," gives the subsets and/or modifications of ASCII as they are used for these five programming languages (Footnote 1):

	NO. OF CHARACTERS USABLE	
Language	Subset of ASCII	Other
APL	57	32
Minimal BASIC	60	0
COBOL	51	0
FORTRAN	49	0
PL/I	55	2

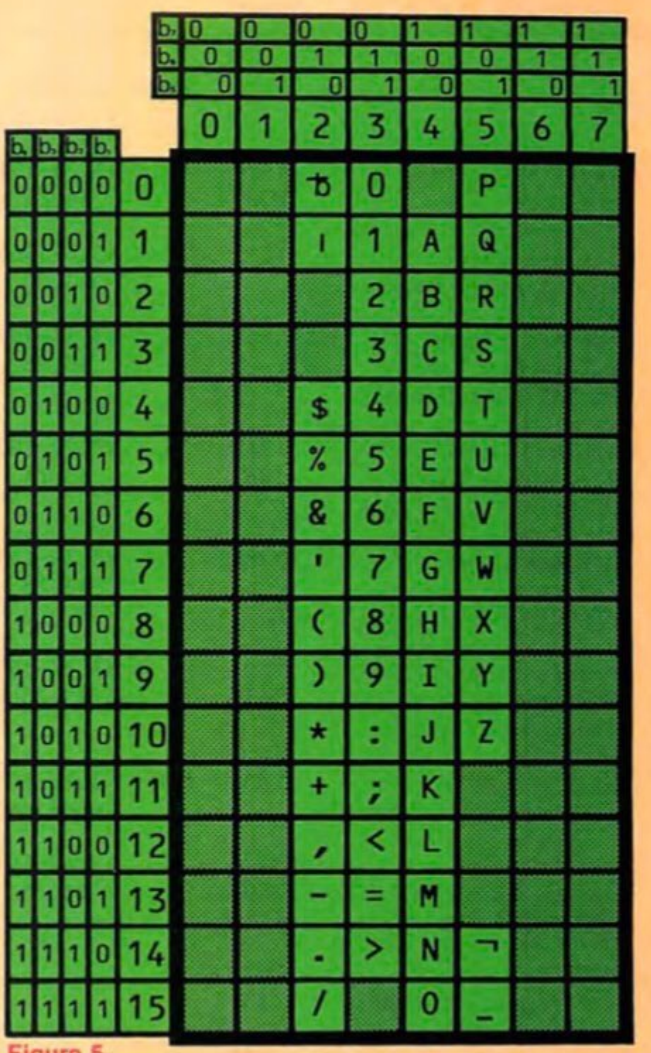
Figures 1 through 5 are the character sets for these languages as given in ECMA-53. They show the only characters permissible for use in source programs, except for:

non-numeric literals	in COBOL
comment-entries	"
comment lines	"
character constants	in FORTRAN
comments	"



b4 b3 b2 b1	0	1	2	3	4	5	6	7
0 0 0 0	0		SP	0		P		
0 0 0 1	1			1	A	Q		
0 0 1 0	2			2	B	R		
0 0 1 1	3			3	C	S		
0 1 0 0	4		$	4	D	T		
0 1 0 1	5			5	E	U		
0 1 1 0	6			6	F	V		
0 1 1 1	7		'	7	G	W		
1 0 0 0	8		(	8	H	X		
1 0 0 1	9		)	9	I	Y		
1 0 1 0	10		*	:	J	Z		
1 0 1 1	11		+		K			
1 1 0 0	12		,		L			
1 1 0 1	13		-	=	M			
1 1 1 0	14		.		N			
1 1 1 1	15		/		O			

Figure 4.



b4 b3 b2 b1	0	1	2	3	4	5	6	7
0 0 0 0	0		↑	0		P		
0 0 0 1	1		!	1	A	Q		
0 0 1 0	2			2	B	R		
0 0 1 1	3			3	C	S		
0 1 0 0	4		$	4	D	T		
0 1 0 1	5		%	5	E	U		
0 1 1 0	6		&	6	F	V		
0 1 1 1	7		'	7	G	W		
1 0 0 0	8		(	8	H	X		
1 0 0 1	9		)	9	I	Y		
1 0 1 0	10		*	:	J	Z		
1 0 1 1	11		+	;	K			
1 1 0 0	12		,	<	L			
1 1 0 1	13		-	=	M			
1 1 1 0	14		.	>	N	¬		
1 1 1 1	15		/		O	_		

Figure 5.

character-string-constants in PL/I
comments ''

For these purposes only, other ASCII characters may be used, providing there is agreement between the sender and receiver for any interchange of source programs.

The TEX language has gone farther than this general caution. There the specific characters have permanent names. For example, one could say:

linefeed = ''
'' (actual line feed inside the quotes)

if If:eqs:linefeed ....

and it would be true, because "*If" is the permanent name of Line Feed. The control characters have names that are the letters from the ASCII chart, preceded by the asterisk to show that they are read-only variables with permanent content. TEX can in fact operate upon all 256 characters of ASCII in an 8-bit byte, all 512 in a 9-bit byte.

Specific Notes on the Figures

APL -Sticks 6 and 7 (ordinarily lower case alphabet) are replaced entirely except for the DELete position.
-Space is nonprinting, although the symbol shown is SP.
-Ampersand (2/6) is not used for writing source programs, except as the last character of a line if that line is to be continued on the next line.

PL/I -In position 2/1, the exclamation point is replaced by a vertical bar for OR.
-In position 5/14, the circumflex is replaced by the symbol shown, for NOT.
-If you have to use your terminal for both PL/I and some other programming language, forget that foolishness. You can get by with the exclamation point as OR, and the circumflex as NOT. The important point in source program interchange is to have the encoded representations of the characters exchanged correctly.

(all) -Although the character BLANK (space) is shown as the flagged lower case "b" in the FORTRAN and PL/I sets, there is no printing graphic to indicate it. For all practical purposes, it is really the Space of ASCII (2/0).
-Four of these five languages (not APL) have the "$" shown in 2/4. When the International Reference Version of the code is used, this becomes the universal currency symbol, which is also acceptable.
-Minimal BASIC uses "#", which is the International Reference Version symbol. The national symbols, such as the English pound sign, are also acceptable.

ASCII AND MEDIA

ASCII and Punch Cards

Reading and punching equipment for punch cards, being very mechanical, is so expensive that microcomputer people are unlikely to use them. So you might ask why we bother here with the representation of ASCII on this medium? I can think of at least three reasons:

•A scientist at the U.S. National Bureau of Standards said once that if punch cards were on the way out, it was the only product he ever saw dying on an upward usage curve. Thus they are likely to be around for a long

	ISO	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Hollerith Punched Card Code	1679 2021	44	X3.26-1970 $4.25	14	Z243.14 .36	4636/3 /4	1063			
Track Assignment - 25.4 mm Perf. Tape	1113	10	X3.6-1965 $3.00	2	Z243.8	3880/3	1062		C6221	
Track Assignment - 12.7 mm Mag Tape 200 cpi NRZI 9-track	1862	5	X3.14-1973 $3.25			3968	1008			
Track Assignment - 12.7 mm Mag Tape 800 cpi NRZI 9-track	962 1863	12	X3.22-1973 $3.75	3-1		4503/1	1009		C6222	
Track Assignment - 12.7 mm Mag Tape 1600 cpi PE 9-track	3788	36	X3.39-1973 $3.75	25		4503/2				
Track Assignment - 12.7 mm Mag Tape 6250 cpi GCR 9-track	DP 5652		X3.54-1976 $5.25	50						
Labeling & File Structure - 12.7 mm MT	1001	13	X3.27-1977 (unpriced)		Z243.7	4732	1068			
Track Assignment - Magtape Cassette 3.81 mm, 32 bpmm	3275 3407	34	X3.48-1977 $5.75	51		5079/1				
Labeling & File Struct. - 3.81 Magtape Cassette	DIS 4341	41								
Track Assignment - 6.35 mm Cartridge Tape 64 bpmm PE	DIS 4057	46	X3.56-1977 $4.24							

Table 1a. Standards for ASCII on Physical Media.

	ISO	ECMA	ANSI	FIPS PUB	CSA	BS	AS	CCITT	JIS	GOST
Bit Sequencing in Serial Transmission			X3.15-1976 $3.00	16-1				V.4 X.4		
Char. Structure & Parity Sense - Serial-by-Bit			X3.16-1976 $3.50	17-1				V.4 X.4		
Char. Structure & Parity Sense - Parallel-by-Bit			X3.25-1976 $3.50	18-1				V.4 X.4		
Procedures for Using Commun. Control Chars.	1745	16	X3.28-1976 $10.50		Z243.13	4505/1	1484/1			
Message Heading Formats	1745		X3.57-1977 $5.25							
Advanced Data Commun. Control Procedures			BSR X3.66							

Table 1b. Standards for ASCII in Communications.

