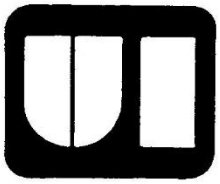
Computer-based Education Research Laboratory
University of Illinois at Urbana-Champaign



CERL Report X-52

December 1991

Level 1

of the

Extended ASCII Protocol

Dale Sinder
James Kurt Mahan
John Hegarty
Monica Fortner

Level 1 of the Extended ASCII Protocol

Dale Sinder
James Kurt Mahan
John Hegarty
Monica Fortner

December 1991

Computer-based Education Research Laboratory

University of Illinois at Urbana-Champaign

This document describes an Enhanced ASCII Protocol for use on the NovaNET® system. It is a superset of the standard PLATO® ASCII protocol and is compatible with Control Data PLATO® systems. The protocol is designed to take advantage of the processing power and display capabilities available on modern micro computers.

Computer-based Education Research Laboratory

Level 1 of the Extended ASCII Protocol

July 1990
September 1990
December 1991

1. General

1.1. Introduction

This document describes "Level 1 of the Extended ASCII Protocol" used for communication with terminals connected to the NovaNET® System. This protocol is a superset of the standard ASCII protocol used for communication with PLATO® systems. Terminal programs implementing this protocol will be compatible with both NovaNET and PLATO systems. This protocol provides for the following improvements on the NovaNET system:

- Full support for XON/XOFF flow control
- Author control of terminal color palettes
- Graphics styles
- Text styles
- In-terminal generation of
  - circles
  - ellipses
  - arcs
  - boxes
  - thick lines
  - dashed lines
  - filled regions
- In-terminal windowing of graphics
- In-terminal display centering
- Pixel resolution pointing
- Author initiated screen printing
- File upload/download capability with KERMIT server
- Local text editing assistance features
- Saving and restoring bitmaps of the screen
- Trapping displays

1.2. Applicable Documents

1.2.1. File "s0ascers";

Sections 1 through 3.4

Appendix A
Appendix B
Appendix D

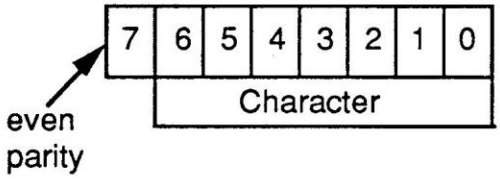
The file "s0ascers" is available for on-line inspection on PLATO systems and NovaNET systems. It describes the standard ASCII protocol.

1.2.2. KERMIT A File Transfer Protocol, Frank da Cruz, Digital Press, 1987

1.3. Definitions

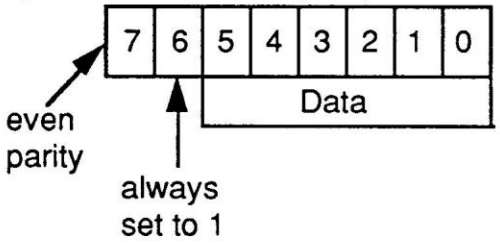
1.3.1. Character

A "Character" contains eight bits and is used to transmit a seven-bit character. The lower seven bits contain the character. The top bit is set for even parity.



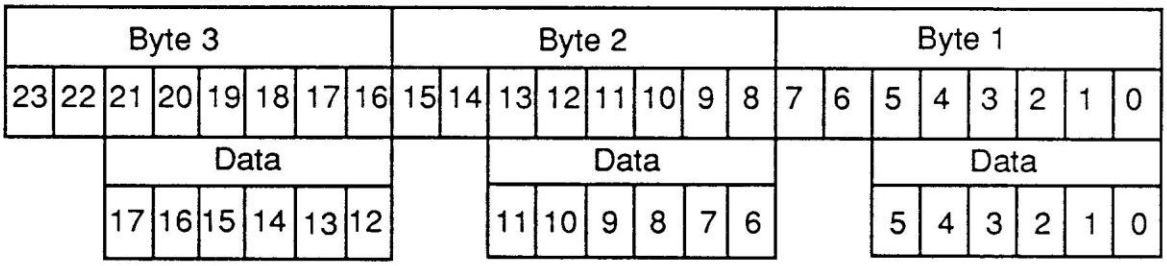
1.3.1. Byte

A "Byte" contains eight-bits and is used to transmit six-bit data values. The lower six bits are the data. The sixth bit is always set to 1. The top bit is set for even parity.



