A Proposed Set of Mnemonic Symbolic Glyphs for the Visual Representation of C0 Controls and Other Nonprintable ASCII Characters

Michael P. Frank
FAMU-FSU College of Engineering
Thursday, September 14, 2006

Abstract. In this document, we propose some single-character mnemonic, symbolic representations of the nonprintable ASCII characters based on Unicode characters that are present in widely-available fonts. The symbols chosen here are intended to be evocative of the originally intended meaning of the corresponding ASCII character, or, in some cases, a widespread *de facto* present meaning. In many cases, the control pictures that we suggest are similar to those specified in ANSI X3.32/ISO 2047, which has not been uniformly conformed to for representing the control codes in all of the widely available Unicode fonts. In some cases, we suggest alternative glyphs that we believe are more intuitive than ISO 2047's choices.

1. Introduction

The most current official standard specification defining the 7-bit ASCII character set is ANSI X3.4-1986, presently distributed as ANSI/INCITS 4-1986 (R2002). This character set includes 34 characters that have no visible, non-blank graphical representation, namely the 33 control characters (codes 00_{16}-$1F_{16}$ and $7F_{16}$) and the space character (code 20_{16}). In some situations, for readability and compactness, it may be desired to display or print concise graphic symbolic representations of these characters, consisting of only a single glyph per character, rather than a multi-character character sequence.

According to ANSI X3.4-1986, graphic representations of control characters may be found in *American National Standard Graphic Representation of the Control Characters of American National Standard Code for Information Interchange*, ANSI X3.32-1973. We have not yet been able to obtain this particular document online, but an appendix to ANSI/INCITS 4-1986 states that the X3.32 standard is identical to ISO 2047, which we did examine and will discuss below.

The Unicode standard (version 5.0, also some earlier versions) provides a number of code points (2400_{16}-2421_{16}) that are intended for use in graphically representing the C0 control characters and the space and delete characters; however, Unicode itself does not specify how these characters should look. The most widely-supported glyphs for these code points consist merely of the standard abbreviations for the control characters' names rendered in a narrow-width font within a single glyph. For example, in the Arial Unicode MS font distributed with MS Office, the code points intended for depictions of the nonprintable ASCII characters are rendered as follows (shown here in an 18-point bold version, for clarity):

DC_1 DC_2 DC_3 DC_4 NA_K SY_N ET_B CA_N E_M SU_B ES_C F_S G_S R_S U_S S_P DE_L

whereas in the Lucida Sans Unicode font, the same code points are rendered somewhat less compactly, like this:

NUL SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI DLE

DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US SP DEL

In both cases, the available glyphs provide no graphic mnemonic or symbolic depiction of the meaning of the characters, beyond what is already present in their multi-character textual abbreviations, and so these symbols are not much of an improvement beyond simply writing out the abbreviations NUL, SOH, STX, and so forth.

There is actually an available standard which specifies a more mnemonic set of graphical representations: INCITS/ISO 2047-1975, which is indentical to and apparently derived from the earlier ANSI X3.32-1973. Figure 1 is a code table that was found on a random website (without attribution) that shows one rendering of the ISO 2047 standard glyphs for characters 00_{16}-$1F_{16}$. In most cases, the pictures shown here are almost identical to ISO's images, except that ISO's backspace arrow doesn't curve quite as far around. Figure 2 shows an excerpt from the official ISO 2047 standard document (p. 6) showing their recommended glyphs for SP (space) and DEL (delete).

