WIKIPEDIA

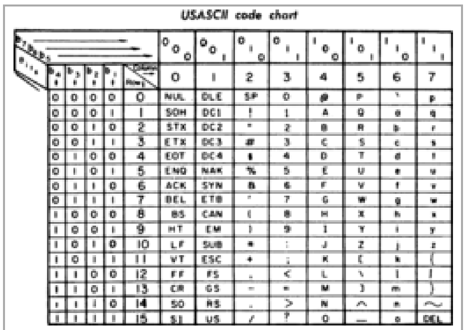
ASCII

ASCII (/ˈæskiː/ (listen) *ASS-kee*),[3]:6 abbreviated from **American Standard Code for Information Interchange**, is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

The Internet Assigned Numbers Authority (IANA) prefers the name **US-ASCII** for this character encoding.[2]

ASCII is one of the IEEE milestones.

ASCII



ASCII chart from a pre-1972 printer manual

MIME / IANA	us-ascii
Alias(es)	ISO-IR-006,[1] ANSI_X3.4-1968, ANSI_X3.4-1986, ISO_646.irv:1991, ISO646-US, us, IBM367, cp367[2]
Language(s)	English
Classification	ISO 646 series
Extensions	▪ Unicode ▪ ISO/IEC 8859 (series) ▪ KOI-8 ▪ OEM (series) ▪ Windows-125x (series) ▪ Others
Preceded by	ITA 2, FIELDATA
Succeeded by	ISO 8859, Unicode

Contents

Overview

ASCII was developed from telegraph code. Its first commercial use was as a seven-bit teleprinter code promoted by Bell data services. Work on the ASCII standard began on October 6, 1960, with the first meeting of the American Standards Association's (ASA) (now the American National Standards Institute or ANSI) X3.2 subcommittee. The first edition of the standard was published in 1963,[4][5] underwent a major revision during 1967,[6][7] and experienced its most recent update during 1986.[8] Compared to earlier telegraph codes, the proposed Bell code and ASCII were both ordered for more convenient sorting (i.e., alphabetization) of lists, and added features for devices other than teleprinters.

The use of ASCII format for Network Interchange was described in 1969.[9] That document was formally elevated to an Internet Standard in 2015.[10]

Originally based on the English alphabet, ASCII encodes 128 specified characters into seven-bit integers as shown by the ASCII chart above.[11] Ninety-five of the encoded characters are printable: these include the digits *0* to *9*, lowercase letters *a* to *z*, uppercase letters *A* to *Z*, and punctuation symbols. In addition, the original ASCII specification included 33 non-printing control codes which originated with Teletype machines; most of these are now obsolete,[12] although a few are still commonly used, such as the carriage return, line feed and tab codes.

For example, lowercase *i* would be represented in the ASCII encoding by binary 1101001 = hexadecimal 69 (*i* is the ninth letter) = decimal 105.

History

The American Standard Code for Information Interchange (ASCII) was developed under the auspices of a committee of the American Standards Association (ASA), called the X3 committee, by its X3.2 (later X3L2) subcommittee, and later by that subcommittee's X3.2.4 working group (now INCITS). The ASA became the United States of America Standards Institute (USASI)[3]:211 and ultimately the American National Standards Institute (ANSI).

ASCII (1963). Control pictures of equivalent controls are shown where they exist, or a grey dot otherwise.

With the other special characters and control codes filled in, ASCII was published as ASA X3.4-1963,[5][13] leaving 28 code positions without any assigned meaning, reserved for future standardization, and one unassigned control code.[3]:66, 245 There was some debate at the time whether there should be more control characters rather than the lowercase alphabet.[3]:435 The indecision did not last long: during May 1963 the CCITT Working Party on the New Telegraph Alphabet proposed to assign lowercase characters to *sticks*[a][14] 6 and 7,[15] and International Organization for Standardization TC 97 SC 2 voted during October to incorporate the change into its draft standard.[16] The X3.2.4 task group voted its approval for the change to ASCII at its May 1963 meeting.[17] Locating the lowercase letters in *sticks*[a][14] 6 and 7 caused the characters to differ in bit pattern from the upper case by a single bit, which simplified case-insensitive character matching and the construction of keyboards and printers.

The X3 committee made other changes, including other new characters (the brace and vertical bar characters),[18] renaming some control characters (SOM became start of header (SOH)) and moving or removing others (RU was removed).[3]:247–248 ASCII was subsequently updated as USAS X3.4-1967,[6][19] then USAS X3.4-1968, ANSI X3.4-1977, and finally, ANSI X3.4-1986.[8][20]

Revisions of the ASCII standard:

- ASA X3.4-1963[3][5][19][20]
- ASA X3.4-1965 (approved, but not published, nevertheless used by IBM 2260 & 2265 Display Stations and IBM 2848 Display Control)[3]:423, 425–428, 435–439[21][19][20]
- USAS X3.4-1967[3][6][20]
- USAS X3.4-1968[3][20]
- ANSI X3.4-1977[20]
- ANSI X3.4-1986[8][20]
- ANSI X3.4-1986 (R1992)
- ANSI X3.4-1986 (R1997)
- ANSI INCITS 4-1986 (R2002)[22]
- ANSI INCITS 4-1986 (R2007)[23]
- (ANSI) INCITS 4-1986[R2012][24]
- (ANSI) INCITS 4-1986[R2017][25]

In the X3.15 standard, the X3 committee also addressed how ASCII should be transmitted (least significant bit first),[3]:249–253[26] and how it should be recorded on perforated tape. They proposed a 9-track standard for magnetic tape, and attempted to deal with some punched card formats.

Design considerations

Bit width

The X3.2 subcommittee designed ASCII based on the earlier teleprinter encoding systems. Like other character encodings, ASCII specifies a correspondence between digital bit patterns and character symbols (i.e. graphemes and control characters). This allows digital devices to communicate with each other and to process, store, and communicate character-oriented information such as written language. Before ASCII was developed, the encodings in use included 26 alphabetic characters, 10 numerical digits, and from 11 to 25 special graphic symbols. To include all these, and control characters compatible with the Comité Consultatif International Téléphonique et Télégraphique (CCITT) International Telegraph Alphabet No. 2 (ITA2) standard of 1924,[27][28] FIELDATA (1956), and early EBCDIC (1963), more than 64 codes were required for ASCII.

ITA2 were in turn based on the 5-bit telegraph code Émile Baudot invented in 1870 and patented in 1874.[28]

The committee debated the possibility of a shift function (like in ITA2), which would allow more than 64 codes to be represented by a six-bit code. In a shifted code, some character codes determine choices between options for the following character codes. It allows compact encoding, but is less reliable for data transmission, as an error in transmitting the shift code typically makes a long part of the transmission unreadable. The standards committee decided against shifting, and so ASCII required at least a seven-bit code.[3]:215, 236 §4

The committee considered an eight-bit code, since eight bits (octets) would allow two four-bit patterns to efficiently encode two digits with binary-coded decimal. However, it would require all data transmission to send eight bits when seven could suffice. The committee voted to use a seven-bit code to minimize costs associated with data transmission. Since perforated tape at the time could record eight bits in one position, it also allowed for a parity bit for error checking if desired.[3]:217, 236 §5 Eight-bit machines (with octets as the native data type) that did not use parity checking typically set the eighth bit to 0.[29] In some printers, the high bit was used to enable Italics printing.

