C0 and C1 stability for Unicode and 10646

Kent Karlsson
Stockholm
2022-01-10

Individual contribution

Introduction

Lately there have been a number of quite destructive, from a standard and standards use perspective, suggestions and proposals for the so-called C0 and C1 areas. Effectively, some propose to regard these areas as wholly private use, and one can assign whatever ill-designed set of control codes there; some do not even have line feed... This is very strange as this goes against the very grain of Unicode and ISO/IEC 10646. These areas cannot be regarded as some kind of private-use areas. There are many important characters allocated in these areas, that are never expected to change, characters that are either near-universally supported, or has special properties in Unicode (for bidi and line-breaking at least). Just about every process that handles Unicode (or indeed many other encodings) rely on that several of the C0 (in particular) and C1 characters do not suddenly get some other semantics.

No-one is expecting that U+0009 would be anything else than CHARACTER TABULATION, no-one is expecting that U+000A be anything else than LINE FEED, etc., in a Unicode encoding, no matter how weird or old the "original encoding" (to have a mapping to Unicode, perhaps proposing addition of some so-called "printable" characters to the Unicode standard). But there are weird encodings. Teletext has no line feed, no carriage return, no character tabulation, … Instead, the *entire* C0 (yes, C0, not C1) is repurposed to have graphic characters with control functionality (code page switching, colour change, …).

True, most C0 and C1 ("control") characters are today generally uninterpreted. And some of them are really obsolete, especially in a Unicode context. Like SHIFT IN and SHIFT OUT, which make sense only in a code page switching context, or the INFORMATION SEPARATORs, which were intended for a kind of data structure linearisation that is, AFAIK, not used anymore (but they do have non-default properties in the Unicode character database).

There are actually some "private-use" control codes in the C0/C1 area: DEVICE CONTROLs, though DC1 and DC3 are de-facto standardized for flow control in terminal (emulators), leaving DC2 and DC4, and then also PRIVATE USE 1 and 2.

But there are some character encodings that are "a bit crazy" when it comes to control codes. They override all of C0 (or C1) with something that cannot even be regarded as pure control characters. In particular Teletext (but there are apparently others, also apart from EBCDIC encodings). Teletext has in the C0 range "control codes" where most of them serve three purposes at once 1) a graphic character (usually a SPACE, but under some conditions it could be a "mosaic" character), 2) code page switching often to the default "alphanumeric" code page (set elsewhere in the Teletext

1

protocol), or the "mosaic" code page (but there are some that do other cope page switching), 3) sets a colour display mode. The Teletext encodings have no CHARACTER TABULATION, HT is just not used. And in Teletext there is no LINE FEED (or any other NLF), the Teletext *protocol* instead has explicit line numbers.

L2/21-235 'Proposal to add further characters from legacy computers and teletext' suggests that:

"Control characters from microcomputer platforms and teletext were also determined to be out of scope for the UCS. These characters were located in what would today be considered the C0 control range (0x00–0x1F) or the C1 control range (0x7F–0x9F). Processes that need to interchange these codes should simply interchange the binary C0 or C1 value, extended to the UCS code space but without further mapping. Emulators should treat these control codes as appropriate for the targeted environment."

This must be the *absolutely worst* suggestion in the history of Unicode by an unspeakably enormous margin. Literally! Firstly, there is no such thing as an (isolated) target environments in this solar system. In particular, Teletext is not an isolated target system at all; it is very widespread and has connections to web pages and smart phone apps. The suggestion is detrimental to interoperability, for all systems that use Unicode and to the Unicode standard itself. Secondly, no-one, *absolutely no-one*, has "processes" that expects the C0/C1 areas to be private-use areas in any way whatsoever.

Teletext

The suggestion above is an especially bad idea for converting Teletext text to Unicode. Teletext has "control codes" (they aren't quite control codes, they are spacing *graphic* characters, with two other functionalities; they are *usually* rendered as a SPACE, but in some circumstances rendered as a "mosaic" character) in the C0 area (or moved to the C1 area, but that does not work in Teletext, there is no 8-bit code page in Teletext, all are 7-bit using the 8th bit as parity). For instance, 0x0A, which is used for "End box" spacing control (usually rendered as a SPACE, and used for subtitling via Teletext, as well as news flash functionality), would be mapped to U+000A, LINE FEED, which absolutely no-one expects to be anything else than LINE FEED in any context. And 0x00 which is used for "Alpha black", which switches to the G0, sets the foreground colour to black, and is (usually) displayed as a SPACE; 0x00 would be turned into U+0000, NULL, which absolutely no-one expects to be anything else than NULL, which in addition is often used as string terminator; which is inaccurate but extremely widespread. Indeed, the entire C0 is repurposed in Teletext (the one called ESC is not an escape, it switches to an alternative G0/G2).

So, what about LINE FEED (or CARRIAGE RETURN)? Those C0 codes are mapped to something entirely different in Teletext. How represent multiple lines (in a page)? Is there some other code used for LINE FEED or similar? No. In addition to page numbers (and subpage numbers), the Teletext *protocol* has explicit line numbers (within the page). (The Teletext protocol is a quite complicated protocol; not a design I would recommend…)

So, is this a disaster for Teletext when outside of the box (TV set)? No, of course not. Nobody, absolutely nobody, follows the disastrous suggestion quoted above. Teletext is displayed via several handfuls of web pages (and smartphone apps). Generally, the Teletext pages are converted to HTML pages, with styled text for the text in the pages; though some use generated images (more and more it seems, but not a technical necessity).

This image is taken from one such web page:

Teletext colours

L2/21-235 'Proposal to add further characters from legacy computers and teletext' states:

"==4. Teletext. Teletext was a service invented in the United Kingdom in the early 1970s for broadcasting pages of information, generally text and simple block graphics, to analog television receivers via the vertical blanking interval. Teletext found its greatest popularity in Europe, where it was commonplace until the adoption of digital television; almost all analog television sets sold in Europe since the early 1980s had built-in teletext decoders.=="

While most of that is true, the use of *past tense is not quite accurate*, Teletext is still in use. Nor is the use of "until". Teletext is supported in all manufactured TV sets and "TV boxes", also for digital television. In addition, Teletext pages are commonly converted to web pages (using HTML, usually this was with styled text, rarely with images, but the latter has become more common), and smart phone apps (again using HTML). Teletext pages can also be converted to ECMA-48 (fast-tracked ISO version is ISO/IEC 6429) using control sequences for the colour changes, both the ones inline and the ones out of line (in the Teletext protocol, for additional colours).