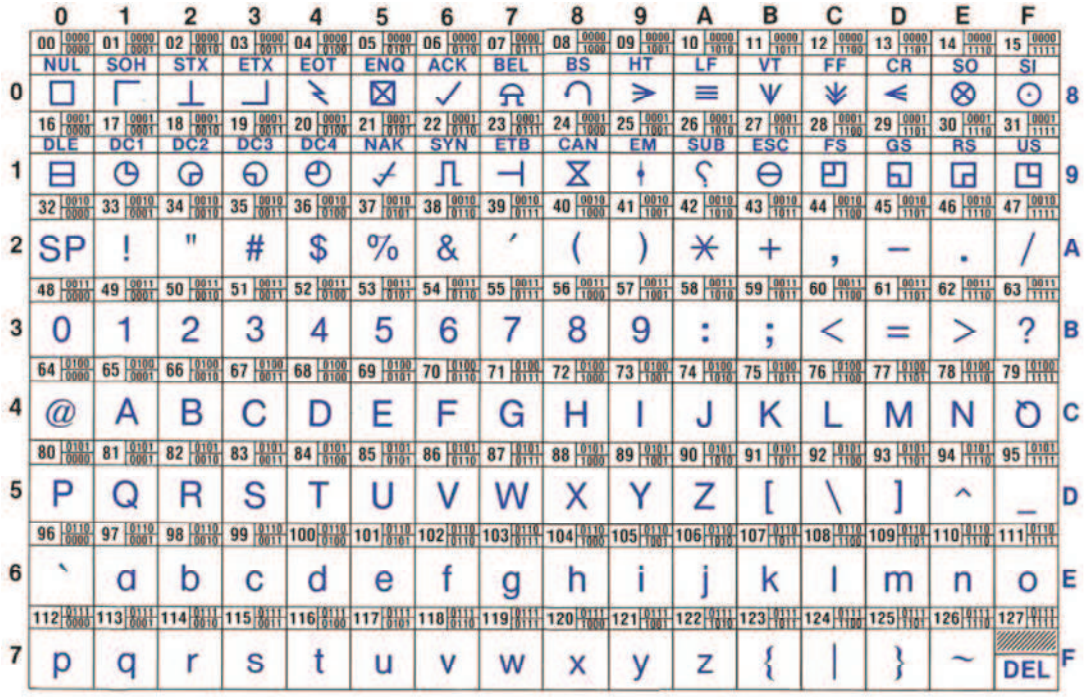


Figure 1. ASCII table including the ISO 2047 standard representations of the C0 controls 00_{16}-$1F_{16}$.

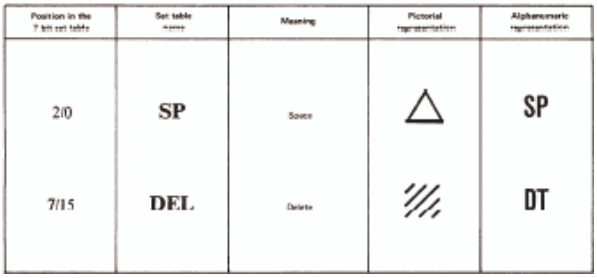


Figure 2. Pictorial and two-character representations of the SP (20_{16}) and DEL ($1F_{16}$) characters, excerpted from ISO 2047.

Now, you might like or dislike ISO 2047's particular choice of symbols, but the bigger problem is that these symbols themselves are not supported (in my experience, at least) by the widely available software for text entry and display. For example, in the Microsoft Word 2003 word processing environment, the only two fonts that came preinstalled that appear to support a large subset of Unicode are Arial Unicode MS and Lucida Sans Unicode. Not only do these fonts fail to use the ISO 2047 symbols to represent the standard Unicode codes for control pictures, as we mentioned earlier, but even worse, most of the ISO 2047 symbols are not to be found anywhere in these fonts at all. In fact, as far as I have been able to tell, quite a few of the ISO 2047 symbols are not specifically mandated to be present even in the complete, latest versions of Unicode. One could try searching for ISO 2047's symbols among available non-Unicode fonts, but this hardly seemed worthwhile, since for maximum portability, we would prefer to stick with widely available fonts that are based on a well-supported standard such as Unicode.

Therefore, it was decided that a worthwhile exercise would be to see if we could come up with an alternative set of glyphs (that is, not conforming to ISO 2047) for representing the nonprintable ASCII characters, using only symbols that are available within the Unicode fonts that are distributed with Microsoft Office, namely the fonts mentioned above. This will permit representations of all nonprinting ASCII characters to be easily entered and depicted within popular Microsoft programs such as Word and Powerpoint, by using the Insert Symbol menu command.

Without further ado, let's proceed to the next section, which presents our suggested symbols. Then, section 3 will present a discussion to justify these choices.

2. The Proposed Symbols

Table 1 below lists the nonprintable 7-bit ASCII characters and the suggested symbols for them. Generally, when we suggest more than one symbol for a given character, the symbols are given in an order that corresponds roughly to the degree to which they are preferred, from most to least preferred. Section 3 explains our choices in detail.