Internal organization

The code itself was patterned so that most control codes were together and all graphic codes were together, for ease of identification. The first two so-called *ASCII sticks*[a][14] (32 positions) were reserved for control characters.[3]:220, 236 8,9) The "space" character had to come before graphics to make sorting easier, so it became position 20_{hex};[3]:237 §10 for the same reason, many special signs commonly used as separators were placed before digits. The committee decided it was important to support uppercase 64-character alphabets, and chose to pattern ASCII so it could be reduced easily to a usable 64-character set of graphic codes,[3]:228, 237 §14 as was done in the DEC SIXBIT code (1963). Lowercase letters were therefore not interleaved with uppercase. To keep options available for lowercase letters and other graphics, the special and numeric codes were arranged before the letters, and the letter *A* was placed in position 41_{hex} to match the draft of the corresponding British standard.[3]:238 §18 The digits 0–9 are prefixed with 011, but the remaining 4 bits correspond to their respective values in binary, making conversion with binary-coded decimal straightforward.

Many of the non-alphanumeric characters were positioned to correspond to their shifted position on typewriters; an important subtlety is that these were based on *mechanical* typewriters, not *electric* typewriters.[30] Mechanical typewriters followed the standard set by the Remington No. 2 (1878), the first typewriter with a shift key, and the shifted values of `23456789-` were `"#$%_&'()` – early typewriters omitted *0* and *1*, using *O* (capital letter *o*) and *l* (lowercase letter *L*) instead, but `1!` and `0)` pairs became standard once 0 and 1 became common. Thus, in ASCII `!"#$%` were placed in the second stick,[a][14] positions 1–5, corresponding to the digits 1–5 in the adjacent stick.[a][14] The parentheses could not correspond to *9* and *0*, however, because the place corresponding to *0* was taken by the space character. This was accommodated by removing _ (underscore) from *6* and shifting the remaining characters, which corresponded to many European typewriters that placed the parentheses with *8* and *9*. This discrepancy from typewriters led to bit-paired keyboards, notably the Teletype Model 33, which used the left-shifted layout corresponding to ASCII, not to traditional mechanical typewriters. Electric typewriters, notably the IBM Selectric (1961), used a somewhat different layout that has become standard on computers – following the IBM PC (1981), especially Model M (1984) – and thus shift values for symbols on modern keyboards do not correspond as closely to the

ASCII table as earlier keyboards did. The /? pair also dates to the No. 2, and the ,< .> pairs were used on some keyboards (others, including the No. 2, did not shift , (comma) or . (full stop) so they could be used in uppercase without unshifting). However, ASCII split the ;: pair (dating to No. 2), and rearranged mathematical symbols (varied conventions, commonly -* =+) to :* ;+ -=.

Some common characters were not included, notably ½¼¢, while ^`~ were included as diacritics for international use, and <> for mathematical use, together with the simple line characters \| (in addition to common /). The @ symbol was not used in continental Europe and the committee expected it would be replaced by an accented *À* in the French variation, so the @ was placed in position 40_{hex}, right before the letter A.[3]:243

The control codes felt essential for data transmission were the start of message (SOM), end of address (EOA), end of message (EOM), end of transmission (EOT), "who are you?" (WRU), "are you?" (RU), a reserved device control (DC0), synchronous idle (SYNC), and acknowledge (ACK). These were positioned to maximize the Hamming distance between their bit patterns.[3]:243–245

Character order

ASCII-code order is also called *ASCIIbetical* order.[31] Collation of data is sometimes done in this order rather than "standard" alphabetical order (collating sequence). The main deviations in ASCII order are:

- All uppercase come before lowercase letters; for example, "Z" precedes "a"
- Digits and many punctuation marks come before letters

An intermediate order converts uppercase letters to lowercase before comparing ASCII values.

Character groups

Control characters

ASCII reserves the first 32 codes (numbers 0–31 decimal) for control characters: codes originally intended not to represent printable information, but rather to control devices (such as printers) that make use of ASCII, or to provide meta-information about data streams such as those stored on magnetic tape.

For example, character 10 represents the "line feed" function (which causes a printer to advance its paper), and character 8 represents "backspace". RFC 2822 (https://tools.ietf.org/html/rfc2822) refers to control characters that do not include carriage return, line feed or white space as non-whitespace control characters.[32] Except for the control characters that prescribe elementary line-oriented formatting, ASCII does not define any mechanism for describing the structure or appearance of text within a document. Other schemes, such as markup languages, address page and document layout and formatting.

The original ASCII standard used only short descriptive phrases for each control character. The ambiguity this caused was sometimes intentional, for example where a character would be used slightly differently on a terminal link than on a data stream, and sometimes accidental, for example with the meaning of "delete".

Probably the most influential single device on the interpretation of these characters was the Teletype Model 33 ASR, which was a printing terminal with an available paper tape reader/punch option. Paper tape was a very popular medium for long-term program storage until the 1980s, less costly and in some ways less fragile than magnetic tape. In particular, the Teletype Model 33 machine assignments for codes 17 (Control-Q, DC1, also known as XON), 19 (Control-S, DC3, also known as XOFF), and 127 (Delete) became de facto standards. The Model 33 was also notable for taking the description of Control-G (code 7, BEL, meaning audibly alert the operator) literally, as the unit contained an actual bell which it rang when it received a BEL character. Because the keytop for the O key also showed a left-arrow symbol (from ASCII-1963, which had this character instead of underscore), a noncompliant use of code 15 (Control-O, Shift In) interpreted as "delete previous character" was also adopted by many early timesharing systems but eventually became neglected.

When a Teletype 33 ASR equipped with the automatic paper tape reader received a Control-S (XOFF, an abbreviation for transmit off), it caused the tape reader to stop; receiving Control-Q (XON, "transmit on") caused the tape reader to resume. This technique became adopted by several early computer operating systems as a "handshaking" signal warning a sender to stop transmission because of impending overflow; it persists to this day in many systems as a manual output control technique.

On some systems Control-S retains its meaning but Control-Q is replaced by a second Control-S to resume output. The 33 ASR also could be configured to employ Control-R (DC2) and Control-T (DC4) to start and stop the tape punch; on some units equipped with this function, the corresponding control character lettering on the keycap above the letter was TAPE and ~~TAPE~~ respectively.[33]

Delete & Backspace

The Teletype could not move the head backwards, so it did not put a key on the keyboard to send a BS (backspace). Instead there was a key marked RUB OUT that sent code 127 (DEL). The purpose of this key was to erase mistakes in a hand-typed paper tape: the operator had to push a button on the tape punch to back it up, then type the rubout, which punched all holes and replaced the mistake with a character that was intended to be ignored.[34] Teletypes were commonly used for the less-expensive computers from Digital Equipment Corporation, so these systems had to use the available key and thus the DEL code to erase the previous character.[35][36] Because of this, DEC video terminals (by default) sent the DEL code for the key marked "Backspace" while the key marked "Delete" sent an escape sequence, while many other terminals sent BS for the Backspace key. The Unix terminal driver could only use one code to erase the previous character, this could be set to BS *or* DEL, but not both, resulting in a long period of annoyance where users had to correct it depending on what terminal they were using (shells that allow line editing, such as ksh, bash, and zsh, understand both). The assumption that no key sent a BS caused Control+H to be used for other purposes, such as the "help" prefix command in GNU Emacs.[37]

Escape

Many more of the control codes have been given meanings quite different from their original ones. The "escape" character (ESC, code 27), for example, was intended originally to allow sending other control characters as literals instead of invoking their meaning. This is the same meaning of "escape" encountered in URL encodings, C language strings, and other systems where certain characters have a reserved meaning. Over time this meaning has been co-opted and has eventually been changed. In modern use, an ESC sent to the terminal usually indicates the start of a command sequence usually in the form of a so-called "ANSI escape code" (or, more properly, a "Control Sequence Introducer") from ECMA-48 (1972) and its successors, beginning with ESC followed by a "[" (left-bracket) character. An ESC sent from the terminal is most often used as an out-of-band character used to terminate an operation, as in the TECO and vi text editors. In graphical user interface (GUI) and windowing systems, ESC generally causes an application to abort its current operation or to exit (terminate) altogether.