Below is an excerpt from [Teletext] regarding colours (CLUT is colour lookup table, 2-bit reference number, Entry number (in the CLUT) has a 3-bit reference number). (Leaving out the details of how these can used in the Teletext protocol, as that is out of scope for this proposal, and quite intricate). Note that there is one added (not part of the quote) column, for colour swatches. Some of the colour entries can be modified (at conformity/presentation level 3.5), and thus corresponds to an ECMA-48 colour palette entry; a converter to HTML/CSS need to keep more directly track of the settings of the Teletext CLUTs, and that approach of course can be used for a converter to ECMA-48. *Some* of the colours can be invoked by using inline spacing "controls", but only within the Teletext protocol. We

will basically ignore the complications of "conformity levels" ("presentation levels"). It is obvious that the Teletext protocol is overcomplicated, and nothing anyone should try to mimic in any replacement, in particular not in Unicode.

The quote below is included here just to establish that Teletext (at conformity levels greater than 2) has more colours than the ones available via the spacing "control" codes (which aren't really control codes, but serves three functions in one). The additional colours are used via a kind of out of line formatting objects, that reference substrings of the page. At conformity level 3.5 there is a mechanism for redefining the colours in this colour palette. All text colour settings (background and foreground) are reset to default colours (black (or transparent) background, white foreground/text) at end of line. For details, see [Teletext].

12.4 Colour Map

Table 30: Colour Map

CLUT	Entry Number	Default Colour	Default Values			Swat-ches	Comments
			R	G	B		
0	0	Black	0	0	0		Fixed at Levels 1, 1.5 and 2.5 Re-definable using X/28/4 or M/29/4 at Level 3.5
	1	Red	15	0	0		
	2	Green	0	15	0		
	3	Yellow	15	15	0		
	4	Blue	0	0	15		
	5	Magenta	15	0	15		
	6	Cyan	0	15	15		
	7	White	15	15	15		
1	0	Transparent	-	-	-		Valid at Levels 2.5 and 3.5 (fixed) Valid at Levels 2.5 and 3.5 Fixed at Level 2.5 Re-definable using X/28/4 or M/29/4 at Level 3.5
	1	Half red	7	0	0		
	2	Half green	0	7	0		
	3	Half yellow	7	7	0		
	4	Half blue	0	0	7		
	5	Half magenta	7	0	7		
	6	Half cyan	0	7	7		
	7	Grey	7	7	7		
2	0		15	0	5		Valid at Levels 2.5 and 3.5 Re-definable using X/28/0 Format 1 or M/29/0
	1		15	7	0		
	2		0	15	7		
	3		15	15	11		
	4		0	12	10		
	5		5	0	0		
	6		6	5	2		
	7		12	7	7		
3	0		3	3	3		Valid at Levels 2.5 and 3.5 Re-definable using X/28/0 Format 1 or M/29/0
	1		15	7	7		
	2		7	15	7		
	3		15	15	7		
	4		7	7	15		
	5		15	7	15		
	6		7	15	15		
	7		13	13	13		
NOTE:	The individual R, G, B levels are variable in 16 equally spaced steps. A value of 0 represents zero intensity and a value of 15 (decimal) represents full intensity. The levels are not gamma corrected.						

Teletext styles (other than colour)

Teletext in origin has only fixed-width, upright, "normal" weight style. But at conformity level ("presentation level") 3.5 there is also support for bold, italics and proportional spacing. There is also support for underlined text, but not quoting that section here.

This text styling is done via out of line formatting objects, that reference a starting point in the line.

01110	**Font Style**								
	This command specifies the appearance of the text with respect to italics, bold and proportional spacing. The address field defines the column at which the style(s) starts. The effect of this attribute persists to the end of a display row unless overridden by a further Font Style command.								
		D6	D5	D4	D3	D2	D1	D0	
		R2	R1	R0	Reserved	Italics	Bold	Proportional Spacing	
	The functions controlled by bits D0, D1 and D2 are enabled when the bit is set to '1', and cancelled when it is set to '0'. Bits D6 - D4 allow the font style to be extended to the following 0 to 7 (maximum) rows, at the same column positions.								

The quote above is included here just to establish that Teletext (at conformity levels greater than 3) has more styles than just upright, fixed-width, "normal" weight. This styling is not mediated by control characters, but out of line with the text, in formatting "objects". For details see [Teletext].

Teletext protocol and code page switching

Teletext relies heavily on code page switching via the "control" characters (again, three functions in one code), and setting of code pages in the Teletext protocol. All the code pages are 7-bit, using the 8th bit for parity. The G2 page is a "mosaic graphics" page. The "alphanumerics" page is set in the Teletext protocol to one of the "language variants" of ISO/IEC 646, or a Hebrew page, or an Arabic page.

Teletext "control codes" to ECMA-48 mapping

Teletext pages are commonly converted to HTML/CSS. This could be done (and was done) by using the styling mechanisms of CSS. But now they are often converted to images (embedded in HTML), presumably for greater fidelity with the "look" of Teletext pages on TV sets.

But an alternative is to convert the Teletext styling to ECMA-48 (ISO/IEC 6429) styling. Most of which do not really require any additions to ECMA-48, except for specifying the colours better. There is one functionality in particular that *does* require a substantial addition to ECMA-48 styling: for the Start box and End box. These are used for the subtitling (and news flash) functionality of Teletext, and has no close corresponding control sequences in ECMA-48. (An addition is also needed for more directly mapping Teletext's strange way of setting background colour.) Here is a quote from my proposal to extend the ECMA-48 styling functionality, focussing on converting Teletext text to use ECMA-48 styling:

11. Converting Teletext styling to ECMA-48 styling

Teletext is still in common use around the world, especially for optional subtitling, though the use of Teletext for news pages has declined or even been abandoned. Teletext allows for certain style settings, mostly to do with colour. There are also control bits in the Teletext protocol for handling bold, italic and underline. The colour codes also select which set of characters to use (either alphanumeric or "mosaic" characters, the latter are Teletext specific symbols used to build up larger

(and crude…) graphics). Note that the coding for "alphanumeric" characters have multiple variants for G0/G2, various Latin subsets, Greek, Cyrillic, Arabic and Hebrew. These are selected by control bits in the Teletext protocol, in addition **ESC** can be used to switch between a primary and a secondary G0+G2 set. Further, the "mosaic" characters (G1) have two variants, a "contiguous" form (default, and selected by **0x19**) and a "separated" form (selected by **0x1A**). From a modern encoding point of view the latter are separate characters (not yet in Unicode). *Teletext also allows for dynamically defined ("bitmapped") fonts of unspecified charset.*

The table is a rough sketch only. Teletext overrides (italics, bold, underline, more colours, prop. font, G3 chars.) are not covered in the sketch mapping here, though the functionality is covered. Character sets are specified in control bits in the Teletext protocol, that is not covered here.