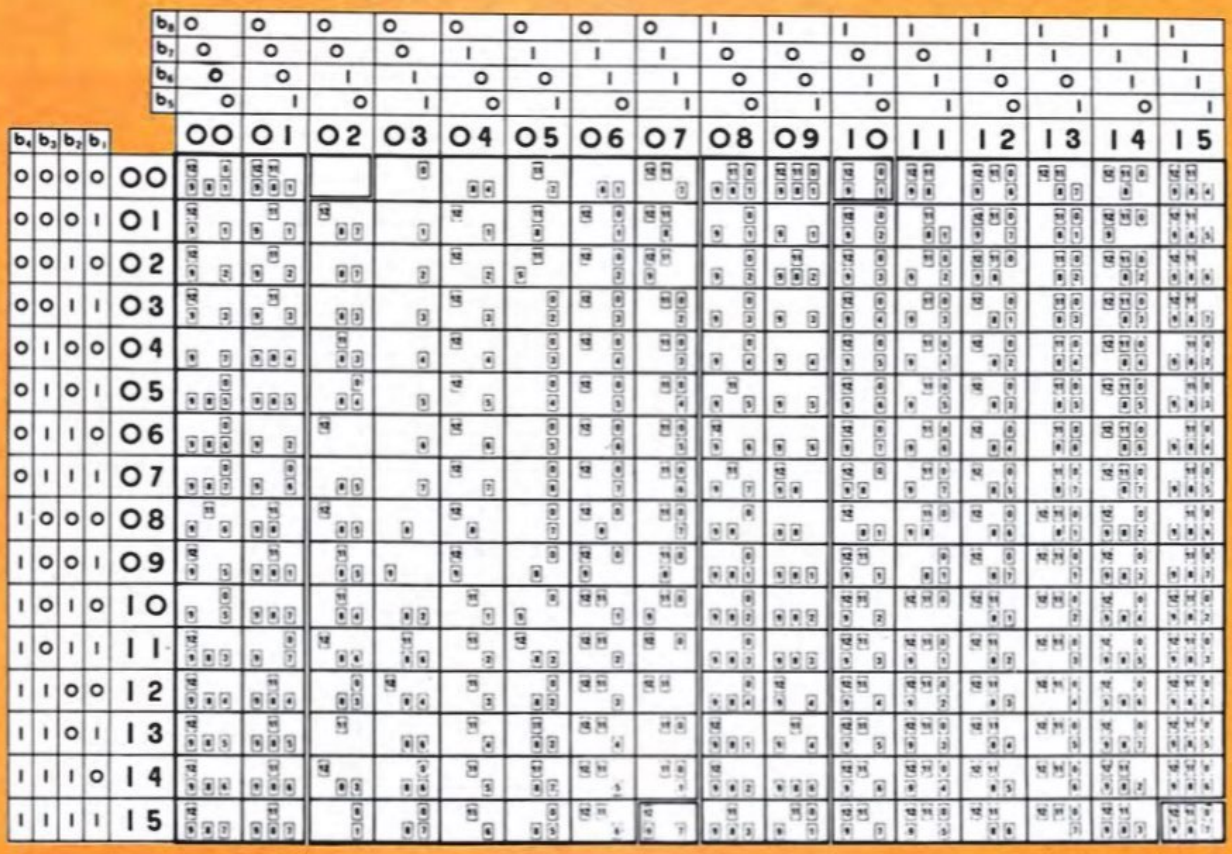


Figure 6.

	02	11	0		02 0	02 11	11 0	02 11 0
	02/06	02/13	03/00	02/00	07/11	07/12	07/13	11/10
1	04/01	04/10	02/15	03/01	06/01	06/10	07/14	13/09
2	04/02	04/11	05/03	03/02	06/02	06/11	07/03	13/10
3	04/03	04/12	05/04	03/03	06/03	06/12	07/04	13/11
4	04/04	04/13	05/05	03/04	06/04	06/13	07/05	13/12
5	04/05	04/14	05/06	03/05	06/05	06/14	07/06	13/13
6	04/06	04/15	05/07	03/06	06/06	06/15	07/07	13/14
7	04/07	05/00	05/08	03/07	06/07	07/00	07/08	13/15
8	04/08	05/01	05/09	03/08	06/08	07/01	07/09	14/00
9	04/09	05/02	05/10	03/09	06/09	07/02	07/10	14/01
02	05/11	05/13	05/12	03/10	12/04	12/11	13/02	14/02
03	02/14	02/04	02/12	02/03	12/05	12/12	13/03	14/03
04	03/12	02/10	02/05	04/00	12/06	12/13	13/04	14/04
05	02/08	02/09	05/15	02/07	12/07	12/14	13/05	14/05
06	02/11	03/11	03/14	03/13	12/08	12/15	13/06	14/06
07	02/01	05/14	03/15	02/02	12/09	13/00	13/07	14/07

02	11	0		02 0	02 11	11 0	02 11 0	
10/08	11/01	11/09	06/00	12/03	12/10	13/01	13/08	0 1
00/01	01/01	08/01	09/01	10/00	10/09	09/15	11/11	9 1
00/02	01/02	08/02	01/06	10/01	10/10	11/02	11/12	9 2
00/03	01/03	08/03	09/03	10/02	10/11	11/03	11/13	9 3
09/12	09/13	08/04	09/04	10/03	10/12	11/04	11/14	9 4
00/09	08/05	00/10	09/05	10/04	10/13	11/05	11/15	9 5
08/06	00/08	01/07	09/06	10/05	10/14	11/06	12/00	9 6
07/15	08/07	01/11	00/04	10/06	10/15	11/07	12/01	9 7
09/07	01/08	08/08	09/08	10/07	11/00	11/08	12/02	9 8
08/13	01/09	08/09	09/09	00/00	01/00	08/00	09/00	9 8 1
08/14	09/02	08/10	09/10	14/08	14/14	15/04	15/10	9 8 2
00/11	08/15	08/11	09/11	14/09	14/15	15/05	15/11	9 8 3
00/12	01/12	08/12	01/04	14/10	15/00	15/06	15/12	9 8 4
00/13	01/13	00/05	01/05	14/11	15/01	15/07	15/13	9 8 5
00/14	01/14	00/06	09/14	14/12	15/02	15/08	15/14	9 8 6
00/15	01/15	00/07	01/10	14/13	15/03	15/09	15/15	9 8 7

Figure 7.

time, and you may need to transfer some of those files to other media that you do use.

- There is some likelihood that microcomputers could be used in the reading and punching equipment itself, to make it less expensive.
- ASCII users are going to be confronted for a while yet with one of the several versions of IBM's EBCDIC, and the punch card assignments provide the only legitimate link for conversion of EBCDIC files to ASCII.

So Figure 6 defines the hole patterns for the binary encodings. And Figure 7 defines the encodings for the hole patterns. Don't worry about the inconsistency in the relationships. Nothing can be done about it now, because it started with Herman Hollerith's first U.S. Census machines in 1890. At first only digits and + and − signs were used. Then the code was expanded to the upper case alphabet. And other special characters for commercial use. When FORTRAN came along in 1964, it turned out that the limited capability of the subset of a 6-bit set would not permit the graphics needed for scientific work. For a long while there were dual graphic representations for several of the punch card code combinations, and this carried over into printer chains, and so on.

The only logic that the patterns follow is that they do or do not have a punch from among these six possibilities:

12-punch (top row)
11-punch (next to the top row)
0-punch
8-punch
9-punch (bottom row)
a punch from among the digits 1 through 7

Including the no-punch-at-all combination (NUL), this gives 256 combinations, just right for the 8-bit code. Although ASCII was technically only a 7-bit code at the time this rule was formulated, it was felt necessary to plan ahead a little.

ASCII and Magnetic Tape

Figure 8 gives a compact representation of several relationships, among which is the assignment of ASCII bit pattern to 9-track magnetic tape. The jumbled assignment may remind you of the "firing order" for the cylinders of an automobile engine. In fact, we used to call it just that. It was intentional for increased reliability. As in so many cases, better technology has removed the need for peculiar design, but the assignments are unchangeable because of data file investment.

There is no parallelism in recording and reading on cassettes and cartridges. The ASCII bits are recorded serially in the track. Thus Figure 8 does not consider these media.

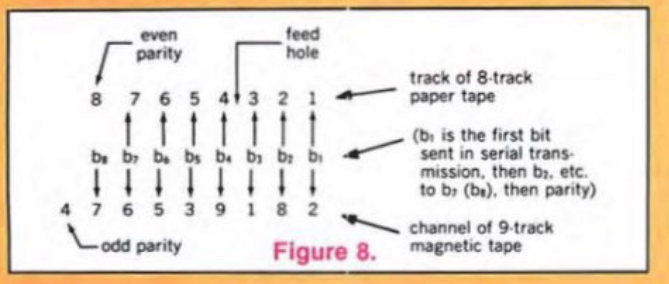


Figure 8.

ASCII and Communications

Not only is the topic of ASCII and communications a very complex and large dissertation for this article — it is also undergoing substantial rethinking, enlargement, and invention. You will have to follow on your own the workings of the CCITT, the various networking systems of the several large and many small manufacturers of computer systems, and the offerings of the common carriers — either on the local distribution system (via ATT) or direct distribution (via Satellite Business Systems).

Many of the existing standards are listed in Table 1b. Many more are under development. Arguments are raging internationally on the merits of packet switching, byte protocols, value-added systems, open-working systems, tariffs, data movement across national borders, the X.25 protocol, etc., etc. ATT is offering a new service because they suddenly discovered data-undervoice (DUV). All I can tell you now is that it is all based upon ASCII, and the proposed protocols are all dependent upon the ASCII control characters in stick 0 and 1. It will take years for this to shake out, and for now all one can do is get on the CBEMA mailing list (see reference, Part I, INTERFACE AGE, May, 1978).