End of Line

The inherent ambiguity of many control characters, combined with their historical usage, created problems when transferring "plain text" files between systems. The best example of this is the newline problem on various operating systems. Teletype machines required that a line of text be terminated with both "Carriage Return" (which moves the printhead to the beginning of the line) and "Line Feed" (which advances the paper one line without moving the printhead). The name "Carriage Return" comes from the fact that on a manual typewriter the carriage holding the paper moved while the position where the typebars struck the ribbon remained stationary. The entire carriage had to be pushed (returned) to the right in order to position the left margin of the paper for the next line.

DEC operating systems (OS/8, RT-11, RSX-11, RSTS, TOPS-10, etc.) used both characters to mark the end of a line so that the console device (originally Teletype machines) would work. By the time so-called "glass TTYs" (later called CRTs or terminals) came along, the convention was so well established that backward compatibility necessitated continuing the convention. When Gary Kildall created CP/M he was inspired by some command line interface conventions used in DEC's RT-11. Until the introduction of PC DOS in 1981, IBM had no hand in this because their 1970s operating systems used EBCDIC instead of ASCII and they were oriented toward punch-card input and line printer output on which the concept of carriage return was meaningless. IBM's PC DOS (also marketed as MS-DOS by Microsoft) inherited the convention by virtue of being loosely based on CP/M,[38] and Windows inherited it from MS-DOS.

Unfortunately, requiring two characters to mark the end of a line introduces unnecessary complexity and questions as to how to interpret each character when encountered alone. To simplify matters plain text data streams, including files, on Multics[39] used line feed (LF) alone as a line terminator. Unix and Unix-like systems, and Amiga systems, adopted this convention from Multics. The original Macintosh OS, Apple DOS, and ProDOS, on the other hand, used carriage return (CR) alone as a line terminator; however, since Apple replaced these operating systems with the Unix-based macOS operating system, they now use line feed (LF) as well. The Radio Shack TRS-80 also used a lone CR to terminate lines.

Computers attached to the ARPANET included machines running operating systems such as TOPS-10 and TENEX using CR-LF line endings, machines running operating systems such as Multics using LF line endings, and machines running operating systems such as OS/360 that represented lines as a character count followed by the characters of the line and that used EBCDIC rather than ASCII. The Telnet protocol defined an ASCII "Network Virtual Terminal" (NVT), so that connections between hosts with different line-ending conventions and character sets could be supported by transmitting a standard text format over the network. Telnet used ASCII along with CR-LF line endings, and software using other conventions would translate between the local conventions and the NVT.[40] The File Transfer Protocol adopted the Telnet protocol, including use of the Network Virtual Terminal, for use when transmitting commands and transferring data in the default ASCII mode.[41][42] This adds complexity to implementations of those protocols, and to other network protocols, such as those used for E-mail and the World Wide Web, on systems not using the NVT's CR-LF line-ending convention.[43][44]

End of File/Stream

The PDP-6 monitor,[35] and its PDP-10 successor TOPS-10,[36] used Control-Z (SUB) as an end-of-file indication for input from a terminal. Some operating systems such as CP/M tracked file length only in units of disk blocks and used Control-Z to mark the end of the actual text in the file.[45] For these reasons, EOF, or end-of-file, was used colloquially and conventionally as a three-letter acronym for Control-Z instead of SUBstitute. The end-of-text code (ETX), also known as Control-C, was inappropriate for a variety of reasons, while using Z as the control code to end a file is analogous to it ending the alphabet and serves as a very convenient mnemonic aid. A historically common and still prevalent convention uses the ETX code convention to interrupt and halt a program via an input data stream, usually from a keyboard.

In C library and Unix conventions, the null character is used to terminate text strings; such null-terminated strings can be known in abbreviation as ASCIZ or ASCIIZ, where here Z stands for "zero".

Binary	Oct	Dec	Hex	Abbreviation			[b]	[c]	[d]	Name (1967)
				1963	1965	1967				
000 0000	000	0	00	NULL	NUL		␀	^@	\0	Null
000 0001	001	1	01	SOM	SOH		␁	^A		Start of Heading
000 0010	002	2	02	EOA	STX		␂	^B		Start of Text
000 0011	003	3	03	EOM	ETX		␃	^C		End of Text
000 0100	004	4	04	EOT			␄	^D		End of Transmission
000 0101	005	5	05	WRU	ENQ		␅	^E		Enquiry
000 0110	006	6	06	RU	ACK		␆	^F		Acknowledgement
000 0111	007	7	07	BELL	BEL		␇	^G	\a	Bell
000 1000	010	8	08	FE0	BS		␈	^H	\b	Backspace[e][f]
000 1001	011	9	09	HT/SK	HT		␉	^I	\t	Horizontal Tab[g]
000 1010	012	10	0A	LF			␊	^J	\n	Line Feed
000 1011	013	11	0B	VTAB	VT		␋	^K	\v	Vertical Tab
000 1100	014	12	0C	FF			␌	^L	\f	Form Feed
000 1101	015	13	0D	CR			␍	^M	\r	Carriage Return[h]
000 1110	016	14	0E	SO			␎	^N		Shift Out
000 1111	017	15	0F	SI			␏	^O		Shift In
001 0000	020	16	10	DC0	DLE		␐	^P		Data Link Escape
001 0001	021	17	11	DC1			␑	^Q		Device Control 1 (often XON)
001 0010	022	18	12	DC2			␒	^R		Device Control 2
001 0011	023	19	13	DC3			␓	^S		Device Control 3 (often XOFF)
001 0100	024	20	14	DC4			␔	^T		Device Control 4
001 0101	025	21	15	ERR	NAK		␕	^U		Negative Acknowledgement
001 0110	026	22	16	SYNC	SYN		␖	^V		Synchronous Idle
001 0111	027	23	17	LEM	ETB		␗	^W		End of Transmission Block

001 1000	030	24	18	S0	CAN		␘	^X		Cancel
001 1001	031	25	19	S1	EM		␙	^Y		End of Medium
001 1010	032	26	1A	S2	SS	SUB	␚	^Z		Substitute
001 1011	033	27	1B	S3	ESC		␛	^[	\e[i]	Escape[j]
001 1100	034	28	1C	S4	FS		␜	^\		File Separator
001 1101	035	29	1D	S5	GS		␝	^]		Group Separator
001 1110	036	30	1E	S6	RS		␞	^^[k]		Record Separator
001 1111	037	31	1F	S7	US		␟	^_		Unit Separator
111 1111	177	127	7F	DEL			␡	^?		Delete[l][f]

Other representations might be used by specialist equipment, for example ISO 2047 graphics or hexadecimal numbers.

Printable characters

Codes 20_{hex} to $7E_{hex}$, known as the printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols. There are 95 printable characters in total.[m]

Code 20_{hex}, the "space" character, denotes the space between words, as produced by the space bar of a keyboard. Since the space character is considered an invisible graphic (rather than a control character)[3]:223[46] it is listed in the table below instead of in the previous section.