1.3.2. Word

A "Word" is a three-byte sequence that is reduced to 18 bits of data. Each byte received has the same format. The lower six bits are the data. The next to the highest order bit is one. The top bit is set for even parity. The first byte received has the lower six bits of data. The second byte has the middle six bits, and the third byte the most significant six bits of data.



1.3.3. Coordinate

A "Coordinate" is used to specify an x/y location on the screen. A standard coordinate can be from one to four bytes in length. Each time the terminal receives a standard coordinate it must remember the value of each of the one to four bytes received. Later coordinates may omit some of the bytes if they are the same as in the last transmitted standard coordinate. The x/y coordinates are formatted into a sequence of bytes according to the following table:

	bit 6	bit 5	bits 4 to 0
high y	0	1	5 MSB of y
low y	1	1	5 LSB of y
high x	0	1	5 MSB of x
low x	1	0	5 LSB of x

Bits five and six uniquely identify the low x and low y. The high x and y must be inferred from context. The bytes are transmitted in the order shown in the table above. The low x is the only required byte. Each x/y standard coordinate is an unsigned ten-bit quantity.

1.3.4. Color

A "Color" is a 24-bit data item transmitted in four bytes. Each byte contains six bits of data as in the "Word" type. The first byte received contains the lower six bits; the second byte contains bits 6 to 11; the third byte contains bits 12 to 17; and the fourth byte contains the high six bits. In each byte, bit 6 is set, and the top bit is set for even parity.

The upper eight bits of the color data item represent the intensity of the red component of the color. The middle eight bits represent the green component of the color. The lower eight bits represent the blue intensity of the color.

EXAMPLES:

Color	Hex value R G B
black	00 00 00
red	FF 00 00
green	00 FF 00
blue	00 00 FF
yellow	FF FF 00
magenta	FF 00 FF
cyan	00 FF FF
white	FF FF FF
low red	0F 00 00
low green	00 0F 00
low blue	00 00 0F
grey	0F 0F 0F

1.3.5. Slot

A "Slot" is a two-byte sequence containing a 12-bit color palette slot selector. The lower six bits of the first byte are the low-order six bits of the palette slot number. The lower six bits of the second byte are the high-order bits of the palette slot number. The next to the top bit is set in each of the two bytes and the top bit is set for even parity.

1.4. Screen Mapping

Level 1 of the Extended ASCII Protocol is expected to be implemented on various Micro computers which have less than the 512 dot vertical resolution required by NovaNET and PLATO systems. NovaNET and PLATO systems continue to think of the screen as being 512 dots in height. The author of the program implementing the protocol is expected to remap the screen as needed to simulate a 512-dot high display. The loadable character set must also be compressed either when loaded or when displayed to fit into the real vertical dot space. When reporting pointing device locations, the actual pixel must be unmapped and uncentered before reporting.

1.5. Flow Control

Level 1 programs must implement flow control even if they will not lose data at any implemented communication rates. Flow control will permit the program to be used on XON/XOFF networks. Failure to implement flow control will cause the program to fail on those networks. (See Appendix H.)

2. Downline Extensions

2.1. Screen Modes

The following escape sequences are used to set the screen modes in the extended protocol:

ESC d [1B 64 hex]	**Inverse Screen Mode**
ESC e [1B 65 hex]	**Write Screen Mode**
ESC f [1B 66 hex]	**Erase Screen Mode**
ESC g [1B 67 hex]	**Rewrite Screen Mode**
ESC h [1B 68 hex]	**Exclusive OR Screen Mode**

The following table explains how the Exclusive OR screen mode works in monochrome mode. The first column shows whether the pixel on the screen is turned on or off. The second column shows whether the pixel to be plotted is turned on or off. The third column shows whether the pixel will be on or off after the object is plotted in Exclusive Or Mode.

pixel already on screen	pixel to be plotted	after plotting in Exclusive OR
turned on	turned off	turned on
turned on	turned on	turned off
turned off	turned off	turned off
turned off	turned on	turned on

The Exclusive OR mode will behave differently for different color machines. However, all implementations should be sure that after two successive Exclusive OR operations, the screen should be returned to how it looked initially. (See also section 3.2 of this manual.)

The use of these sequences permits the program to operate on XON/XOFF flow control networks.

The old sequences (ESC DC1..DC4) should also be recognized for compatibility with other PLATO systems UNTIL an ESC q w is received (see section 2.14).