ASCII AND THE METRIC SYSTEM

The full ASCII graphic set (both cases) is sufficient to indicate all symbols and prefixes of the SI (International System of Units, the new metric system), with three exceptions. They are the Greek letters "omega" for "ohm", and "mu" for "micro", and the degree symbol for Celsius temperature. These three characters will be provided in 8-bit ASCII (see Part III, next month). Meanwhile, for these, and also for such equipment that has only a single case, there is a standard way of representing the SI units and prefixes. This is given in International Standard 2955, "Representations of SI Units and Other Units for Use in Systems with Limited Character Sets", and also in American Standard X3.50-1976.

To keep the record straight, let's first look at the characters used for the prefixes. They're shown in Table 2, which indicates multiples from 10 to the − 18 up to 10 to the + 18:

10 + i	i	10 − i
exa (E)	18	atto (a)
peta (P)	15	femto (f)
tera (T)	12	pico (p)
giga (G)	9	nano (n)
mega (M)	6	micro (μ)
kilo (k)	3	milli (m)
hecto (h)	2	centi (c)
deka (da)	1	deci (d)

Table 2. Metric Prefixes

Above 3 there are no powers except multiples of 3. This practice breeds better comprehension, like marking off three's in writing numbers of many digits. Also, as a memory convenience, all symbols are upper case for powers greater than + 3. And there are no conflicts with the symbols for the units of measurement.

Now, again for the record, here are the ASCII character(s) used as symbols for the units:

A	ampere	cd	candela
Bq	becquerel	d	day
C	coulomb	g	gram
°C	degree celsius	h	hour
F	farad	l	litre
Gy	gray	lm	lumen
H	henry	lx	lux
J	joule	μ	micro
K	kelvin	m	metre
N	newton	min	minute (time)
Ω	ohm	mol	mole
Pa	pascal	rad	radian
S	siemens	s	second (time)
T	tesla	sr	steradian
V	volt	t	tonne/metric ton/ megagram
W	watt		
Wb	weber		

Table 3. Metric Units

Table 3 shows the rules clearly. Units not named after people are all lower case, as shown in the righthand column (although I do know a Mr. Day). In the lefthand col-

umn are the units that are named after people. The names of the units are not capitalized at all, but the symbols begin with an upper case letter.

I said previously that there were no conflicts between unit and prefix symbols. But you've probably noticed "d" for both "day" and "deci", "h" for both "hour" and "hecto", "m" for both "metre" and "milli", and "T" for both "tesla" and "tera". OK. But there isn't any confusion in actual usage, because the prefix precedes the unit:

dd	is a deciday (2.4 hours)
hh	is a hectohour (100 hours)
hH	is a hectohenry (but don't ever use the term)
mm	is a millimetre
Mm	is a megametre (1/300 the speed of light)
TT	is a teratesla (Wow!)

I am not suggesting that the prefixes should be applied to other than the primary metric units (the second is the primary time unit; hour and day are not), even though the timesharing system I customarily use figures my time in millihours. But when you get accustomed, the prefixes are very valuable in other ways. For example, an American billion is a kilomillion, whereas the British billion is a megamillion! And my metric teaching program understands such things as kilofathoms.

The "space" character is also vital to correct SI usage. It must occur between values and units, like 123.6 mm, and 22 °C.

And don't forget another peculiarity of ASCII as an international alphabet: (1/14) is absolutely not defined as a "decimal point" (nor is it defined as "period", which in the United Kingdom is "full stop"). For most of the rest of the world, the comma (1/12) is the decimal marker, and the period is used to mark off threes. That's why the recommended practice for marking off threes is to use the space, not either comma or period. E.g., "1 234 567 mm".

To save you the bother of looking up the standards for use with limited character sets, here is the algorithm:

1. If you have ASCII with both cases of alphabet, the three missing symbols are handled as:

ohm	for Ω
Cel (initial cap)	for °C
u (lower case)	for μ (micro)

2. If you have only one case of alphabet (either upper or lower), use it, and these three replacements remain as:

OHM		ohm
CEL	or	cel
U		u

And in addition:

S (siemens)		SIE		sie
h (hour)	become	HR	or	hr
t (tonne)		TNE		tne

Examples:

16 UOHM is 16 μΩ

373.15 K = 100 Cel

Notice that no plurals are used in symbol combinations — MICROOHMS, but UOHM.

ASCII AND KEYBOARDS

Technically, a keyboard is an ASCII keyboard if it generates the proper codes for the full set of ASCII graphic and control characters. Moreover, none of the graphic characters should have any control properties.

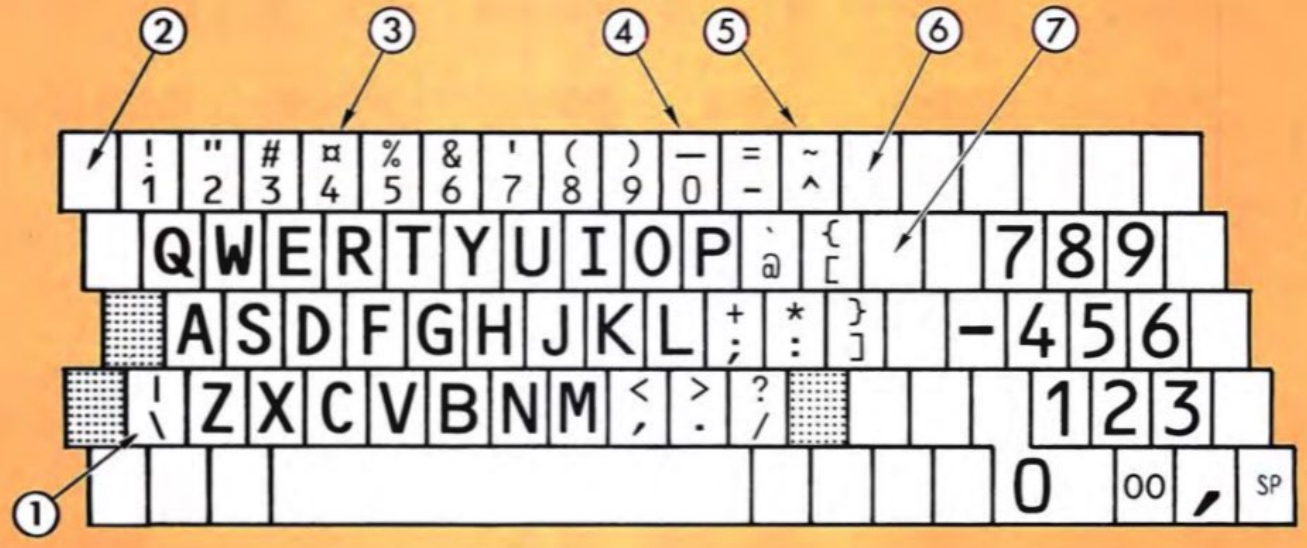
There are many types of special keyboards — Dvorak, a two-sided one used like an accordion with the hands in a vertical plane, Touch-Tone and its derivatives, etc. There are no formal standards to relate these keyboards to ASCII. For typewriter-style keyboards, however, there are two versions given in the American National Standard. One is derived from the usual electric typewriter keyboard, the other is called the "bit-paired" keyboard. Only the bit-paired keyboard will be shown and discussed here, because the other form is the subject of proposals for extensive change due to the growth of Word Processing. ANSI Committee X4A12 is studying this now.

The bit-paired keyboard was designed for minimum circuitry cost. Thus the "at" symbol (4/0) is paired with the accent grave (6/0), "A" (4/1) with "a" (6/1), and "+" (2/11) with ";" (3/11). Thus the shift key affects each other key by only a 1-bit change.

This keyboard is shown in Figure 9. It is the interchange keyboard of ECMA-33. The numbered arrows key to the notes on changes that would make this ECMA keyboard into the ANSI keyboard for ASCII. It is also equivalent to the keyboard of ISO Standard 2530-1975 (Footnote 2).

Notes for Figure 9

1. *For and ANSI keyboard, this key is put to the right of the circumflex key, on the top row (see Note 6). The Shift Key is put in its place.*
2. *If this key exists and is available, the ECMA and ISO standards put the underscore here, removing it from the "zero" key.*
3. *The ANSI keyboard of course puts a "$" here in place of the international currency symbol.*
4. *This is where the underscore is removed for the 48-key keyboard (see Note 2).*