Code $7F_{hex}$ corresponds to the non-printable "delete" (DEL) control character and is therefore omitted from this chart; it is covered in the previous section's chart. Earlier versions of ASCII used the up arrow instead of the caret ($5E_{hex}$) and the left arrow instead of the underscore ($5F_{hex}$).[5][47]

Binary	Oct	Dec	Hex	Glyph		
				1963	1965	1967
010 0000	040	32	20	space		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043	35	23	#		
010 0100	044	36	24	$		
010 0101	045	37	25	%		
010 0110	046	38	26	&		
010 0111	047	39	27	'		
010 1000	050	40	28	(		
010 1001	051	41	29	)		
010 1010	052	42	2A	*		
010 1011	053	43	2B	+		
010 1100	054	44	2C	,		
010 1101	055	45	2D	-		
010 1110	056	46	2E	.		
010 1111	057	47	2F	/		
011 0000	060	48	30	0		
011 0001	061	49	31	1		
011 0010	062	50	32	2		
011 0011	063	51	33	3		
011 0100	064	52	34	4		
011 0101	065	53	35	5		
011 0110	066	54	36	6		
011 0111	067	55	37	7		
011 1000	070	56	38	8		
011 1001	071	57	39	9		
011 1010	072	58	3A	:		
011 1011	073	59	3B	;		
011 1100	074	60	3C	<		
011 1101	075	61	3D	=		
011 1110	076	62	3E	>		
011 1111	077	63	3F	?		
100 0000	100	64	40	@	`	@
100 0001	101	65	41	A		
100 0010	102	66	42	B		
100 0011	103	67	43	C		
100 0100	104	68	44	D		
100 0101	105	69	45	E		
100 0110	106	70	46	F		
100 0111	107	71	47	G		
100 1000	110	72	48	H		
100 1001	111	73	49	I		

100 1010	112	74	4A	J		
100 1011	113	75	4B	K		
100 1100	114	76	4C	L		
100 1101	115	77	4D	M		
100 1110	116	78	4E	N		
100 1111	117	79	4F	O		
101 0000	120	80	50	P		
101 0001	121	81	51	Q		
101 0010	122	82	52	R		
101 0011	123	83	53	S		
101 0100	124	84	54	T		
101 0101	125	85	55	U		
101 0110	126	86	56	V		
101 0111	127	87	57	W		
101 1000	130	88	58	X		
101 1001	131	89	59	Y		
101 1010	132	90	5A	Z		
101 1011	133	91	5B	[		
101 1100	134	92	5C	\	~	\
101 1101	135	93	5D	]		
101 1110	136	94	5E	↑	^	
101 1111	137	95	5F	←	_	
110 0000	140	96	60		@	`
110 0001	141	97	61		a	
110 0010	142	98	62		b	
110 0011	143	99	63		c	
110 0100	144	100	64		d	
110 0101	145	101	65		e	
110 0110	146	102	66		f	
110 0111	147	103	67		g	
110 1000	150	104	68		h	
110 1001	151	105	69		i	
110 1010	152	106	6A		j	
110 1011	153	107	6B		k	
110 1100	154	108	6C		l	
110 1101	155	109	6D		m	
110 1110	156	110	6E		n	
110 1111	157	111	6F		o	
111 0000	160	112	70		p	
111 0001	161	113	71		q	
111 0010	162	114	72		r	
111 0011	163	115	73		s	
111 0100	164	116	74		t	
111 0101	165	117	75		u	

111 0110	166	118	76		v	
111 0111	167	119	77		w	
111 1000	170	120	78		x	
111 1001	171	121	79		y	
111 1010	172	122	7A		z	
111 1011	173	123	7B		{	
111 1100	174	124	7C	ACK	¬	|
111 1101	175	125	7D		}	
111 1110	176	126	7E	ESC	|	~

Character set

Points which represented a different character in previous versions (the 1963 version and/or the 1965 draft) are shown boxed. Points assigned since the 1963 version but otherwise unchanged are shown lightly shaded relative to their legend colours.

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_ 0	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_ 16	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_ 32	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_ 48	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ 64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ 80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6_ 96	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ 112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Letter Number Punctuation Symbol Other Undefined Character changed from 1963 version and/or 1965 draft

Use

ASCII was first used commercially during 1963 as a seven-bit teleprinter code for American Telephone & Telegraph's TWX (TeletypeWriter eXchange) network. TWX originally used the earlier five-bit ITA2, which was also used by the competing Telex teleprinter system. Bob Bemer introduced features such as the escape sequence.[4] His British colleague Hugh

McGregor Ross helped to popularize this work – according to Bemer, "so much so that the code that was to become ASCII was first called the *Bemer–Ross Code* in Europe".[48] Because of his extensive work on ASCII, Bemer has been called "the father of ASCII".[49]

On March 11, 1968, U.S. President Lyndon B. Johnson mandated that all computers purchased by the United States Federal Government support ASCII, stating:[50][51][52]

> I have also approved recommendations of the Secretary of Commerce [Luther H. Hodges] regarding standards for recording the Standard Code for Information Interchange on magnetic tapes and paper tapes when they are used in computer operations. All computers and related equipment configurations brought into the Federal Government inventory on and after July 1, 1969, must have the capability to use the Standard Code for Information Interchange and the formats prescribed by the magnetic tape and paper tape standards when these media are used.

ASCII was the most common character encoding on the World Wide Web until December 2007, when UTF-8 encoding surpassed it; UTF-8 is backward compatible with ASCII.[53][54][55]

Variants and derivations

As computer technology spread throughout the world, different standards bodies and corporations developed many variations of ASCII to facilitate the expression of non-English languages that used Roman-based alphabets. One could class some of these variations as "ASCII extensions", although some misuse that term to represent all variants, including those that do not preserve ASCII's character-map in the 7-bit range. Furthermore, the ASCII extensions have also been mislabelled as ASCII.

7-bit codes

From early in its development,[56] ASCII was intended to be just one of several national variants of an international character code standard.

Other international standards bodies have ratified character encodings such as ISO 646 (1967) that are identical or nearly identical to ASCII, with extensions for characters outside the English alphabet and symbols used outside the United States, such as the symbol for the United Kingdom's pound sterling (£). Almost every country needed an adapted version of ASCII, since ASCII suited the needs of only the US and a few other countries. For example, Canada had its own version that supported French characters.

Many other countries developed variants of ASCII to include non-English letters (e.g. é, ñ, ß, Ł), currency symbols (e.g. £, ¥), etc. See also YUSCII (Yugoslavia).

It would share most characters in common, but assign other locally useful characters to several code points reserved for "national use". However, the four years that elapsed between the publication of ASCII-1963 and ISO's first acceptance of an international recommendation during 1967[57] caused ASCII's choices for the national use characters to seem to be de facto standards for the world, causing confusion and incompatibility once other countries did begin to make their own assignments to these code points.

ISO/IEC 646, like ASCII, is a 7-bit character set. It does not make any additional codes available, so the same code points encoded different characters in different countries. Escape codes were defined to indicate which national variant applied to a piece of text, but they were rarely used, so it was often impossible to know what variant to work with and, therefore, which character a code represented, and in general, text-processing systems could cope with only one variant anyway.

Because the bracket and brace characters of ASCII were assigned to "national use" code points that were used for accented letters in other national variants of ISO/IEC 646, a German, French, or Swedish, etc. programmer using their national variant of ISO/IEC 646, rather than ASCII, had to write, and thus read, something such as

```
ä aÄiÜ = 'Ön'; ü
```

instead of

```
{ a[i] = '\n'; }
```

C trigraphs were created to solve this problem for ANSI C, although their late introduction and inconsistent implementation in compilers limited their use. Many programmers kept their computers on US-ASCII, so plain-text in Swedish, German etc. (for example, in e-mail or Usenet) contained "{, }" and similar variants in the middle of words, something those programmers got used to. For example, a Swedish programmer mailing another programmer asking if they should go for lunch, could get "N{ jag har sm|rg}sar" as the answer, which should be "Nä jag har smörgåsar" meaning "No I've got sandwiches".

