quired that (1) each I_j be a proper subset of L, and (2) N be a proper subset of each I_j.

In set-theoretic notation,

$$L \supsetneq I_1 \cup I_2 \cup \ldots \cup I_m$$

$$N \subsetneq I_1 \cap I_2 \cap \ldots \cap I_m \cap L$$

where "$\cup$" denotes the union of features and "$\cap$" denotes the intersection of features (selection of only those common to both the two operands).

11.2 Modular Specification of Subsets

Modular specification of subsets is an equivalent method for determining the intermediate subsets of a language by other than explicit enumeration.

In the modular approach, there is a nucleus, N, and a set of modules, M_i, where each module is a collection of language additions to N. The nucleus has no feature in common with any module. (That is, $N \cap M_i = \varphi$ (empty).)

The standard subsets are the nucleus in combination with any one or more of the modules:

$$I_i = N \cup M_{i_1} \cup M_{i_2} \cup \ldots \cup M_{i_p}$$

with

$$L \supseteq N \cup M_1 \cup M_2 \cup \ldots \cup M_n$$

for a language having n modules.

With the module approach, it might be possible to achieve L only by inclusion of more than one module. Secondly, there may be nested, overlapping, or mutually exclusive modules. That is, it is not necessary that either

$$L = N \cup M_i \quad \text{for some i,}$$

or that

$$\varnothing = M_i \cap M_j \quad \text{for all i, j} \quad \text{where} \quad i \neq j.$$

Note that given N, L, and some intermediate subsets, then a set of modules capable of generating precisely those subsets (and L) can be determined.

When there are to be multiple subsets, specification in terms of modularity is preferable.

11.3 Methods of Specification

(To be submitted for approval at a later date.)

11.4 Some Restrictions

As with standards in general, a standard programming language represents a floor, not a ceiling. Thus L or a subset may be the basis of a more extensive programming language.

Suppose some programming language X purports to include I (a subset of L), but not all of L. In order for X to be an acceptable extension of I, X must not contradict L. That is, X must not contain a syntactic form or syntactic feature of L to which X ascribes a different meaning from the one ascribed by L. It is permissible, however, for X to provide a new syntactic form to achieve an effect already obtainable in L or even in I. In other words, any extensions made to a subset of L must be permissible extensions to L itself.

If the definition of L involves modules which are defined to be mutually exclusive, L as such is excluded from the standard. However, it must be possible to form L as an extension to the standard. In other words, modules must not be mutually exclusive on the basis of contradiction but only on the basis of appropriateness.

PROPOSED USA STANDARD

Code Extension Procedures for Information Interchange*

Editor's Note

This proposed American Standard has been accepted for publication for a four-month period followed by a six-week letter ballot by USA Standards Committee X3 Computers and Information Processing. In order that the final version of the proposed standard reflect the largest public consensus, X3 authorized publication of this document to elicit comment, and general public reaction with the understanding that such a working document is an intermediate result in the standardization process and is subject to change, modification, or withdrawal in part or in whole. Comments should be addressed to the X3 Secretary, Business Equipment Manufacturers Association, 235 East 42 Street, New York, NY 10017.—E. L.

Key Words and Phrases: standard code, code, information interchange, characters, shift out, shift in, escape, data link escape, control functions, standard procedures, code extension, code table, bit pattern

CR Categories: 1.0, 2.0, 2.43, 3.20, 3.24, 3.50, 3.51, 3.52, 3.53, 3.54, 3.55, 3.56, 3.57 3.70, 3.71, 3.72, 3.73, 3.74, 3.75, 3.80, 3.81, 3.82, 3.83. 5.0, 5.1, 6.2, 6.20, 6.21, 6.22

* USASI Document X3.4/248, June 14, 1968

Foreword

(This foreword is not a part of the USA Standard Code Extension Procedures for Information Interchange, X3. . . .)

The proposed USA Standard Code for Information Interchange (ASCII-USASI X3.4-19) provides coded representations for a set of graphic and control characters having general utility in information interchange. In some applications it may be desirable to augment the standard repertoire of characters with additional graphic symbols or control functions.

The Code includes several special characters intended to facilitate the representation of such additional symbols or functions, a process known as *code extension*. Although the basic nature of code extension—providing for encoding of information beyond the standard—limits the degree to which it may be standardized, there are advantages to adherence to certain standard rules of procedure. These advantages include minimized risk of conflict between systems required to interoperate, and the possibility of including advance provision for code extension in the design of general purpose data handling systems.