2.2. Word Coordinate Mode

In order to be able to set the terminal x and y screen locations to some point off screen (perhaps a negative value) in preparation for a graphics function such as a circle or an arc, a "word coordinate" mode is available in the Extended Protocol.

ESC 4 — **Enter word coordinate mode**
[1B 34 hex]

ESC 5 — **Return to standard coordinate mode**
[1B 35 hex]

In "word coordinate" mode each x/y coordinate is specified as a pair of "Word"s. The first "Word" is the x value. The second "Word" is the y value. This pair of words may be stored as 16-bit twos-complement signed integers.

A switch to or from "word coordinate" mode may occur at any time except in the middle of receiving a coordinate. A command or mode which accepts more than one coordinate may receive one coordinate in "word coordinate" mode and other(s) in standard mode.

2.3. Centering

The program is responsible for centering all display objects on the screen. The central system communicates the offsets by which to center the display.

ESC q x ("Word") **Add "Word" to each x coordinate received**
[1B 71 78 hex)

ESC q y ("Word") **Add "Word" to each y coordinate received**
[1B 71 79 hex)

These offset values affect all aspects of terminal operation.

The program should initially start with centering set to assume a 512-dot wide screen.

2.4. Windowing

The terminal program is responsible for clipping all graphics objects EXCEPT text and fills (mode 4).

ESC q z ("Coordinate1" "Coordinate2")
[1B 71 7A hex)

Specifies the corners of the clipping rectangle. If both coordinates are zero, default clipping is used. With default clipping, graphics objects should still be clipped according to the x and y centering offsets. For example, if the x centering offset is 64, no lines, dots, or other graphics objects should be displayed on the left and right 64 dots of the physical display.

2.5. Palette Support

Lesson authors may control the assignment of colors to palette slots using the NovaNET -palette- command. This section describes the protocol involving color palettes. (See also section 3.2 of this manual.)

2.5.1. Load Palette Slot

ESC n ("Slot" "Color")
[1B 6E hex)

This command instructs the program to load "Color" into "Slot". Slot zero is always "black" and may not be loaded.

A "palette hold flag" is also set when this command is received. When the "palette hold flag" is set, the color palette is not cleared on full-screen erase as is the normal case.

2.5.2. Palette Clear

ESC 6
[1B 36 hex)

This command instructs the terminal to clear the "palette hold flag." While the palette hold flag is set, the palette is not cleared on full-screen erase.

2.5.3. Set Foreground Color

ESC l ("Slot") (l as in letter "el")
[1B 6C hex)

This command causes the color in "Slot" to become the current foreground color. "Slot" also becomes the current foreground slot.

2.5.4. Set Background Color

ESC m ("Slot")
[1B 6D hex)

This command causes the color in "Slot" to become the current background color. "Slot" also becomes the current background slot.

2.6. Abort Input

SOH
[01 hex)

When this byte is detected by the input interrupt handler, all data in the job stack should be discarded. This byte is sent before a full-screen erase which included the aborting of output if the central system does not believe there is any protected output pending. This only works if an ESC q w has been received (see section 2.14).

2.7. Delay

ESC 8 ("Word")
[1B 38 hex)

This command causes the terminal to pause in its display of output for "Word" milliseconds. The terminal program should come as close as it can in the measurement of the time. It is recognized that many machines do not provide facility for resolving time this finely.

2.8. Printing

2.8.1. Print Graphic Screen

ESC v A
[1B 76 41 hex)

This command causes the terminal screen to be printed as graphic data. At the completion of the screen print, the terminal should send an echo of:

0	successful
1	unsuccessful (no printer attached or error occurred)

2.8.2. Entire Text Screen

ESC v B
[1B 76 42 hex)

This command causes the entire terminal screen to be sent as text to the text output file. (See also section 3.2 of this manual.)

2.8.3. Partial Text Screen

ESC v C (Word1, Word2, Word3, Word4)
[1B 76 43 hex)

This command causes part of the terminal screen to be sent as text to the text output file. (See also section 3.2 of this manual.)

"Word1". X location of first corner of area
"Word2". Y location of first corner of area
"Word3". X location of second corner of area
"Word4". Y location of second corner of are

2.9. Charset Loading Mode

ESC s
[1B 73 hex)

This sequence sets the terminal to "Character set load by slot" mode. This is similar to the standard protocol character set load mode except that the Memory Address Register is used to point to a character set slot number rather than a memory address. Character slots 0 - 63 are mapped to character memory M2 ("s0ascers" 3.2.3.1.2.4.1) and character slots 64-127 are mapped to character memory M3.