Teletext	ECMA-48 (as extended here)
0x00 Alpha black (and sets "alpha mode", G0) (default)	**SP**? **CSI 30m**
0x01 Alpha red (and sets "alpha mode", G0)	**SP**? **CSI 91m**
0x02 Alpha green (and sets "alpha mode", G0)	**SP**? **CSI 92m**
0x03 Alpha yellow (and sets "alpha mode", G0)	**SP**? **CSI 93m**
0x04 Alpha blue (and sets "alpha mode", G0)	**SP**? **CSI 94m**
0x05 Alpha magenta (and sets "alpha mode", G0)	**SP**? **CSI 95m**
0x06 Alpha cyan (and sets "alpha mode", G0)	**SP**? **CSI 96m**
0x07 Alpha white (and sets "alpha mode", G0)	**SP**? **CSI 37m**
0x08 Flash	**SP**? **CSI 5m** or **SP**? **CSI 6m**
0x09 Steady (default)	**CSI 25m SP**?
0x0A End box (default)	**SP**? **CSI 112:0m** (allow 100% fading)
0x0B Start box (news flashes; also used for subtitles)	**SP**? **CSI 112:1m** (block fading)
0x0C Normal size (default)	**SP**? **CSI 73:1:1m**
0x0D Double height (note: background & foreground overshadows text on the next line)	**SP**? **CSI 73:2:1m** (overshadowing is implementation defined)
0x0E Double width	**SP**? **CSI 73:1:2m**
0x0F Double size (note: extends down one "row" (overshadows text on the next line))	**SP**? **CSI 73:2:2m** (might not overshadow text on the next line)
0x10 Mosaics black (and sets "mosaics mode", G1)	**SP**? **CSI 30m**
0x11 Mosaics red (and sets "mosaics mode", G1)	**SP**? **CSI 91m**
0x12 Mosaics green (and sets "mosaics mode", G1)	**SP**? **CSI 92m**
0x13 Mosaics yellow (and sets "mosaics mode", G1)	**SP**? **CSI 93m**
0x14 Mosaics blue (and sets "mosaics mode", G1)	**SP**? **CSI 94m**
0x15 Mosaics magenta (and sets "mosaics mode", G1)	**SP**? **CSI 95m**
0x16 Mosaics cyan (and sets "mosaics mode", G1)	**SP**? **CSI 96m**
0x17 Mosaics white (and sets "mosaics mode", G1)	**SP**? **CSI 37m**
0x18 Conceal (ignored on the currently displayed page when user presses a 'reveal' button)	**CSI 38:8m SP**? (*not* **CSI 8m**)
0x19 ("mosaics mode") Contiguous mosaic graphics (default); note: does *not* unset underlined in "alpha mode"	(converter change) **SP**?
0x1A ("mosaics mode") Separated mosaic graphics; note: does *not* set underlined in "alpha mode"	(converter change) **SP**?
0x1B Escape (toggles between two G0 code pages)	**SP**? (converter change)
0x1C Black background (default)	**CSI 40m SP**?
0x1D New background (foreground to background)	**CSI 48:6m SP**?

0x1E Hold mosaics (details of hold/release mosaics is beyond the scope of this paper)	(**SP***?* **:=** *<mos. char>*; **SP***?*
0x1F Release mosaics (details of hold/release mosaics is beyond the scope of this paper)	**SP***?*; **SP***?* **:= SP**
line break (implicit; Teletext has numbered "rows") (style settings are reset at the beginning of each row)	*line break* (CRLF/CR/LF), **CSI 0m**; **SP***?* **:= SP**

As you can see, the "triple functionality" control codes can be converted to using Unicode with ECMA-48 control sequences for the colour changes as well as "subtitle boxing" (with a proposed extension to ECMA-48). There is no need for new control characters, or redefining any control characters. The Teletext "control" characters are so deeply rooted in the Teletext *protocol*, which uses code page switching extensively, that they do not even make sense outside of that protocol, and must be converted anyway.

I do not know the storage format that broadcaster's Teletext systems use. But it is surely not in the Teletext *protocol*, but something *converted* to that protocol *when the pages are sent*. The storage formats may well use "escapes" (in the programming language sense; like \g for alpha green, say). There is no need for "Teletext control codes" in the storage formats. The entire idea of using Teletext control codes outside of the Teletext *protocol* is bogus and ill-conceived. But if converted to use ECMA-48 styling, then *that* is a possible storage (archival) format.

Teletext and optional subtitling (and news flashes)

Subtitles (in Teletext) are rarely, if ever (haven't seen that), converted to web pages. Subtitling needs timing information (when to show the subtitles). This is not part of the Teletext protocol but must be part of the storage format for Teletext subtitling (again, I'm not familiar with the storage formats used, but they (or it) are likely proprietary for a particular system).

News flashes, a good idea, but haven't seen that used, are more one-off (show now, …, stop showing), not related to running time of the TV program. But on the Teletext protocol side news flashes would use the same "box" controls as subtitling (plus that there are protocol bits saying whether the page a "normal" page, a subtitling page, or a news flash page).

Teletext and digital TV broadcast/multicast (DVB, IP-TV)

The original Teletext data was transmitted in the so-called synch lines of the analogue TV signal. For the digital TV broadcasts (and multicasts, a.k.a. IP-TV), the Teletext data is embedded in the digital TV signal, but not in the synch lines as there are no sync lines, indeed no scan lines at all. See reference [Teletext in DVB] for details of that embedding of Teletext in DVB (Digital Video Broadcasting).

Teletext references

Teletext is not dead, quite yet. But it is admittedly on the decline. There are few and fewer broadcasters that provide any Teletext news service. Teletext is still used for optional subtitling, for accessibility reasons. It may continue to be in use for optional subtitling for several years to come. In DVB there is a "newer" mechanisms for subtitling (apart from Teletext). But has still to catch on, it seems.

[Teletext]	*Enhanced Teletext specification*, ETSI EN 300 706 V1.2.1, https://www.etsi.org/deliver/etsi_en/300700_300799/300706/01.02.01_60/en_300706v010201p.pdf, 2003.