8-bit codes

Eventually, as 8-, 16- and 32-bit (and later 64-bit) computers began to replace 12-, 18- and 36-bit computers as the norm, it became common to use an 8-bit byte to store each character in memory, providing an opportunity for extended, 8-bit relatives of ASCII. In most cases these developed as true extensions of ASCII, leaving the original character-mapping intact, but adding additional character definitions after the first 128 (i.e., 7-bit) characters.

Encodings include ISCII (India), VISCII (Vietnam). Although these encodings are sometimes referred to as ASCII, true ASCII is defined strictly only by the ANSI standard.

Most early home computer systems developed their own 8-bit character sets containing line-drawing and game glyphs, and often filled in some or all of the control characters from 0 to 31 with more graphics. Kaypro CP/M computers used the "upper" 128 characters for the Greek alphabet.

The PETSCII code Commodore International used for their 8-bit systems is probably unique among post-1970 codes in being based on ASCII-1963, instead of the more common ASCII-1967, such as found on the ZX Spectrum computer. Atari 8-bit computers and Galaksija computers also used ASCII variants.

The IBM PC defined code page 437, which replaced the control characters with graphic symbols such as smiley faces, and mapped additional graphic characters to the upper 128 positions. Operating systems such as DOS supported these code pages, and manufacturers of IBM PCs supported them in hardware. Digital Equipment Corporation developed the Multinational Character Set (DEC-MCS) for use in the popular VT220 terminal as one of the first extensions designed more for international languages than for block graphics. The Macintosh defined Mac OS Roman and Postscript also defined a set, both of these contained both international letters and typographic punctuation marks instead of graphics, more like modern character sets.

The ISO/IEC 8859 standard (derived from the DEC-MCS) finally provided a standard that most systems copied (at least as accurately as they copied ASCII, but with many substitutions). A popular further extension designed by Microsoft, Windows-1252 (often mislabeled as ISO-8859-1), added the typographic punctuation marks needed for traditional text printing. ISO-8859-1, Windows-1252, and the original 7-bit ASCII were the most common character encodings until 2008 when UTF-8 became more common.[54]

ISO/IEC 4873 introduced 32 additional control codes defined in the 80–9F hexadecimal range, as part of extending the 7-bit ASCII encoding to become an 8-bit system.[58]

Unicode

Unicode and the ISO/IEC 10646 Universal Character Set (UCS) have a much wider array of characters and their various encoding forms have begun to supplant ISO/IEC 8859 and ASCII rapidly in many environments. While ASCII is limited to 128 characters, Unicode and the UCS support more characters by separating the concepts of unique identification (using natural numbers called *code points*) and encoding (to 8-, 16- or 32-bit binary formats, called UTF-8, UTF-16 and UTF-32).

ASCII was incorporated into the Unicode (1991) character set as the first 128 symbols, so the 7-bit ASCII characters have the same numeric codes in both sets. This allows UTF-8 to be backward compatible with 7-bit ASCII, as a UTF-8 file containing only ASCII characters is identical to an ASCII file containing the same sequence of characters. Even more importantly, forward compatibility is ensured as software that recognizes only 7-bit ASCII characters as special and does not alter bytes with the highest bit set (as is often done to support 8-bit ASCII extensions such as ISO-8859-1) will preserve UTF-8 data unchanged.[59]

See also

- 3568 ASCII, an asteroid named after the character encoding

- Alt codes
- Ascii85
- ASCII art
- ASCII Ribbon Campaign
- Basic Latin (Unicode block) (ASCII as a subset of Unicode)
- Extended ASCII
- HTML decimal character rendering
- Jargon File, a glossary of computer programmer slang which includes a list of common slang names for ASCII characters
- List of computer character sets
- List of Unicode characters

Notes

a. The 128 characters of the 7-bit ASCII character set are divided into eight 16-character groups called *sticks* 0–7, associated with the three most-significant bits.[14] Depending on the horizontal or vertical representation of the character map, *sticks* correspond with either table rows or columns.
b. The Unicode characters from the area U+2400 to U+2421 reserved for representing control characters when it is necessary to print or display them rather than have them perform their intended function. Some browsers may not display these properly.
c. Caret notation is often used to represent control characters on a terminal. On most text terminals, holding down the Ctrl key while typing the second character will type the control character. Sometimes the shift key is not needed, for instance ^@ may be typable with just Ctrl and 2.
d. Character escape sequences in C programming language and many other languages influenced by it, such as Java and Perl (though not all implementations necessarily support all escape sequences).
e. The Backspace character can also be entered by pressing the ← Backspace key on some systems.
f. The ambiguity of Backspace is due to early terminals designed assuming the main use of the keyboard would be to manually punch paper tape while not connected to a computer. To delete the previous character, one had to back up the paper tape punch, which for mechanical and simplicity reasons was a button on the punch itself and not the keyboard, then type the rubout character. They therefore placed a key producing rubout at the location used on typewriters for backspace. When systems used these terminals and provided command-line editing, they had to use the "rubout" code to perform a backspace, and often did not interpret the backspace character (they might echo "`^H`" for backspace). Other terminals not designed for paper tape made the key at this location produce Backspace, and systems designed for these used that character to back up. Since the delete code often produced a backspace effect, this also forced terminal manufacturers to make any Delete key produce something other than the Delete character.
g. The Tab character can also be entered by pressing the Tab ↹ key on most systems.
h. The Carriage Return character can also be entered by pressing the ↵ Enter or Return key on most systems.
i. The `\e` escape sequence is not part of ISO C and many other language specifications. However, it is understood by several compilers, including GCC.
j. The Escape character can also be entered by pressing the Esc key on some systems.
k. ^^ means Ctrl + ^ (pressing the "Ctrl" and caret keys).
l. The Delete character can sometimes be entered by pressing the ← Backspace key on some systems.
m. Printed out, the characters are:

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

References

1. ANSI (1975-12-01). *ISO-IR-006: ASCII Graphic character set* (https://www.itscj.ipsj.or.jp/iso-ir/006.pdf) (PDF). ITSCJ/IPSJ.
2. "Character Sets" (https://www.iana.org/assignments/character-sets). *Internet Assigned Numbers Authority (IANA)*. 2007-05-14. Retrieved 2019-08-25.

