ASA
Reg. U. S. Pat. Off.
X3.4-1963
UDC 681.3 (Tentative)

→ Superseded by X3.4-1965

American Standard Code for Information Interchange

Sponsor

Business Equipment Manufacturers Association

Approved June 17, 1963

AMERICAN STANDARDS ASSOCIATION
INCORPORATED

American Standard

Registered United States Patent Office

An American Standard implies a consensus of those substantially concerned with its scope and provisions. An American Standard is intended as a guide to aid the manufacturer, the consumer, and the general public. The existence of an American Standard does not in any respect preclude anyone, whether he has approved the standard or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standard. American Standards are subject to periodic review and users are cautioned to obtain the latest editions. Producers of goods made in conformity with an American Standard are encouraged to state on their own responsibility in advertising, promotion material, or on tags or labels, that the goods are produced in conformity with particular American Standards.

Published by

AMERICAN STANDARDS ASSOCIATION
INCORPORATED
10 East 40th Street, New York 16, N. Y.

Universal Decimal Classification 681.3 (Tentative)

Printed in USA

D10M763/1

Foreword

(This Foreword is not a part of American Standard Code for Information Interchange, X3.4-1963.)

This American Standard presents the standard coded character set to be used for information interchange among information processing systems, communication systems, and associated equipment.

Subsequent standards will prescribe the means of implementing this standard in the principal media, such as perforated tape, punched cards, and magnetic tape. Other standards will deal with collating and error control considerations. These standards will facilitate the interchange of digital information.

The 7-bit coded character set was developed by a group of highly qualified and experienced specialists in information processing and communication.* Past work in the field was reviewed, and a comprehensive program of original research and code design completed. Careful consideration has been given to the several conflicting code set requirements, and their resolution achieved in the standard code.

This standard was approved as American Standard by the American Standards Association on June 17, 1963.

Suggestions for improvement gained in the use of this standard will be welcome. They should be sent to the American Standards Association, Incorporated, 10 East 40th Street, New York 16, N. Y.

The ASA Sectional Committee on Computers and Information Processing, X3, which developed this standard, had the following personnel at the time of approval:

C. A. PHILLIPS, *Chairman* R. E. UTMAN, *Secretary*

Air Transport Association of America	F. C. WHITE
American Bankers Association	G. W. FREY
American Gas Association and Edison Electric Institute (jointly)	R. E. HARBAUGH
American Management Association	G. N. STILIAN
American Petroleum Institute	F. A. GITZENDANNER
Association for Computing Machinery	SAUL GORN
Association of Consulting Management Engineers	LANSDALE BOARDMAN
Business Equipment Manufacturers Association	F. W. BAUER W. H. BURKHART R. F. CLIPPINGER G. T. CROFT R. W. GREEN R. J. MINDLIN B. W. POLLARD C. E. SHINN A. H. STILLMAN
Data Processing Management Association	J. M. ADAMS
Electronic Industries Association	H. L. THOLSTRUP
Engineers Joint Council	W. M. CARLSON
General Services Administration	R. A. RAUP
The Institute of Electrical and Electronics Engineers	R. W. FERGUSON G. W. PATTERSON
Insurance Accounting and Statistical Association	CARL ORKILD
Joint Users Group	H. N. CANTRELL
Life Office Management Association	EUGENE BOULANGER
National Retail Merchants Association	ETHEL LANGTRY
Telephone Group	L. W. CLAUSSEN
U. S. Department of Defense	G. L. BOWLBY

* Operating under ASA Project X3, Computers and Information Processing. The Business Equipment Manufacturers Association serves as sponsor of the X3 project.

At the time the proposal was developed and processed through ASA X3.2 subcommittee, the subcommittee membership was as follows:

C. E. Macon, *Chairman*
Burroughs Corporation

I. Liggett, *Former Chairman*
International Business Machines Corporation

R. E. Utman, *Secretary*
Sperry Rand Corporation

J. F. Auwaerter
Teletype Corporation

J. E. Bartelt
IBM Corporation

L. Bloom
National Cash Register Company

R. E. Blue
IBM Corporation

T. H. Bonn
Sperry Rand Corporation

J. B. Booth
Teletype Corporation

T. R. Bousquet
Minneapolis-Honeywell Regulator Company

J. F. Chesterman
Bell Telephone Laboratories

L. L. Griffin
U. S. Department of Defense

R. Gryb
American Telephone & Telegraph Corporation

H. Kleinberg
Radio Corporation of America

W. J. Leubbert
U.S. Military Academy

M. Pivovonsky
Monroe Calculating Machine Company

R. W. Reach
Minneapolis-Honeywell Regulator Company

H. J. Smith, Jr
IBM Corporation

H. Tholstrup
Friden, Inc

A. J. Ungar
International Electric Corporation

A. L. Whitman
Bell Telephone Laboratories

It should be recognized that although X3.2 members are variously affiliated, work on an ASA subcommittee is achieved primarily on an individual competence and experience basis. The membership above has with some exceptions been continuously active from the beginning of X3.2 work in 1960.

Contents

American Standard Code for
Information Interchange

1. Scope

This coded character set is to be used for the general interchange of information among information processing systems, communication systems, and associated equipment.