2.10. Graphics Styles

The extended protocol includes provision for a number of styles that may be imposed on graphics objects. These include:

Patterns
Thickness
Dashes
Fills
Line Caps
Line Joints

The terminal program must return all graphics styles to their default at a full screen erase.

2.10.1. Patterns

ESC i A ("Byte")
[1B 69 41 hex)

This command sets the current "pattern" to pattern number "Byte." The default "pattern" is number 1. Thirty-two patterns numbered 0 to 31 are supported at level 1. The "pattern" should be masked to six bits. The other bit is reserved for future use. See appendix A for the pattern definitions.

The current "pattern" is used for all graphics objects except characters, fills (mode 4), and dots.

The colors written are affected by the current foreground and background colors and the current screen mode according to the table below:

Screen mode	On dots	Off dots
rewrite	foreground	background
inverse	background	foreground
write	(unchanged)	(unchanged)
erase	(unchanged)	(unchanged)
Xor	Xor with current	(unchanged)

See Section 2.10.11 for notes about mode write and mode erase interactions with patterns.

2.10.2. User Defined Patterns

ESC i G ("Byte" "Word1" "Word2" "Word3" "Word4")
[1B 69 47 hex)

This command loads a user defined pattern into slot "Byte" with each word being two horizontal lines of the pattern. "Byte" is equal to slot number (where 32 ≤ slot number ≤ 63). (See also section 3.2 of this manual.)

The following table indicates where a specific bit within a specific word will show up in the 8 x 8 pattern:

Word 1 Bit 7	Word 1 Bit 6	Word 1 Bit 5	Word 1 Bit 4	Word 1 Bit 3	Word 1 Bit 2	Word 1 Bit 1	Word 1 Bit 0
Word 1 Bit 15	Word 1 Bit 14	Word 1 Bit 13	Word 1 Bit 12	Word 1 Bit 11	Word 1 Bit 10	Word 1 Bit 9	Word 1 Bit 8
Word 2 Bit 7	Word 2 Bit 6	Word 2 Bit 5	Word 2 Bit 4	Word 2 Bit 3	Word 2 Bit 2	Word 2 Bit 1	Word 2 Bit 0
Word 2 Bit 15	Word 2 Bit 14	Word 2 Bit 13	Word 2 Bit 12	Word 2 Bit 11	Word 2 Bit 10	Word 2 Bit 9	Word 2 Bit 8
Word 3 Bit 7	Word 3 Bit 6	Word 3 Bit 5	Word 3 Bit 4	Word 3 Bit 3	Word 3 Bit 2	Word 3 Bit 1	Word 3 Bit 0
Word 3 Bit 15	Word 3 Bit 14	Word 3 Bit 13	Word 3 Bit 12	Word 3 Bit 11	Word 3 Bit 10	Word 3 Bit 9	Word 3 Bit 8
Word 4 Bit 7	Word 4 Bit 6	Word 4 Bit 5	Word 4 Bit 4	Word 4 Bit 3	Word 4 Bit 2	Word 4 Bit 1	Word 4 Bit 0
Word 4 Bit 15	Word 4 Bit 14	Word 4 Bit 13	Word 4 Bit 12	Word 4 Bit 11	Word 4 Bit 10	Word 4 Bit 9	Word 4 Bit 8

2.10.3. Thickness

ESC i B ("Word")
[1B 69 42 hex]

This command sets the current "thickness" to "Word" dots. The default "thickness" is 1. The terminal should use only odd thicknesses. Even thicknesses should be reduced by one. The "thickness" applies to all graphics objects except characters, fills, and dots. The "thickness" is not applied when the "filled object flag" is set. See the section on "Polylines" (2.10.10) for the details of the handling of lines.

NOTE: If the pen size is greater than 1, all lines received will not be drawn immediately but held in a data structure. The lines will then be drawn under the following situations:

1. receive an -at- command (skip in a -draw-)
2. exit line mode
3. any line style change
4. any color change

2.10.4. Dash

ESC i D ("Byte")
[1B 69 44 hex]

This command sets the "dash" style to number "Byte." The default is number 0. The system defined dash styles are numbered 0 to 7, and the user defined dash styles are numbered 8 to 15. The "dash" should be masked to four bits. The other bit is reserved for future use. The "dash" style does not pertain to graphic objects other than lines. Also, the "dash" style does not pertain to thick lines or lines with a pattern which is not equal to 1. See the section on "Polylines" (2.10.10) for the details of the handling of lines.