3. Mackenzie, Charles E. (1980). *Coded Character Sets, History and Development* (https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf) (PDF). *The Systems Programming Series* (1 ed.). Addison-Wesley Publishing Company, Inc. pp. 6, 66, 211, 215, 217, 220, 223, 228, 236–238, 243–245, 247–253, 423, 425–428, 435–439. ISBN 978-0-201-14460-4. LCCN 77-90165 (https://lccn.loc.gov/77-90165). Archived (https://web.archive.org/web/20160526172151/https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf) (PDF) from the original on May 26, 2016. Retrieved August 25, 2019.
4. Brandel, Mary (1999-07-06). "1963: The Debut of ASCII" (http://edition.cnn.com/TECH/computing/9907/06/1963.idg/). CNN. Archived (https://web.archive.org/web/20130617155922/http://edition.cnn.com/TECH/computing/9907/06/1963.idg/) from the original on 2013-06-17. Retrieved 2008-04-14.
5. "American Standard Code for Information Interchange, ASA X3.4-1963" (https://www.sr-ix.com/Archive/CharCodeHist/X3.4-1963/index.html). American Standards Association (ASA). 1963-06-17. Retrieved 2020-06-06.
6. "USA Standard Code for Information Interchange, USAS X3.4-1967". United States of America Standards Institute (USASI). 1967-07-07.
7. Jennings, Thomas Daniel (2016-04-20) [1999]. "An annotated history of some character codes or ASCII: American Standard Code for Information Infiltration" (https://www.sr-ix.com/Archive/CharCodeHist/index.html#ASCII-1967). *Sensitive Research (SR-IX)*. Retrieved 2020-03-08.
8. "American National Standard for Information Systems — Coded Character Sets — 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986". American National Standards Institute (ANSI). 1986-03-26.
9. Vint Cerf (1969-10-16). *ASCII format for Network Interchange* (https://tools.ietf.org/html/rfc20). IETF. doi:10.17487/RFC0020 (https://doi.org/10.17487%2FRFC0020). RFC 20 (https://tools.ietf.org/html/rfc20).
10. Barry Leiba (2015-01-12). "Correct classification of RFC 20 (ASCII format) to Internet Standard" (https://datatracker.ietf.org/doc/status-change-rfc20-ascii-format-to-standard/). IETF.
11. Shirley, R. (August 2007), *Internet Security Glossary, Version 2* (https://tools.ietf.org/html/rfc4949), RFC 4949 (https://tools.ietf.org/html/rfc4949), archived (https://web.archive.org/web/20160613203815/https://tools.ietf.org/html/rfc4949) from the original on 2016-06-13, retrieved 2016-06-13
12. Maini, Anil Kumar (2007). *Digital Electronics: Principles, Devices and Applications* (https://books.google.com/?id=NQSpNAEACAAJ&pg=PA28). John Wiley and Sons. p. 28. ISBN 978-0-470-03214-5. "In addition, it defines codes for 33 nonprinting, mostly obsolete control characters that affect how the text is processed."
13. Bukstein, Ed (July 1964). "Binary Computer Codes and ASCII" (https://web.archive.org/web/20160303180933/http://www.swtpc.com/mholley/ElectronicsWorld/Jul1964/EW_Jul1964.htm). *Electronics World*. **72** (1): 28–29. Archived from the original (http://www.swtpc.com/mholley/ElectronicsWorld/Jul1964/EW_Jul1964.htm) on 2016-03-03. Retrieved 2016-05-22.
14. Bemer, Robert William (1980). "Chapter 1: Inside ASCII" (https://web.archive.org/web/20160827000956/http://dlx.bookzz.org/genesis/772000/c80a62495acf1e1a5b966de23c1f989a/_as/%5BInterface_Age_Staff%5D_Best_of_Interface_Age%2C_Volum%28BookZZ.org%29.pdf) (PDF). *General Purpose Software*. Best of Interface Age. **2**. Portland, OR, USA: dilithium Press. pp. 1–50. ISBN 978-0-918398-37-6. LCCN 79-67462 (https://lccn.loc.gov/79-67462). Archived from the original (http://bookzz.org/dl/1210234/1105c6) on 2016-08-27. Retrieved 2016-08-27, from:
    - Bemer, Robert William (May 1978). "Inside ASCII – Part I". *Interface Age*. **3** (5): 96–102.
    - Bemer, Robert William (June 1978). "Inside ASCII – Part II". *Interface Age*. **3** (6): 64–74.
    - Bemer, Robert William (July 1978). "Inside ASCII – Part III". *Interface Age*. **3** (7): 80–87.
15. Brief Report: Meeting of CCITT Working Party on the New Telegraph Alphabet, May 13–15, 1963.
16. Report of ISO/TC/97/SC 2 – Meeting of October 29–31, 1963.
17. Report on Task Group X3.2.4, June 11, 1963, Pentagon Building, Washington, DC.
18. Report of Meeting No. 8, Task Group X3.2.4, December 17 and 18, 1963
19. Winter, Dik T. (2010) [2003]. "US and International standards: ASCII" (https://web.archive.org/web/20100116001012/http://homepages.cwi.nl/~dik/english/codes/stand.html#ascii). Archived from the original (http://homepages.cwi.nl/~dik/english/codes/stand.html#ascii) on 2010-01-16.
20. Salste, Tuomas (January 2016). "7-bit character sets: Revisions of ASCII" (http://www.aivosto.com/vbtips/charsets-7bit.html#body). Aivosto Oy. urn:nbn:fi-fe201201011004 (http://nbn-resolving.de/urn:nbn:fi-fe201201011004). Archived (https://web.archive.org/web/20160613145224/http://www.aivosto.com/vbtips/charsets-7bit.html#body) from the original on 2016-06-13. Retrieved 2016-06-13.
21. "Information". *Scientific American* (special edition). **215** (3). September 1966. JSTOR e24931041 (https://www.jstor.org/stable/e24931041).
22. Korpela, Jukka K. (2014-03-14) [2006-06-07]. *Unicode Explained – Internationalize Documents, Programs, and Web Sites* (2nd release of 1st ed.). O'Reilly Media, Inc. p. 118. ISBN 978-0-596-10121-3.

