DEC
STD
110
REV. B

ESCAPE SEQUENCES

TITLE: DEC STANDARD FOR ESCAPE SEQUENCES

ABSTRACT: Indiscriminate echoing for ESC as (33)8 is prohibited. Where it is desirable to print some displayable character to provide visible confirmation that ESC has been received by the program, then that character must be single dollar sign ($;(44)8).

ESC is the character which initially delimits an ESC sequence and ESC may carry no other meaning, even though ESC currently has many other meanings. Applies to all new DEC terminals.

Reviewed and reaffirmed Jul 1975

DATE	ECO #	AUTHOR	APPROVED	REV	SEC	PAGES
01-Jul-74	--	K. Fine		A	--	See Index
07-Mar-75	00001	K. Fine	R. Amann	B	--	See Index

Size	Code	Number	Rev.
A	DS	EL00110-00	B

INDEX: DEC Standard for ESCAPE Sequences - text

*1 JULY 74 DRAFT CORRESPONDED TO REV A

PAGE REVISION CONTROL

PAGE NO.	PAGE REVISIONS	PAGE NO.	PAGE REVISIONS
1	B*		
2	B		
3	B		
4	B		
5	B		
6	B		
7	B		
8	B		
9	B		

SEC. REV	--	SEC. REV	
STD REV	B	STD REV	

1.0 DEFINITIONS

NOTE

Throughout this section, "device" means either a terminal or a program.

1.1 Break Sequence

A break sequence is a sequence of one or more characters, which, when acted upon, causes a device to take some action which cannot be reversed as a result of a single erase character, such as RUBOUT. In addition, no data character which preceeded a break sequence can be rubbed out.

Examples:

1. CTRL/C on the PDP-10.
2. Carriage Return when entering text via the console terminal under DOS-11.

1.2 ASCII/Binary Mode

A device is in ASCII Mode when the device associates meanings with certain characters it receives; a device is in Binary Mode when it treats all characters it receives as pure data with no meanings associated.

1.3 Escape

Insofar as this standard is concerned, ESCAPE (ESC) is the ASCII character whose 7-bit octal representation is 33.

During the last several years, the ASCII Standard has changed. The ESC key replaces the ALTMODE key. ALTMODE transmitted 175(8), and ESC was specified to transmit 176(8). On some older terminals still in use, ESC does transmit 176(8). On newer terminals, ESC transmits 33(8).

This standard is not intended to provide a universally accepted algorithm for recognizing ESC. A way which will work is:

1. If the terminal transmits upper case only, then 175(8), 176(8), and 33(8) should all be interpreted to be ESC, and 175(8) and 176(8) should be converted to 33(8) before further processing commences.

2. If the terminal transmits both upper case and lower case, then ESC is 33(8) only.

For the rest of this document, it is assumed that the only representation of ESC is 33(8). In particular, 175(8) and 176(8) are assumed to have been converted to 33(8). If necessary, before any character processing at all takes place, and that includes echoing.

1.4 ESC Sequence

An ESC sequence is a sequence of two or more characters, the first of which is invariably ESC. The remaining character(s) select a specific function from a potentially very large set.

Examples:

1. ESC A causes the VT50 terminal to do a "cursor up" function.

2. ESC 3 causes a red shift on a model 37 Teletype.

1.5 Function State

A device is in function state if the device will invariably associate some special meaning to the next character it receives.

Examples:

1. If a VT05 terminal is not already in function state, then it enters function state as a result of receiving a 16(8), (CTRL/N). The VT05 assumes that the next character it receives will be the identifier of the new cursor Y-coordinate address.

2. If a VT50 terminal is not already in function state, then it enters function state as a result of receiving a 33(8), (ESC). The VT50 assumes that the next character it receives will be part of an ESC Sequence.

2.0 ESCAPE STANDARD

1. Indiscriminate echoing of ESC, as 33(8), is prohibited. For example, a terminal device driver program is not allowed to implement an "echo-first-think-later" algorithm.

2. Where it is desirable to print some displayable character, solely to provide visible confirmation that ESC has been received by the program, then that displayable character must be a single dollar sign, ($; 44(8)).

