

# Appendix C - ASCII and CHR\$ Tables

---

Intuition Engine uses standard 7-bit ASCII for character codes and 8-bit bytes for the body of every string. The function `CHR$(n)` returns a one-character string whose single byte equals `n`. The function `ASC(s$)` returns the byte value of the first character of `s$`. These are exact inverses: `ASC(CHR$(n))` is always `n`, for every `n` in the range 0 through 255.

There is no character translation between BASIC strings and the terminal. The byte you put in is the byte that goes out, and the byte the terminal sends in is the byte you get back.

## C.1 Control codes (0-31)

---

These bytes have no printed glyph. Three of them have meaning to the terminal; the rest pass through to `TERM_OUT` (Chapter 38) and have no visible effect.

Dec	Hex	Name	Effect on the terminal
0	\$00	NUL	Terminates a stored line and a string. Do not embed in a string.
1	\$01	SOH	(no effect)
2	\$02	STX	(no effect)
3	\$03	ETX	(no effect)
4	\$04	EOT	(no effect)
5	\$05	ENQ	(no effect)
6	\$06	ACK	(no effect)
7	\$07	BEL	(no effect)
8	\$08	BS	Move cursor back one column (does not erase). Used by <code>INPUT</code> to delete.
9	\$09	HT	(no effect)
10	\$0A	LF	Move to start of new line. Resets the column counter.
11	\$0B	VT	(no effect)
12	\$0C	FF	(no effect)
13	\$0D	CR	Carriage return. Resets the column counter.
14	\$0E	SO	(no effect)
15	\$0F	SI	(no effect)
16	\$10	DLE	(no effect)
17	\$11	DC1	(no effect)
18	\$12	DC2	(no effect)
19	\$13	DC3	(no effect)
20	\$14	DC4	(no effect)
21	\$15	NAK	(no effect)

Dec	Hex	Name	Effect on the terminal
22	\$16	SYN	(no effect)
23	\$17	ETB	(no effect)
24	\$18	CAN	(no effect)
25	\$19	EM	(no effect)
26	\$1A	SUB	(no effect)
27	\$1B	ESC	(no effect)
28	\$1C	FS	(no effect)
29	\$1D	GS	(no effect)
30	\$1E	RS	(no effect)
31	\$1F	US	(no effect)

PRINT always emits CR followed by LF at the end of a statement that does not end with a semicolon or comma. INPUT reads characters until it receives CR or LF, and treats BS as "delete the previous character".

## C.2 Printable characters (32-127)

The byte value of each printable character equals its ASCII code. The table below also gives the keyword form of the character for use in PRINT and string assignments.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	\$20	(sp)	56	\$38	8	80	\$50	P	104	\$68	h
33	\$21	!	57	\$39	9	81	\$51	Q	105	\$69	i
34	\$22	"	58	\$3A	:	82	\$52	R	106	\$6A	j
35	\$23	#	59	\$3B	;	83	\$53	S	107	\$6B	k
36	\$24	\$	60	\$3C	<	84	\$54	T	108	\$6C	l
37	\$25	%	61	\$3D	=	85	\$55	U	109	\$6D	m
38	\$26	&	62	\$3E	>	86	\$56	V	110	\$6E	n
39	\$27	'	63	\$3F	?	87	\$57	W	111	\$6F	o
40	\$28	(	64	\$40	@	88	\$58	X	112	\$70	p
41	\$29	)	65	\$41	A	89	\$59	Y	113	\$71	q
42	\$2A	*	66	\$42	B	90	\$5A	Z	114	\$72	r
43	\$2B	+	67	\$43	C	91	\$5B	[	115	\$73	s
44	\$2C	,	68	\$44	D	92	\$5C	\	116	\$74	t
45	\$2D	-	69	\$45	E	93	\$5D	]	117	\$75	u
46	\$2E	.	70	\$46	F	94	\$5E	^	118	\$76	v
47	\$2F	/	71	\$47	G	95	\$5F	_	119	\$77	w
48	\$30	0	72	\$48	H	96	\$60	`	120	\$78	x

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
49	\$31	1	73	\$49	I	97	\$61	a	121	\$79	y
50	\$32	2	74	\$4A	J	98	\$62	b	122	\$7A	z
51	\$33	3	75	\$4B	K	99	\$63	c	123	\$7B	{
52	\$34	4	76	\$4C	L	100	\$64	d	124	\$7C	\
53	\$35	5	77	\$4D	M	101	\$65	e	125	\$7D	}
54	\$36	6	78	\$4E	N	102	\$66	f	126	\$7E	~
55	\$37	7	79	\$4F	O	103	\$67	g	127	\$7F	DEL

Code 127 (\$7F, DEL) is emitted to the terminal like any ordinary byte: the terminal advances one column and the active video font draws whatever glyph it assigns to byte 127 (usually a blank or a small square). DEL is not BS; INPUT does not delete on receiving DEL.

The double-quote character at \$22 cannot appear inside a quoted string literal in source code. To include it in a string, use:

```
10 A$ = CHR$(34)
20 PRINT "HE SAID "; A$; "HELLO"; A$
```

### C.3 High codes (128-255)

Bytes 128 through 255 carry no defined character meaning. IE64 BASIC stores them in strings without modification and the terminal emits them as-is. The glyph (if any) is whatever the active video mode chooses to draw for that byte (see Chapter 5 for VGA text modes, Chapter 8 for ULA, Chapter 7 for ANTIC). The byte value round-trips through ASC and CHR\$ unchanged:

```
10 FOR N = 128 TO 255
20 IF ASC(CHR$(N)) <> N THEN PRINT "FAIL AT"; N
30 NEXT
```

This loop should print nothing.

### C.4 Useful CHR\$ idioms

Expression	Value	Use
CHR\$(13)	CR	Force a carriage return without LF
CHR\$(10)	LF	Advance one line at the current column
CHR\$(13) + CHR\$(10)	CRLF	Standard line ending
CHR\$(8)	BS	Back up one column
CHR\$(34)	"	Embed a double quote in a string
CHR\$(7)	BEL	Writes byte 7 to the terminal and advances the column counter by one. No bell.

## C.5 Worked CHR\$ / ASC examples

---

```
10 PRINT ASC("A")
RUN
65

20 PRINT CHR$(65)
RUN
A

30 FOR N = 32 TO 47 : PRINT N; CHR$(N); : NEXT : PRINT
RUN
32  33 !  34 "  35 #  36 $  37 %  38 &  39 '
40 (  41 )  42 *  43 +  44 ,  45 -  46 .  47 /
```

ASC("") returns 0 (the null terminator of the empty string). CHR\$(n) does not range-check its argument: only the low eight bits of the integer value are stored, so CHR\$(256) produces the same single-byte string as CHR\$(0), and CHR\$(-1) produces CHR\$(255). Keep n in the range 0 through 255 if you want the result you typed.

## C.6 Why there is no PETSCII or alternate code table

---

Intuition Engine BASIC strings are byte sequences. The screen mode in use decides how a non-control byte is drawn, but the byte itself is not translated. PRINT "A" sends byte 65; the active font in the video mode draws byte 65 as "A" (or whatever shape that mode assigns to slot 65). To use an alternate code page, choose a font whose glyphs match the codes you wish to display; the codes themselves never change. See Chapter 5 for VGA fonts and Chapter 4 for character RAM on VideoChip.