2. Standard Code

b_7				0	0	0	0	1	1	1	1
b_6				0	0	1	1	0	0	1	1
b_5				0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1								
0	0	0	0	NULL	DC_0	b	0	@	P		
0	0	0	1	SOM	DC_1	!	1	A	Q		
0	0	1	0	EOA	DC_2	"	2	B	R		
0	0	1	1	EOM	DC_3	#	3	C	S		
0	1	0	0	EOT	DC_4 (STOP)	$	4	D	T		
0	1	0	1	WRU	ERR	%	5	E	U		
0	1	1	0	RU	SYNC	&	6	F	V		
0	1	1	1	BELL	LEM	(APOS)	7	G	W		
1	0	0	0	FE_0	S_0	(	8	H	X		
1	0	0	1	HT / SK	S_1	)	9	I	Y		
1	0	1	0	LF	S_2	*	:	J	Z		
1	0	1	1	V_{TAB}	S_3	+	;	K	[		
1	1	0	0	FF	S_4	(COMMA)	<	L	\		
1	1	0	1	CR	S_5	−	=	M	]		
1	1	1	0	SO	S_6	.	>	N	↑		
1	1	1	1	SI	S_7	/	?	O	←		

UNASSIGNED

UNASSIGNED

ACK

(1)

ESC

DEL

3. Positional Order and Notation

Standard 7-bit set code positional order and notation are shown below with b_7 the high-order, and b_1 the low-order, bit position.

	b_7	b_6	b_5	b_4	b_3	b_2	b_1
EXAMPLE: The code for "R" is:	1	0	1	0	0	1	0

4. Legend

NULL	Null/Idle	DC_1-DC_3	Device control
SOM	Start of message	DC_4(Stop)	Device control (stop)
EOA	End of address	ERR	Error

Legend continued on following page

Legend Continued

EOM	End of message		
EOT	End of transmission	SYNC	Synchronous idle
WRU	"Who are you?"	LEM	Logical end of media
RU	"Are you . . .?"	S_0-S_7	Separator (information)
BELL	Audible signal	b	Word separator (space, normally non-printing)
FE_0	Format effector		
HT	Horizontal tabulation	<	Less than
SK	Skip (punched card)	>	Greater than
LF	Line feed	↑	Up arrow (Exponentiation)
V_{TAB}	Vertical tabulation	←	Left arrow (Implies/ Replaced by)
FF	Form feed		
CR	Carriage return	\	Reverse slant
SO	Shift out	ACK	Acknowledge
SI	Shift in	①	Unassigned control
DC_0	Device control reserved for data link escape	ESC	Escape
		DEL	Delete/Idle

NOTE: Expanded definitions of some of the above terms may be found in the appendixes.

5. Qualifications

5.1 This standard does not define the means by which the coded set is to be recorded in any physical medium. The standard code does not include any redundancy or define techniques for error control. Further, it does not specify a standard collating sequence.

5.2 Deviations from the standard may create serious difficulties in information interchange and should be used only with full cognizance of the parties involved.

5.3 Unassigned codes are reserved for future standardization. Their use in information interchange prior to such standardization is a deviation from the standard.

5.4 The appendixes to this standard cover code design considerations and criteria, related subsets, extensions and deviations.

Appendixes

(These Appendixes are not a part of American Standard Code for Information Interchange, X3.4-1963, but are included to facilitate its use.)

Appendix A
Design Considerations for the Coded Character Set

A1. Introduction

A1.1 The standard coded character set is intended for the interchange of information among information processing systems, communication systems, and associated equipment.

A1.2 Work will continue in the following areas (not necessarily listed in order of priority):

(1) Representation of the coded character set in the principal media (perforated tape, magnetic tape, and punched cards)
(2) Error control considerations
(3) Collating conventions
(4) Relation of the standard set to other sets
(5) Assignment of meaning to presently unassigned codes as required
(6) Relationship to other standards

A2. Considerations Affecting the Standard

There were many considerations that determined the standard's set size, set structure, character selection, and character placement. Among these were (not listed in order of priority):

(1) Need for adequate number of graphics
(2) Need for adequate number of device controls and format effectors
(3) Desire for a non-ambiguous code, i.e., one in which every character has a unique meaning independent of other characters
(4) Physical limitations of media and facilities
(5) Error control
(6) Special interpretation of the all-zeroes and all-ones codes
(7) Ease in the identification of classes of characters
(8) Data manipulation requirements
(9) Collating conventions
    (a) Logical
    (b) Historical
(10) Keyboard conventions
    (a) Logical
    (b) Historical
(11) Other set sizes
(12) International considerations
(13) Programming languages
(14) Existing coded character sets

A3. Set Size

A 7-bit set is the minimum size that will meet the requirements for graphics and control in applications involving general information interchange. Both a 6-bit and an 8-bit set were considered and rejected—the 6-bit, providing only 64 graphics, could not accommodate essential format effectors, such as "carriage return," "line feed," "horizontal tab," etc; the 8-bit because it provides far more characters than are now needed in general applications.

A4. Set Structure

A4.1 In discussing the set structure it is convenient to divide the set into 8 columns of 16 characters each, as indicated in the standard.

A4.2 It was considered essential to have a dense 64-character subset which contained only graphics. For ease of identification this graphic subset was placed in 4 contiguous columns.

A4.3 Placement of the graphic subset was dictated by the requirement that the all-zeroes character be reserved for the "Null/Idle" function, and the all-ones character for the "Delete/Idle" function. Since the first and last columns contain these characters, the next logical choice for the graphic subset was the middle four columns of the code. Although this placement complicates the identification of the graphic subset since two bits must be examined (a one-bit test would have been sufficient had these graphics been placed in the first four or last four columns), this disadvantage is outweighed by the advantages of the dense graphic subset.

A4.4 The character set was structured to enable the easy identification of classes of graphics and controls.

A5. Choice of Graphics

Included in the set are the digits, a single case of the alphabetic letters A through Z, and those punctuation, mathematical, and business symbols considered most useful. The set includes the characters commonly encountered in programming languages. In particular, the COBOL graphics are included. It is not practicable to include all of the ALGOL graphics (which number 120).

A6. Graphic Subset Structure

A6.1 The basic structure of the dense graphic subset was influenced by logical collating considerations, the requirements of simply related 6-bit sets, and the needs of typewriter-like devices. For information processing, it is desirable that the characters be arranged in such a way as to minimize both the operating time and the hardware components required for ordering and sequencing operations. This requires that the relative order of characters, within classes, be such that a simple comparison of the binary codes will result in information being ordered in a desired sequence.