[Teletext in DVB]	Digital Video Broadcasting (DVB); *Specification for conveying ITU-R System B Teletext in DVB bitstreams*, ETSI EN 300 472 V1.4.1, https://www.etsi.org/deliver/etsi_en/300400_300499/300472/01.04.01_60/en_300472v010401p.pdf, 2017.
[TeletextWebList]	http://teletext.mb21.co.uk/live.shtml (many links are now defunct, due to the decline of use of Teletext for news pages).
[WikiTeletext]	https://en.wikipedia.org/wiki/Teletext, https://en.wikipedia.org/wiki/Teletext_character_set

Some other legacy encodings for C0/C1 according to INTERNATIONAL REGISTER OF CODED CHARACTER SETS TO BE USED WITH ESCAPE SEQUENCES

The register summary is given at: https://www.itscj-ipsj.jp/custom_contents/cms/linkfile/ISO-IR.pdf.

This does not cover all "old" control codes. For instance, "PETSCII" and "ATASCII" are not covered (but see below). But for these much of the special control codes have to do with colour changes and arrow key presses, which is all covered by standard control sequences in ECMA-48.

Here is a walk-through of the registered C0 and C1 variant for ISO/IEC 2022.

- 1 **C0 Set of ISO 646** https://www.itscj-ipsj.jp/ir/001.pdf 4/0 4.2
  A bit cryptic, but this should probably be seen as the "normal" C0, but with one major typo: the first ETX should be STX.
- 7 **NATS, C0 Set** https://www.itscj-ipsj.jp/ir/007.pdf 4/1 4.9
  Intended for "newspaper typography". Seems to be mappable to ECMA-48 styling (plus word delete…).
- 26 **IPTC, C0 Set** https://www.itscj-ipsj.jp/ir/026.pdf 4/3 4.14
  "International press council"; has similarities with entry 7. Seems to be mappable to ECMA-48 styling (plus word delete…), plus maybe some APC…ST (the description in the registration is incomplete), which is private-use in ECMA-48.
- 36 **C0 Set of ISO 646 with SS2 instead of IS4** https://www.itscj-ipsj.jp/ir/036.pdf 4/4 4.18
  The title says it all. Easily mapped to ECMA-48 standard C0/C1. However, that is for SINGLE SHIFT, and thus not relevant for Unicode, only for conversion (maybe to Unicode).
- 48 **INIS, Control Set** https://www.itscj-ipsj.jp/ir/048.pdf 4/2 4.29
  "International Nuclear Information System"/"Bibliographic data interchange"; Just declaring that they (supposedly) only use ESC, GS (IS3) and RS (IS2) out of the "normal" C0 codes.
- 74 **C0 for Japanese standard JIS C 6225-1979** https://www.itscj-ipsj.jp/ir/074.pdf 4/6 4.48
  "Normal" C0 except that IS4 is replaced by CEX (CONTROL EXTENSION). CEX should probably be mapped either to DLE or some ECMA-48 control sequence (depending on what CEX is…).
- 104 **Minimum C0 Set for ISO 4873** https://www.itscj-ipsj.jp/ir/104.pdf 4/7 4.62
  Ordinary C0 limited to ESC…
- 106 **Teletex primary set of Control Functions CCITT Rec. T.61** https://www.itscj-ipsj.jp/ir/106.pdf 4/5 4.68
  *(**Not to be confused with Teletext.**) C0 easily mapped to ECMA-48 standard C0/C1. However, that is for LOCKING SHIFTs, and thus not relevant for Unicode, only for conversion (maybe to Unicode).*
- 130 **C0 Set of ISO 646 without SI and SO ASMO-662 and COMECON ST SEV 358** https://www.itscj-ipsj.jp/ir/130.pdf 4/8 4.80

Normal (ECMA-48) C0 but excluding SI and SO (LS1 and LS0). These two controls are basically always uninterpreted today (and must be uninterpreted for Unicode) anyway.

- 132 **Primary Control Set of Data Syntax I of CCITT Rec. T.101** https://www.itscj-ipsj.jp/ir/132.pdf 4/9 4.85
  Easily mappable to ECMA-48 standard control codes and standard control sequences.
- 134 **Primary Control Set of Data Syntax II of CCITT Rec. T.101** https://www.itscj-ipsj.jp/ir/134.pdf 4/10 4.93
  Mostly easily mappable to standard ECMA-48 control codes/sequences, except for CURSOR ON/OFF and REPEAT, which may need some minor extension to ECMA-48 or the use of private use control codes/sequences.
- 135 **Primary Control Set of Data Syntax III of CCITT Rec. T.101** https://www.itscj-ipsj.jp/ir/135.pdf 4/11 4.97
  Easily mappable to ECMA-48 standard control codes and standard control sequences.
- 140 **C0 Set of ISO 646 with EM replaced by SS2 - Czechoslovak 4/12 4.110 Standard CSN 369102** https://www.itscj-ipsj.jp/ir/140.pdf 4/12 4.110
  The title says it all. Easily mapped to ECMA-48 standard C0/C1.
- 40 **Additional Control Functions for Bibliographic Use according to DIN 31626** https://www.itscj-ipsj.jp/ir/040.pdf 4/5 4.25
  Has a number of control codes for affecting bibliographic sorting of strings. These can be mapped to private-use control sequences as per ECMA-48, or private-use Unicode characters.
- 56 **Attribute Control set for UK Videotex British Telecom** https://www.itscj-ipsj.jp/ir/056.pdf 4/0 4.32
  *(**Videotex was a precursor to Teletext.**) Neither the 7-bit encoding (using escape sequences) nor the 8-bit encoding of that registration can be used in Teletext (and probably nor in Videotex): there are no escape sequences in Teletext, nor are there any 8-bit encodings so there is no C1 area. These "controls" are also inappropriate for anything outside of the Teletext protocol, since these (like the real C0 in Teletext) assume a particular code page setup particular to Teletext. In addition, the descriptions are wrong (or at least incomplete); these are <u>spacing</u> graphic characters (despite being called control characters), usually rendered as SPACE (this likely holds also for Videotex). And… there are no CSI nor ESC (despite the name) controls in Teletext (and likely likewise for Videotex). So this is just…very wrong.*
- 67 **Additional Control Functions for Bibliographic Use according to ISO 6630** https://www.itscj-ipsj.jp/ir/067.pdf 4/2 4.38
  See registration nr 124 below. Has a number of control codes for affecting bibliographic sorting of strings. These can be mapped to private-use control sequences as per ECMA-48, or private-use Unicode characters.
- 73 **Attribute Control Set for Videotex CCITT** https://www.itscj-ipsj.jp/ir/073.pdf 4/1 4.42
  *(**Videotex was a precursor to Teletext.**) In <u>addition</u> to the problems listed with entry 56 (see above), this one is incorrect relative to the (Enhanced) Teletext standard w.r.t. the set of "controls", even when ignoring the issues listed above (for entry 56). So this is just…very, very wrong (and likely wrong also w.r.t. Videotex).*
- 77 **C1 Control Set of ISO 6429-1983** https://www.itscj-ipsj.jp/ir/077.pdf 4/3 4.55
  This seems to the normal C1 controls.
- 105 **Minimum C1 Set for ISO 4873** https://www.itscj-ipsj.jp/ir/105.pdf 4/7 4.65
  "Normal" C1 limited to SS2 and SS3…