The definition for the dash styles is given in terms of NovaNET pixels for run length (pixels turned on) and skip length (pixels turned off).

The definitions are:

Dash Style	run/skip pixel pattern
0	run n (n is the number of pixels in the line) (solid line)
1	run 64, skip 12
2	run 32, skip 16
3	run 16, skip 16
4	run 8, skip 8
5	run 4, skip 4
6	run 2, skip 2
7	run 32, skip 8, run 8, skip 8

2.10.5. User Defined Dash

ESC i H ("Byte1" "Byte2" "Word1" "Word2".."Wordn")
[1B 69 48 hex]

This command loads a user defined dash style into "Byte1". (See also section 3.2 of this manual.)

"Byte1" is equal to slot number (where 8 ≤ slot number ≤ 15). "Byte2" is equal to the number of words that will be included with the command (where number of words ≤ 8). The words "Word1" through "Wordn" (where n = "Byte2") contain the run-skip length in pixels.

For example:

ESC i H(8,2,12,3)

would define the following dash pattern with a run/skip pattern of 12/3 and would place it in slot #8.

2.10.6. Fill

ESC i F ("Byte")
[1B 69 46 hex]

This command sets or clears the "filled object flag." If "Byte" is 0 the flag is cleared. If "Byte" is 1 the flag is set. The "fill object flag" should be masked to one bit. The other bits are reserved for future use. When the "filled object flag" is set, all graphics primitives except dots and characters are interpreted as filled regions. The "Graphics Modes" section describes each of these regions. See the section on "Polylines" (2.10.10) for the details of the handling of lines.

2.10.7. Cap

ESC i C ("Byte")
[1B 69 43 hex]

This command sets the cap style for lines. The default is 0 = flat. Other cap styles are: 1 = round and 2 = square. The cap style should be masked to two bits. The other bits are reserved for future use. See Appendix B for examples. See the section on "Polylines" (2.10.10) for the details of the handling of lines.

2.10.8. Join

ESC i E ("Byte")
[1B 69 45 hex]

This command sets the join style for Polylines. The default is the round joint (2). Other join styles are: 0 = mitre and 1 = bevel. The join style should be masked to two bits. The other bits are reserved for future use. See the section on "Polylines" (2.10.10) for the details of the handling of lines. See Appendix C for examples.

2.10.9. Clear

ESC 7
[1B 37 hex]

This command returns all graphics styles to the defaults. A full-screen erase also returns the graphics styles to the defaults.

2.10.10. Polylines: Lines and Styles

It is necessary to buffer data received in line mode when the "thickness" is greater than one, when the dash style is non-zero, or when the "fill object flag" is set. This allows the joints to be properly made, and allows the program to determine the region to be filled, before the filling process begins. The buffer of line data should be displayed when any of the following appear in the input stream:

An -at- command (skip in a draw)
A terminal mode command including set to line mode
A style command
A color or palette command

If the "fill object flag" is set, the beginning and ending points of the continuous draw should be joined and the resulting area filled with the current "pattern." If the area is complex such that subareas are enclosed multiple times, the subarea should be filled only if it is enclosed an odd number of times. See Appendix D for an example.

Some styles apply to other objects as well. For example, if the fill style is in use and an ARC command is sent to the terminal, a pie slice is drawn using the current pattern, color etc.

Lines are drawn according to the current pattern, size, cap, join, and frame/fill styles. Thus one can draw "nicely flowing" thick, patterned lines with nice joints. Also, if the fill style is in use, the area enclosed by the arbitrary lines will be filled with the current pattern.

2.10.11. Mode Write/Erase and Patterns

Level one does not support modes write and erase (transparent) for certain styled operations. If the operation is not supported, the requested graphics function is ignored. It is expected that modes write and erase will be supported in higher levels. These operations are ignored in level 1 in order to force authors to use modes rewrite and inverse. This insures that lessons will continue to function when higher levels do support the transparent modes.

A graphics object is to be ignored if patterns apply and:

((MODE = WRITE OR MODE = ERASE) AND PATTERN ≠ 1)

Since pattern 1 is solid foreground, it may be used.

A graphics object will not be ignored when stretching or trapping.

2.11. Graphics Modes and Commands

This section describes the graphics objects that have been added to level 1 and the effect of the various styles on these objects.