A6.2 Conventional usage requires that the word separator (space) be ahead of any other symbol in a collatable set. This permits a name such as "Johns" to collate ahead of a name such as "Johnson." The requirement that punctuation symbols also collate ahead of the alphabet ("Johns, A" should also collate before "Johnson") established the special symbol locations, including space, in the first column of the graphic subset.

A6.3 To simplify the design of typewriter-like devices, it is desirable that there be only a common 1-bit difference between characters normally paired on keytops. This, together with the requirement for a contiguous alphabet, the collating requirements outlined above, and international considerations, resulted in the placement of the alphabet in the last two columns of the graphic subset. This left the second column of the graphic subset for the numerals.

A6.4 Although the resultant structure of "specials" (S), "digits" (D), and "alphabetics" (A) does not conform to the most prevalent collating convention (SAD), it must be recognized that simple binary rules for collation do not necessarily apply between classes of characters.

A6.5 The need for a simple transformation from the set sequence to the prevalent collating convention was recognized, and dictated the placement of some of the "specials" within the set. Specifically, those special symbols, viz, ampersand (&), asterisk (*), comma (,), hyphen (-), period (.), and slant (/), which are most often used as identifiers for ordering information and which normally collate ahead of both the alphabet and the numerals, were not placed in the column containing the numbers, so that the entire numeric column could be rotated via relatively simple computer logic to a position higher than the alphabet. The sequence of the aforementioned "specials" was also established to the extent practical to conform to the prevalent collating convention.

A6.6 The need to adapt a useful 4-bit numeric set from the 6-bit graphic subset also played a role in the placement of characters. Such a 4-bit set, including the digits, asterisk, plus (+), comma, hyphen, period, and slant, can easily be derived from the standard.

A6.7 To further international standardization, and provide the 4-bit set mentioned in A6.6, the structure of the graphic subset precludes (logically) the historic keyboard association of colon (:) with semicolon (;). However, the dual character key assignment of the question mark (?) with the slant was maintained, as it was with a majority of the numerals and commonly associated symbols.

A6.8 Considerations of other domestic code sets, including the Department of Defense Standard 8-bit data transmission code (1961) as well as international requirements, played an important role in deliberations that resulted in the standard. The selection and grouping of the symbols dollar sign ($), percent sign (%), ampersand, and apostrophe (') and the symbols less than (<), equal (=), and greater than (>) permit contraction to either a business ($ % &) or scientific (< = > ') 6-bit subset. The sequence of these latter symbols and of the symbols comma, hyphen, period, and slant permitted an advantageous pairing on the keyboard. The historic pairing of question mark and slant is preserved and the less than and greater than symbols, which have comparatively low usage, are paired with period and comma so that in dual-case keyboard devices where it is desired to have period and comma in both cases the less than and greater than symbols are the ones displaced. Provision was made for the accommodation of alphabets containing more than 26 letters and for 6-bit contraction by the location of low-usage characters in the area following the alphabet. In addition, the requirement for the digits 10 and 11 used in sterling monetary areas was considered in the placement of the asterisk, plus, semicolon, and colon.

A7. Choice of Controls

A7.1 The control characters included in the set are those required for the control of terminal devices, input and output devices, format, or transmission and switching on a general enough basis to justify inclusion in a standard set.

A7.2 A group of eight codes has been reserved for information separators which when implemented in

a system shall bear a hierarchical relationship. They identify boundaries of various elements of information.

A7.3 Information separators are machine-oriented controls having two characteristics that differentiate them from human-oriented separators (word separator, punctuation, etc). First, machine-oriented separators are hierarchical in nature, whereas human-oriented separators have no fixed hierarchy. Second, machine-oriented separators must serve rigidly defined functions in a system, whereas proper interpretation of human-oriented separators requires knowledge of the context in which they are used.

A8. Control Subset Structure

A8.1 The first two columns were chosen for most of the assigned controls because there are more codes in the last two columns with a high probability of being inadvertently generated during an idle line condition than there are in the first two columns. "Acknowledge" was placed where its code could be generated by simple means. The "Escape" was placed so as to conform with the "special" function of the DOD standard 8-bit code and to facilitate the 6-bit contraction.

A8.2 The controls that were selected logically fall into four groups:

(1) Transmission controls
(2) Format effectors
(3) Device controls
(4) Information separators

Within each group the controls are ordered so that the binary and hierarchical order are directly related. This structure facilitates the contraction of the standard to related 6-bit sets and permits logical comparisons of related controls in the ordering of information. In particular, the placement of the format effectors and information separators facilitates their dual usage when contracting to a related 6-bit set.

Appendix B
Related Subsets and Adaptations

B1. Introduction

B1.1 The standard code was developed to provide for information interchange among information processing systems, communications systems, and associated equipment. Its structure facilitates conversion from the standard code to adaptations usable internally in a variety of equipments. In a system consisting of several equipments each with its local or native code, maximum flexibility will be achieved if each of the native codes is translated to the standard whenever information interchange is desired.

B1.2 Within any particular equipment or closed system it may be necessary to substitute characters. For example, some systems may require special graphic symbols and some devices may require special control codes. Design efforts on the standard code included consideration of these types of adaptations and the possibility that the parties employing them may develop a need to interchange information with others. If certain simple rules are followed in making such adaptations, minimum difficulty will be encountered in conversion to the standard code.

B1.3 The material below describes possible adaptations and logically related sets.

B2. Character Substitutions

B2.1 When a nonstandard character is introduced, only the code position where the substitution is made shall be affected.

B2.2 It is recommended that graphic substitutions be made only in the graphic area and control substitutions only in the control area. Any substitution involving a control should be made only with full cognizance of all possible operational effects.