- 107 **Teletex Supplementary Set of Control Functions CCITT Rec. T.61** https://www.itscj-ipsj.jp/ir/107.pdf 4/8 4.72
  (Not to be confused with Teletext.) "Normal" C1 limited to PLU, PLD, and CSI.
- 124 **Upward Compatible Version of ISO 6630 (Registration 67)** https://www.itscj-ipsj.jp/ir/124.pdf 4/0 4/2 4.76
  Has a number of control codes for affecting bibliographic sorting of strings. These can be mapped to private-use control sequences as per ECMA-48, or private-use Unicode characters.
- 133 **Supplementary Control Set of Data Syntax I of CCITT Rec. T.101** https://www.itscj-ipsj.jp/ir/133.pdf 4/4 4.89
  (Another registration for Videotex, but different set of controls. See above. This one has likely never been used.)
- 136 **Supplementary Control Set of Data Syntax III of CCITT Rec. T.101** https://www.itscj-ipsj.jp/ir/136.pdf 4/6 4.102
  (Yet another registration for Videotex, but different set of controls. See above. This one has likely never been used.)

Note that there is no registration for Teletext "controls"; and the registration for "normal" C0 has a major typo.

It is high time to scrap all this for the purposes of ISO/IEC 10646 (which mistakenly still references it). The C0/C1 need to be fixed, and fixed to what is in ISO/IEC 6429. Again, this does not mean that implementations suddenly must interpret all or any of C0/C1 characters according to ISO/IEC 6429. All can stay as is in just about all modern implementations w.r.t. conformity to the Unicode standard and the ISO/IEC 10646 standard. Most of this is either completely outdated (since decades), or plain wrong (w.r.t. the relevant standard; Teletext in particular). Data using these registrations may still occur in archival data, but should be converted to follow ISO/IEC 6429 (ECMA-48) with suitable bit padding for UTF-16 and UTF-32, and appropriate bit encoding for C1 in UTF-8. If there is no direct mapping to ECMA-48 control codes or standard control sequences, there are a few control codes that are private use, a large number of control sequences that are private use (and one could use appropriate extensions to the standard ones as well, though that would be a matter of at least proposed standard extensions), and all control strings (APC…ST, OSC…ST, DCS…ST) are all private use. So there is no lack of appropriate ways of representing "odd" (and old) control codes in a standards compliant manner according to ISO/IEC 6429.

ATASCII, PETSCII, …
The character encoding for many old computer systems do **not** follow the ISO/IEC 2022 architecture, and their control codes do not follow ISO/IEC 6429 and sometimes allocate control codes in what would otherwise be "graphic" areas. However, many of the "special" control codes (regardless of allocation) can mostly easily be mapped to standard control sequences in ISO/IEC 6429. A few may need to be mapped to some minor extensions or the use of private use control sequences.

Several of the ATASCII and PETSCII control codes are for arrow keys (there are ISO/IEC 6429 standard control sequences for that), or (foreground?) colour changes (there are ISO/IEC 6429 standard control sequences for that). So much, if not all, of those legacy control codes can be mapped to ISO/IEC 6429 standard control sequences. (And mapped back, if needed.)

No new control codes in Unicode or ISO/IEC 10646
There is no need to encode any new control codes in Unicode. ISO/IEC 6429 provided for plenty of private use possibilities (a few C0/C1 control codes that are private-use, lots of private use control

sequences, and all control strings are private-use), and Unicode has private use areas. So, for control codes from legacy character sets that cannot be mapped closely to ISO/IEC 6429 C0/C1 controls codes or control sequences (many can be so mapped), there are plenty of private-use options. Some (in particular, box start/end in Teletext) may need extension to the SGR functionality of ISO/IEC 6429 which is not formally private-use.

Nit: Teletext has no tabs, and no line feed (lines are handled by having line numbers in the Teletext protocol), and overrides the *entire* C0 (yes, C0, not C1, there is no C1 in Teletext) controls with graphic characters (SPACE or a "mosaic" character) that have control functions (often two each).

C0/C1 stability

C0/C1 needs to be stable (except for the handful of private-use control codes), not vary like a private use area, like some have, quite destructively (from a standards and standards use perspective), proposed. Below are some text changes to the Unicode standard text relevant to this. There may be more changes needed.

Conclusion and recommendations

The goal of this paper is not to preserve Teletext, even though quite a lot of is has been devoted to certain aspects of Teletext. But that has been in order to give a background for why it is a bad idea to regard C0/C1 areas as some kind of private use area(s), like suggested in L2/21-235 'Proposal to add further characters from legacy computers and teletext' which suggests that:

"Control characters from microcomputer platforms and teletext were also determined to be out of scope for the UCS. These characters were located in what would today be considered the C0 control range (0x00–0x1F) or the C1 control range (0x7F–0x9F). Processes that need to interchange these codes should simply interchange the binary C0 or C1 value, extended to the UCS code space but without further mapping. Emulators should treat these control codes as appropriate for the targeted environment."

This must not happen. It is a senseless idea for Teletext (as explained at length above), and is a very very bad idea also for other, less extreme, settings. The C0/C1 needs to be fixed in semantics. That does not mean that Teletext and other "old" systems are left stranded. For Teletext it is perfectly possible to convert to another styling system that is valid outside of the Teletext protocol. For other odd-ball old control codes one should map each to the closest corresponding ISO/IEC 6429 control code (there are even 4 private-use ones), escape sequence, control sequence (CSI.....; there are plenty of private use ones), or control string (often application program control string; APC....ST; all of which is private-use).