23. *ANSI INCITS 4-1986 (R2007): American National Standard for Information Systems – Coded Character Sets – 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)* (http://sliderule.mraiow.com/w/images/7/73/ASCII.pdf) (PDF), 2007 [1986], archived (https://web.archive.org/web/20140207170900/http://sliderule.mraiow.com/w/images/7/73/ASCII.pdf) (PDF) from the original on 2014-02-07, retrieved 2016-06-12
24. "INCITS 4-1986[R2012]: Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)" (https://standards.incits.org/apps/group_public/project/details.php?project_id=796). 2012-06-15. Archived (https://web.archive.org/web/20200228222040/https://standards.incits.org/apps/group_public/project/details.php?project_id=796) from the original on 2020-02-28. Retrieved 2020-02-28.
25. "INCITS 4-1986[R2017]: Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)" (https://standards.incits.org/apps/group_public/project/details.php?project_id=1829). 2017-11-02 [2017-06-09]. Archived (https://web.archive.org/web/20200228221645/https://standards.incits.org/apps/group_public/project/details.php?project_id=1829) from the original on 2020-02-28. Retrieved 2020-02-28.
26. *Bit Sequencing of the American National Standard Code for Information Interchange in Serial-by-Bit Data Transmission*, American National Standards Institute (ANSI), 1966, X3.15-1966
27. "BruXy: Radio Teletype communication" (http://bruxy.regnet.cz/web/hamradio/EN/radio-teletype-communication/). 2005-10-10. Archived (https://web.archive.org/web/20160412130035/http://bruxy.regnet.cz/web/hamradio/EN/radio-teletype-communication/) from the original on 2016-04-12. Retrieved 2016-05-09. "The transmitted code use International Telegraph Alphabet No. 2 (ITA-2) which was introduced by CCITT in 1924."
28. Smith, Gil (2001). "Teletype Communication Codes" (http://www.baudot.net/docs/smith--teletype-codes.pdf) (PDF). Baudot.net. Archived (https://web.archive.org/web/20080820043949/http://www.baudot.net/docs/smith--teletype-codes.pdf) (PDF) from the original on 2008-08-20. Retrieved 2008-07-11.
29. Sawyer, Stanley A.; Krantz, Steven George (1995). *A TeX Primer for Scientists* (https://books.google.com/books?id=bXLDwmIJNkUC&pg=PA13). CRC Press, LLC. p. 13. Bibcode:1995tps..book.....S (https://ui.adsabs.harvard.edu/abs/1995tps..book.....S). ISBN 978-0-8493-7159-2. Archived (https://web.archive.org/web/20161222151907/https://books.google.com/books?id=bXLDwmIJNkUC&pg=PA13) from the original on 2016-12-22. Retrieved 2016-10-29.
30. Savard, John J. G. "Computer Keyboards" (http://www.quadibloc.com/comp/kybint.htm). Archived (https://web.archive.org/web/20140924183236/http://www.quadibloc.com/comp/kybint.htm) from the original on 2014-09-24. Retrieved 2014-08-24.
31. "ASCIIbetical definition" (https://www.pcmag.com/encyclopedia_term/0,2542,t=ASCIIbetical&i=38025,00.asp). *PC Magazine*. Archived (https://web.archive.org/web/20130309183509/http://www.pcmag.com/encyclopedia_term/0%2C2542%2Ct%3DASCIIbetical%26i%3D38025%2C00.asp) from the original on 2013-03-09. Retrieved 2008-04-14.
32. Resnick, P. (April 2001), *Internet Message Format* (https://tools.ietf.org/html/rfc2822), RFC 2822 (https://tools.ietf.org/html/rfc2822), archived (https://web.archive.org/web/20160613203505/https://tools.ietf.org/html/rfc2822) from the original on 2016-06-13, retrieved 2016-06-13 (NB. NO-WS-CTL.)
33. McConnell, Robert; Haynes, James; Warren, Richard. "Understanding ASCII Codes" (http://www.nadcomm.com/ascii_code.htm). Archived (https://web.archive.org/web/20140227190425/http://www.nadcomm.com/ascii_code.htm) from the original on 2014-02-27. Retrieved 2014-05-11.
34. Barry Margolin (2014-05-29). "Re: editor and word processor history (was: Re: RTF for emacs)" (http://lists.gnu.org/archive/html/help-gnu-emacs/2014-05/msg00448.html). *help-gnu-emacs* (Mailing list). Archived (https://web.archive.org/web/20140714133149/http://lists.gnu.org/archive/html/help-gnu-emacs/2014-05/msg00448.html) from the original on 2014-07-14. Retrieved 2014-07-11.
35. "PDP-6 Multiprogramming System Manual" (http://bitsavers.trailing-edge.com/pdf/dec/pdp6/DEC-6-0-EX-SYS-UM-IP-PRE00_Multiprogramming_System_Manual_1965.pdf) (PDF). Digital Equipment Corporation (DEC). 1965. p. 43. Archived (https://web.archive.org/web/20140714140253/http://bitsavers.trailing-edge.com/pdf/dec/pdp6/DEC-6-0-EX-SYS-UM-IP-PRE00_Multiprogramming_System_Manual_1965.pdf) (PDF) from the original on 2014-07-14. Retrieved 2014-07-10.
36. "PDP-10 Reference Handbook, Book 3, Communicating with the Monitor" (http://bitsavers.org/pdf/dec/pdp10/1970_PDP-10_Ref/1970PDP10Ref_Part3.pdf) (PDF). Digital Equipment Corporation (DEC). 1969. p. 5-5. Archived (https://web.archive.org/web/20111115083418/http://www.bitsavers.org/pdf/dec/pdp10/1970_PDP-10_Ref/1970PDP10Ref_Part3.pdf) (PDF) from the original on 2011-11-15. Retrieved 2014-07-10.
37. "Help - GNU Emacs Manual" (https://www.gnu.org/software/emacs/manual/html_node/emacs/Help.html). Archived (https://web.archive.org/web/20180711223750/https://www.gnu.org/software/emacs/manual/html_node/emacs/Help.html) from the original on 2018-07-11. Retrieved 2018-07-11.

38. Tim Paterson (2007-08-08). "Is DOS a Rip-Off of CP/M?" (http://dosmandrivel.blogspot.com/2007/08/is-dos-rip-off-of-cpm.html). *DosMan Drivel*. Archived (https://web.archive.org/web/20180420075137/http://dosmandrivel.blogspot.com/2007/08/is-dos-rip-off-of-cpm.html) from the original on 2018-04-20. Retrieved 2018-04-19.
39. Ossanna, J. F.; Saltzer, J. H. (November 17–19, 1970). "Technical and human engineering problems in connecting terminals to a time-sharing system" (http://www.multicians.org/jhs-jfo-terminals.pdf) (PDF). *Proceedings of the November 17–19, 1970, Fall Joint Computer Conference (FJCC)*. p. 357: AFIPS Press. pp. 355–362. Archived (https://web.archive.org/web/20120819085101/http://www.multicians.org/jhs-jfo-terminals.pdf) (PDF) from the original on 2012-08-19. Retrieved 2013-01-29. "Using a "new-line" function (combined carriage-return and line-feed) is simpler for both man and machine than requiring both functions for starting a new line; the American National Standard X3.4-1968 permits the line-feed code to carry the new-line meaning."
40. O'Sullivan, T. (1971-05-19), *TELNET Protocol* (https://tools.ietf.org/html/rfc158), Internet Engineering Task Force (IETF), pp. 4–5, RFC 158 (https://tools.ietf.org/html/rfc158), archived (https://web.archive.org/web/20160613202223/https://tools.ietf.org/html/rfc158) from the original on 2016-06-13, retrieved 2013-01-28
41. Neigus, Nancy J. (1973-08-12), *File Transfer Protocol* (https://tools.ietf.org/html/rfc542), Internet Engineering Task Force (IETF), RFC 542 (https://tools.ietf.org/html/rfc542), archived (https://web.archive.org/web/20160613202314/https://tools.ietf.org/html/rfc542) from the original on 2016-06-13, retrieved 2013-01-28
42. Postel, Jon (June 1980), *File Transfer Protocol* (https://tools.ietf.org/html/rfc765), Internet Engineering Task Force (IETF), RFC 765 (https://tools.ietf.org/html/rfc765), archived (https://web.archive.org/web/20160613202356/https://tools.ietf.org/html/rfc765) from the original on 2016-06-13, retrieved 2013-01-28
43. "EOL translation plan for Mercurial" (https://www.mercurial-scm.org/wiki/EOLTranslationPlan). Mercurial. Archived (https://web.archive.org/web/20160616235536/https://www.mercurial-scm.org/wiki/EOLTranslationPlan) from the original on 2016-06-16. Retrieved 2017-06-24.
44. Bernstein, Daniel J. "Bare LFs in SMTP" (http://cr.yp.to/docs/smtplf.html). Archived (https://web.archive.org/web/20111029013105/http://cr.yp.to/docs/smtplf.html) from the original on 2011-10-29. Retrieved 2013-01-28.
45. *CP/M 1.4 Interface Guide* (http://www.bitsavers.org/pdf/digitalResearch/cpm/1.4/CPM_1.4_Interface_Guide_1978.pdf) (PDF). Digital Research. 1978. p. 10. Archived (https://web.archive.org/web/20190529055800/http://bitsavers.org/pdf/digitalResearch/cpm/1.4/CPM_1.4_Interface_Guide_1978.pdf) (PDF) from the original on 2019-05-29. Retrieved 2017-10-07.
46. Cerf, Vinton Gray (1969-10-16), *ASCII format for Network Interchange* (https://tools.ietf.org/html/rfc20), Network Working Group, RFC 20 (https://tools.ietf.org/html/rfc20), archived (https://web.archive.org/web/20160613203624/https://tools.ietf.org/html/rfc20) from the original on 2016-06-13, retrieved 2016-06-13 (NB. Almost identical wording to USAS X3.4-1968 except for the intro.)
47. Haynes, Jim (2015-01-13). "First-Hand: Chad is Our Most Important Product: An Engineer's Memory of Teletype Corporation" (http://ethw.org/First-Hand:Chad_is_Our_Most_Important_Product:_An_Engineer's_Memory_of_Teletype_Corporation). Engineering and Technology History Wiki (ETHW). Archived (https://web.archive.org/web/20161031223347/http://ethw.org/First-Hand%3AChad_is_Our_Most_Important_Product%3A_An_Engineer%27s_Memory_of_Teletype_Corporation) from the original on 2016-10-31. Retrieved 2016-10-31. "There was the change from 1961 ASCII to 1968 ASCII. Some computer languages used characters in 1961 ASCII such as up arrow and left arrow. These characters disappeared from 1968 ASCII. We worked with Fred Mocking, who by now was in Sales at Teletype, on a type cylinder that would compromise the changing characters so that the meanings of 1961 ASCII were not totally lost. The underscore character was made rather wedge-shaped so it could also serve as a left arrow."
48. Bemer, Robert William. "Bemer meets Europe (Computer Standards) – Computer History Vignettes" (https://web.archive.org/web/20131017062722/http://www.trailing-edge.com/~bobbemer/EUROPE.HTM). Trailing-edge.com. Archived from the original (http://www.trailing-edge.com/~bobbemer/EUROPE.HTM) on 2013-10-17. Retrieved 2008-04-14. (NB. Bemer was employed at IBM at that time.)
49. "Robert William Bemer: Biography" (http://www.thocp.net/biographies/bemer_bob.htm). 2013-03-09. Archived (https://web.archive.org/web/20160616081929/http://www.thocp.net/biographies/bemer_bob.htm) from the original on 2016-06-16.
50. Johnson, Lyndon Baines (1968-03-11). "Memorandum Approving the Adoption by the Federal Government of a Standard Code for Information Interchange" (http://www.presidency.ucsb.edu/ws/index.php?pid=28724). The American Presidency Project. Archived (https://web.archive.org/web/20070914121230/http://www.presidency.ucsb.edu/ws/index.php?pid=28724) from the original on 2007-09-14. Retrieved 2008-04-14.
51. Richard S. Shuford (1996-12-20). "Re: Early history of ASCII?" (https://groups.google.com/d/msg/alt.folklore.computers/gbg5YVFaT48/wIVFfJ2j4hYJ). Newsgroup: alt.folklore.computers (news:alt.folklore.computers). Usenet: Pine.SUN.3.91.961220100220.13180C-100000@duncan.cs.utk.edu (news:Pine.SUN.3.91.961220100220.13180C-100000@duncan.cs.utk.edu).
52. Folts, Harold C.; Karp, Harry, eds. (1982-02-01). *Compilation of Data Communications Standards* (2nd revised ed.). McGraw-Hill Inc. ISBN 978-0-07-021457-6.