These standard procedures were developed after extensive study of various potential applications and of trends expected in system design.

1. Scope

This standard specifies a set of procedures for the representation, by characters of ASCII[1], of graphic symbols or control functions, not directly represented in ASCII, which may be required for a specific application or system. This standard does not make specific assignment of such characters or functions.

[1] USA Standard Code for Information Interchange

2. General

2.1 The characters provided in ASCII for code extension purposes are:

SO	Shift Out
SI	Shift In
ESC	Escape
DLE	Data Link Escape

2.2 SO and SI are intended for use in extension of the graphic repertoire to symbols not assigned in the code proper. The standard procedures for their use are described in Section 3.
2.3 ESC is principally intended for extension of the control repertoire of the code to control functions not assigned in the code proper, other than communication control functions. Standard procedures for its use are described in Section 4.
2.4 DLE is intended for extension of the control repertoire of the code to communication control functions not assigned in the code proper. Standard procedures for its use are to be covered in standards for data communication control procedures. The basic principles of these procedures are described, for reference only, in the Appendix, Section A5.
2.5 The promulgation of these standard procedures is in no way meant to deprecate the use of other code extension procedures, so long as the implications of such usage upon system compatibility are recognized (see also Appendix, Section A2.4).
2.6 This standard does not make specific assignment of additional graphic symbols or control functions to be represented through code extension. The possibility of standardizing some such assignments in the future is still under study. Users of these procedures are advised to seek the latest information in this regard should their use be one potentially impacted by such standardization.
2.7 The ASCII code table is shown for reference in the Appendix as Figure A1.

3. Graphic Set Extension: Use of SO (Shift Out) and SI (Shift In)

3.1 The characters SO and SI are used to select which of two sets of graphic symbols is to be associated with the 95 "graphic" bit patterns of the standard code.
3.2 SO indicates that the standard set of graphics is to be replaced with an "alternate" set.
3.3 SI indicates that the graphic bit patterns of the code are to again be associated with the standard set of graphic characters.
3.4 The use of this procedure requires agreement between the parties to the interchange as to the assignment of characters to the alternate set (see 2.6).
3.5 There is no implication that all 95 positions of the alternate graphic set be different from the corresponding members of the standard set, nor even that all 95 are assigned.
3.6 The new characters of the alternate set may be entirely different symbols, or they may differ from those of the standard set only in size, style, or other "typographical" attributes.
3.7 The set of 32 control characters in the code and the character DEL (Delete) should not be affected by the "shift out" operation.
3.8 If more than one "alternate" set is required, SO should be utilized to select the principal such set, and *escape sequences* (see Section 4) used to select each of the other such sets.
3.9 Alternatively, in such a situation *escape sequences* may be assigned to select which of the available alternate sets is to be subsequently put into force by the use of SO.
3.10 In either case, SI restores the use of the standard graphic set.
3.11 It is recommended that terminal devices and other such equipment be arranged to automatically revert to the use of the standard graphic set whenever the association of the terminal with another terminal or system has been discontinued or suspended: that is, at the end of a *call*, *transmission*, or whatever is appropriate.

4. Control Set Extension: Use of ESC (Escape)

4.1 The character ESC is used as a prefix to a sequence of one or more additional ASCII characters used to represent a control function not directly represented within the code.
4.2 An escape sequence is considered to include its associated ESC character, and its length is defined accordingly.
4.3 Such sequences—known as "escape sequences"—should not be used to represent additional communication control functions (see Appendix, Section A5).
4.4 This standard provides a uniform method for the definition of code extension sequences of any length (two characters or greater).
4.5 The use of this procedure requires agreement between the parties to the interchange as to the assignment of functions to specific escape sequences (see 2.6).
4.6 The means of marking the end of a sequence depends upon division of the characters of the code into two classes, known as "intermediate" and "final" characters, respectively.
4.7 A standard variable length sequence begins with ESC, continues, if necessary, with any number of "intermediate" characters, and invariably ends with one "final" character. Two-character sequences, therefore, contain no "intermediate" characters but consist of ESC followed by one "final" character.
4.8 The *final* characters are those in columns 0, 1, and 3–7 of the ASCII code table, that is, the control characters, the numerics, both the uppercase and lowercase letters of the alphabet, and certain special graphics, *except* as noted in Section 4.9.
4.9 The *intermediate* characters are those in column 2 of the code table, that is, *space* and the bulk of the special graphics.
4.10 The following characters should be excluded from assignment in escape sequences (see Appendix, Section A4.4.):

