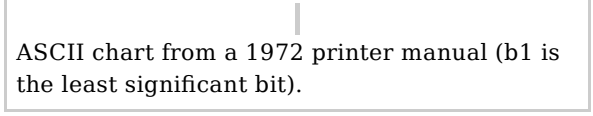
ASCII

From Wikipedia, the free encyclopedia



ASCII chart from a 1972 printer manual (b1 is the least significant bit).

ASCII (Listen[i]/ˈæski/ ***ASS**-kee*), abbreviated from **American Standard Code for Information Interchange**,[1] is a character-encoding scheme (the IANA prefers the name **US-ASCII**[2]). ASCII codes represent text in computers, communications equipment, and other devices that use text. Most modern character-encoding schemes are based on ASCII, though they support many additional characters. ASCII was the most common character encoding on the World Wide Web until December 2007, when it was surpassed by UTF-8, which is fully backward compatible to ASCII.[3][4][5]

ASCII was developed from telegraphic codes. 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) X3.2 subcommittee. The first edition of the standard was published during 1963,[6][7] underwent a major revision during 1967,[8][9] and experienced its most recent update during 1986.[10] 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.

Originally based on the English alphabet, ASCII encodes 128 specified characters into seven-bit integers as shown by the ASCII chart on the right.[11] The characters encoded are numbers *0* to *9*, lowercase letters *a* to *z*, uppercase letters *A* to *Z*, basic punctuation symbols, control codes that originated with Teletype machines, and a space. For example, lowercase *j* would become binary 1101010 and decimal 106. ASCII includes definitions for 128 characters: 33 are non-printing control characters (many now obsolete)[12] that affect how text and space are processed[13] and 95 printable characters, including the space (which is considered an invisible graphic[14][15]:223).

Contents

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. The ASA became the United States of America Standards Institute or USASI[15]:211 and ultimately the American National Standards Institute.

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,[16][17] 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.[17]

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.[15]: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.[15]: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.[18]

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 columns (32 positions) were reserved for control characters.[15]:220, 236 § 8,9) The "space" character had to come before graphics to make sorting easier, so it became position 20_{hex};[15]: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,[15]: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.[15]:238 § 18 The digits 0–9 were arranged so they correspond to values in binary prefixed with 011, 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.[19] 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 second column, rows 1–5, corresponding to the digits 1–5 in the adjacent column. 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 left, 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 more recently introduced 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.[15]: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.[15]:243–245

Publication

With the other special characters and control codes filled in, ASCII was published as ASA X3.4-1963, leaving 28 code positions without any assigned meaning, reserved for future standardization, and one unassigned control code.[15]:66, 245 There was some debate at the time whether there should be more control characters rather than the lowercase alphabet.[15]: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 columns 6 and 7,[20] and International Organization for Standardization TC 97 SC 2 voted during October to incorporate the change into its draft standard.[21] The X3.2.4 task group voted its approval for the change to ASCII at its May 1963 meeting.[22] Locating the lowercase letters in columns 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),[23] renaming some control characters (SOM became start of header (SOH)) and moving or removing others (RU was removed).[15]:247–248 ASCII was subsequently updated as USASI X3.4-1967, then USASI X3.4-1968, ANSI X3.4-1977, and finally, ANSI X3.4-1986 (the first two are occasionally retronamed ANSI X3.4-1967, and ANSI X3.4-1968).

The X3 committee also addressed how ASCII should be transmitted (least significant bit first), 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.

Use

ASCII itself 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.[6] 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".[24] Because of his extensive work on ASCII, Bemer has been called "the father of ASCII."[25]

On March 11, 1968, U.S. President Lyndon B. Johnson mandated that all computers purchased by the United States federal government support ASCII, stating:[26]

> I have also approved recommendations of the Secretary of Commerce 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.

Other standards

Other international standards bodies have ratified character encodings such as ISO/IEC 646 (1972) 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 USA and a few other countries. For example, Canada had its own version that supported French characters. Other adapted encodings include ISCII (India), VISCII (Vietnam), and YUSCII (Yugoslavia). Although these encodings are sometimes referred to as ASCII, true ASCII is defined strictly only by the ANSI standard.