Table 1. ASCII control characters and other nonprintable characters (space, delete) with some proposed new mnemonic symbolic glyph renderings of them. Column 1 shows the hexadecimal code of the character. Column 2 shows the character that, if typed while holding down the Control key, will generate the character (at least on some systems). The third column shows the official abbreviation for the character, from the ANSI X3.4 spec. The fourth column shows the Unicode control picture for the character, as rendered in the Arial Unicode MS font. The fifth column shows one or more proposed choices of suggestive and widely-available Unicode characters that be used to represent the control characters, shown in a proposed ordering from most to least preferred. The next column gives the official

full name of the control character. Finally is the character type, also represented in the background color of the table row, according to OC=other control (magenta), TC=transmission control (red), FE=format effector (green), CE=code extension (yellow), DC=device control (cyan), IS=information separator (blue).

Hex	Ctl	Abbr	UC	Syms	Name of control character	Type
00	@	NUL	NUL	∅	null	OC
01	A	SOH	SOH	⌜ ⌜	start of heading	TC
02	B	STX	STX	⊥	start of text	TC
03	C	ETX	ETX	⌟ ⌟	end of text	TC
04	D	EOT	EOT	↯ □	end of transmission	TC
05	E	ENQ	ENQ	⍰ ʔ ⁉ ¿	enquiry	TC
06	F	ACK	ACK	☑ ✓ ‼ ☺	acknowledge	TC
07	G	BEL	BEL	☎ ☏ ⌓	bell	OC
08	H	BS	BS	⌫ ↶	backspace	FE
09	I	HT	HT	↦	horizontal tabulation	FE
0A	J	LF	LF NL	↓	linefeed, newline	FE
0B	K	VT	VT	↧	vertical tabulation	FE
0C	L	FF	FF	↡ ↓	form feed	FE
0D	M	CR	CR	↲	carriage return	FE
0E	N	SO	SO	⊙	shift out	CE
0F	O	SI	SI	⊗	shift in	CE
10	P	DLE	DLE	⌥ ⌘	data link escape	TC
11	Q	DC1	DC1	◎ ①	device control one (XON)	DC
12	R	DC2	DC2	⊛ ②	device control two	DC
13	S	DC3	DC3	⊖ ③	device control three (XOFF)	DC
14	T	DC4	DC4	⊖ ④ ⊡	device control four	DC
15	U	NAK	NAK	⊠ ☹	negative acknowledge	TC
16	V	SYN	SYN	⏲ ⧖	synchronous idle	TC
17	W	ETB	ETB	⊡ ⊣	end of transmission block	TC
18	X	CAN	CAN	✖ ⊠	cancel	OC
19	Y	EM	EM	■ ▮	end of medium	OC
1A	Z	SUB	SUB	⍰ ¿	substitute	OC
1B	[	ESC	ESC	⇑ ↑	escape	CE
1C	\	FS	FS	𝍣	file separator	IS
1D	]	GS	GS	𝍤	group separator	IS
1E	^	RS	RS	𝍥	record separator	IS
1F	_	US	US	∫	unit separator	IS

Hex	Ctl	Abbr	UC	Syms	Name of control character	Type
20		SP	SP	␣ ƀ	space	FE
7F	?	DEL	DEL	⌦ ▩	delete, rubout	OC/FE

Next, Table 2 below gives a complete table showing the most-preferred glyphs for all the ASCII characters in a convenient two-dimensional code-table format.

Table 2. ASCII code table using special proposed mnemonic glyphs for the nonprintable characters. The color code used is the same as in Table 1.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL ∅	SOH ⌜	STX ⊥	ETX ⌟	EOT ↯	ENQ ⍰	ACK ☑	BEL ☎	BS ⌫	HT ↦	LF ↓	VT ↧	FF ↡	CR ↵	SO ⊙	SI ⊗
1	DLE ⌥	DC1 ◎	DC2 ⊛	DC3 ⊖	DC4 ⊜	NAK ☒	SYN ⍉	ETB ▣	CAN ✖	EM ■	SUB ⍰	ESC ⇑	FS ⌗	GS ⌗	RS ∮	US ∫
2	SP ␣	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL ⌦

Finally, Table 3 below gives the Unicode code points and names of all of our suggested symbols, as well as others that are similar to ISO 2047's pictures (highlighted in yellow). This table may be useful to help the readers of this document in locating these symbols using Microsoft Office's Insert Symbol dialog box, or an equivalent component of their preferred text entry environment.

Table 3. Unicode hex codes and symbol names for some suggested symbolic representations of the nonprinting ASCII control codes. For each character, they are ordered from most to least preferred. Symbols highlighted in yellow are the available symbols that were deemed to correspond most closely to the ISO 2047 proposals. In this document, we do not advocate using all of ISO 2047's suggestions, even where they are available, for the reasons discussed in section 3. However, the available ISO 2047 symbols are included in this chart nevertheless, for purposes of completeness.