B2.3 Any such substitution will result in a nonstandard coded character set.

B3. Unassigned Codes

A meaning was not assigned to a code unless that meaning was of sufficiently general use to warrant standardization. This resulted in some codes being unassigned. These codes are subject to future standardization. Where an unassigned code is given

meaning in a particular system, such meaning is nonstandard and use of this code in information interchange is hazardous.

B4. Illustrative Nonstandard Codes

Code sets obtained by modifying the standard as shown below, or by other replacements, are nonstandard.

B4.1 European Alphabets. The five graphics immediately following the letter Z can be replaced by the additional letters required for complete expression of certain European alphabets. Further, the single position preceding the letter A can be used for those alphabets requiring 32 characters. In most cases, only three additional letters will be required.

B4.2 Base 12 Numeric Digits. For those applications requiring use of the sterling monetary system or duodecimal arithmetic, the digits 10 and 11 can replace the two graphics immediately following the digit 9.

B5. Related Larger and Smaller Sets

Consideration has been given to the relationship between the standard set and sets of other sizes. A number of straightforward logical transforms are possible which result in a variety of sets related to the standard. None of the transformed sets should be considered standard.

Appendix C
Specific Criteria

C1. Introduction

C1.1 This Appendix contains the criteria on which the design of the code was based. Not all criteria have been entirely satisfied. Some are conflicting, and the characteristics of the set represent acceptable compromises of these divergent criteria.

C1.2 The standard has been designated a code for information interchange and not necessarily for internal use in information processing equipment. However, many of the criteria used in establishing the set are processor-oriented since simplicity in deriving logical and consistent processing sets was considered mandatory.

C2. Criteria

C2.1 All codes in the set shall consist of the same number of binary positions (bits).

C2.2 The standard set shall be so structured as to facilitate derivation of larger or smaller sets.

C2.3 Each character code shall consist of n binary bits. All possible 2^n combinations of ones and zeroes will be permitted and considered valid.

C2.4 The number of bits, n, shall be sufficient to provide for the alphabetic and numeric characters, commonly encountered punctuation marks, and other special symbols, along with those controls required for interchange of information.

C2.5 The numerals 0 through 9 shall be included in a 4-bit subset.

C2.6 The numerals 0 through 9 shall be so coded that the four low-order bits shall be the binary-coded-decimal form of the particular numeral that the code represents. In the selection of the two characters immediately succeeding the numeral 9, consideration shall be given to their replacement by the graphics 10 and 11 to facilitate the adoption of the code in the sterling monetary area.

C2.7 The interspersion of control codes among the graphic codes shall be avoided. The codes devoted to controls shall be easily separable from those devoted to graphics.

C2.8 Within the standard set, each character and its corresponding code shall stand by itself and not depend on surrounding characters for interpretation. The "mode shift" characters (SO, SI, or Escape) in an information stream shall signal a departure from the standard set.

C2.9 The alphabet A through Z shall be included in a 5-bit subset. Consideration shall be given to the need for as many as 32 characters in some alphabets.

C2.10 With the letters of the alphabet in their conventional order, A through Z, the codes shall be

assigned in continuous increasing binary order. This criterion prevents interspersion of non-alphabetic characters within the alphabet.

C2.11 Suitable control codes required for communication and information processing shall be included.

C2.12 Escape functions that provide for departures from the standard set shall be incorporated.

C2.13 A simple binary comparison shall be sufficient to determine the order within each class of characters. (For the purpose of this standard, the special graphics, the numerals, and the alphabet are each defined as distinct classes.) Simple binary rules do not necessarily apply between classes when ordering information.

C2.14 The "word separator" (i.e., the space between words) must collate ahead of all other graphics.

C2.15 Special symbols used in the ordering of information must collate ahead of both the alphabet and the numerals.

C2.16 Insofar as possible the special symbols shall be grouped according to their functions; for example, punctuation, mathematical symbols, and shorthand abbreviations. Further, the set shall be so organized that the simplest possible test shall be adequate to distinguish and identify the basic alphabet, numeric, and special symbol subsets.

C2.17 Special symbols shall be placed in the set so as to simplify their generation by typewriters and similar keyboard devices.

This criterion means, in effect, that the codes for pairs of characters that normally appear on the same keytops on a typewriter shall differ only in a common single-bit position.

C2.18 The set shall contain the graphic characters of the principal programming languages.

C2.19 The codes for all control characters shall contain a common, easily recognizable, bit pattern.

C2.20 The null, idle, and delete control functions shall be provided.

American Standards

The standard in this booklet is one of over 2000 standards approved to date by the American Standards Association, Incorporated.

The ASA provides the machinery for creating voluntary standards. It serves to eliminate duplication of standards activities and to weld conflicting standards into single, nationally accepted standards under the designation "American Standard."

Each standard represents general agreement among maker, seller, and user groups as to the best current practice with regard to some specific problem. Thus the completed standards cut across the whole fabric of production, distribution, and consumption of goods and services. Manufacturers, consumers, technical organizations, and governmental agencies — all substantially interested and affected groups — are represented on the committees which develop and regularly revise American Standards. The completed standards are used widely by industry and commerce and often by municipal, state, and federal governments.

The ASA, under whose auspices this work is being done, is the American clearinghouse for standards activity on the national level. Founded in 1918, it is a federation of more than 100 trade associations, technical societies, professional groups, and consumer organizations. Some 2200 companies are affiliated with the ASA as company members.

ASA is the United States member of the International Organization for Standardization (ISO). Through this channel American industry makes its position felt on the international level. American Standards are on file in the libraries of the national standards bodies of more than 40 countries.