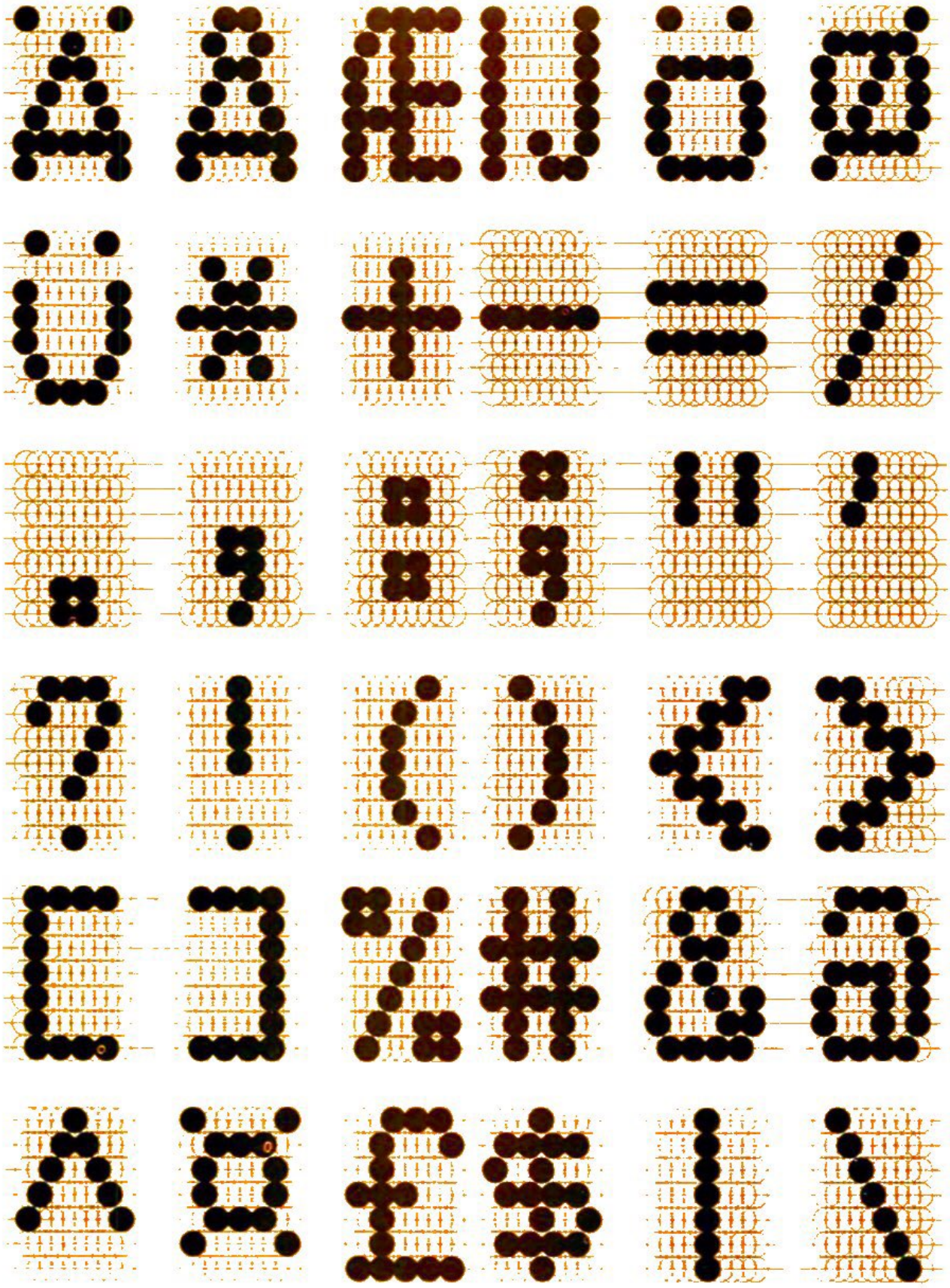


Figure 10.

5. *Here ECMA and ISO show the "overline" instead of the "tilde." It's a question of styling.*
6. *The ANSI keyboard has the reverse slash and vertical bar here, rather than between the shift key and "Z" (see Note 1).*
7. *The ANSI keyboard specifies the underscore here, in both shifts, rather than the positions shown as options in Notes 2 and 4. Practically no keyboards follow this. In fact, as I am entering this text, this is the* ***only*** *key where my Infoton Vistar deviates from the ANSI standard. It has Line Feed there, with Return to its right — a very sensible arrangement.*

Customarily, the Control Key is also tied to bit-pairing in such keyboards. The standards recommend that characters created in combination with the Control Key should use the graphic key in sticks 4 or 6 units higher. Thus "X" (5/8) or "x" (7/8) in combination with the Control Key produce CAN (1/8). Unfortunately this also means that Control-C generates ETX (0/3). And whereas Control-X as CAN is used frequently, to erase an input line of text, ETX is not often wanted. Yet it is a common miskeying to hit C rather than X. In many timesharing systems you will get a disconnect rather than a line delete.

Control and Function Keys

The so-called "QWERTY" arrangement is prevalent throughout the world. Even the French "AZERTY" set is being considered for change. But on top of these basics there are hundreds of keyboard varieties. Some of them have "dead keys" (i.e., the platen or printing element is not advanced when they are hit). This avoids having to use BS for accented letters, but it also creates difficulties in code generation.

There are some general good practices that ASCII keyboards should follow. To facilitate usage by those experienced with typewriters, all controls not used with typewriters should be located outside the customary touch-typing area. As a specific example, the Break/Interrupt key should be located where it is a definite effort to reach it (not mixed in with the keyboard). ISO 3244 may be consulted for these considerations.

Function Keys are those that generate sequences of more than one ASCII character. Examples are cursor keys, Erase-to-EOL, etc. They should be located in special clusters. Most importantly, they must all generate ASCII codes for transmission when in character-at-a-time mode. I know of video terminals where the cursors do not generate codes, as they should not while in full page buffered mode; but they still operate in line mode without generating codes. In this case the screen is alterable, but there is no way of detecting it in the computer.

Many keyboards will have some function keys that are unlabeled, for do-it-yourself assignment. These should also be clustered separately, and generate code sequences when in line mode.

ASCII AND DISPLAY PRINTING

When ASCII characters are displayed, it may be on a video screen, paper, or COM (microfiche).

On the video screen there are a number of methods to form the characters, mostly at the manufacturer's preference. They are usually at pica (constant-width) spacing for economy, so an approximation of graphic quality (such as typesetting) is not obtainable. When lower case is available, the risers and tails extend above and below the line for some screens. In others, they fall within the boundary lines of the upper case characters. They may be shown in inverse video (light background block), or highlighted by different brightness or blinking. Controls for this work will be taken up in Part III of this article.

For paper copy one usually finds either direct impact of a formed letter, or stylus printing. Either method is suitable to proportional spacing if desired. Recently there has been a general trend toward using the 7x9 dot matrix shapes of ECMA Standard 42 for stylus printers. This set of graphics is shown in Figure 10.

For hard print elements, of course, one can get a nearly infinite variety of styles and fonts. There are only two, however, specifically associated with computers — OCR-A and OCR-B. "OCR" stands for "Optical Character Recognition", meaning that the shapes are so styled that a computer-controlled scanner can read the characters as printed on paper, and encode them directly from their shapes.

OCR-A is not suitable for human reading. It's the funny looking one with the diamond-shaped letter "Oh." I won't dignify it by showing the font here. It was thought formerly, with technology of that day, that making humans work harder to read letters would make it easier and thus cheaper for computers to read them. This argument turned out to be specious, and with today's technology there is no need to use anything other than OCR-B.

OCR-B is specified in ISO 1073/2, ECMA-30, and ANSI X3.49. It is the font shown in Figure 11. I have it on my IBM golfball typewriter at home, and on my daisywheel element at the office. So it should be available for most hard elements, including the carousel type.

The first six rows correspond to ASCII sticks 2-7. In the first row, the pound and universal currency symbol are for replacement as needed. In the fourth row, the underline is discontinuous; a continuous form is shown in the auxiliary set. This set also contains a matching accent acute instead of single quote, the real circumflex (not an up arrowhead), a cedilla, and an "m" of better proportion. □

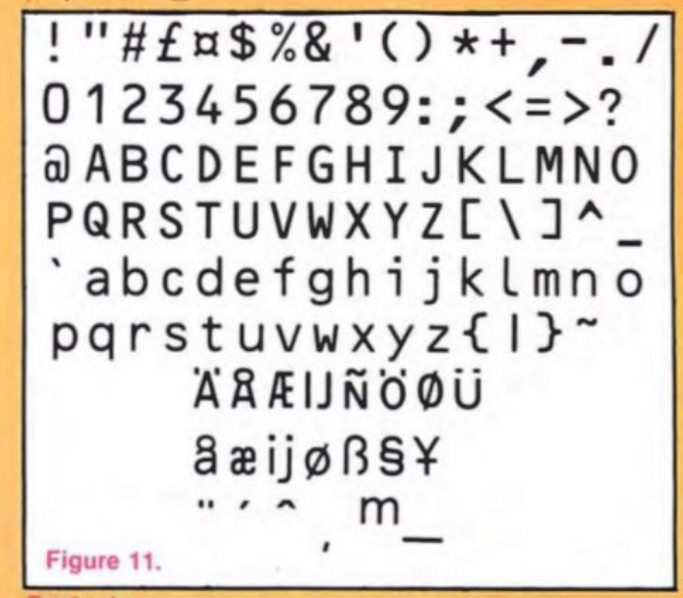


Figure 11.

Footnotes

1. With the distribution, ECMA said "ECMA-53 is an attempt to improve portability of programs. It links the language character sets defined by the language standards, their coded representatives by means of the 7-bit code and the implementations on data carriers (punched tape, punched cards, magnetic tape and magnetic tape cassettes and cartridges). It is a standard of a new type in which already standardized features are assembled in a new standardized combination aimed at supporting interchange and decreasing implementation dependency."
2. ISO 2530 is for the alphanumeric area of the keyboard only. It is augmented by ISO 3243-1975 — Keyboards for Countries whose Languages have Alphabetic Extenders, Guidelines for Harmonizations, and also by ISO 3244-1974 — Principles Governing the Positioning of Control Keys on Keyboards.
   The fact that these are "guidelines" and "principles" indicate the complexity of the subject. Typewriter manufacturers now supply over a hundred different keyboard arrangements, as their catalog will indicate.

INSIDE ASCII

CODE EXTENSION — GENERAL PRINCIPLES

Over ten years ago it was recognized that ASCII was the basis for codification of the various symbols used throughout the world. Through it, libraries could store encoded books as well as printed books. And while electronic mail may be quite simple with ASCII and its Roman alphabet, that's not the alphabet of all countries. The USSR uses Cyrillic, the Japanese use Katakana, and the Arab world uses its own semi-script alphabet. Moreover, to send a mathematics textbook by electronic mail one would have to be able to encode the formulas and special symbols peculiar to mathematics, which includes many Greek characters!