Hex	Name	Sym	Unicode	Symbol Name
00	NUL	∅	2205	empty set
		□	2610	ballot box
01	SOH	⌜	2308	left ceiling

Hex	Name	Sym	Unicode	Symbol Name
		⌜	231C	top left corner
02	STX	⊥	22A5	up tack
03	ETX	⌋	230B	right floor
		⌟	231F	bottom right corner
04	EOT	↯	21AF	downwards zigzag arrow
		□	25A1	white square
05	ENQ	⍰	2370	APL functional symbol quad question
		⸘	203D	interrobang
		؟	061F	arabic question mark
		¿	00BF	inverted question mark
		✠	2720	maltese cross
		☒	22A0	squared times
06	ACK	☑	2611	ballot box with check
		✓	2713	check mark
		☺	263A	white smiling face
		☻	263B	black smiling face
07	BEL	☎	260E	black telephone
		☏	260F	white telephone
		⌓	2313	segment
08	BS	⌫	232B	erase to the left
		↶	21B6	anticlockwise top semicircle arrow
09	HT	↦	21A6	rightwards arrow from bar
		➔	2794	heavy wide-headed rightwards arrow
0A	LF	↓	2193	downwards arrow
		≡	2261	identical to
0B	VT	↧	21A7	downwards arrow from bar
0C	FF	⇟	21DF	downwards arrow with double stroke
		↡	21A1	downwards two headed arrow
0D	CR	↲	21B5	downwards arrow with corner leftwards

Hex	Name	Sym	Unicode	Symbol Name
		←	2190	leftwards arrow
0E	SO	⊙	2299	circled dot operator
		⊗	2297	circled times
0F	SI	⊗	2297	circled times
		⊙	2299	circled dot operator
10	DLE	⌥	2325	option key
		⌘	2318	place of interest sign
		⊟	229F	squared minus
11	DC1	◎	229A	circled ring operator
		①	2460	circled digit one
		◔	25D4	circle with upper right quadrant black
12	DC2	⊛	229B	circled asterisk operator
		②	2461	circled digit two
13	DC3	⊜	229C	circled equals
		③	2462	circled digit three
14	DC4	⊖	229D	circled dash
		④	2463	circled digit four
		◕	25D5	circle with all but upper left quadrant black
15	NAK	☒	2612	ballot box with X
		☹	2639	white frowning face
16	SYN	⌚	231A	watch
		⌛	231B	hourglass
17	ETB	▣	25A3	white square containing small black square
		⊣	22A3	left tack
18	CAN	✖	2716	heavy multiplication X
		☒	2327	X in a rectangle box
		⌛	231B	hourglass
19	EM	■	25A0	black square
1A	SUB	🞋	FFFD	replacement character

Hex	Name	Sym	Unicode	Symbol Name
		؟	061F	arabic question mark
		¿	00BF	inverted question mark
1B	ESC	⇑	21D1	upwards double arrow
		↑	2191	upwards arrow
		⊖	2296	circled minus
1C	FS	∰	2230	volume integral
1D	GS	∯	222F	surface integral
1E	RS	∮	222E	contour integral
1F	US	∫	222B	integral
20	SP	␣	2423	open box
		␢	2422	blank symbol
		△	25B3	increment
7F	DEL	⌦	2326	erase to the right
		▩	25A9	square with diagonal crosshatch fill
		▧	25A7	square with upper left to lower right fill

3. Discussion

In this section, we discuss and attempt to justify our choices and recommended order of preference for all of the control code pictures that we listed above.

Null. In many fields, the concept of nullity or nothingness is represented by ∅, the empty set symbol used in mathematical set theory. Since, in 7-bit ASCII, the NUL (code 00_{16}) character is intended to be used as meaningless filler for the empty space or time in between valid symbols, it seemed appropriate to represent it using a symbol that literally means "nothing." Meanwhile, ISO 2047's suggestion of using an outlined square for NUL is already widely used by many software programs to depict otherwise unprintable characters. In this document, we suggest that a square might be better used to represent *end of transmission*, as discussed below.

Start of header. ISO 2047's suggestion for this is good; a pair of lines across the upper and left sides of the character space. This image suggests the upper-left corner of a page. If one were to write a message on a piece of paper, one might naturally start by writing the message header information in the upper-left corner of the paper. Therefore, it seems natural (at least in languages that read left-to-right) to use this image for SOH. In Unicode, we found two symbols that look similar to ISO's, namely ⌈ (left ceiling) and

⌐ (top left corner). We prefer the left-ceiling symbol mostly because the horizontal line is closer to the top of the character space.