2.11.1. Modes

2.11.1.1. Arcs (circular)

ESC q B ("Word1" "Word2" "Word3")
[1B 71 42 hex]

The first three bytes set the terminal to "Circular Arc" mode. In this mode three "Words" are interpreted as:

"Word1". The radius in dots.
"Word2". The starting angle in tenths of degrees (0..3600).
"Word3". The Arc length in tenths of degrees (-3600..3600).

The current screen mode, "thickness", and "pattern" apply unless the"fill object flag" is set. When the "fill object flag" is set, the current screen mode and "pattern" apply and a pie slice is drawn.

2.11.1.2. Arcs (elliptical)

ESC q A ("Word1" "Word2" "Word3" "Word4")
[1B 71 41 hex]

The first three bytes set the terminal to "Elliptical Arc" mode. In this mode four "Words" are interpreted as:

"Word1". X radius
"Word2". Y radius
"Word3". The starting angle in tenths of degrees (0..3600).
"Word4". The Arc length in tenths of degrees (-3600..3600).

The current screen mode, "thickness", and "pattern" apply unless the "fill object flag" is set. When the "fill object flag" is set, the current screen mode and "pattern" apply and a pie slice is drawn.

Note that, in PLATO and NovaNET, angles are distorted when objects are sized. The Arc defined here does NOT distort the angles when the x and y radii are not equal. PLATO or NovaNET must recompute the angles when sending this type of Arc.

2.11.1.3. Circles

ESC q C ("Word")
[1B 71 43 hex]

The first three bytes set the terminal to "Circle" mode. In this mode a "Word" is interpreted as a radius. The terminal x/y is left at the center of the circle.

The current screen mode, "thickness", and "pattern" style apply unless the "fill object flag" is set. When the "fill object flag" is set, the current screen mode and"pattern" apply and a disk is drawn.

2.11.1.4. Ellipses

ESC q D ("Word1" "Word2")
[1B 71 44 hex]

The first three bytes set the terminal to "Ellipse" mode. In this mode two words are interpreted as:

"Word1". The X radius
"Word2". The Y radius

The terminal x/y is left at the center of the ellipse.

The current screen mode, "thickness", and "pattern" style apply unless the "fill object flag" is set. When the "fill object flag" is set, the current screen mode and "pattern" apply and an elliptical disk is drawn.

2.11.1.5. Boxes

ESC q K ("Coordinate1" "Coordinate2" "Word")
[1B 71 4B hex]

The first three bytes set the terminal to "Box" mode. In this mode two "Coordinates" and a "Word" are received:

"Coordinate1". One corner of the box
"Coordinate2". The opposite corner
"Coordinate3". The box thickness

The terminal x/y is left at the inner lower left-hand corner.

The current screen mode, and "pattern" style apply unless the "fill object flag" is set. When the "fill object flag" is set the current screen mode and "pattern" apply and the rectangular region is filled. Boxes are the single exception to the odd thickness rule. Boxes may have even or odd thicknesses.

2.11.2 Commands for Stretching

2.11.2.1 Stretch Endline

ESC q L ("Word1" "Word2" "Word3" "Word4")
[1B 71 4C hex]

This command is used to stretch a line from its endpoint. The four words are interpreted as:

"Word1". X location of first endpoint
"Word2". Y location of first endpoint
"Word3". X location of second endpoint (cursor is attached here)
"Word4". Y location of second endpoint

The cursor is attached to the second endpoint, and the stretching is done from that location.

2.11.2.2 Stretch Midline

ESC q M ("Word1" "Word2" .. "Word6")
[1B 71 4D hex]

This command is used to stretch a line from its midpoint. The six words are interpreted as:

"Word1". X location of the first endpoint
"Word2". Y location of the first endpoint
"Word3". X location of the midpoint (cursor is attached here)
"Word4". Y location of the midpoint
"Word5". X location of the second endpoint
"Word6". Y location of the second endpoint

The cursor is attached to the midpoint, and the stretching is done from that location.

2.11.2.3. Stretch Circle

ESC q N ("Word1")
[1B 71 4E hex]

This command is used to stretch a circle. The word is interpreted as:

"Word1". length of radius (cursor is attached here)

The cursor is attached to the endpoint of the radius and the stretching is done from that location.

2.11.2.4. Stretch Ellipse

ESC q O ("Word1", "Word2") (X direction)
[1B 71 4F hex]
ESC q P ("Word1", "Word2") (Y direction)
[1B 71 50 hex]

These commands are used to stretch an ellipse in either the X direction or the Y direction. The three words are interpreted as:

"Word1". X radius (cursor is attached here for the X direction)
"Word2". Y radius (cursor is attached here for the Y direction)

The cursor is attached to the endpoint of the radius and the stretching is done from that location. The center of the eclipse does not change when this operation is done.