Designation	*Name*	*Code table Column/Row*
NUL	Null	0/0
SOH	Start of Heading	0/1
STX	Start of Text	0/2
ETX	End of Text	0/3
EOT	End of Transmission	0/4
ENQ	Enquiry	0/5
ACK	Acknowledge	0/6
DLE	Data Link Escape	1/0
NAK	Negative Acknowledge	1/5
SYN	Synchronous Idle	1/6
ETB	End of Transmission Block	1/7
CAN	Cancel	1/8
SUB	Substitute	1/10
ESC	Escape (except as first character)	1/11
DEL	Delete	7/15

APPENDIX

[*This appendix is not a part of the USA Standard Code Extension Procedures for Information Interchange but is included to facilitate its use.*]

A1. Introduction

This appendix to the USA Standard Code Extension Procedures for Information Interchange. Figure A.1, contains a discussion of the objectives, criteria, and other considerations that were used in the development of the standard, as well as supplementary information to facilitate the effective application of these procedures.

b7 b6 b5 / Bits b4 b3 b2 b1 / COLUMN / ROW	0 (0 0 0)	1 (0 0 1)	2 (0 1 0)	3 (0 1 1)	4 (1 0 0)	5 (1 0 1)	6 (1 1 0)	7 (1 1 1)
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

FIG. A.1. USA Standard Code for Information Interchanges (USASCII) per X3.4-1967

A2. General

A2.1 Background. In the establishment of a general purpose code such as the USA Standard Code for Information Interchange (ASCII), or its international counterpart, the ISO 7-bit code, a fundamental decision must be made as to the size of the code. In making such a decision there is usually a conscious effort to avoid the most obvious problems with a code which is either too large or too small. Should the number of characters included be too small, many individual users will find their needs not accommodated and will be forced to adopt "parochial" codes for their applications. Should the number of characters be too large, many potential users will find the standard code disproportionately costly to implement, or untenably inefficient in transmission or storage, and will

again be driven to the use of some other code. Thus, either extreme in code sizing will reduce the generality of application of the code, defeating the very purpose of standardization in this field.

The 7-bit size (128 characters) adopted for ASCII is thought to be near optimum at present with respect to the above considerations. Nevertheless, there will doubtlessly be numerous applications with requirements that are not accommodated by a code of this size, or at least not by the specific characters assigned within it. Still it is hoped that many of these applications can be served by the use of the standard code augmented in some appropriate manner. Through such an approach the user may be able to implement much of his system with standard hardware or software. More significantly, perhaps, he will thereby be able to retain compatibility with other systems for the interchange of that information which can adequately be directly represented by the standard code.

The concept of augmenting the standard code for such purposes may be spoken of in a generic way as "code extension".

A2.2 Standardization of Procedures. The codes with which we are concerned contain four characters whose definitions indicate their relationship to code extension. They are:

SO	(Shift Out)
SI	(Shift In)
DLE	(Data Link Escape)
ESC	(Escape)

The use of these characters is not treated in detail in the code standards. Actually the very nature of code extension inherently limits the degree to which standards for it may be constructed: it is a means of operating "beyond the standard." Nevertheless, there are several advantages to establishing a standard general procedure.

First, such standardization can prevent undesirable conflict between independently contrived applications of code extension. For example, a code extension procedure used by a data communication terminal device should be inherently free from any hazard of conflict with a code extension procedure used in a communications system which may be called upon to serve the terminal.

Second, the availability of such standards can provide guidance to system designers to facilitate the advance inclusion of general provisions for code extension operations in information handling equipment.

A2.3 Application of Standard Procedures. The standard procedures are directed at the application of code extension to those portions of a system where the use of the standard code itself would ordinarily be appropriate, that is, in what is spoken of as "information interchange."

Naturally there are other functions within many information interchange systems for which an extremely unusual usage of the standard code, or some entirely different representation of information (e.g. the points of a character matrix), may be entirely appropriate. Such functions are often thought of as being internal to some autonomous system component. Just as the code standard is not presumed to be appropriate for such functions, it is not presumed that the procedures of this standard are appropriate for them.