Start of text. ISO 2047's suggestion for this looks like it could be a separator between the message header and the message text, which is what STX is defined to be. It is also an upside-down "T", which mentally associates it with the word Text. Therefore, we propose adopting this character as a good choice for STX. It is available in Unicode as ⊥ (up tack), a symbol that is widely used in mathematics, for example to represent the relation "perpendicular to" in geometry.

End of text. This is logically the complement of SOH, as it is intended to end a message. Therefore it is appropriate that it should have the opposite symbol to SOH.

End of transmission. This is a tough one. At first, we thought that a white square might be appropriate, analogously to how mathematicians sometimes use this symbol to end a proof. But then, we decided that we rather liked ISO's lightning-bolt symbol. The lightning bolt suggests an electrical shock, and electricity is what is often used to send transmissions, and an electric spark might result from breaking a wire to end a transmission. However, no lightning bolt symbol was to be found in our subset of Unicode. However, we did find a squiggly downwards-arrow symbol ↯, whose downwards orientation seems to suggest an ending. Therefore, we decided to adopt this symbol, in preference to the white square.

Enquiry. Clearly, ASCII's ENQ symbol, which is used for "Are you there?" polling between sender and receiver, ought to be represented using some sort of question mark. In contrast, ISO's suggestion to use a Maltese cross (✠) or, if this is not available, a square with an X in it (☒) appears to make no logical sense whatsoever. The available symbols containing question marks in Unicode were ⍰ (a question mark inside a white square, which is included due to its use as an operator in the APL programming language), ؟ (arabic question mark, which is backwards to the right-to-left text direction of arabic), ⁈ (interrobang, a rarely-used hybrid of a question mark and exclamation point), and ¿ (upside-down question mark used in Spanish). At first, we liked the interrobang, since the ENQ is in the nature of a question that demands an answer. But, it was decided that the APL quad-question operator ⍰ should be preferred, due to its similarity in appearance to our recommended symbols for ACK and NAK. The three characters ENQ, ACK, and NAK are intended to be used together in handshaking protocols. The square represents the check-box or ballot box where the answer to the "Are you there?" question should be filled in. When there's a question mark there, that's the ENQ character. When the receiver answers yes (ACK) or no (NAK), that's a check-mark or X, respectively, thrown into the ballot box. Thus, our recommendations for these three symbols go together. Other possibilities include the Arabic question mark, although this can cause difficulties in typing because it is intended to be used in right-to-left text; in some environments, it may cause the cursor to move in the wrong direction. Also, ISO suggests a backwards question mark to represent SUB instead, and we'd like to avoid confusion. The inverted question mark of spanish is also a possibility, but the problem there is that it is a page-0 code point in Unicode ($00BF_{16}$) and so it is already used to

represent character number BF_{16} in internationalized 8-bit ASCII character sets. We'd prefer to avoid using any existing 8-bit ASCII symbols in our proposal, so that our symbol selection can be used unambiguously to represent control codes appearing in 8-bit ASCII character streams, as well as the usual 7-bit ASCII. Thus, ¿ is also ruled out, and so interrobang remains the best alternative if quad-question isn't available.

Acknowledge. We approve of ISO's quite logical suggestion to use a check-mark to represent this transmission protocol function. However, we suggest that even better is to put the check-mark inside the same white square that we propose using around the symbols for ENQ and NAK; this visually ties the three symbols together. See the discussion of ENQ above. Other logical possibilities for representing the positive, yes semantics of ACK include smiley-face characters, of which there are two in Unicode, the white smiling face ☺ and the black smiling face ☻. However, we felt that these looked too silly, and the black face especially stands out too strongly. Also, in some IBM extensions to ASCII, used in old EGA text-based CRT drivers, the white and black smiley faces are how the ASCII characters 01_{16} and 02_{16} (SOH and STX) are rendered on a monitor screen! When possible, we'd prefer to avoid any confusion with such pre-existing 8-bit ASCII extensions.

Bell. ISO's suggestion for this is decent in that it looks vaguely like a bell, but it isn't an available symbol in Unicode. The closest thing to it is ⌓ (code 2313_{16}), which is the top half of a circle. However, the similarity is not too close, and ⌓ is anyway perhaps too abstract-looking. We suggest instead using Unicode's black telephone character ☎. It stands out visually, but this is appropriate, since BEL is a character that is meant to stand out, and to physically ring a bell. There is also the cute coincidence that Alexander Graham Bell is the inventor of the telephone, and that old telephones rang by actually ringing a physical bell. The white telephone ☏ is also a possibility, but it doesn't stand out as well.