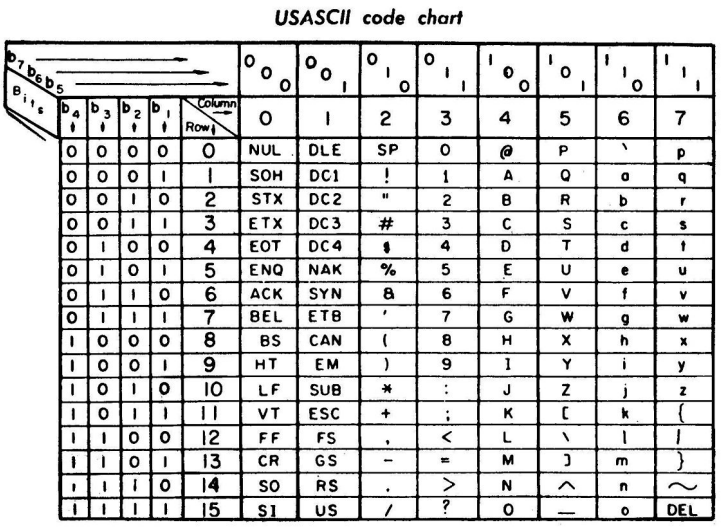
For a free list of all American Standards or information about membership in the ASA write:

AMERICAN STANDARDS ASSOCIATION
INCORPORATED
10 EAST 40th STREET
NEW YORK 16, NEW YORK

ASCII

The **American Standard Code for Information Interchange** (**ASCII** /ˈæski/ *ASS-kee*) is a character-encoding scheme originally based on the English alphabet that encodes 128 specified characters - the numbers 0-9, the letters a-z and A-Z, some basic punctuation symbols, some control codes that originated with Teletype machines, and a blank space - into the 7-bit binary integers.[1]

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.



Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row ↓ / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

A chart of ASCII from a 1972 printer manual

ASCII 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, a major revision during 1967, and the most recent update during 1986. 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.

ASCII includes definitions for 128 characters: 33 are non-printing control characters (many now obsolete) that affect how text and space are processed[2] and 95 printable characters, including the space (which is considered an invisible graphic[3][4]).

The IANA prefers the name **US-ASCII** to avoid ambiguity. ASCII was the most commonly used character encoding on the World Wide Web until December 2007, when it was surpassed by UTF-8, which includes ASCII as a subset.

History

The American Standard Code for Information Interchange (ASCII) was developed under the auspices of a committee of the American Standards Association, 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[5] and ultimately the American National Standards Institute.

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, Fieldata, and early EBCDIC, more than 64 codes were required for ASCII.

The committee debated the possibility of a shift key function (like the Baudot code), which would allow more than 64 codes to be represented by six bits. 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; 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.[6]

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.[7] Eight-bit machines (with octets as the native data type) that did not use parity checking typically set the eighth bit to 0.

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.[8]} The "space" character had to come before graphics to make sorting easier, so it became position 20_{hex};[9] for the same reason, many special signs commonly used as separators were placed before digits. The committee decided it was important to support upper case 64-character alphabets, and chose to pattern ASCII so it could be reduced easily to a usable 64-character set of graphic codes.[10] Lower case letters were therefore not interleaved with upper case. To keep options available for lower case 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.[11] 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. Thus #, $ and % were placed to correspond to 3, 4, and 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. Since many European typewriters placed the parentheses with 8 and 9, those corresponding positions were chosen for the parentheses. 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} next to the letter A.[12]

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.[13]

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.[14] There was some debate at the time whether there should be more control characters rather than the lower case alphabet.[15] The indecision did not last long: during May 1963 the CCITT Working Party on the New Telegraph Alphabet proposed to assign lower case characters to columns 6 and 7,[16] and International Organization for Standardization TC 97 SC 2 voted during October to incorporate the change into its draft standard.[17] The X3.2.4 task group voted its approval for the change to ASCII at its May 1963 meeting.[18] 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 line characters),[19] renaming some control characters (SOM became start of header (SOH)) and moving or removing others (RU was removed).[20] 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

forms of punched card formats.

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 Baudot code, which was also used by the competing Telex teleprinter system. Bob Bemer introduced features such as the escape sequence. 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".[21] Because of his extensive work on ASCII, Bemer has been called "the father of ASCII."

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

> 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.[22]

Other international standards bodies have ratified character encodings such as ISO/IEC 646 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 character set as the first 128 symbols, so the ASCII characters have the same numeric codes in both sets. This allows UTF-8 to be backward compatible with ASCII, allowing software that handles UTF-8 data to treat ASCII data as UTF-8 data, and software made for ASCII treat UTF-8 data as ASCII data, rather than distinguishing between ASCII and UTF-8.

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.[23] 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. 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.

The use of Control-S (XOFF, an abbreviation for transmit off) as a "handshaking" signal warning a sender to stop transmission because of impending overflow, and Control-Q (XON, "transmit on") to resume sending, 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.

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. 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.

As video terminals began to replace printing ones, the value of the "rubout" character was lost. DEC systems, for example, interpreted "Delete" to mean "remove the character before the cursor" and this interpretation also became common in Unix systems. Most other systems used "Backspace" for that meaning and used "Delete" 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") 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 keys 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 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 it 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; it used ASCII, along with CR-LF line endings, and software using other conventions would translate between the local conventions and the NVT. 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. 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.[24]