2.11.2.5. Stretch Box

ESC q Q ("Word1" .. "Word5") (Stretch Corner)
[1B 71 51 hex]
ESC q R ("Word1" .. "Word5") (Stretch Thickness)
[1B 71 52 hex]

These commands are used to stretch a box from the corner or stretch the thickness of a box. The five words are interpreted as:

"Word1". X location of first corner of box
"Word2". Y location of first corner of box
"Word3". X location of second corner (cursor is attached here)
"Word4". Y location of second corner of box
"Word5". Thickness of box

The cursor is attached to the second corner, and the stretching is done from that location.

2.11.2.6. Stretch Fill

ESC q S ("Word1" "Word2" "Word3" "Word4")
[1B 71 53 hex]

This command stretches a fill area. The four words are interpreted as:

"Word1". X location of first corner
"Word2". Y location of first corner
"Word3". X location of second corner (the cursor is attached here)
"Word4". Y location of second corner

The cursor is attached to the second corner, and the stretching is done from that location.

2.11.2.7 Stop Stretch

ESC q Z
[1B 71 5A hex]

This command stops the stretch action.

2.12. Text Styles

The extended protocol includes provision for a number of text style operations which may be performed on a character before it is plotted. The terminal should maintain a 16-bit word to identify which styles are in effect. The bits in that word have the following meanings:

bit	meaning
0	thicken the character in the horizontal direction
1	thicken the character in the vertical direction
2	apply an italic like slant to the character
3	underline all but class 2 characters
4	underline all characters
5	strikeout all but class 2 characters
6	strikeout all characters
7	apply a checkerboard mask to the character
8	superscript character marker (screen text editor)
9	subscript character marker (screen text editor)
10	invert character before plotting

The following are defined as class 2 characters:

(space) ! , . : ; ? ' " tilde universal delimiter arrow char umlaut acute accent cedilla hacek (All M3 characters.)

If the corresponding bit is on, the style should be applied.

Underlining should add a full width horizontal line to the lowest pixel of the character.

Strikeout should add a full width horizontal line at the middle of the character.

Styles should be applied in order, bit zero to bit ten.

2.12.1. Text Style Commands

These commands change the 16-bit word which controls the text styles:

2.12.1.1. Logical "OR"

ESC i a ("Word")
[1B 69 61 hex]

Logical "OR" of "Word" with the old style bits.

2.12.1.2. Exclusive "OR"

ESC i b ("Word")
[1B 69 62 hex]

Exclusive "OR" of "Word" with the old style bits.

2.12.1.3. Style

ESC i c ("Word")
[1B 69 63 hex]

Set the style bits to "Word".

2.12.1.4. Logical "AND"

ESC i d ("Word")
[1B 69 64 hex]

Logical "AND" the complement of "Word" with the old style bits.

2.12.1.5. Clear bit

ESC i e	**Clear bit 0.**	**[1B 69 65 hex]**
ESC i f	**Set bit 0.**	**[1B 69 66 hex]**
ESC i g	**Clear bit 1.**	**[1B 69 67 hex]**
ESC i h	**Set bit 1.**	**[1B 69 68 hex]**
ESC i i	**Clear bit 2.**	**[1B 69 69 hex]**
ESC i j	**Set bit 2.**	**[1B 69 6A hex]**
ESC i k	**Clear bit 3.**	**[1B 69 6B hex]**
ESC i l	**Set bit 3.**	**[1B 69 6C hex]**
ESC i m	**Clear bit 4.**	**[1B 69 6D hex]**
ESC i n	**Set bit 4.**	**[1B 69 6E hex]**
ESC i o	**Clear bit 5.**	**[1B 69 6F hex]**
ESC i p	**Set bit 5.**	**[1B 69 70 hex]**
ESC i q	**Clear bit 6.**	**[1B 69 71 hex]**
ESC i r	**Set bit 6.**	**[1B 69 72 hex]**
ESC i s	**Clear bit 7.**	**[1B 69 73 hex]**
ESC i t	**Set bit 7.**	**[1B 69 74 hex]**
ESC i u	**Clear bit 8.**	**[1B 69 75 hex]**
ESC i v	**Set bit 8.**	**[1B 69 76 hex]**
ESC i w	**Clear bit 9.**	**[1B 69 77 hex]**
ESC i x	**Set bit 9.**	**[1B 69 78 hex]**
ESC i y	**Clear bit 10.**	**[1B 69 79 hex]**
ESC i z	**Set bit 10.**	**[1B 69 7A hex]**

2.13. External Mode

ESC q u
[1B 71 75 hex]

This sequence causes the terminal to enter external data mode. Data words received in this mode are treated as external data. (See Appendix H.)