This is where the ESCape character and ESCape sequences come in. You can get the whole complicated story from ISO Standard 2022 (or ECMA-35) on Code Extension Procedures. But it will be easier to think of reproducing many ASCII Code Tables on the pages of a book, then replacing the ASCII symbols on all but the first page with the other alphabets we need.

Then we make sure that everyone in the world has the same (code) book. (The resemblance to military code books is intentional.) That's done by registering the page number assignment to characters (actually either a control set or a graphic set, but not both) with the French Standards Body AFNOR. That's the Association Francaise de Normalisation, Tour Europe Cedex 7, 92080 Paris La Defense, FRANCE. But you'll find it perhaps easier to get it from ANSI (see data in the first installment, INTERFACE AGE, May 1978).

The registration procedure is spelled out in ISO Standard 2375. It is carefully controlled to prevent frivolity and cluttering up the assignment books, for that all costs money. But the important control and graphic sets of the world may be registered and assigned their own unique ESCape sequence for calling or invoking them.

CODE EXTENSION — BASIC RULES

The control ESC, when encountered in a datastream, means that all characters following it, up to and including the first character from sticks 3 to 7, have special interpretation. The delimiting character is called a "final" (F). Those between ESC and the final are called "intermediates" (I). All of the codes in stick 2 can serve as intermediate characters in ESCape sequences of 3 or more characters in length. The entire group of characters from ESC through the final is called an ESCape sequence.

ESCape sequences obviously require buffers for interpretation, for we cannot know, when they begin, how long they will be. Sequences of length 2 are for single controls. If the character following ESC is from stick 3, the sequences are for private usage, of the class Fp. If it is from sticks 4 or 5, they mean single controls, of the class Fe, from an appropriate set of 32. If from sticks 6 or 7 (except 7/15), they are of the class Fs, composed of single controls. This is elementary extension.

A more complex type of extension is the simulation of one or more 8-bit character sets by alternating between two 7-bit sets. The home base set consists of the C0 (32 controls) set and the G0 (94 graphics plus space and DEL). The alternate sets consist of the C1 (32 controls) set and and the G1 (94 graphics plus space and DEL). The 8-bit set (it doesn't have to be just theoretical if you have a full 8-bit capability) consists of the four parts C0-G0-C1-G1.

These four types of sets are all invoked (designated) by 3-character ESCape sequences in this manner, where F is the final (3rd) character:

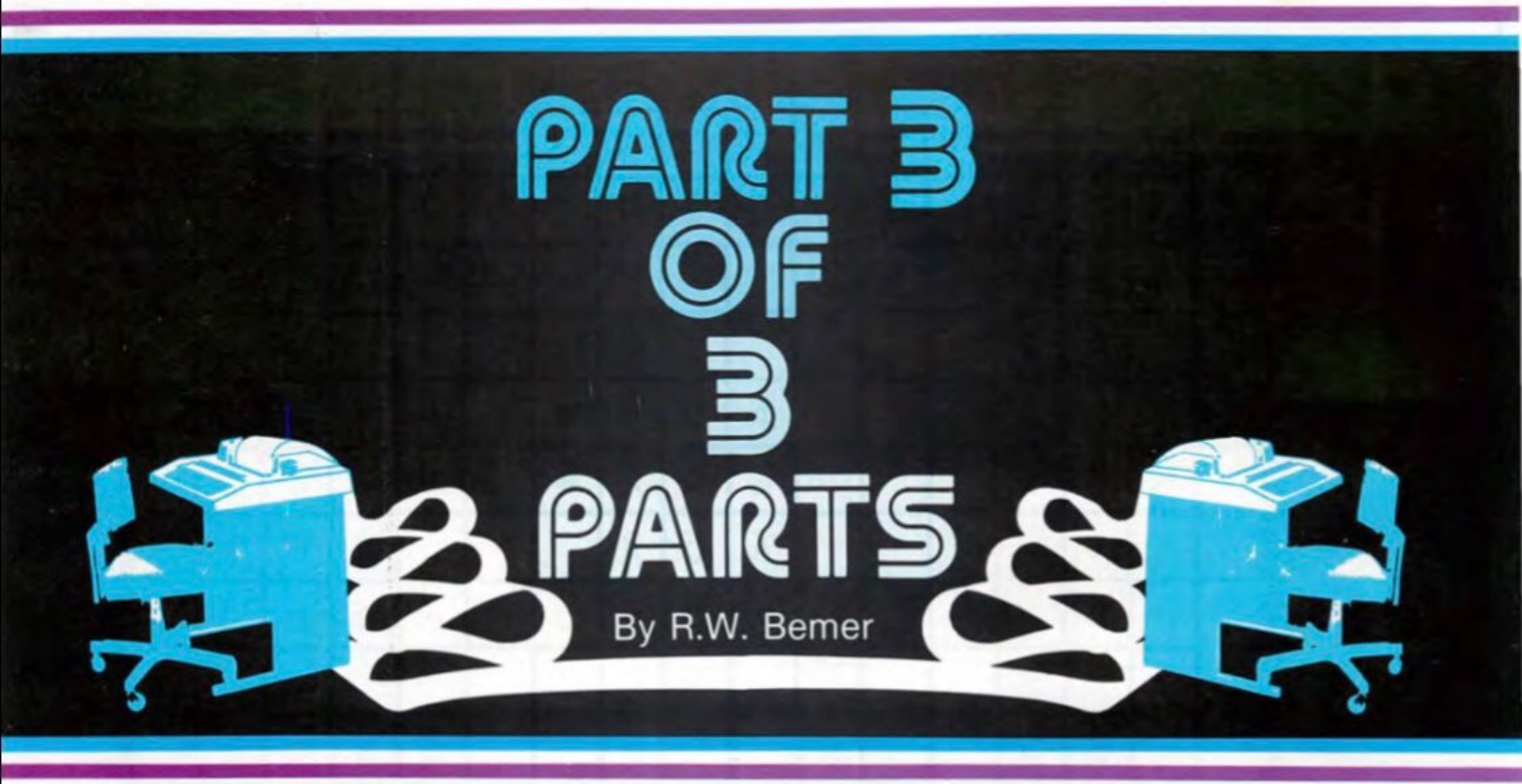
Sequence	Invokes Set Type
ESC 2/1 F	C0
ESC 2/2 F	C1
ESC 2/8 (or 2/12) F	G0
ESC 2/9 (or 2/15) F	G1

The final character "F" selects the particular set to invoke. Once invoked, encountering or entering an SO shifts to the G1 set in force; an SI shifts to the G0 set in force. SO and SI do not affect the control set.

ISO Standard 2022 defines these matters in far more detail, but that is enough for here. That document is complicated and ingenious, and deserves substantial study.

THE CODE EXTENSION REGISTRY

Table 1 identifies the graphic sets registered to date. Table 2 identifies the control sets registered to date. Re-



member that these assignments, once registered, may never be changed!

Regis. No.	Final Char.	Name
002	4/0	IRV (Intl. Reference Version) Graphics
004	4/1	UK Graphics
006	4/2	US Graphics (ASCII)
008-1	4/3	NATS Main Graphic Set (Finland, Sweden)
008-2	4/4	NATS Additional Set (Finland, Sweden)
009-1	4/5	NATS Main Graphic Set (Denmark, Norway)
009-2	4/6	NATS Additional Set (Denmark, Norway)
010	4/7	Swedish Basic Graphics
011	4/8	Swedish Graphics for Names
013	4/9	JIS Katakana Graphics
014	4/10	JIS Roman Graphics
015	5/9	Italian Graphics
017	5/10	Spanish Graphics
018	5/11	Greek Graphics
019	5/12	Latin-Greek Graphics
021	4/11	German Graphics
025	5/2	French Graphics
027	5/5	Latin-Greek Mixed Graphics (Greek Capitals only)
031	5/8	Greek Alphabet Set for Bibliographic Use

For a G0 set the ESCape sequence is
ESC 2/8 plus the final shown.
For a G1 set the ESCape sequence is
ESC 2/9 plus the final shown.

Table 1. Registered Graphic Character Sets

Regis. No.	Final Char.	Name
001	4/0	ISO 646 Controls
007	4/1	Scandinavian Newspaper Controls
026	4/3	IPTC Controls

The ESCape sequence for a C0 set is
ESC 2/1 plus the final shown.