Backspace. ISO's symbol (for which the closest Unicode analogue we found is ↶) is not too bad, since it visually depicts the action of moving the cursor or print head backwards one space. However, in most modern computer-based text entry systems, and even on many electric typewriters having whiteout ribbons, the backspace key not only moves the cursor backwards, but furthermore actually erases the previous character. Therefore, we prefer the symbol ⌫ (erase to the left), which literally means this, and is actually how the backspace key is labeled on some keyboards.

Horizontal tabulation. Unicode's symbol looks a lot like →, which is a wide rightwards-pointing arrow character from the Wingdings 3 font, and which can be created in Word by typing the simple character sequence --> (dash, dash, greater than). However, this is not a Unicode character. There are right arrows in Unicode, but they are not as fat. Therefore, we instead decided to use ↦, rightwards arrow from bar, where the small vertical bar suggests movement that involves a definite position, *i.e.*, a tab stop. It would be a more accurate depiction of HT's function if the little vertical bar were at the head of the arrow rather than its tail, but hey, we do what we can. If this character is not

available, or if one prefers to stick closer to ISO's suggestions, we suggest ➔, the "heavy wide-headed rightwards arrow," although this one may stand out too much.

Line feed. ISO's suggestion appears to be an abstract depiction of the horizontal lines of text on a printer. Although vaguely appropriate, it does not suggest actual movement of the print head or cursor position, which is what LF does. Therefore, we suggest using a simple downwards-pointing arrow ↓ (code 2193_{16}), since this is the direction that the print head is supposed to move relative to the paper, and is also the direction that the cursor moves relative to the text in a more modern CRT terminal or word processor program that interprets LF characters. If you'd rather use ISO's symbol when possible, the "identical to" operator ≡ looks pretty similar to it.

Vertical tabulation. The symbol for this should obviously be similar to the symbol for HT, due their analogous function. Conveniently, there is a ↧ (downwards arrow from bar) symbol in Unicode. This goes with our HT symbol, and justifies out not having used a plain rightwards arrow for HT, since the analogous plain downwards arrow is already being used for LF.

Form feed. ISO's double-headed downwards arrow symbol for this makes sense, since form feed moves the print head down a lot. And there is a Unicode symbol ↡ (code point $21DF_{16}$) that looks similar. However, we thought that an even better representation would be ↡, "downwards arrow with double stroke." The double stroke can represent the bottom of one page and the top of the next, and the arrow is the print head moving from near the end of the first page to near the start of the next. I can't think of a more direct depiction of FF's intended function.

Carriage return. The obvious choice for this is the ↵ symbol that already decorates the RETURN or ENTER key of many keyboards. The actual semantics of CR in ASCII does not specify moving downwards (only left), but there are enough systems that move the cursor down simultaneously, at least during text entry, that this representation seems reasonable. If it is not available, or if one's system doesn't advance the cursor upon CR, then one may wish to use a plain left arrow ← instead, although this suggests an overly strong analogy with the function of LF (↓); the CR moves *all the way* left, whereas LF only moves *one line* down. The ideal symbol would be a leftwards arrow that approaches a vertical line representing the left edge of the page, but this is not available.

Shift out. For the shift out and shift in characters, which invoke alternate character sets, ISO's suggestions of ⊗ (circled X) ⊙ (circled dot) are pretty good, but I would suggest swapping them. The reason is that in physics, a circled dot standardly represents a magnetic field vector coming out of the page, wheras a circled X represents a vector going into the page. Thus, ISO's suggestions as to which symbol means "in" and which means "out" seem backwards. With this fixed, I endorse ISO's suggestions, and they are already available in Unicode.

Data link escape. This character is intended to initiate escape sequences meant for advanced transmission control functions. ISO's suggestion for this is a square with a

horizontal line across the middle. This seems rather abstract and does not evoke for me the concept of what DLE is supposed to do. I thought a better choice might be a symbol that is already associated with the concept of invoking new options. This brought to mind the symbol ⌥ that decorates the Option key on some Macintosh keyboards. It's especially suggestive of "data link escape" because the broken horizontal line at the top suggests the "data link" whereas the diagonal part coming down could represent "escaping" from it to process some special escape sequence. Anyway, it's the best thing I found. Another possibility is Apple's Command key logo ⌘ (place of interest sign), but it does not seem quite as appropriate.