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 (TLA) for Control-Z instead of SUBstitute. For a variety of reasons, the end-of-text code, ETX aka Control-C, was inappropriate and using Z as the control code to end a file is analogous to it ending the alphabet, a very convenient mnemonic aid. An historic common, and still prevalent, convention uses the ETX aka Control-C 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	Oct	Dec	Hex	Abbr	[25]	[26]	[27]	Name
000 0000	000	0	00	NUL		^@	\0	Null character
000 0001	001	1	01	SOH		^A		Start of Header
000 0010	002	2	02	STX		^B		Start of Text
000 0011	003	3	03	ETX		^C		End of Text
000 0100	004	4	04	EOT		^D		End of Transmission
000 0101	005	5	05	ENQ		^E		Enquiry
000 0110	006	6	06	ACK		^F		Acknowledgment
000 0111	007	7	07	BEL		^G	\a	Bell
000 1000	010	8	08	BS		^H	\b	Backspace[28]
000 1001	011	9	09	HT		^I	\t	Horizontal Tab[29]
000 1010	012	10	0A	LF		^J	\n	Line feed
000 1011	013	11	0B	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[30]
000 1110	016	14	0E	SO		^N		Shift Out

000 1111	017	15	0F	SI		^O		Shift In
001 0000	020	16	10	DLE		^P		Data Link Escape
001 0001	021	17	11	DC1		^Q		Device Control 1 (oft. XON)
001 0010	022	18	12	DC2		^R		Device Control 2
001 0011	023	19	13	DC3		^S		Device Control 3 (oft. XOFF)
001 0100	024	20	14	DC4		^T		Device Control 4
001 0101	025	21	15	NAK		^U		Negative Acknowledgement
001 0110	026	22	16	SYN		^V		Synchronous idle
001 0111	027	23	17	ETB		^W		End of Transmission Block
001 1000	030	24	18	CAN		^X		Cancel
001 1001	031	25	19	EM		^Y		End of Medium
001 1010	032	26	1A	SUB		^Z		Substitute
001 1011	033	27	1B	ESC		^[	\e[31]	Escape[32]
001 1100	034	28	1C	FS		^\		File Separator
001 1101	035	29	1D	GS		^]		Group Separator
001 1110	036	30	1E	RS		^^[33]		Record Separator
001 1111	037	31	1F	US		^_		Unit Separator
111 1111	177	127	7F	DEL		^?		Delete[34][]

[1] RFC 4949

[2] International Organization for Standardization (December 1, 1975). " The set of control characters for ISO 646 (http://www.itscj.ipsj.or.jp/ISO-IR/001.pdf)". *Internet Assigned Numbers Authority Registry*. Alternate U.S. version: (http://www.itscj.ipsj.or.jp/ISO-IR/006.pdf). Accessed 2008-04-14.

[3] "RFC 20: ASCII format for Network Interchange" (http://tools.ietf.org/html/rfc20), ANSI X3.4-1968, October 16, 1969.

[4] Mackenzie 1980, p. 223.

[5] Mackenzie 1980, p. 211.

[6] Mackenzie 1980, p. 215, Decision 4.

[7] Mackenzie 1980, p. 217, Decision 5.

[8] Mackenzie 1980, p. 220, Decisions 8,9.

[9] Mackenzie 1980, p. 237, Decision 10.

[10] Mackenzie 1980, p. 228, Decision 14.

[11] Mackenzie 1980, p. 238, Decision 18.

[12] Mackenzie 1980, p. 243.

[13] Mackenzie 1980, pp. 243-245.

[14] Mackenzie 1980, pp. 66, 245.

[15] Mackenzie 1980, p. 435.

[16] Brief Report: Meeting of CCITT Working Party on the New Telegraph Alphabet, May 13–15, 1963.

[17] Report of ISO/TC/97/SC 2 – Meeting of October 29–31, 1963.

[18] Report on Task Group X3.2.4, June 11, 1963, Pentagon Building, Washington, DC.

[19] Report of Meeting No. 8, Task Group X3.2.4, December 17 and 18, 1963

[20] Mackenzie 1980, p. 247–248.

[21] 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

[22] 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.
[23] RFC 2822 (April 2001). "NO-WS-CTL".
[24] EOL translation plan for Mercurial (http://mercurial.selenic.com/wiki/EOLTranslationPlan)
[25] 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.
[26] Caret notation often used to represent control characters on a terminal. On most text terminals, holding down the 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.
[27] 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).
[28] The Backspace character can also be entered by pressing the key on some systems.
[29] The Tab character can also be entered by pressing the key on most systems.
[30] The Carriage Return character can also be entered by pressing the or key on most systems.
[31] The '\e' escape sequence is not part of ISO C and many other language specifications. However, it is understood by several compilers, including GCC.
[32] The Escape character can also be entered by pressing the key on some systems.
[33] ^^ means (pressing the "Ctrl" and caret keys).
[34] The Delete character can sometimes be entered by pressing the key on some systems.

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

ASCII 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.

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)[4] and thus would not normally be visible, it is represented here by Unicode character U+2420 "␠"; Unicode characters U+2422 "␢" and U+2423 "␣" are also available for use when a visible representation of a space is necessary.

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 ($5F_{hex}$).[1]

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

Reverse List:
~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#"!

Binary	Oct	Dec	Hex	Glyph	
010 0000	040	32	20	space (punctuation)	
010 0001	041	33	21	[[Exclamation mark	]]
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[2] and the Internet Assigned Numbers Authority registry of character sets 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
  - (with ASCII-7 and ASCII-8 variants)Wikipedia:Verifiability
- ISO646-US
- US-ASCII (preferred MIME name)[]
- us
- IBM367
- cp367
- 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.

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,[3] 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[4] 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.

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 0-31 with more graphics. 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, this was 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 attractive text printing.

ISO-8859-1, Windows-1252, and the original 7-bit ASCII were the most common character encodings until the late 2000s, nowadays UTF-8 is becoming more common.

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. 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" 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.

References

Footnotes

[1] ASA X3.4-1963 (http://www.wps.com/projects/codes/X3.4-1963/index.html).
[2] RFC 1345 (June 1992).
[3] "Specific Criteria," attachment to memo from R. W. Reach, "X3-2 Meeting – September 14 and 15," September 18, 1961
[4] R. Maréchal, ISO/TC 97 – Computers and Information Processing: Acceptance of Draft ISO Recommendation No. 1052, December 22, 1967

Bibliography

- Mackenzie, Charles E. (1980). *Coded Character Sets, History and Development*. Addison-Wesley. ISBN 0-201-14460-3.