C0/C1 should be protected by conformity clauses, stability clauses, and a total rewrite of section 23.1 (Control Codes; see below for a suggested rewrite) of the Unicode standard, and a rewrite of at least section 13.4 (Identification of control function set) of ISO/IEC 10646 (nobody, absolutely nobody, implements *those* escape sequences (for C0/C1) anyway).

C0/C1 needs to be as stable as the ASCII part of the Unicode and ISO/IEC 10646 standards. This does not mean that implementations must start supporting lots of control codes not supported today. Many of them will simply be uninterpreted, as they are today. Note that not even A-Z, a-z, 0-9 are required to be supported by conforming implementations. (All do, but that is for practical reasons, not conformity reasons.)

Requesters that want "new" control codes for handling some (presumably old) character sets when mapped to Unicode, should be directed to ISO/IEC 6429, which in addition to many standard control sequences, also has room for private-use control sequences and control strings. There is no need to add any "new" control code to Unicode for any "old" character set, while still having C0/C1 areas stable.

Proposed text changes, and property changes for the Unicode standard

Old (https://www.unicode.org/versions/Unicode14.0.0/UnicodeStandard-14.0.pdf):
Control Codes. Sixty-five code points (U+0000..U+001F and U+007F..U+009F) are defined specifically as control codes, for compatibility with the C0 and C1 control codes of the ISO/IEC 2022 framework. A few of these control codes are given specific interpretations by the Unicode Standard. (See Section 23.1, Control Codes.)

Proposed (6429 instead of 2022):
Control Codes. Sixty-five code points (U+0000..U+001F and U+007F..U+009F) are defined specifically as control codes, for compatibility with the C0 and C1 control codes of ISO/IEC 6429:1992 *Information technology — Control functions for coded character sets*. A few of these control codes are given specific interpretations by the Unicode Standard. (See Section 23.1, Control Codes.)

Old (https://www.unicode.org/versions/Unicode14.0.0/UnicodeStandard-14.0.pdf):
Control Codes

In addition to the special characters defined in the Unicode Standard for a number of purposes, the standard incorporates the legacy control codes for compatibility with the ISO/IEC 2022 framework, ASCII, and the various protocols that make use of control codes. Rather than simply being defined as byte values, however, the legacy control codes are assigned to Unicode code points: U+0000..U+001F, U+007F..U+009F. Those code points for control codes must be represented consistently with the various Unicode encoding forms when they are used with other Unicode characters. For more information on control codes, see *Section 23.1, Control Codes*.

Proposed (6429 instead of 2022):
Control Codes

In addition to the special characters defined in the Unicode Standard for a number of purposes, the standard incorporates the legacy control codes for compatibility with ISO/IEC 6429:1992 *Information technology — Control functions for coded character sets*, ASCII, and the various protocols that make use of control codes. Rather than simply being defined as byte values, however, the legacy control codes are assigned to Unicode code points: U+0000..U+001F, U+007F..U+009F. Those code points for control codes must be represented consistently with the various Unicode encoding forms when they are used with other Unicode characters. For more information on control codes, see *Section 23.1, Control Codes*.

Old (https://www.unicode.org/versions/Unicode14.0.0/UnicodeStandard-14.0.pdf):

23.1 Control Codes

There are 65 code points set aside in the Unicode Standard for compatibility with the C0 and C1 control codes defined in the ISO/IEC 2022 framework. The ranges of these code points are U+0000..U+001F, U+007F, and U+0080..U+009F, which correspond to the 8- bit controls 00_{16} to $1F_{16}$ (C0 controls), $7F_{16}$ (delete), and 80_{16} to $9F_{16}$ (C1 controls), respectively. For example, the 8-bit legacy control code character tabulation (or tab) is the byte value 09_{16}; the Unicode Standard encodes the corresponding control code at U+0009.

The Unicode Standard provides for the intact interchange of these code points, neither adding to nor subtracting from their semantics. The semantics of the control codes are generally determined by the application with which they are used. However, in the absence of specific application uses, they may be interpreted according to the control function semantics specified in ISO/IEC 6429:1992.

In general, the use of control codes constitutes a higher-level protocol and is beyond the scope of the Unicode Standard. For example, the use of ISO/IEC 6429 control sequences for controlling bidirectional formatting would be a legitimate higher-level protocol layered on top of the plain text of the Unicode Standard. Higher-level protocols are not specified by the Unicode Standard; their existence cannot be assumed without a separate agreement between the parties interchanging such data.

Representing Control Sequences

There is a simple, one-to-one mapping between 7-bit (and 8-bit) control codes and the Unicode control codes: every 7-bit (or 8-bit) control code is numerically equal to its corresponding Unicode code point. For example, if the ASCII line *feed control* code ($0A_{16}$) is to be used for line break control, then the text "WX<LF>YZ" would be transmitted in Unicode plain text as the following coded character sequence: <0057, 0058, 000A, 0059, 005A>.

Control sequences that are part of Unicode text must be represented in terms of the Unicode encoding forms. For example, suppose that an application allows embedded font information to be transmitted by means of markup using plain text and control codes. A font tag specified as "^ATimes^B", where ^A refers to the C0 control code 01_{16} and ^B refers to the C0 control code 02_{16}, would then be expressed by the following coded character sequence: <0001, 0054, 0069, 006D, 0065, 0073, 0002>. The representation of the control codes in the three Unicode encoding forms simply follows the rules for any other code points in the standard:

    UTF-8: <01 54 69 6D 65 73 02>
    UTF-16: <0001 0054 0069 006D 0065 0073 0002>
    UTF-32: <00000001 00000054 00000069 0000006D 00000065 00000073 00000002>

Escape Sequences. Escape sequences are a particular type of protocol that consists of the use

13

of some set of ASCII characters introduced by the *escape* control code, 1B_16, to convey extra-textual information. When converting escape sequences into and out of Unicode text, they should be converted on a character-by-character basis. For instance, "ESC-A" <1B 41> would be converted into the Unicode coded character sequence <001B, 0041>. Interpretation of U+0041 as part of the escape sequence, rather than as latin capital letter a, is the responsibility of the higher-level protocol that makes use of such escape sequences. This approach allows for low-level conversion processes to conformantly convert escape sequences into and out of the Unicode Standard without needing to actually recognize the escape sequences as such.

If a process uses escape sequences or other configurations of control code sequences to embed additional information about text (such as formatting attributes or structure), then such sequences constitute a higher-level protocol that is outside the scope of the Unicode Standard.