Device controls one through four. ISO's symbols for device controls 1-4 look like 25%/75% pie charts with the 25% part in different quadrants. These symbols are very abstract, and they exaggerate the amount of symmetry between these functions. The ASCII spec does not treat DC1-4 completely symmetrically, but rather specifies particular functions that are supposed to be used for each one, if needed. The pie chart symbols are also not available in the Arial Unicode MS font, although they are actually available in full Unicode (in code points 25F4-25F7). Anyway, I greatly prefer to use the convenient four Unicode symbols ◎, ⊛, ⊜, ⊖ which are already available (and are even adjacent and in the right sequence!) in code points 229A-229D.

Device control one. The circled-circle ◎ looks good for DC1 (XON), which is supposed to turn on some device at the receiving end, since it looks like a push-button to turn something on, or we can think of the inner circle as being an "O" for "On."

Device control two. DC2 is supposed to put the device in some special mode. The symbol ⊛ looks good for this, with the asterisk representing "something special."

Device control three. DC3 (XOFF) is a "secondary stop" that is supposed to just pause the receiving device (rather than turning it off completely). Well, the double lines inside ⊜ look a lot like the pause symbol that one sees on audio/video playback devices. Also, the fact that there are two of them suggests a secondary level stop. The only improvement might be if the two lines were vertical (like a pause button) instead of horizontal.

Device control four. DC4 is supposed to turn off the remote device. ⊖ is perfect for this, since like ◎, it also looks like an on-off button, but one where the inner "O" has collapsed or closed shut, like a closed eyelid. Also, the fact that there is only one horizontal line instead of two suggests that it is a "primary stop" in contrast to DC3's "secondary stop."

If these characters are not available, or for applications where the device control functions assigned to DC1-DC4 are more arbitrary than the ones suggested by the ASCII spec, the characters ①, ②, ③, ④ would seem a nice straightforward representation.

Negative acknowledge. ISO suggests a checkmark with a line through it, which is a nice counterpoint to ACK's checkmark, but is unfortunately not available in Unicode. We suggest instead using ☒, which is a no or "X" answer to the ballot box question posed

by ⍰ (ENQ). If smiley-face ☺ is used instead for ACK, then clearly one should use frowny-face ☹ for NAK.

Synchronous idle. ISO's suggestion looks like a square-wave pulse. This suggests the idea of synchronization nicely, but isn't available in Unicode. We suggest instead using the stopwatch (⏱) or hourglass (⧗) icons to represent the idle passage of time, although they are both a little cutesy. Of these two, the stopwatch icon does a better of evoking the idea of discrete-time synchronization between sender and receiver.

End transmission block. ISO's suggestion, a leftwards-pointing tack ⊣, is not too bad; the line coming from the left representing the transmission coming to the edge or end of a particular block of data. It also associates ETB visually with EOT (⊥), which is another transmission control character. So, using this symbol is not a bad choice. But we also liked ⧈, a filled square inside an empty square. The filled square represents the "block" of data, and the empty square around it represents finality, like the QED tombstone □ that ends a proof. Take your pick.

Cancel. ISO's symbol, which looks like an abstract depiction of an hourglass, doesn't seem to make very much sense. CAN is supposed to literally cancel or withdraw some preceding information, such as perhaps the entire content of the current message, transmission block, or maybe even the entire transmission. This is a fairly drastic action, and it requires a bold, direct presentation. The best icon I found seemed to be ✖, the bold, black X. It stands out from the crowd, as a cancel character should.

End of medium. ISO's symbol is a vertical line with a dot in the middle. This doesn't make much sense, and isn't available in Unicode. I thought a better symbol would be a filled square ■, which is a bold statement of finality, a filled tombstone symbol that is sometimes used to end a proof in mathematics. It is a more final version of the empty square □ that might be used to mean "end of transmission." If the medium is over, then that's it! However, EM can be used to mark the end of just the *used* part of the medium, which gives it a close analogy to EOT.

Substitute. ISO suggests a backwards question mark. This is actually available in Unicode as an Arabic question mark, but this character presents problems with cursor movement which we discussed earlier. We suggest instead using the Unicode replacement character  . This is in fact the perfect representation, since the replacement character in Unicode serves exactly the same purpose as SUB is supposed to in ASCII – that is, it is intended to be a replacement for characters that are found to be illegal, invalid, or unknown. Another possibility is the upside-down question mark ¿, but that would be conflict with its existing use in 8-bit ASCII.