A2.4 Related Approaches. The suggested procedures presented here for code extension should in no way be considered to deprecate the practice of using sequences of graphic characters to represent machine instructions, graphic characters not otherwise available, and so forth. Programming languages used in data processing, for example, are based upon such an approach.

A2.5 ASCII. Figure 1 shows the USA Standard Code for Information Interchange and is provided for reference. The code consists of two general categories of characters, *graphics* and *controls*. There are 32 controls, 95 graphics, and the character DEL (Delete) which in reality is neither. The 95 graphics include both uppercase and lowercase letters of the Latin (often called roman) alphabet, the Arabic numerals 0 to 9, a number of punctuation marks and special symbols, and SP (space), the "nonprinting graphic."

A3. Graphic Set Extension: Use of SO and SI

A3.1 Basic Concepts

A3.1.1 There are a number of applications which are not adequately accommodated by the graphic set of the standard code. The most prominent examples are those of systems in which special symbols are required by some scientific discipline or commercial usage (for example, meteorology), and those requiring the use of languages which cannot be directly represented by the Latin alphabet, such as Russian. Of course, these needs could often be met through graphic substitution: that is, by adopting for the system a code which differs from the standard code in that certain standard characters are replaced by the special ones which are required. The displaced standard characters are, however, naturally lost to use by such a system. However, it will often be desirable for the system to have the capability of printing (or otherwise handling) such special graphics, while retaining the ability to communicate with other systems using the standard set of graphics. The procedures of Section 2 provide for this.

A3.1.2 Although the control characters are not to be affected by the "shift out" operation, it is of course possible that the use of a new set of graphics may require a corollary change in the execution of a control within its standard definition. For example, if the alternate graphic set contains characters which are given a larger typographical size than those of the standard set, the character Line Feed may have to produce a larger motion when the alternate set is in use. This is of course not construed as a change in the control character set.

A3.1.3 As pointed out in Section 3.5, there is no implication that the alternate set should be entirely different from the standard set nor that all 95 positions are even assigned. It may contain whatever repertoire of characters are needed for operation in a particular environment. For example, the alternate set might retain the standard letters and numerals but replace certain punctuation marks with weather symbols. In another application, the lowercase alphabet might be replaced in the alternate set by special mathematical symbols, while the uppercase alphabet, the numerals, and the punctuation marks are retained. It is recommended that any symbols common to both the standard and alternate sets be assigned to the same code table position in both. It is also advisable to leave SP (*space*) in the alternate set whether required or not, as many printing mechanisms treat it separately, not actually behaving as if it were a graphic.

A3.2 Application to Devices of Modest Repertoire

A3.2.1 It should be noted that useful application of these principles may in some cases be made in devices having a relatively modest capacity for different graphics. Consider, as an example, the problem of making a terminal device to render messages in both Latin (standard) and Greek (alternate) alphabets and requiring the conventional numerals and punctuation in connection with either. In many situations it would be satisfactory to render all letters in uppercase; that is, the receipt of the coded representation for either "A" or "a" would cause "A" to be printed.[2] Extending this principle to the special symbols coded in the same area of the code with the letters, it is seen that 32 printing characters can suffice for 64 characters of the code. (Actually 63, since DEL, though coded in the graphic region, is not a graphic.) Adding provision for the 10 numerals and the 22 remaining symbols, the machine need have but a 64-character graphic capacity for its work in the Latin alphabet.

A3.2.2 An additional 31 printing characters can serve, in the same manner, for both the uppercase and lowercase Greek alphabets and some associated special symbols when in the alternate set. The 10 standard numerals and the 22 standard punctuation marks are used in the alternate set operation. This postulated application can therefore be implemented in this manner with a terminal device having only the 95-character graphic capacity which would ordinarily be required for full rendition of the standard set.

A3.2.3 Such a device when in its standard mode may receive, without hazard, information containing any of the 95 ASCII graphics. If a graphic set shift were not used in this application, the bilingual capability could only be served with a 95-graphic printer by making the Greek alphabet a *graphic substitution* for the lowercase Latin letters in the code table.

The device could not then be safely used for interchange of information with systems which might use the lowercase Latin letters, since the receipt of these would of course cause the printing of Greek letters.