Specification of Control Code Semantics

Several control codes are commonly used in plain text, particularly those involved in line and paragraph formatting. The use of these control codes is widespread and important to interoperability. Therefore, the Unicode Standard specifies semantics for their use with the rest of the encoded characters in the standard. Table 23-1 lists those control codes.

Table 23-1. Control Codes Specified in the Unicode Standard

Code Point	Abbreviation	ISO/IEC 6429 Name
U+0009	HT	character tabulation (tab)
U+000A	LF	line feed
U+000B	VT	line tabulation (vertical tab)
U+000C	FF	form feed
U+000D	CR	carriage return
U+001C	FS	information separator four
U+001D	GS	information separator three
U+001E	RS	information separator two
U+001F	US	information separator one
U+0085	NEL	next line

The control codes in Table 23-1 have the Bidi_Class property values of S, B, or WS, rather than the default of BN used for other control codes. (See Unicode Standard Annex #9, "Unicode Bidirectional Algorithm.") In particular, U+001C..U+001E and U+001F have the Bidi_Class property values B and S, respectively, so that the Bidirectional Algorithm recognizes their separator semantics.

The control codes U+0009..U+000D and U+0085 have the White_Space property. They also have line breaking property values that differ from the default CM value for other control codes. (See Unicode Standard Annex #14, "Unicode Line Breaking Algorithm.")

U+0000 null may be used as a Unicode string terminator, as in the C language. Such usage is outside the scope of the Unicode Standard, which does not require any particular formal language representation of a string or any particular usage of null.

Newline Function. In particular, one or more of the control codes U+000A line feed, U+000D carriage return, and the Unicode equivalent of the EBCDIC next line can encode a

newline function. A newline function can act like a line separator or a paragraph separator, depending on the application. See Section 23.2, Layout Controls, for information on how to interpret a line or paragraph separator. The exact encoding of a newline function depends on the application domain. For information on how to identify a newline function, see Section 5.8, Newline Guidelines.

Proposed (incl.: HTJ should be treated the same as HT; bidi, lb, and some other prop. fixes):

23.1 Control Codes

There are 65 code points set aside in the Unicode Standard for compatibility with the C0 and C1 control codes defined in ISO/IEC 6429:1992 *Information technology — Control functions for coded character sets*. The ranges of these code points are U+0000..U+001F, U+007F, and U+0080..U+009F, which correspond to the 8- bit controls 00_16 to 1F_16 (C0 controls), 7F_16 (delete), and 80_16 to 9F_16 (C1 controls), respectively. For example, the 8-bit legacy control code *character tabulation* (or tab) is the byte value 09_16; the Unicode Standard encodes the corresponding control code at U+0009.

The Unicode Standard provides for the intact interchange of these code points, neither adding to nor subtracting from their semantics, except as noted below. The semantics of the control codes are generally determined by the application with which they are used. The control function semantics is specified in ISO/IEC 6429:1992. With some exceptions, especially the ones for NLF and tab, many applications do not interpret most of the control codes, and they are then usually rendered as zero-width and glyph-less (something that may cause security issues or strange errors).

In general, the use of control codes constitutes a higher-level protocol and is beyond the scope of the Unicode Standard, except for the control codes mentioned in section *Specification of Control Code Semantics* below.

For example, the use of ISO/IEC 6429 control sequences for controlling bidirectional formatting would be a legitimate higher-level protocol layered on top of the plain text of the Unicode Standard. Note though, that the use of ISO/IEC 6429 control sequences for controlling bidirectional formatting is incompatible with the use of the Unicode bidi algorithm, a different higher-level protocol for the same purpose using control codes that are not in C0/C1. A more common use of ISO/IEC 6429 control sequences is to specify bold, italic, or underline style, as well as background and foreground colours for the text.

Even if a process does not interpret any of the ISO/IEC 6429 escape sequences, control sequences, or control strings, it may render them (in their entirety) as zero-width and glyph-less (if rendering is at all part of the process), or (in their entirety) as default-ignorable. However, that is not required by the Unicode standard.

Higher-level protocols, except for the Unicode bidi algorithm and certain cursive shaping, are not specified by the Unicode Standard; their existence cannot be assumed without a separate agreement between the parties interchanging such data (and likewise for the Unicode bidi algorithm).

Representing Escape Sequences, Control Sequences and Control Strings

15

There is a simple, one-to-one mapping between commonly used 7-bit (and 8-bit) control codes (those conforming to ISO/IEC 6429) and the Unicode control codes: every 7-bit (or 8-bit) control code is numerically equal to its corresponding Unicode code point. For example, if the ASCII *line feed* control code (0A_16) is to be used for line break control, then the text "WX<LF>YZ" would be transmitted in Unicode plain text as the following coded character sequence: <0057, 0058, 000A, 0059, 005A>.

Escape sequences and control sequences, as well as control strings, that are part of Unicode text must be represented in terms of the Unicode encoding forms. For example, suppose that an application allows embedded font information to be transmitted by means of markup using plain text and control codes. A font tag specified as "^ATimes^B", where ^A refers to the C0 control code 01_16 and ^B refers to the C0 control code 02_16, would then be expressed by the following coded character sequence: <0001, 0054, 0069, 006D, 0065, 0073, 0002>. The representation of the control codes in the three Unicode encoding forms simply follows the rules for any other code points in the standard:

    UTF-8: <01 54 69 6D 65 73 02>
    UTF-16: <0001 0054 0069 006D 0065 0073 0002>
    UTF-32: <00000001 00000054 00000069 0000006D 00000065 00000073 00000002>

(Note that this example is an example only; it has no known implementations, nor does it follow ISO/IEC 6429 syntax for control strings (something that does have some implementations). The appropriate control string syntax is **APC…ST**, where the … is private use. It could be "**fnt=Times**".) [It would be preferable to use the latter, which is standards based and realistic, rather than something purely invented for this paragraph…i.e. **APC fnt=Times ST**.]

Escape sequences and control sequences. [That subtitle should probably be moved up, or deleted; there is already a title good enough above.] Escape sequences and control sequences are a particular type of protocol that consists of the use of some set of ASCII characters introduced by the *escape* control code, 1B_16, or the *control sequence introducer* control code, to convey extra-textual information. When converting escape sequences into and out of Unicode text, they should be converted on a character-by-character basis. For instance, "ESC-A" <1B 41> would be converted into the Unicode coded character sequence <001B, 0041>. Interpretation of U+0041 as part of the escape sequence, rather than as latin capital letter a, is the responsibility of the higher-level protocol that makes use of such escape sequences. This approach allows for low-level conversion processes to conformantly convert escape sequences into and out of the Unicode encoding forms without needing to actually recognize the escape sequences as such, provided that the other encoding conforms to ISO/IEC 6429. ISO/IEC 6429 also specifies the general syntax for control strings but does not specify the content of such strings; they are entirely private-use. See example above.