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.[27]

ASCII 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 refers to control characters that do not include carriage return, line feed or white space as non-whitespace control characters.[28] 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 (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.[29]

Code 127 is officially named "delete" but the Teletype label was "rubout". Since the original standard did not give detailed interpretation for most control codes, interpretations of this code varied. The original Teletype meaning, and the intent of the standard, was to make it an ignored character, the same as NUL (all zeroes). This was useful specifically for paper tape, because punching the all-ones bit pattern on top of an existing mark would obliterate it.[30] Tapes designed to be "hand edited" could even be produced with spaces of extra NULs (blank tape) so that a block of characters could be "rubbed out" and then replacements put into the empty space.

Some software assigned special meanings to ASCII characters sent to the software from the terminal. Operating systems from Digital Equipment Corporation, for example, interpreted DEL as an input character as meaning "remove previously-typed input character",[31][32] and this interpretation also became common in Unix systems. Most other systems used BS for that meaning and used DEL to mean "remove the character at the cursor". That latter interpretation is the most common now.

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.

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 cloned RT-11 to create CP/M he followed established DEC convention. 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 a clone of CP/M, 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[33] 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 OS X operating system, they now use line feed (LF) as well.

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.[34] 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.[35][36] 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.[37][38]

Older operating systems such as TOPS-10, along with CP/M, tracked file length only in units of disk blocks and used Control-Z (SUB) to mark the end of the actual text in the file. For this reason, 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".

ASCII control code chart

Binary	Octal	Decimal	Hexadecimal	Abbreviation	Print form[a]	Caret notation[b]	Escape code[c]	Name
000 0000	000	0	00	NUL	$^{N}_{U}{}_{L}$	^@	\0	Null
000 0001	001	1	01	SOH	$^{S}_{O}{}_{H}$	^A		Start of Heading
000 0010	002	2	02	STX	$^{S}_{T}{}_{X}$	^B		Start of Text
000 0011	003	3	03	ETX	$^{E}_{T}{}_{X}$	^C		End of Text
000 0100	004	4	04	EOT	$^{E}_{O}{}_{T}$	^D		End of Transmission
000 0101	005	5	05	ENQ	$^{E}_{N}{}_{Q}$	^E		Enquiry
000 0110	006	6	06	ACK	$^{A}_{C}{}_{K}$	^F		Acknowledgement
000 0111	007	7	07	BEL	$^{B}_{E}{}_{L}$	^G	\a	Bell
000 1000	010	8	08	BS	$^{B}_{S}$	^H	\b	Backspace[d][e]
000 1001	011	9	09	HT	$^{H}_{T}$	^I	\t	Horizontal Tab[f]
000 1010	012	10	0A	LF	$^{L}_{F}$	^J	\n	Line Feed
000 1011	013	11	0B	VT	$^{V}_{T}$	^K	\v	Vertical Tab
000 1100	014	12	0C	FF	$^{F}_{F}$	^L	\f	Form Feed
000 1101	015	13	0D	CR	$^{C}_{R}$	^M	\r	Carriage Return[g]
000 1110	016	14	0E	SO	$^{S}_{S}$	^N		Shift Out
000 1111	017	15	0F	SI	$^{S}_{I}$	^O		Shift In
001 0000	020	16	10	DLE	$^{D}_{L}{}_{E}$	^P		Data Link Escape
001 0001	021	17	11	DC1	$^{D}_{C}{}_{1}$	^Q		Device Control 1 (oft. XON)
001 0010	022	18	12	DC2	$^{D}_{C}{}_{2}$	^R		Device Control 2
001 0011	023	19	13	DC3	$^{D}_{C}{}_{3}$	^S		Device Control 3 (oft. XOFF)
001 0100	024	20	14	DC4	$^{D}_{C}{}_{4}$	^T		Device Control 4
001 0101	025	21	15	NAK	$^{N}_{A}{}_{K}$	^U		Negative Acknowledgement
001 0110	026	22	16	SYN	$^{S}_{Y}{}_{N}$	^V		Synchronous Idle