Escape. ISO suggests a circle with a horizontal line across it. This is similar to their symbol for DLE, which is appropriate, but both symbols are too abstract. We suggest instead an upwards arrow ↑ or upwards double arrow ⇑. The upwards arrow suggests rising up to escape from the context of the literal data stream or application within which one is embedded. Of these two symbols, the double arrow is probably

better, to avoid a false perception of similarity to the single arrows that we propose using to represent the formatting characters.

File separator. The ISO symbols for the information separators, like their symbols for the device controls, give a mistaken impression of some kind of "rotational symmetry" between the four functions. But actually, the ASCII file separators are hierarchical; there is a definite ordering associated to them, a "highest" and a "lowest," which is not apparent in the fourfold symmetry of the ISO symbols. Also, the ISO symbols were not available in my fonts, although they are present in full Unicode, in code points 25F0-25F0. The best representation I found in my subset of Unicode was a series of integral operators. The higher-dimensional integrals correspond to higher levels in the information grouping hierarchy. FS is the highest level, so it gets the triple volume integral ∰. A quadruple integral might have been better, but it wasn't available. A bonus is that the integral symbol looks like the "S" in "separator," which is the case for historical reasons, because it originally stood for "Sum." Integrals sum things up, and the information separators say, "this is/was the sum total of everything contained in this particular information unit." The oval makes it clear that even the triple integral is a single glyph.

Group separator. One hierarchy level below the file separator, the group separator can be represented by a surface integral ∯. The oval "groups" the two integral symbols together.

Record separator. One level below the group separator, RS can be written with a countour integral ∮. The circle suggests the completeness of the record's information about its contents.

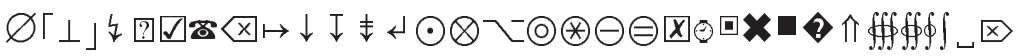
Unit separator. At the lowest level, we have the unit separator, represented by a plain integral ∫. This is the most common separator, and it has the simplest symbol.

Space. In my experience, the most common visible typographic symbol used for representing a space is the open box, ␣. The alternate symbol ␢ (blank) is also provided by Unicode, but seems less intuitive. ISO suggests using the triangle or increment symbol △, which may make some kind of sense, since the space character increments the cursor position, but it is visually less evocative of empty space than is the open box.

Delete. Code point $7F_{16}$ originally meant RUBOUT, or the manual nullification of a character by poking out all the holes in its area of a punch card or paper tape to change it into a 1111111_2. ISO's hatched box ▧, or something like it, has actually been used in many systems to depict the rubout character. However, nowadays, $7F_{16}$ is called DEL and, when used for text entry, it is more likely to delete the next character to the right. Therefore, the Unicode ⌦ (erase to the right) character seemed most appropriate for it. However, I wouldn't mind if you decided to use ▧ instead. There's also a darker-looking double-crosshatched box, ▩, which looks a little bit more like most of the RUBOUT glyphs that I've seen used in practice.

4. Conclusion

In this document, we proposed that, when necessary and possible, the traditionally non-printable ASCII characters 00_{16}-$1F_{16}$, 20_{16}, and $7F_{16}$ ought to be represented, in systems supporting Unicode subsets that are at least as complete as is the Arial Unicode MS font, by the following symbols, respectively:



These symbols were chosen based on the criteria of (1) availability in the Arial Unicode MS font, which is widely distributed as part of Microsoft Office 2003 (as well as some earlier versions), (2) appropriateness for conveying the control characters' semantics as specified by ANSI, (3) semantic appropriateness of the visual relationships between the characters, and (4) their similarity to the standard ANSI X3.32/ISO 2047 symbols where those symbols make sense, with these criteria being applied in roughly the given order of priority. We also explained in detail our rationale for each choice of symbol.

Improvements on this proposed system are doubtless possible; and this system may in fact be obsoleted by future standards, and/or by extensions to Unicode and/or the character sets that are supported on widely-available platforms. Furthermore, many of the control characters (particularly the transmission control characters) in the original 7-bit ASCII code are themselves largely obsolete today, being either rarely used, or used for some purpose that is completely unrelated to the one originally dictated by the ANSI X3.4 spec. Still, a certain fraction of legacy systems (and even some occasional new systems) may continue to use these old codes for some time to come, and for purposes of describing these systems' behavior using modern word processors, the symbol set proposed above may be useful, at least for a while.