A3.3 Multiple Graphic Sets. In many applications there will be a need for many alternative graphic sets. Applications in the graphic arts industry will often be of this class. It has been frequently suggested that, to cater to such needs, provisions should be made for the use of a suffix after the character SO to indicate which alternate set is desired. Actually, however, such a procedure appears to be neither necessary nor desirable. Therefore, SO is reserved for use, by itself, to select the single alternate set in systems having but two sets[3] and for selecting the principal alternate set in systems having several sets. The additional alternate sets, if provided, should be invoked according to the procedures of Section 3.7 or 3.8.

This approach avoids any possible need for one device to be capable of handling two types of control-representing sequences, one type prefixed with Escape and the other with SO.

A4. Control Set Extension: Use of ESC (Escape)

A4.1 General

A4.1.1 The expected requirements for additional controls beyond those assigned in the code are somewhat different from those for additional graphics. It is typical of systems requiring additional graphics that the graphics may often be used in groups and for an extended period, such as when printing text in a foreign language. On the other hand, it is more typical of controls that they appear sparsely throughout the information. For this reason the standard procedures for obtaining additional controls do not provide for replacing the standard set of control characters with an alternative one but rather for the one-at-a-

[2] This technique is already widely in use where 64-graphic printers are used in systems which utilize all 95 graphic characters.

[3] Such systems may be of appreciable commercial significance. A prominent example is that of a message handling system in a country having a nonLatin national alphabet, where the national and standard (and therefore international) alphabets are both useful.

time representation of additional controls by sequences of existing characters, called "escape sequences."

A4.1.2 In order that a code extension sequence may invariably be identifiable as such, each such sequence begins with the prefix character ESC (Escape), which has no other use. (The name Escape is perhaps a little misleading in this respect: the character was initially established as a signal that subsequent operation was to be "not in the standard code.")

A4.2 Sequence Length

A4.2.1 It was at one time proposed that code extension sequences should be standardized as always consisting of ESC and a single-following character. While this would be adequate for many applications, there are a number of considerations which may make longer sequences desirable in many cases. One such consideration is just that of having an adequate number of sequences available for the functions required in one system, or in a number of systems requiring nonconflicting function representations. Another consideration is that it is sometimes desirable to represent a critical function by a long sequence to gain security against accidental or malicious operation. A third consideration is the desire, in some systems, to have a mnemonic relationship between the character sequence and the designation of the function to be controlled.

A4.2.2 In many systems it is very useful to have a doctrine which allows sequences of various lengths to coexist in the same system.

Paramount among the requirements for a variable-length doctrine is the need to have a simple means for a device to determine the end of each sequence which it receives, that is, how many of the characters following ESC are associated with it. This is necessary so that the device may avoid giving the normal interpretation to individual characters of a code extension sequence, even when the specific sequence is not to be recognized and acted upon.

A4.2.3 The procedures of Section 4 provide this flexibility without requiring the use of an "ending" character in each sequence, which carries no other information.

A4.3 Partition of the Code

A4.3.1 There are a number of criteria which affected the way in which the characters of the code were divided into "intermediate" and "final" groups. Among the significant ones were

1. "Intermediates" should be distinguishable from "finals" by a simple logical test, preferably by the sense of 1 bit in the coded representation.
2. A given class of characters, such as alphabetic, and numeric, should be entirely within one group.
3. Uppercase and lowercase of any specific alphabetic characters should be in the same group. This allows a system designer to assign sequences so that no distinction is made on the basis of case, if desired.
4. A number of 2-character (i.e. ESC-plus-one-"final") sequences should be available which use only letters or numerals, because such sequences are convenient for use by humans.
5. The "final" group should contain some characters which are likely to occur with reasonable frequency in a stream of data. This ensures that, should the legitimate final character of a sequence be lost or mutilated, the system will soon be restored to its normal mode of character interpretation.

A4.3.2 These criteria led to partition of the code into "intermediate" and "final" characters as follows (See section 4.6):

Columns 0, 1, and 3 through 7 of the code table contain *final* characters ($b_7b_6b_5 \neq 010$).

Column 2 of the code table contains *intermediate* characters ($b_7b_6b_5 = 010$).

(See A4.4 below for restrictions.)

This partition is felt to produce the most useful balance in the degree to which these criteria are satisfied.

A4.4 Restrictions. The restrictions of Section 4.5 were imposed in order to avoid certain potentially serious problems.