001 0111	027	23	17	ETB	$^{E}_{T_B}$	^W		End of Transmission Block
001 1000	030	24	18	CAN	$^{C}_{A_N}$	^X		Cancel
001 1001	031	25	19	EM	$^{E}_{M}$	^Y		End of Medium
001 1010	032	26	1A	SUB	$^{S}_{U_B}$	^Z		Substitute
001 1011	033	27	1B	ESC	$^{E}_{S_C}$	^[	\e[h]	Escape[i]
001 1100	034	28	1C	FS	$^{F}_{S}$	^\		File Separator
001 1101	035	29	1D	GS	$^{G}_{S}$	^]		Group Separator
001 1110	036	30	1E	RS	$^{R}_{S}$	^^[j]		Record Separator
001 1111	037	31	1F	US	$^{U}_{S}$	^_		Unit Separator
111 1111	177	127	7F	DEL	$^{D}_{E_L}$	^?		Delete[k][e]

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

ASCII printable characters

Codes 20_{16} to $7E_{16}$, known as the printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols. There are 95 printable characters in total.[l]

Code 20_{16}, 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) [14][15]:223 it is listed in the table below instead of in the previous section.

Code $7F_{16}$ 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 ($5E_{16}$) and the left arrow instead of the underscore ($5F_{16}$).[39]

ASCII printable code chart

Binary	Oct	Dec	Hex	Glyph
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	?

Binary	Oct	Dec	Hex	Glyph
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	_

Binary	Oct	Dec	Hex	Glyph
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	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

Aliases

A June 1992 RFC[40] and the Internet Assigned Numbers Authority registry of character sets[2] recognize the following case-insensitive aliases for ASCII as suitable for use on the Internet: ANSI_X3.4-1968 (canonical name), iso-ir-6, ANSI_X3.4-1986, ISO_646.irv:1991, ASCII, ISO646-US, US-ASCII (preferred MIME name),[2] us, IBM367, cp367, and csASCII.

Of these, the IANA encourages use of the name "US-ASCII" for Internet uses of ASCII (even if it is a redundant acronym, but the US is needed because of abuse of the ASCII term). One often finds this in the optional "charset" parameter in the Content-Type header of some MIME messages, in the equivalent

"meta" element of some HTML documents, and in the encoding declaration part of the prologue of some XML documents.

Variants

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.

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

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.

7-bit

From early in its development,[41] ASCII was intended to be just one of several national variants of an international character code standard, ultimately published as ISO/IEC 646 (1972), which 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[42] 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, was a 7-bit character set. It did 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

Eventually, as 8-, 16- and 32-bit (and later 64-bit) computers began to replace 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.

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 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.[4]

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.[43]

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).

To allow backward compatibility, the 128 ASCII and 256 ISO-8859-1 (Latin 1) characters are assigned Unicode/UCS code points that are the same as their codes in the earlier standards. Therefore, ASCII can be considered a 7-bit encoding scheme for a very small subset of Unicode/UCS, and ASCII (when prefixed with 0 as the eighth bit) is valid UTF-8.

Order

ASCII-code order is also called *ASCIIbetical* order.[44] 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" comes before "a"
- Digits and many punctuation marks come before letters; for example, "4" precedes "one"
- Numbers are sorted naïvely as strings; for example, "10" precedes "2"

An intermediate order—readily implemented—converts uppercase letters to lowercase before comparing ASCII values. Naïve number sorting can be averted by zero-filling all numbers (e.g. "02" will sort before "10" as expected), although this is an external fix and has nothing to do with the ordering itself.

See also

- 3568 ASCII, an asteroid named after the character encoding
- Ascii85
- ASCII art

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

Notes

a. 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.
b. 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.
c. Character escape codes in C programming language and many other languages influenced by it, such as Java and Perl (though not all implementations necessarily support all escape codes).
d. The Backspace character can also be entered by pressing the ← Backspace key on some systems.
e. 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.
f. The Tab character can also be entered by pressing the Tab ↹ key on most systems.
g. The Carriage Return character can also be entered by pressing the ↵ Enter or Return key on most systems.
h. The \e escape sequence is not part of ISO C and many other language specifications. However, it is understood by several compilers, including GCC.
i. The Escape character can also be entered by pressing the Esc key on some systems.
j. ^^ means Ctrl+^ (pressing the "Ctrl" and caret keys).
k. The Delete character can sometimes be entered by pressing the ← Backspace key on some systems.
l. Printed out, the characters are:

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

References

1. "Pronunciation for ASCII". *Merriam Webster* (audio). Retrieved 2008-04-14.
2. Internet Assigned Numbers Authority (May 14, 2007). "Character Sets (http://www.iana.org/assignments/character-sets)". Accessed 2008-04-14.
3. Dubost, Karl (May 6, 2008). "UTF-8 Growth On The Web". *W3C Blog*. World Wide Web Consortium. Retrieved 2010-08-15.
4. Davis, Mark (May 5, 2008). "Moving to Unicode 5.1". *Official Google Blog*. Google. Retrieved 2010-08-15.
5. Davis, Mark (Jan 28, 2010). "Unicode nearing 50% of the web". *Official Google Blog*. Google. Retrieved 2010-08-15.
6. Brandel, Mary (July 6, 1999). "1963: The Debut of ASCII". CNN. Retrieved 2008-04-14.
7. "American Standard Code for Information Interchange, ASA X3.4-1963". American Standards Association. June 17, 1963. Retrieved 2014-05-23.
8. "USA Standard Code for Information Interchange, USAS X3.4-1967". United States of America Standards Institute. July 7, 1967.
9. Jennings, Tom. "An annotated history of some character codes". *World Power Systems*. Retrieved 2015-01-22.

10. "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. March 26, 1986.
11. Shirley, R. (August 2007). "RFC 4949". Retrieved 2013-12-23.
12. Maini, Anil Kumar (2007). *Digital Electronics: Principles, Devices and Applications*. 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. International Organization for Standardization (December 1, 1975). "The set of control characters for ISO 646 (http://kikaku.itscj.ipsj.or.jp/ISO-IR/001.pdf)". *Internet Assigned Numbers Authority Registry*. Alternate U.S. version: [1] (http://kikaku.itscj.ipsj.or.jp/ISO-IR/006.pdf). Accessed 2008-04-14.
14. "RFC 20: ASCII format for Network Interchange" (http://tools.ietf.org/html/rfc20), ANSI X3.4-1968, October 16, 1969.
15. Mackenzie, Charles E. (1980). *Coded Character Sets, History and Development*. *The Systems Programming Series* (1 ed.) (Addison-Wesley Publishing Company, Inc.). pp. 166, 211, 215, 217, 220, 223, 228, 236–238, 243–245, 247–248, 435. ISBN 0-201-14460-3. LCCN 77-90165.
16. "BruXy: Radio Teletype communication". 2005-10-10. Retrieved 2016-05-09. "The transmitted code use International Telegraph Alphabet No. 2 (ITA-2) which was introduced by CCITT in 1924."
17. Smith, Gil (2001). "Teletype Communication Codes" (PDF). Baudot.net. Retrieved 2008-07-11.
18. Sawyer, Stanley A.; Krantz, Steven George (1995). *A TeX Primer for Scientists*. CRC Press, LLC. p. 13. ISBN 978-0-8493-7159-2.
19. Savard, John J. G. "Computer Keyboards". Retrieved 2014-08-24.
20. Brief Report: Meeting of CCITT Working Party on the New Telegraph Alphabet, May 13–15, 1963.
21. Report of ISO/TC/97/SC 2 – Meeting of October 29–31, 1963.
22. Report on Task Group X3.2.4, June 11, 1963, Pentagon Building, Washington, DC.
23. Report of Meeting No. 8, Task Group X3.2.4, December 17 and 18, 1963
24. Bob Bemer (n.d.). Bemer meets Europe (http://www.trailing-edge.com/~bobbemer/EUROPE.HTM). *Trailing-edge.com*. Accessed 2008-04-14. Employed at IBM at that time
25. "Biography of Robert William Bemer".
26. Lyndon B. Johnson (March 11, 1968). 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*. Accessed 2008-04-14.
27. "utf-8(7) - Linux manual page". Man7.org. 2014-02-26. Retrieved 2014-04-21.
28. RFC 2822 (April 2001). "NO-WS-CTL".
29. McConnell, Robert; Haynes, James; Warren, Richard. "Understanding ASCII Codes". Retrieved 2014-05-11.
30. "Re: editor and word processor history (was: Re: RTF for emacs)".
31. "PDP-6 Multiprogramming System Manual" (PDF). Digital Equipment Corporation. 1965. p. 43.
32. "PDP-10 Reference Handbook, Book 3, Communicating with the Monitor" (PDF). Digital Equipment Corporation. 1969. p. 5-5.
33. Ossanna, J. F.; Saltzer, J. H. (November 17–19, 1970). "Technical and human engineering problems in connecting terminals to a time-sharing system" (PDF). *Proceedings of the November 17–19, 1970, Fall Joint Computer Conference*. p. 357: AFIPS Press. pp. 355–362. "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 X3.4-19687 permits the line-feed code to carry the new-line meaning."
34. T. O'Sullivan (May 19, 1971). *TELNET Protocol* (https://tools.ietf.org/html/rfc158). Internet Engineering Task Force (IETF). pp. 4-5. RFC 158. https://tools.ietf.org/html/rfc158. Retrieved 2013-01-28.
35. *File Transfer Protocol* (https://tools.ietf.org/html/rfc542). Internet Engineering Task Force (IETF). Aug 12, 1973. RFC 542. https://tools.ietf.org/html/rfc542. Retrieved 2013-01-28.
36. Jon Postel (June 1980). *File Transfer Protocol* (https://tools.ietf.org/html/rfc765). Internet Engineering Task Force (IETF). RFC 765. https://tools.ietf.org/html/rfc765. Retrieved 2013-01-28.
37. "EOL translation plan for Mercurial". Mercurial. Retrieved 2014-10-28.
38. Bernstein, Daniel J. "Bare LFs in SMTP". Retrieved 2013-01-28.
39. ASA X3.4-1963 (http://www.worldpowersystems.com/projects/codes/X3.4-1963/index.html).
40. RFC 1345 (June 1992).
41. "Specific Criteria," attachment to memo from R. W. Reach, "X3-2 Meeting – September 14 and 15," September 18, 1961
42. R. Maréchal, ISO/TC 97 – Computers and Information Processing: Acceptance of Draft ISO Recommendation No. 1052, December 22, 1967
43. The Unicode Consortium (2006-11-03). "Chapter 13: Special Areas and Format Characters" (PDF). In Allen, Julie D. *The Unicode standard, Version 5.0*. Upper Saddle River, NJ: Addison-Wesley. p. 314. ISBN 0321480910. Retrieved 13 March 2015.
44. "ASCIIbetical definition". *PC Magazine*. Retrieved 2008-04-14.