3. ESC is the character which initially delimits an ESC sequence; and ESC may carry no other meaning, even though ESC currently has many other meanings.

3.0 SCOPE OF THE STANDARD

3.1 Hardware

This standard applies to the VT50 project and to all future CRT terminals designed and built by DIGITAL. If one of these terminals is in ASCII Mode and not in function state, then the terminal will enter function state as a result of receiving 33(8), (ESC). The terminal will then invariably assume that the next character it receives, whether sent by the host or received form the keyboard in half duplex mode, is part of an ESC sequence. ESC sequences which are longer than two characters can be handled, as indicated in section 5.

In other words, if a terminal is in ASCII Mode and not in function state, then the terminal invariably thinks that ESC is the initial delimiter of an ESC sequence.

3.2 Software

All software supported by Software Engineering for which another release is planned, must conform to 2.0(1). Conformance is required by the next planned release.

All new software (specifically, software which has not yet had a first release) must, in addition, conform to 2.0(2) and 2.0(3).

Existing software, which is not intended to support the CRT-specific features of the VT50 and future CRT terminals (see sec. 5.0(8)), need not conform to 2.0(2) or 2.0(3). Such software may conform, and appropriate modification is encouraged in those instances where conformity is cheap, easy, and of relatively low user impact.

Existing software, which will support any of the CRT-specific features of the VT50 terminal (see sec. 5.0(8)), must conform to 2.0(2) and 2.0(3).

4.0 IMPLEMENTATION SUGGESTIONS

It appears as if there will have to be (at least), two modes of terminal I/O: one mode to support the existing ESC conventions, called $-Mode; and a second mode to support this standard, called ESC-Mode. The basic difference between $-Mode and ESC-Mode is that

ESC, by itself, is allowed to be a break sequence in S-Mode, and this is prohibited in ESC-Mode.

The most general S-Mode requirements are:

1. ESC, by itself, may be a break sequence. For example, TECO uses ESC for this purpose.

2. A program must be able to receive ESC as 33(8) in the proper position in the data stream.

3. If there is some central echoing facility, (e.g., a terminal device driver), and if this facility supports a visible confirmation that ESC has been read, then $ must be echoed, as specified by 2.0(2).

4. If ESC, by itself, is not a break sequence, then the ESC character can be "rubbed out".

The most general ESC-Mode Requirements Are:

1. ESC is used solely to prefix an ESC sequence. Thus, ESC, by itself, cannot be a break sequence. An ESC sequence, however, can be a break sequence.

2. A program must be able to receive ESC as 33(8) in the proper position in the data stream.

3. The user program selects one of two echoing modes for ESC sequences:

   a. Characters are echoed as they are received. In this mode, the user program need not concern itself with echoing, if there is a central terminal driver facility in the operating system.

   b. No part of an ESC sequence is echoed until the program receives the entire ESC sequence. This mode can be used to inhibit printing undesirable data on the teleprinter of a terminal which does not recognize ESC sequences, such as a model 33 Teletype. This mode also may be used with terminals which have function keys which generate complete ESC sequences. The user program can then echo an appropriate character string in response to the ESC sequence.

4. The program selects the echoing mode dynamically; that is, the program may freely switch between the two during execution.

5. Note that a typical small system implementation of passing (just about) all characters, unechoed, to the program, satisfies this requirement.

   a. The program must be able to output ESC as 33(8).

b. If it is desirable to provide visible assurance that ESC has been read, the $ must be echoed, as specified by 2.0(2).

c. If an ESC sequence is not a break sequence, the it can be "rubbed out". (Example: an escape sequence of a user program using the reserved characters. See sec. 5.0(3).)

5.0 ESCAPE SEQUENCE SEMANTICS

1. Every ESC sequence begins with exactly one ESC character, and terminates with exactly one final character, (FC). A final character is any character whose 7-bit octal ASCII representation is 60 through 176, inclusive. An intermediate character, (IC), is any character whose 7-bit octal ASCII representation is 40 through 57, inclusive.

   a. In order to support Parameterized ESC Sequences, it is likely that there will be a set of (IC)s which cause certain (FC)s (e.g., the decimal digits) to be treated as parameters, rather than as (FC)s. This standard will be modified as required to support the ANSI Standard for Parameterized ESC Sequences.