Table Vb. Registered Control Character Sets

Table 2. Registered Control Character Sets

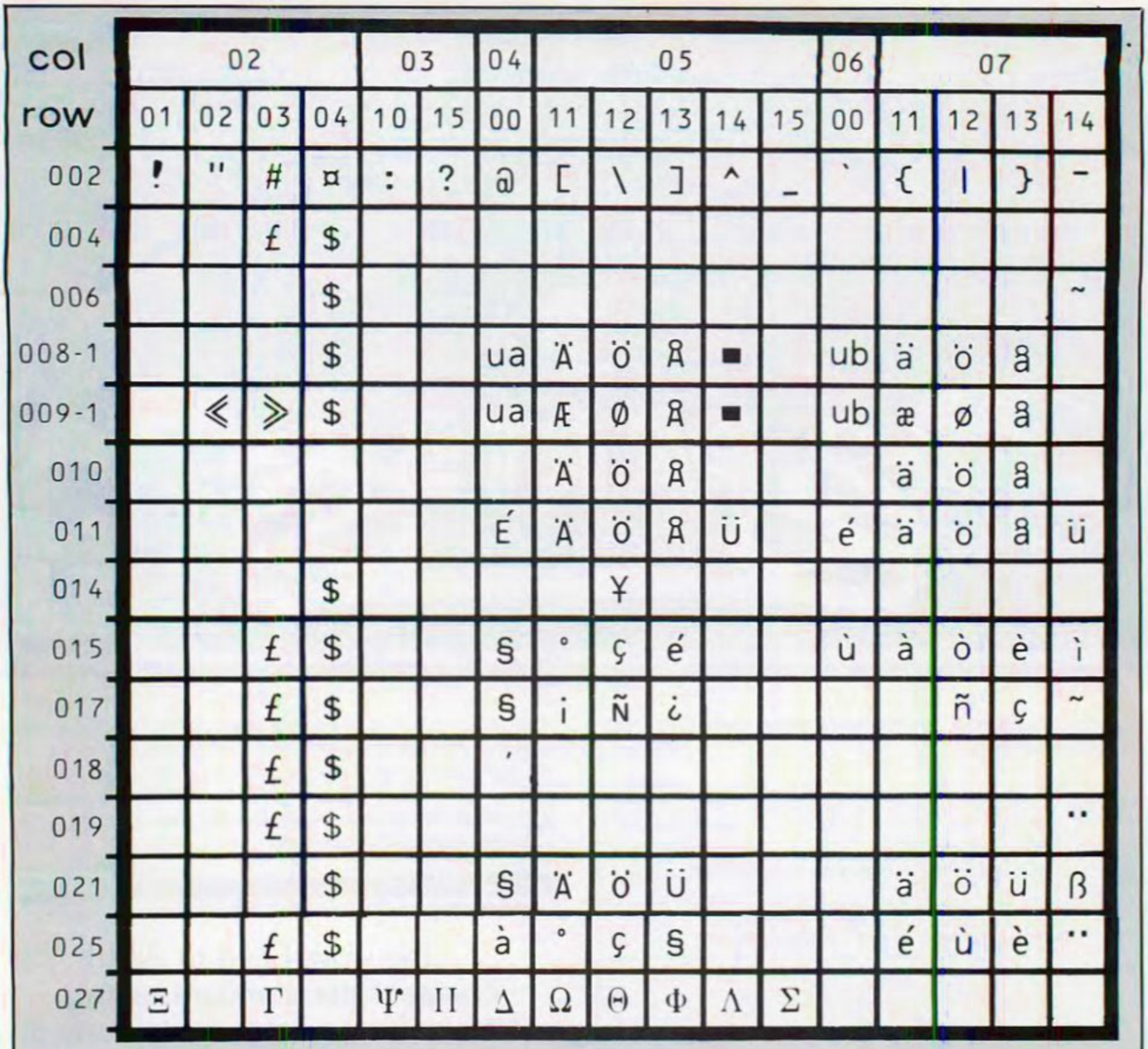
The registry set is available from AFNOR for approximately 172 French francs, say $35. It would be vital for an equipment or software manufacturer to have it, and it comes in a beautiful 4-ring binder symbolizing worldwide interchange compatibility. But the summary provided here will fill most needs.

. . .the work I had to do to compact the standard, trying to make it understandable, turned up more than unreadability. So it's back to the drawing board, perhaps for a considerable period of time. . .it's sometimes useful to have symbols whose meaning you can reassign without harm to programming languages. . .

CONTENT OF THE EXTENDED SETS

Figure 1a shows, against the ISO Code, International Reference Version, how the other graphic sets differ in the column/row positions shown. The rows are keyed to Table 1, reminding you that ASCII is "006", or "ISO 646-006".

From this figure we can see that many countries need accented letters as individual characters, not compound via BS (BackSpace). This is particularly true for the double sets 008 and 009, for Scandinavian newspaper transmission, which have characters that cannot be made from ASCII in compound form. For example — Ring-A, a solid, and the angle open and closed quotes.



col	02				03		04	05					06	07			
row	01	02	03	04	10	15	00	11	12	13	14	15	00	11	12	13	14
002	!	"	#	¤	:	?	@	[	\	]	^	_	`	{	|	}	¯
004			£	$													
006				$													~
008-1				$			ua	Ä	Ö	Å	■		ub	ä	ö	å	
009-1		≪	≫	$			ua	Æ	Ø	Å	■		ub	æ	ø	å	
010								Ä	Ö	Å				ä	ö	å	
011							É	Ä	Ö	Å	Ü		é	ä	ö	å	ü
014				$					¥								
015			£	$			§	°	ç	é			ù	à	ò	è	ì
017			£	$			§	¡	Ñ	¿					ñ	ç	~
018			£	$			'										
019			£	$													¨
021				$			§	Ä	Ö	Ü				ä	ö	ü	ß
025			£	$			à	°	ç	§				é	ù	è	¨
027	Ξ		Γ		Ψ	Π	Δ	Ω	Θ	Φ	Λ	Σ					

Figure 1a. Registered Graphic Character Substitution

008-2 and 009-2 are shown in Figure 1b. Here these are not exceptions from the IRV, but rather the only graphics assigned in the set. The additions are necessary to set type for newspapers throughout Scandinavia. See the Crossbar-D, Crossbar-O, the A-E ligature, and the Icelandic Thorn.

col	04			05				06			07			
row	01	04	05	00	05	11	12	01	04	05	00	05	11	12
008-2	À	Ð	É	Þ	Ü	Æ	Ø	à	ð	é	þ	ü	æ	ø
009-2	À	Ð	É	Þ	Ü	Ä	Ö	à	ð	é	þ	ü	ä	ö

Figure 1b. Registered Additional Graphic Sets

Figure 1a doesn't show Set 031 because it deviates more and is not of that much general interest. It doesn't show the Japanese Katakana set because that is completely different from the IRV. In fact, Japanese Industrial Standard C6220-1969 is an 8-bit coded set with the IRV (see Set 014 for the dollar and yen signs) in the lower (bit 8 = 0) portion, and Set 013 in the higher portion, with space reserved for future additional controls. This Set 013 is shown in Figure 2. It is shown in its high-order position, to indicate the card codes at the same time.

Figure 2 also shows the Cyrillic set of the USSR state standard GOST 13052-67, but it is not half of an 8-bit set as the Japanese do it. Rather it is another page of extensions. After SO (Shift Out) is used, the Russian register is operative. Following SI (Shift In) it is the IRV. Although this set has no registry number now, it was submitted recently by ECMA, and we expect an assignment soon. By the way, both Katakana and Cyrillic are shown in their OCR font.

Figure 3 shows the contents of the registered control sets. Set 007 serves as control set for the graphic sets 008-1,2 and 009-1,2, for Scandinavian newspaper transmission. And set 026 is the control set for the worldwide newspaper transmission, defined by the IPTC (International Press Telecommunications Council). The 18 control positions not shown, and those where there is no entry, are the same as in the International Reference Version (646-001).

These newspapers are driving composition equipment, not line printers, so they don't need VT and FF. Their set is already defined, so they don't need SO and SI. They have (properly) assigned meaning to three device controls. And they're probably not doing payroll, so they don't need the four information separators. But they do transmit, and instead of choosing their own functions and placement they have chosen to be a registered variant of the ISO Code. And all variants within this controlled and registered cluster can at least recognize each other, even if they can't print it!

b8 b7 b6 b5	b4 b3 b2 b1	COL / ROW	1000 8	1001 9	1010 10	1011 11	1100 12	1101 13	1110 14	1111 15
	0000	0				–	ダ Ю	ヨ П	Ю	П
	0001	1			｡	ｱ	ﾁ а	ﾑ Я	А	Я
	0010	2			｢	ｲ	ﾂ б	ﾒ Р	Б	Р
	0011	3			｣	ｳ	ﾃ Ц	ﾓ С	Ц	С
	0100	4			､	ｴ	ﾄ Д	ﾔ Т	Д	Т
	0101	5			･	ｵ	ﾅ е	ﾕ У	Е	У
	0110	6		ｦ	ｦ	ｶ	ﾆ Ф	ﾖ Ж	Ф	Ж
	0111	7		ｧ	ｷ	ｸ	ﾇ Г	ﾗ В	Г	В
	1000	8		ｨ	ｸ	ｹ	ﾈ Х	ﾘ Ъ	Х	Ъ
	1001	9		ｩ	ｹ	ｺ	ﾉ И	ﾙ Ы	И	Ы
	1010	10		ｪ	ｺ	ｻ	ﾊ Й	ﾚ З	Й	З
	1011	11		ｫ	ｻ	ｼ	ﾋ К	ﾛ Ш	К	Ш
	1100	12		ｬ	ｼ	ｽ	ﾌ Л	ﾜ Э	Л	Э
	1101	13		ｭ	ｽ	ｾ	ﾍ М	ﾝ Щ	М	Щ
	1110	14		ｮ	ｾ	ﾀ	ﾎ Н	ﾞ Ч	Н	Ч
	1111	15		ｯ	ｿ	ﾁ	ﾏ О	ﾟ —	О	

Figure 2. Katakana and Cyrillic Sets

Position	IRV 001	007	026	
0/09	HT	FO	FO	Format Control
0/11	VT	ECD	ECD	End (a typographical) CommanD
0/12	FF	SCD	SCD	Start (") CommanD
0/13	CR	QL	QL	Quad Left
0/14	SO	UR		Upper Rail
0/15	SI	LR		Lower Rail
1/01	DC1		Font 1	Change to normal
1/02	DC2		Font 2	Change to italic
1/03	DC3		Font 3	Change to bold
1/08	CAN	KW	KW	Kill Word (through previous space)
1/12	FS	SS	SS	SuperShift
1/13	GS	QC	QC	Quad Center
1/14	RS	QR	QR	Quad Right
1/15	US	JY	JY	JustifY

Figure 3. Registered Control Character Substitution

CODE EXTENSION IN ACTION

To illustrate the operation of code extension, let's imagine some equipment that may not exist now:

- A microfiche reader with automatic location controls.
- A microfiche with ASCII (the 8-bit form) on the first two pages, the other pages containing other sets such as Katakana, Cyrillic, Arabic, Greek, Hebrew, mathematical symbols, astronomical symbols, etc. Also, symbol sets for selecting typestyles, weights, rotations, sizes, and elongations.
- A display screen for the microfiche; it is touch-sensitive and generates 7-bit codes according to location touched on the display.
- As an alternative, keyboard tops with fibre optic bundles molded in as a matrix, so that the keytops can be lighted with different symbols as selected.