Further reading

- Bemer, R. W. (1960). "A Proposal for Character Code Compatibility". *Communications of the ACM* **3** (2): 71–72. doi:10.1145/366959.366961.
- Bemer, R. W. (1980). "Inside ASCII". *General Purpose Software*. Best of Interface Age **2**. Portland, OR, USA: Dilithium Press. Chapter 1. ISBN 0-918398-37-1, from:
  - Bemer, R. W. (May 1978). "Inside ASCII - Part I". *Interface Age* (Portland, OR, USA: Dilithium Press) **3** (5): 96–102.
  - Bemer, R. W. (June 1978). "Inside ASCII - Part II". *Interface Age* (Portland, OR, USA: Dilithium Press) **3** (6): 64–74.
  - Bemer, R. W. (July 1978). "Inside ASCII - Part III". *Interface Age* (Portland, OR, USA: Dilithium Press) **3** (7): 80–87.
- Bemer, R. W. (2003-05-23). "The Babel of Codes Prior to ASCII: The 1960 Survey of Coded Character Sets: The Reasons for ASCII". Archived from the original on 2013-10-17. Retrieved 2016-05-09, from:
  - Bemer, R. W. (December 1960). "Survey of coded character representation". *Communications of the ACM* **3** (12): 639–641. doi:10.1145/367487.367493.
  - Smith, H. J.; Williams, F. A. (December 1960). "Survey of punched card codes". *Communications of the ACM* **3** (12): 642. doi:10.1145/367487.367491.
- Robinson, G. S. & Cargill, C. (1996). "History and impact of computer standards". *Computer* **29** (10): 79–85. doi:10.1109/2.539725.
- *American National Standard Code for Information Interchange*. American National Standards Institute. 1977.

External links

- The ASCII subset (http://www.unicode.org/charts/PDF/U0000.pdf) of Unicode
- Fischer, Eric. "The Evolution of Character Codes, 1874-1968".
- Scanned copy of American Standard Code for Information Interchange ASA standard X3.4-1963 (http://worldpowersystems.com/archives/codes/X3.4-1963/index.html)

Wikimedia Commons has media related to ***ASCII***.