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 (http://dx.doi.org/10.1145/366959.366961).
- Bemer, R. W (May 23, 2003). "The Babel of Codes Prior to ASCII: The 1960 Survey of Coded Character Sets: The Reasons for ASCII" (http://www.trailing-edge.com/~bobbemer/SURVEY.HTM). from:
  - Bemer, R. W. (December 1960), "Survey of coded character representation", *Communications of the ACM* **3** (12): 639–641, doi: 10.1145/367487.367493 (http://dx.doi.org/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 (http://dx.doi.org/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 (http://dx.doi.org/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
- The Evolution of Character Codes, 1874–1968 (http://www.pobox.com/~enf/ascii/ascii.pdf)
- Scanned copy of American Standard Code for Information Interchange ASA standard X3.4-1963 (http://wps.com/projects/codes/X3.4-1963/index.html)

Article Sources and Contributors

ASCII *Source*: https://en.wikipedia.org/w/index.php?oldid=583162613 *Contributors*: 126pk, 24ip, 28bytes, 32X, 4twenty42o, 7Keypad, 9thbit, A. di M., A12n, A8UDI, ABCD, AMR, Abdull, AdamRaizen, Addihockey10, Aditya, Adriatikus, AgentPeppermint, Ahoerstemeier, Alansohn, Aleksandar Šušnjar, Allstarecho, Amire80, Andrejj, Andyhowlett, Angela, Animum, Anomie, AnonMoos, Apokrif, Arda Xi, Arjun G. Menon, Ashershow1, Asmeurer, Augietrautz, AxelBoldt, Axlrosen, AzaToth, B2382F29, B4hand, BCube, BD2412, BDE1982, BIL, Baa, Badon, Barabuski, Basil.bourque, Bbx, Bender235, Benjamin Mako Hill, Bennytheboy, Betacommand, Bevo, Bgwhite, Bisqwit, Bitsecure, Bjmullan, Bobblewik, Bobo192, Bongwarrior, BonsaiViking, Boujois, Brendan Moody, Brianski, Brighterorange, Brouhaha, Bryan Derksen, Bryant1410, CLW, CRGreathouse, CTZMSC3, Cadby Waydell Bainbrydge, Caffolote, Caltrop, Can't sleep, clown will eat me, Cbdorsett, Cburnett, Cema, Ceyockey, Chaosrxn, Cherlin, Chris Chittleborough, Chris D Heath, Chris is me, ChrisGualtieri, ChrisTek, Christian List, Chrsclmn, Clappingsimon, Cleared as filed, Closedmouth, Cobaltcigs, Codename Lisa, Codysnider, ColbertTOLDMETODOIT, Comp.arch, Computer97, Conversion script, Coroboy, Corti, Crimsunwulf, Curps, Cyanoa Crylate, Cybercobra, CyborgTosser, DHN, DVD R W, David Gerard, DavidCary, Dbachmann, Dcanright, DeadEyeArrow, Deflective, Dendodge, DennisWithem, DerHexer, DexDor, Dfrankow, Dfrg.msc, Diabolical mdog, Direvus, Discospinster, Dispenser, Dl2000, Dmeranda, Dmlandfair, DocWatson42, Domthedude001, Donaldgdavis, Donarreiskoffer, Doomdayx, Dragonscales, Dreftymac, DropDeadGorgias, Drphilharmonic, Dwiakigle, Dysprosia, ERcheck, Ed g2s, Edward Z. Yang, Egil, Eigenlambda, Ekohlwey, El C, Electron9, Eleeeeeephan, Elektron, Ellers, Elliskev, Elm-39, Eloquence, Elpollo, Elwikipedista, Emijrp, Emperorbma, Enf, Epbr123, Eric N Fischer, Eubulides, Eugenwpg, Evertype, Everyking, Extropian314, Eyreland, FFraenz, Fabartus, Fastily, Favonian, Finell, Fish and karate, FrankGinzoFella, Freedomlinux, FreplySpang, Fresheneesz, Frungi, Fudoreaper, Fullstop, Furrykef, Fuzheado, Fæ, G1234, Gail, Gaius Cornelius, Galwhaa, Gancheff, GatesPlusPlus, Gauss, Gazibara, Gene Thomas, Geo Swan, George The Dragon, GeorgeLouis, Gerbrant, Ghettoblaster, Giftlite, Gilliam, Gimboid13, Gimmetoo, Gimmetrow, Giobe2000, Giulio.orru, Glaisher, Glloq, Gracenotes, GraemeL, Graham87, GreatWhiteNortherner, GreggTownsend, Gtg204y, Gutworth, Gutza, Guy Harris, Gwalla, Hadal, HairyWombat, Hajhouse, Harland1, Herbee, Hevelius, HexaChord, Hhielscher, Hobart, Hotcrocodile, Hotlorp, Hu12, Hydrargyrum, I am One of Many, IAMDESU, IMSoP, Iamzemasterraf, Icairns, Icey, Ignaciomella, Ike, Ilikepie00, Ilya, Indefatigable, Intgr, Iridescent, J Di, J.delanoy, JDBravo, JLaTondre, JMyrleFuller, Jagged 85, JakeVortex, Jallan, Jasonglchu, Java13690, Java7837, Jay, Jbmurray, Jeandré du Toit, Jeanhaney, Jeff G., Jeffrey Henning, Jeffthejiff, Jengod, Jesdisciple, Jesseili, Jfioeawfjdls453, Jianhui67, Jjk, Jmajeremy, Johnteslade, Jonathan