Now imagine that we are writing an astrology book:

- Type

```
Those of you born under the sign of Aries (
```

- Depress the "astro" key on the special keyboard
- Notice the shift in display for the fiche screen and/or the keytop lighting
- Touch the Aries symbol on the screen (or the keytop)
- Depress SI (Shift In) on the special keyboard
- And return to typing the rest of the sentence

```
) will find this month ...
```

Now imagine what a computer would do to the input stream in driving photocomposition equipment. The "astro" key generated an ESCape sequence for an astronomical graphic symbol set that would have been registered by AFNOR. When the input parser recognizes ESC, it analyzes the following characters, and then calls this set of character formation methods from the backup store, generates the character shape for Aries according to the character code after the final character, notices SI, and returns to normal mode.

Now we can envision how all of the world's printed material can be stored in machine-readable form, and interchanged recognizably!

ALTERNATE CONTROLS

Work has been in progress for several years to develop a companion standard for controls for devices such as CRT terminals. In the US this is contained in the ANSI document BSR X3.64, Additional Controls for Character Imaging. In a similar form, this C1 set is before the Codes Committee of ISO Technical Committee 97 (Computers and Information Processing) as document 2 N 868, for consideration at its 1978 May 24-26 meeting.

I had hoped to give the essence of this work in this installment. There were only two negative votes in X3, which one could presume might be answered. Unfortunately, the work I had to do to compact the standard, trying to make it understandable, turned up more than unreadability. It turned up many logical flaws and ambiguities. So it's back to the drawing board, perhaps for a considerable period of time.

Figures 4a through 4e will give, however, some flavor of the controls under consideration.

Figure 4 shows the controls of Format Type (FT) 1 and 2. Format 1 is either the single character of the 8-bit set, shown in the first column as "Ce", or the 2-character sequence of the type "ESC Fe", where Fe is a final character taken from 4/00 to 5/15, and whose column designation is 4 less than Ce. I.e., in an 8-bit code, INDex would be 8/04. In a 7-bit code it would be ESC 4/04. Format 2 is of the type "ESC Fs", where Fs is a final taken from 6/00 to 7/14.

Figures 4b through 4e show controls with formats beginning with the control "CSI", defined in Figure 4a to be either 9/11 (in the 8-bit set) or "ESC [" (in the 7-bit sets). The six possible formats are:

```
3a = CSI Pn F          4a = CSI Pn I F
3b = CSI Pn ; Pn F     4b = CSI Pn ; Pn I F
3c = CSI Ps F          4c = CSI Ps I F
```

Pn stands for numeric parameter(s), Ps for a variable number of selective parameters separated by semicolons. The type 4 formats differ from type 3 only in inserting the intermediate character 2/00 just prior to the final.

In the figures, the parameter value enclosed in parentheses is the default value. That is, if the parameters are not actually inserted, i.e., being null, then the effect is the same as if the default value(s) were inserted.

To give an example of how these controls operate, look in Figure 4d for the second mnemonic, SGR (Select Graphic Rendition). It is represented first by CSI, the Control Sequence Introducer, the parameter, and the final 6/13. This means that when the 4-character string

ESC [6 m

is encountered, it should turn on rapid blink in the field(s) specified on your video screen.

```
AL  = Active Line (containing AP)
AP  = Active Position (where the cursor is)
EF  = Editor Function
FE  = Format Effector
HT  = Horizontal Tabulation
IN  = INtroducer
PAD = Primary Auxiliary Device
RD  = Received Datastream
SAD = Secondary Auxiliary Device
SD  = String Delimiter
VT  = Vertical Tabulation
QA  = Qualified Area (defined by DAQ, SPA, EPA)
rfs = reserved for future standardization
```

Abbreviations for Figures 4a through 4e.

Ce	FT	Type	Param	Mnem	Name
8/00-03	1				(rfs)
8/04	1	FE		IND	INDex
8/05	1	FE		NEL	NExt Line
8/06	1			SSA	Start of Selected Area
8/07	1			ESA	End of Selected Area
8/08	1	FE		HTS	Horizontal Tabulation Set
8/09	1	FE		HTJ	Horiz. Tabul. with Justification
8/10	1	FE		VTS	Vertical Tabulation Set
8/11	1	FE		PLD	Partial Line Down
8/12	1	FE		PLU	Partial Line Up
8/13	1	FE		RI	Reverse Index
8/14	1	IN		SS2	Single Shift 2
8/15	1	IN		SS3	Single Shift 3
9/00	1	SD		DCS	Device Control String
9/01	1			PU1	Private Use 1
9/02	1			PU2	Private Use 2
9/03	1			STS	Set Transmit State
9/04	1			CCH	Cancel CHaracter
9/05	1			MW	Message Waiting
9/06	1			SPA	Start of Protected Area
9/07	1			EPA	End of Protected Area
9/08-10	1				(rfs)
9/11	1	IN		CSI	Control Sequence Introducer
9/12	1	SD		ST	String Terminator
9/13	1	SD		OSC	Operating System Command
9/14	1	SD		PM	Privacy Message
9/15	1	SD		APC	Application Program Command

Fs	FT	Mnem	Name
6/00	2	DMI	Disable Manual Input
6/01	2	INT	INTerrupt
6/02	2	EMI	Enable Manual Interrupt
6/03	2	RIS	Reset to Initial State

Figure 4a. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
4/00	3a	EF	(1)	ICH	Insert CHaracter
4/01	3a	EF	(1)	CUU	CUrsor Up
4/02	3a	EF	(1)	CUD	CUrsor Down
4/03	3a	EF	(1)	CUF	CUrsor Forward
4/04	3a	EF	(1)	CUB	CUrsor Backward
4/05	3a	EF	(1)	CNL	Cursor Next Line
4/06	3a	EF	(1)	CPL	Cursor Preceding Line
4/07	3a	EF	(1)	CHA	Cursor Horizontal Absolute
4/08	3b	EF	(1;1)	CUP	CUrsor Position
4/09	3a	EF	(1)	CHT	Cursor Horizontal Tabulation
4/10	3c	EF		ED	Erase in Display
			(0)		From AP to end (inclusive)
			1		From start to AP (inclusive)
			2		All of display
4/11	3c	EF		EL	Erase in Line
			(0)		From AP to end (inclusive)
			1		From start to AP (inclusive)
			2		All of line
4/12	3a	EF	(1)	IL	Insert Line
4/13	3a	EF	(1)	DL	Delete Line
4/14	3c	EF		EF	Erase in Field
			(0)		From AP to end (inclusive)
			1		From start to AP (inclusive)
			2		All of field
4/15	3c	EF		EA	Erase in Area
			(0)		From AP to end (inclusive)
			1		From start to AP (inclusive)
			2		All of QA
5/00	3a	EF	(1)	DCH	Delete CHaracter
5/01	3c			SEM	Select editing Extent Mode
			(0)		Edit in display
			1		Edit in AL
			2		Edit in field
			3		Edit in QA
5/02	3b		(1;1)	CPR	Cursor Position Report
5/03	3a	EF	(1)	SU	Scroll Up
5/04	3a	EF	(1)	SD	Scroll Down
5/05	3a	EF	(1)	NP	Next Page
5/06	3a	EF	(1)	PP	Preceding Page
5/07	3c	EF		CTC	Cursor Tabulation Control
			(0)		Set HT stop at AP
			1		Set VT stop at AL
			2		Clear HT stop at AP
			3		Clear VT stop at AL
			4		Clear all HT stops in AL
			5		Clear all HT stops in device
			6		Clear all VT stops in device
5/08	3a	EF	(1)	ECH	Erase CHaracter
5/09	3a	EF	(1)	CVT	Cursor Vertical Tabulation
5/10	3a	EF	(1)	CBT	Cursor Backward Tabulation