If a process uses escape sequences or other configurations of control code sequences to embed additional information about text (such as formatting attributes or structure), then such sequences constitute a higher-level protocol that is outside the scope of the Unicode Standard.

Legacy charsets often contain control codes that do not directly match any of the control codes in C0/C1 of ISO/IEC 6429. However, often one can find matching standard control sequences in ISO/IEC 6429. For example, ATASCII's 1C_16 (cursor up) corresponds to ISO/IEC 6429's *CSI 1A*, (cursor up 1 step) and PETSCIIs 1F_16 (blue) corresponds to

ISO/IEC 6429's **CSI 34m** (blue-ish foreground) or maybe better **CSI 94m** (clear blue foreground, a popular extension to ISO/IEC 6429). For Teletext "control" codes, specified in the "Enhanced Teletext" standard, many of them serve three purposes in one, a graphic character, code page switching, and colour change. E.g. Teletext's 00_16 (alpha black), (usually) displays as a SPACE, switches to the current default "alphanumeric" 7-bit codepage, and changes to black foreground colour; despite not being true control characters, they are graphic characters with variable display, they reside in the C0 area of all of the 7-bit codepages of Teletext (aside: Teletext itself has no escape sequences, nor any control sequences, nor any C1 area). Generally, if there is no close match, one may consider using the private-use control codes, the private-use control sequences of ISO/IEC 6429 or even private-use Unicode code points. Unicode does not specify these mappings.

Specification of Control Code Semantics

Several control codes are commonly used in plain text, particularly those involved in line and paragraph formatting. The use of these control codes is widespread and important to interoperability. Therefore, the Unicode Standard specifies semantics for their use with the rest of the encoded characters in the standard. Table 23-1 lists those control codes.

Table 23-1. Control Codes Specified in the Unicode Standard

Code Point	Abbreviation	ISO/IEC 6429 Name
U+0009	HT	character tabulation (tab)
U+000A	LF	line feed
U+000B	VT	line tabulation (vertical tab)
U+000C	FF	form feed
U+000D	CR	carriage return
U+001C	FS	information separator four
U+001D	GS	information separator three
U+001E	RS	information separator two
U+001F	US	information separator one
U+0085	NEL	next line
U+0089	HTJ	character tabulation with justification

The control codes in Table 23-1 have the Bidi_Class property values of S or B, rather than the default of BN used for other control codes. (See Unicode Standard Annex #9, "Unicode Bidirectional Algorithm.") In particular, U+001C..U+001E and U+001F have the Bidi_Class property values B and S, respectively, so that the Bidirectional Algorithm recognizes their separator semantics.

The control codes U+0009..U+000D and U+0085 have the White_Space property. They also have line breaking property values that differ from the default CM value for other control codes. (See Unicode Standard Annex #14, "Unicode Line Breaking Algorithm.")
[Line break properties for U+001C..U+001F…; though these are basically obsolete, they have "special" bidi properties; maybe they should have line break properties consistent with the bidi properties…; though I don't want to encourage "reviving" these (in contrast to HTJ…).]

U+0000 *null* may be used as a Unicode string terminator, as in C, C++ and several other programming languages. Such usage is outside the scope of the Unicode Standard, which does not require any particular formal programming language representation of a string or any particular usage of *null*.

Newline Function. In particular, one or more of the control codes U+000A *line feed*, U+000D *carriage return*, and the Unicode equivalent of the ISO/IEC 6429 *next line* (often used in EBCDIC based encodings) can encode a newline function. A newline function can act like a *line separator* or a *paragraph separator*, depending on the application. See Section 23.2, Layout Controls, for information on how to interpret a line or paragraph separator. The exact encoding of a newline function depends on the application domain. For information on how to identify a newline function, see Section 5.8, Newline Guidelines. Also U+000B (*line tabulation*) and U+000C (*form feed*) can encode new line functions, similar to that of *line separator*.

Proposed, LineBreak.txt, change to have:

```
001C..001E;BK       # Cc      [3] <control-001E>..<control-001F>  [cmp. bidi B]
001F;BA             # Cc          <control-001F>  [cmp. bidi S]
0084;LF             # Cc          <control-0084>  [xterm still interprets this character]
0089;BA             # Cc          <control-0089>  [this is a character tabulation variant]
```

Proposed, UnicodeData.txt, change to have:

```
000C;<control>;Cc;0;S;;;;;N;FORM FEED (FF);;;;
0084;<control>;Cc;0;S;;;;;N;;;;;  [xterm still interprets this character]
0089;<control>;Cc;0;S;;;;;N;CHARACTER TABULATION WITH JUSTIFICATION;;;;
```

Proposed text changes for ISO/IEC 10646

Section 12 correctly states:

"Code extension control functions for the ISO/IEC 2022 code extension techniques (such as designation escape sequences, single shift, and locking shift) shall not be used with this coded character set."

However, section 13.4 incorrectly contradicts that for control characters, harking back to ISO 2022 codepage switching which must not be used with the ISO/IEC 10646 encodings:

"**13.4 Identification of control function set**
When the escape sequences from ISO/IEC 2022 are used, the identification of each set of control functions (see Clause 12) of ISO/IEC 6429 to be used in conjunction with ISO/IEC 10646 shall be an identifier sequence of the type shown below.
          ESC 02/01 04/00 identifies the full C0 set of ISO/IEC 6429
          ESC 02/02 04/03 identifies the full C1 set of ISO/IEC 6429
For other C0 or C1 sets, the final octet F shall be obtained from the International Register of Coded Character Sets. The identifier sequences for these sets shall be
          ESC 02/01 F identifies a C0 set
          ESC 02/02 F identifies a C1 set
If such an escape sequence appears within a code unit sequence conforming to ISO/IEC 2022, it shall consist only of the sequences of bit combinations as shown above.
If such an escape sequence appears within a code unit sequence conforming to this document, it shall be padded in accordance with Clause 12."

That text needs to be deleted. This leftover from before the synchronization with Unicode has made some think that the "shall not be used with this coded character set" from section 12 does not apply to the C0/C1 areas, which of course it does. Indeed, C0/C1 need be fixed to that of ISO/IEC 6429, by reference to that standard.

There may be some other changes needed as well.