2. The most general form of an ESC sequence (as standardized so far by ANSI) is

   ESC(IC) (IC)... (IC) (FC).

   There can be any number of (IC)s, including none. Two character ESC sequences are the most common, and such a sequence is of the form ESC(FC).

3. There is a class of two character ESC sequences which the ANSI X3.41 standard reserves for private use. This is the set whose (FC) is any character whose 7-bit octal ASCII representation is 60 through 77, inclusive. Digital will assign these 16 2-character escape sequences for control functions which are never likely to be standardized by ANSI. They will be registered in a separate standard.

   This class of ESC sequences is that class which X3.41 will never standardize.

4. No characters in the range 0 through 37(8), inclusive, or 177(8) may be part of an ESC sequence, because certain of these characters convey a special meaning to the software or to the terminal. For example, 23(8) is a synchronization character.

   The rule for processing these "control characters" is that

they have a consistent meaning (in ASCII Mode), regardless of where they occur in the data stream. A straightforward way to implement this standard is to check for control characters first, thus processing them consistently, whether in function state or not.

a. For example, if the cursor is on line 5, column 12, then any one of <CR>ESC A, ESC<CR>A, or ESC A<CR>, will cause the cursor to be positioned on line 4, column 0, 0 being the leftmost column. (See Section 5.0(8)).

This standard does not preclude control characters from cancelling ESC Sequences. DEC standard terminals will enable the "ESC Sequence mode" flip-flop upon receipt of the ESC character. Thus, if ESC is received when an ESC sequence character is expected, then that ESC character begins a new ESC sequence, and the ESC sequence which was in progress is cancelled.

5. These ESC sequence semantics are compatible with the ANSI extension proposal.

6. These ESC sequence semantics are incompatible in every respect to those in use on the model 37 Teletype. The model 37 sequences are all two-character sequences, which are in the proposed never-to-be-standardized set.

7. The use of ESC sequences makes the VT50 incompatible with the VT05 with respecto to: Cursor Up, Cursor Right, Cursor Home, Erase Screen from Cursor to End, Erase Line from Cursor to End, and Direct Cursor Addressing. The VT05 uses CTRL codes to perform these functions.

8. The ESC Sequences currently defined for the VT50 are listed below.

****WARNING****

These ESC sequences may be changed when the ANSI standard is approved. The software should thus be table driven so that redefinition can proceed as orderly as possible.

Feature	VT50 ESC SEQ	VT05	Notes
Cursor Up	ESC A	32(8)	(1)
Cursor Down	None Yet	13(8)	(2)
Cursor Right	ESC C	30(8)	(1)
Cursor Left	None Yet	10(8)	(3)
Direct Cursor Addressing	None Yet	16(8)	
Cursor Home	ESC H	35(8)	(1)
Erase Screen from Cursor	ESC J	37(8)	(1)
Erase Line from Cursor	ESC K	36(8)	(1)
Request Terminal Description	ESC Z	N.A.	(4)

Report Terminal Description	ESC / x	N.A.	(4) (5)
Enable Hold Screen Mode	ESC [	N.A.	(4)
Disable Hold Screen Mode	ESC \	N.A.	(4)
Print	ESC]	N.A.	(4)
Enable Auto Print Mode	ESC 136(8)	N.A.	(4)
Disable Auto Print Mode	ESC 137 (8)	N.A.	(4)

Notes:

1. ESC Sequence is compatible with ASCII extension proposal.

2. VT50 uses 13(8) as well.

3. VT50 uses 10(8) as well.

4. New feature, using a previously unassigned ESC sequence. Consistent with ASCII recommendations.

5. Note the three character ESC sequences.
   x = A if the terminal is a VT50 without a printer;
   x = B if the terminal is a VT50 with a printer.