Figure 4b. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
6/00	3a	FE	(1)	HPA	Horizontal Position Absolute
6/01	3a	FE	(1)	HPR	Horizontal Position Relative
6/02	3a		(1)	REP	REPeat
6/03	3a		(0)	DA	Device Attributes
6/04	3a	FE	(1)	VPA	Vertical Position Absolute
6/05	3a	FE	(1)	VPR	Vertical Position Relative
6/06	3b	FE	(1;1)	HVP	Horiz. and Vertical Position
6/07	3c	FE		TBC	TaBulation Clear
			(0)		Clear HT stop at AP
			1		Clear VT stop at AL
			2		Clear all HT stops in AL
			3		Clear all HT stops
			4		Clear all VT stops
6/08	3c			SM	Set Mode
			1	GATM	Guarded Area Transfer Mode
			2	KAM	Keyboard Action Mode
			3	CRM	Control Representation Mode
			4	IRM	Insertion-Replacement Mode
			5	SRTM	Status Reporting Transfer Mode
			6	ERM	ERasure Mode
			7	VEM	Vertical Editing Mode
			8		(rfs)
			9		(rfs)
			10	HEM	Horizontal Editing Mode
			11	PUM	Positioning Unit Mode
			12	SRM	Send-Receive Mode
			13	FEAM	Format Effector Action Mode
			14	FETM	Format Effector Transfer Mode
			15	MATM	Multiple Area Transfer Mode
			16	TTM	Transfer Termination Mode
			17	SATM	Selected Area Transfer Mode
			18	TSM	Tabulation Stop Mode
			19	EBM	Editing Boundary Mode
			20	LNM	Line feed New line Mode
6/09	3c			MC	Media Copy
			(0)		To PAD
			1		From PAD
			2		To SAD
			3		From SAD
			4		Turn OFF copying RD to PAD
			5		Turn ON copying RD to PAD
			6		Turn OFF copying RD to SAD
			7		Turn ON copying RD to SAD

Figure 4c. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
6/10-11					(rfs)
6/12	3c			RM	Reset Mode
					(same parameters as SM)
6/13	3c	FE		SGR	Select Graphic Rendition
			(0)		Primary rendition
			1		Bold, or increased intensity
			2		Faint, decreased intensity, or secondary color
			3		Italic
			4		Underscore
			5		Slow blink (< 2.5/second)
			6		Rapid blink (> 2.5/second)
			7		Negative (reverse) image
			8		(rfs)
			9		(rfs)
			10		Primary Font
			11-19		1st to 9th alt. font (via FNT)
			20		Fraktur
6/14	3c			DSR	Device Status Report
			(0)		Ready, no malfunctions detected
			1		Busy - retry later
			2		Busy - DSR will notify ready
			3		Malfunction - retry
			4		Malfunction - DSR will notify ready
			5		Please report status (DSR or DSC)
			6		Please report AP via CPR
6/15	3c			DAQ	Define Area Qualification
			(0)		Accept all input
			1		Accept no input (protected); do not transmit (guarded)
			2		Accept graphics
			3		Accept numerics
			4		Accept alphabetics
			5		Right justify in area
			6		Zerofill in area
			7		HT stop at start of area (field)
			8		Accept no input (protected); permit transmit (unguarded)
			9		Spacefill in area

Figure 4d. Controls for Character-Imaging Devices

Final	FT	Type	Param	Mnem	Name
4/00	4a	EF	(1)	SL	Scroll Left
4/01	4a	EF	(1)	SR	Scroll Right
4/02	4b	FE	(100;100)	GSM	Graphic Size Modification
4/03	4a	FE		GSS	Graphic Size Selection
4/04	4b	FE	(0;0)	FNT	FoNT selection
			(0;0)		Primary font
			1;0		First alternative font
			...		...
			9;0		Ninth alternative font
4/05	4a	FE		TSS	Thin Space Specification
4/06	4c	FE		JFY	JustiFY
			(0)		Terminate all justify actions
			1		Fill action ON (text to/from other lines)
			2		Interword spacing
			3		Letter spacing
			4		Hyphenation
			5		Flush left margin
			6		Center text between margins
			7		Flush right margin
			8		Italian form (underscore last)
4/07	4b	FE		SPI	SPacing Increment
4/08	4c	FE		QUAD	Quad
			(0)		Flush left
			1		Flush left, fill with leader
			2		Center
			3		Center, fill with leader
			4		Flush right
			5		Flush right, fill with leader

Figure 4e. Controls for Character-Imaging Devices

Code	Symbol	Code	Symbol
10/00	(same as 02/00)	11/00	Large circle
10/01	Opening double quote	11/01	Dagger
10/02	Closing double quote	11/02	Superior (superscript) 2
10/03	Club suit	11/03	Superior (superscript) 3
10/04	Diamond suit	11/04	Rectangle
10/05	Heart suit	11/05	Parallel
10/06	Spade suit	11/06	Partial derivative
10/07	Closing single quote	11/07	Lower left corner, floor
10/08	Is implied by	11/08	Upper left corner, ceiling
10/09	Implies	11/09	Upper right corner
10/10	Multiply	11/10	Lower right corner
10/11	Plus or minus	11/11	Perpendicular
10/12	Nabla, or del	11/12	Less than or equal
10/13	Em dash	11/13	Not equal, other than
10/14	Radix point	11/14	Greater than or equal
10/15	Divide	11/15	Paragraph mark, pilcrow
12/00	Section mark	13/00	Capital pi
12/01	Double dagger	13/01	Capital psi
12/02	Dot bullet	13/02	Square bullet
12/03	Capital theta	13/03	Capital sigma
12/04	Capital delta	13/04	Integral
12/05	At least one exists	13/05	Capital upsilon
12/06	Capital phi	13/06	Therefore
12/07	Capital gamma	13/07	Capital omega
12/08	Upward arrow	13/08	Downward arrow
12/09	Right arrow	13/09	Left arrow
12/10	Dot product	13/10	Approximately equal
12/11	Degree	13/11	Opening angular bracket
12/12	Capital lambda	13/12	Logical AND
12/13	Register	13/13	Closing angular bracket
12/14	Copyright mark	13/14	Logical NOT
12/15	Capital xi	13/15	Infinity
14/00	Opening single quote	15/00	Small pi
14/01	Small alpha	15/01	Small psi
14/02	Small beta	15/02	Small rho
14/03	Small theta	15/03	Small sigma
14/04	Small delta	15/04	Small tau
14/05	Small epsilon	15/05	Small upsilon
14/06	Small phi	15/06	Check mark, radical mark
14/07	Small gamma	15/07	Small omega
14/08	Small eta	15/08	Small chi
14/09	Small iota	15/09	Logical universal quantifier
14/10	Identically equivalent	15/10	Small zeta
14/11	Small kappa	15/11	Cap intersection
14/12	Small lambda	15/12	Logical OR
14/13	Small mu	15/13	Cup, union
14/14	Small nu	15/14	Overbar
14/15	Small xi	15/15	(same as 7/15)

Table 3. Names of the Additional Graphics, 8-bit Set

Figure 5. 8-bit ASCII Proposal

CODE EXPANSION

We have seen how ASCII was *extended* by making many related pages of the 7-bit code. It is also possible to *expand* ASCII into an 8-bit code, or even 9-bit and 10-bit if we wished, for that matter. But an 8-bit code is obviously the most logical one to concentrate on, and this has been under development for several years.

The proposed 8-bit Expanded ASCII Code is shown in Figure 5. The identification of the graphic symbols is given in Table 3.

One can observe many interesting things about this set. For example, it has the entire Greek set of small letters except for "omicron", with eleven capitals to go with others from the Roman capitals to complete the Greek set. But apparently the committee didn't follow 646-031, the Greek alphabet mentioned in Table 1. They didn't use the customary ordering "alpha-beta-gamma", the way we learn our "a-b-c's". I suppose it is argued that this set will never be used for language, only math symbols. And 646-027, shown in Figure 1a, does not demand the special capital "upsilon" shown in position 13/5. If the Greeks can agree to using a Roman capital "Y" for upsilon, could the Americans?

You'll notice some math symbols, but not enough for APL. In fact, the whole set seems highly slanted to mathematics, rather than business. Of course there are the four corner symbols for forms. Presumably the card suits will strike your eye, and you will wonder why so many other useful symbols were ignored in favor of these. Don't worry, they will always come in handy; it's sometimes useful to have symbols whose meaning you can reassign without harm to programming languages, etc. The committee were obviously bridge players, for spades collate high.

This proposal has not had real public scrutiny yet, and it must be considered no more than a proposal. Presumably X3 will agree about July that it should be sent out for formal public review and letter ballot. My guess is that it will not be adopted in just the form you see here.

FUTURE FOR ASCII

The methods are in place for codifying all symbols that people use. They may be language alphabets, signs, drawing symbols, or controls for equipments. Robots, for example. Satellites are augmenting conventional telecommunications systems, so that one can borrow cheaply and permanently from electronic libraries.

To prepare for this, other sets are being developed for registry, many through ISO Technical Committee 46/1, Automated Documentation. A 2-page mathematical symbol set is near submission, as are African sets. Work is started for Arabic, which will take about 5 sets to handle fully, although there is a commercial subset of 94 graphics. Another C1 set is being proposed for bibliographic controls. It contains four types — annotation controls, filing controls, reference controls, and subject designators. Other C1 sets can come from process control, animation and other graphics applications, etc.

West Germany has proposed a new ISO project on text communication, to harmonize teleconnection of the more than one hundred varieties of typewriters (and keyboards) throughout the world. The extension method of multiple 7-bit codes is ideal for this (8-bit codes imply too many keys or shift combinations for people to use easily.

I am convinced that microcomputer users are going to develop some fantastic applications that will become widespread enough for their special graphic and control sets to be registered. How about a control set or two for sewing machines?

In fact, it is very difficult to think of any general application where one could not find a usage for these registered variants and extensions. □