53. Dubost, Karl (2008-05-06). "UTF-8 Growth on the Web" (http://www.w3.org/QA/2008/05/utf8-web-growth.html). *W3C Blog*. World Wide Web Consortium. Archived (https://web.archive.org/web/20160616084132/https://www.w3.org/blog/2008/05/utf8-web-growth/) from the original on 2016-06-16. Retrieved 2010-08-15.
54. Davis, Mark (2008-05-05). "Moving to Unicode 5.1" (http://googleblog.blogspot.com/2008/05/moving-to-unicode-51.html). *Official Google Blog*. Archived (https://web.archive.org/web/20160616084637/https://googleblog.blogspot.de/2008/05/moving-to-unicode-51.html) from the original on 2016-06-16. Retrieved 2010-08-15.
55. Davis, Mark (2010-01-28). "Unicode nearing 50% of the web" (http://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html). *Official Google Blog*. Archived (https://web.archive.org/web/20160616085323/https://googleblog.blogspot.de/2010/01/unicode-nearing-50-of-web.html) from the original on 2016-06-16. Retrieved 2010-08-15.
56. "Specific Criteria", attachment to memo from R. W. Reach, "X3-2 Meeting – September 14 and 15", September 18, 1961
57. Maréchal, R. (1967-12-22), *ISO/TC 97 – Computers and Information Processing: Acceptance of Draft ISO Recommendation No. 1052*
58. The Unicode Consortium (2006-10-27). "Chapter 13: Special Areas and Format Characters" (http://unicode.org/book/ch13.pdf) (PDF). In Allen, Julie D. (ed.). *The Unicode standard, Version 5.0*. Upper Saddle River, New Jersey, US: Addison-Wesley Professional. p. 314. ISBN 978-0-321-48091-0. Retrieved 2015-03-13.
59. "utf-8(7) – Linux manual page" (http://man7.org/linux/man-pages/man7/utf-8.7.html). Man7.org. 2014-02-26. Archived (https://web.archive.org/web/20140422232059/http://man7.org/linux/man-pages/man7/utf-8.7.html) from the original on 2014-04-22. Retrieved 2014-04-21.

Further reading

- Bemer, Robert William (1960). "A Proposal for Character Code Compatibility". *Communications of the ACM*. **3** (2): 71–72. doi:10.1145/366959.366961 (https://doi.org/10.1145%2F366959.366961).
- Bemer, Robert William (2003-05-23). "The Babel of Codes Prior to ASCII: The 1960 Survey of Coded Character Sets: The Reasons for ASCII" (https://web.archive.org/web/20131017062654/http://www.trailing-edge.com/~bobbemer/SURVEY.HTM). Archived from the original (http://www.trailing-edge.com/~bobbemer/SURVEY.HTM) on 2013-10-17. Retrieved 2016-05-09, from:
  - Bemer, Robert William (December 1960). "Survey of coded character representation". *Communications of the ACM*. **3** (12): 639–641. doi:10.1145/367487.367493 (https://doi.org/10.1145%2F367487.367493).
  - Smith, H. J.; Williams, F. A. (December 1960). "Survey of punched card codes" (https://doi.org/10.1145%2F367487.367491). *Communications of the ACM*. **3** (12): 642. doi:10.1145/367487.367491 (https://doi.org/10.1145%2F367487.367491).
- *American National Standard Code for Information Interchange*. American National Standards Institute. 1977.
- Robinson, G. S.; Cargill, C. (1996). "History and impact of computer standards". *Computer*. **29** (10): 79–85. doi:10.1109/2.539725 (https://doi.org/10.1109%2F2.539725).
- Mullendore, Ralph Elvin (1964) [1963]. Ptak, John F. (ed.). "On the Early Development of ASCII – The History of ASCII" (http://longstreet.typepad.com/thesciencebookstore/2012/03/heres-the-link.html). JF Ptak Science Books (published March 2012). Archived (https://web.archive.org/web/20160526181319/http://longstreet.typepad.com/thesciencebookstore/2012/03/heres-the-link.html) from the original on 2016-05-26. Retrieved 2016-05-26.

External links

- "C0 Controls and Basic Latin – Range: 0000–007F" (https://www.unicode.org/charts/PDF/U0000.pdf) (PDF). *The Unicode Standard 8.0*. Unicode, Inc. 2015 [1991]. Archived (https://web.archive.org/web/20160526182105/http://www.unicode.org/charts/PDF/U0000.pdf) (PDF) from the original on 2016-05-26. Retrieved 2016-05-26.
- Fischer, Eric. "The Evolution of Character Codes, 1874–1968". CiteSeerX 10.1.1.96.678 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.678). [1] (https://web.archive.org/web/20120213005708/http://www.transbay.net/~enf/ascii/ascii.pdf)