Drain, Jonathanrcoxhead, Jordan Brown, Jsnx, KJRehberg, KTC, Kaisershatner, Kalidasa 777, Karl Dickman, Kbdank71, Kbolino, KeithTyler, Keka, Kerowyn, KerryVeenstra, Khazaei.mr, KillaDamo, Kjoonlee, Kricxjo, Krótki, Kubanczyk, Kune, Kvng, Kwamikagami, Kwi, Kylegwot, LWChris, LarryGilbert, Latics, LeadSongDog, Lee Daniel Crocker, Lelkesa, LeoNomis, LeonardoRob0t, Leuko, Levin, Lfwlfw, Lgfcd, Liftarn, Lincher, Ling.Nut, LittleOldMe old, Loadmaster, LodeRunner, Logan, Loohcsnuf, Lotje, LouDogg, Lowellian, LuNatic, Luna Santin, Lupin, Maclean25, Magioladitis, MagnaMopus, Magnus Bakken, Mahmudmasri, Mange01, Manop, Mardus, Mark Rosenthal, MarkMLl, MarkOverton222, Martial75, Materialscientist, Matt Britt, MattGiuca, Matthiaspaul, Maury Markowitz, Mav, Maxamegalon2000, Maximaximax, Mblumber, Mboverload, McSly, MeekMark, Mercury, Michael Frind, Michaeln, Mike Rosoft, Mike Schwartz, Mike Selinker, Mikehead, Mikeisgay12345, Mild Bill Hiccup, Mindspillage, Minna Sora no Shita, Mitch Ames, Mjb, Mjec, Mkhadpe, Modulatum, Monedula, Mono .lck, Morgan Leigh, Mothmolevna, Mpolo, Mpwrmnt, MrWeeble, MusikAnimal, Musiphil, Mwtoews, Myleslong, N5iln, NSR, Naelphin, Namazu-tron, Naohiro19, Nate Silva, Nathanaeljones, NatusRoma, Nestea Zen, NevilleDNZ, Nick Comber, Nickj, Nikola Smolenski, Nohat, Noldoaran, Nonky, NotACow, Nwbeeson, ORBIT, Oblivious, Octane, Ohhzone, Ohnoitsjamie, OlEnglish, Oleg Alexandrov, Oli Filth, Omegatron, Onegin1, Onorem, Orayzio, OrgasGirl, Ortolan88, Ost316, Owain, OwenBlacker, Oxymoron83, P0lyglut, PSUMark2006, Pabouk, Panser Born, Patrick, PatrickSauncy, Patsw, Paul Magnussen, Paulschou, Pb30, Pboyd04, Pedant17, Pengo, Per Abrahamsen, Peripitus, PeterThoeny, Pgk, Phantomsteve, PhilHibbs, Phuzion, Pichai Asokan, Piet Delport, Pinethicket, Plugwash, Pne, Poiuyuiop, Porterjoh, Potaco99, Potatoswatter, Prohlep, Psychonaut, Pxma, Python eggs, Quercus solaris, Quintote, R.123, RJaguar3, RTC, Raise exception, Ralos, Raymondwinn, ReallyNiceGuy, Redquark, Reisio, Relaxing, RenesisX, RexNL, Rich Farmbrough, RickK, Rjwilmsi, Rob Hooft, Robbgodshaw, Robert Moyse, RobertG, RockMFR, Rocksissor, Ron1947, Roomoor, Rts.bn.vs, Runlevel1, SMC, ST47, Sade, Saga City, Salix alba, SallyForth123, Salvio giuliano, SandyJax, SchfiftyThree, Scientus, Scovetta, Sean William, Seraphim, Sergent Tooj, Shadowjams, Shanes, Shimonnyman, Shlomital, Shoeofdeath, Sintaku, SirEpicMan, SixSix, Sjoerd visscher, Skarebo, Sl, Slapmaster3000, Slightsmile, Smite-Meister, Snoyes, Soccerimp, SpaceFlight89, Spearhead, Speck-Made, Spel-Punc-Gram, Spitzak, Squorky1, Stassats, Stephen, Stephenb, Stephenchou0722, StevenDH, Sunny256, Superm401, Susvolans, Swtpc6800, T-bonham, Taka, TakuyaMurata, Tangotango, Tannin, Tbhotch, Tbutzon, Tbvdm, Tearsinraine, Tedius Zanarukando, Tempshill, Terr0rist333, The Epopt, The Mark of the Beast, The Rambling Man, The Singing Badger, The Thing That Should Not Be, Thelibragirl, Thesean43, Thewayforward, Tide rolls, TigerShark, Tim Starling, Timwi, Titoxd, Tizio, Tobias Bergemann, Tompsci, Traal, Transfinite, TreasuryTag, Treekids, Tripodics, Twas Now, Tyler, UU, Ultraexactzz, UnknownzD, Uriber, UserGoogol, Vadmium, Valery Beaud, VampWillow, Vanisaac, Vanished user 9i39j3, Vary, Vegpuff, VictorLucas, Vishahu, VishantG, Wa3frp, WaltBusterkeys, Wardog, Wario456, Wavelength, Wayward, Wereon, Werieth, Wernher, Widr, WikiWikiHacker, Wikiklrsc, William Avery, Wimt, Winterst, Wizardman, WorldlyWebster, WormNut, Wtshymanski, Ww, Xaosflux, Xevious, Yacht, Yaksar, Yintan, Yosri, ZX81, ZapThunderstrike, Zenohockey, ZeroEgo, Zoezoo, Zundark, Zzuuzz, Саша Стефановић, 909 anonymous edits

Image Sources, Licenses and Contributors

File:ASCII Code Chart-Quick ref card.jpg *Source*: https://en.wikipedia.org/w/index.php?title=File:ASCII_Code_Chart-Quick_ref_card.jpg *License*: Public Domain *Contributors*: Namazu-tron

License