A4.4.1 The ten communication control characters should never be used in Escape sequences. Such use could cause interference with the control logic of communication systems through which the data may be passed, unless the systems were arranged to detect the sequences and determine their lengths, an unnecessary burden. These ten characters are:

Designation	*Name*	*Code table Column/Row*
SOH	Start of Heading	0/1
STX	Start of Text	0/2
ETX	End of Text	0/3
EOT	End of Transmission	0/4
ENQ	Enquiry	0/5
ACK	Acknowledge	0/6
DLE	Data Link Escape	1/0
NAK	Negative Acknowledge	1/5
SYN	Synchronous Idle	1/6
ETB	End of Transmission Block	1/7

Also, additional communication controls should not be represented by ESC sequences but rather by DLE sequences, as described in Section A5.

A4.4.2 The following characters also should not be assigned in Escape sequences:

Designation	*Name*	*Code table Column/Row*
NUL	Null	0/0
CAN	Cancel	1/8
SUB	Substitute	1/10
ESC	Escape (except as the first character)	1/11
DEL	Delete	7/15

A4.4.2.1 NUL is excluded due to the hazards associated with the lack of clearly established conventions for its use and because some systems may be unable to process this character.

A4.4.2.2 CAN is excluded since its purpose is to "cancel" a portion of the data and may thus appear abruptly in a stream of data and may even be used to "cancel" an Escape sequence.

A4.4.2.3 SUB is similarly excluded because it may be used to replace a character determined to be in error and may thus unpredictably appear in an escape sequence as a result of this process.

A4.4.2.4 ESC is excluded to avoid confrontation with the paradox created by its definition as a "final" character: after the first ESC of a sequence another would mean at once that the sequence was starting *and* ending. It therefore seems better to avoid this problem than to become dependent upon specific resolutions of it in equipment.

A4.4.2.5 Finally, DEL is excluded because in some systems it may unpredictably appear as a result of correction of operator errors in perforated tape, and because some portions of a system may "delete" this character from the data stream.

A4.4.3 The use of the remaining control characters in any ESC sequence should be avoided whenever possible, due to the effects which they may cause if the ESC is lost or mutilated or is not recognized for some other reason (see also A4.6).

A4.5 Printing of ESC. The question is often raised as to whether or not typical terminal devices will print a symbol for ESC. It is expected that there will be printing devices capable of printing symbols for many or all of the control characters. Nevertheless, it is a basic concept of the code that, in the ordinary application, the control characters will be nonprinting. Control-printing devices would thus be primarily for monitoring, maintenance, and similar functions.

When a need is expressed for ESC to print, it is instructive to ask, "If there were a character in the code for this function, would we want *it* to print?" If the answer is "no," then it is reasonable to suggest that the ESC (and the rest of the sequence, for that matter) not print. The nonprinting of ESC will no doubt be inherent on most devices as it is a control character; nonprinting of the rest of the Escape sequence may be easily controlled by virtue of the simple way in which the beginning and end of a sequence are delineated.

If the answer is "yes," then either

(1) the application calls for a "monitoring-type" printer anyway, and when one is provided, the ESC will print; or

(2) the function is in the nature of a program instruction, or an abbreviation for something, and should be represented by the syntactic use of graphics anyway, as is traditional in programming languages.

A4.6 Anomalies

A4.6.1 The response of a system to a second ESC inadvertently introduced into a sequence is not prescribed.

A4.6.2 A sequence should always be considered ended if a final character is recognized, whether or not the character is recognized and regardless of whether the final character is an "allowable" one in escape sequences. If the final character is ESC, it may optionally be considered to start a second sequence (see A4.6.1).

A5. Code Extension for Communication Controls: Use of DLE (Data Link Escape)

Standardization of specific procedures for the use of DLE falls within the jurisdiction of ASA Task Group X3.3.4, Communication Control Procedures. The subject is discussed here only to show the relationships of this use to other aspects of code extension.

It is necessary for a communication system to be able to readily distinguish between the communication controls which are of concern to it and other controls with which it is not concerned. The assignment of specific communication control characters in the code provides this distinction under ordinary circumstances. It is necessary that this distinction be preserved when additional controls of one type or the other are represented by escape sequences. The character DLE (Data Link Escape) is provided for use in lieu of ESC as the first character of sequences used to represent additional communication controls. Thus communication link control logic may ignore ESC entirely, passing it and the characters which follow as any other "text" characters. Code extension sequences of concern to the communication control logic can invariably begin with DLE, to which the logic may be made sensitive.

The prohibition previously expressed against the use of communication control characters in ESC sequences is intended to prevent direct interference with the communication control logic.