2.14. Extension select

ESC q w
[1B 71 77 hex]

This command turns on Level 1 enhancements which are not on by default. These enhancements are not on because they would interfere with operation under the standard protocol. These enhancements include:

Pixel resolution pointing.

Ignoring DC1, DC2, DC3, DC4 in the input stream.

Use of SOH to abort the input stream.

A TTY mode command will turn these enhancements off.

2.15. Report Terminal Characteristics

ESC q v
[1B 71 76 hex]

When this command is received the program should respond as indicated in section 3.2.

2.16. Alert Terminal

ESC q t
[1B 71 74 hex]

When this command is received, the program may assume that a message is about to arrive in the input stream which was sent to this user's station by another station. Examples would be system messages and TERM-talks. In an environment where it is possible to make the program "semi-active", such as a Microsoft® Windows, OS/2, or Macintosh® environment, this command may be used to alert the user that a message has arrived that may be of interest while the user is in another local application. The method of alert used is left to the program author, but might include such options as beeping the terminal, flashing the ICON for the program, or making the program the active program. Authors might additionally consider making the alert action user configurable. However, this command should be ignored unless the program is in a "semi-active" state such as being reduced to an ICON.

2.17. Filename Buffer

A buffer, which holds a filename, is used by the imaging and trap commands. The filename buffer is 100 bytes long. The following commands manipulate the filename buffer.

2.17.1. Clear Filename

ESC q r
[1B 71 72 hex]

This command clears the filename buffer and should be done before the buffer is loaded.

2.17.2. Load filename

ESC q s ("Byte1" "Byte2"..."Byten")
[1B 71 73 hex]

This mode loads the filename. "Bytes1", "Bytes2", through "Bytesn" contain the ASCII representation of the characters of the filename.

2.17.3. Filename sumchk

ESC q i ("Word")
[1B 71 69 hex]

This command sets the filename sumchk to "Word1". The algorithm for determining the sumchk is given below in C:

```
sumchk = 0;
for ( i = 0 ; i < pad.fnlth ; i++)
/* pad.fnlth = # of characters in filename */
   sumchk ^= ( (int) (pad.filename[i]) << ( i & 0x7) );
```

2.18. Run Filename

ESC x a
[1B 78 61 hex]

This command causes the filename in the filename buffer to be executed by the local operating system. (See also section 3.2 of this manual.) This works in conjunction with the TUTOR command -local execute-. The level 1 program should return one of the following:

ESC /	**[1B 2F]**	**successful local activity**
ESC ?	**[1B 3F]**	**failed local activity**

2.19. Trap

The TUTOR -trap- commands are used to trap, save and restore portions of the NovaNET displays. See NovaNET lesson "aid" for information on how to use the TUTOR -trap- commands. The following commands describe the Level 1 protocol for using the TUTOR -trap- commands. (See also section 3.2 of this manual.)

2.19.1. Trap save

ESC q p ("Word1" "Byte1")
[1B 71 70 hex]

This command marks where to begin saving the display. The display is saved into buffer ID "Word1". If "Byte1" is equal to 'A", then do not display while trapping. If "Byte1" is equal to 'B', then display while trapping.

2.19.2. Trap end

ESC q q
[1B 71 71 hex]

This command marks where to end the trapping and returns an echo code.

2.19.3. Trap display

ESC q n ("Word1" "Word2" "Word3")
[1B 71 6E hex]

This command displays the trapped buffer with ID "Word3". The display can be offset from the original position.

In this command, the three "Words" are interpreted as:

"Word1". Buffer ID to display.
"Word2". Offset from X location.
"Word3". Offset from Y location.

2.19.4. Trap delete

ESC q o ("Word")
[1B 71 6F hex]

This command deletes a trap buffer with ID "Word1". If "Word1" is equal to -1, then delete all traps.

2.19.5. Trap drag

ESC q j ("Word1" "Word2" "Word3" "Word4" "Word5")
[1B 71 6A hex]

This command allows a trapped display to be attached to a local pointer (mouse) and dragged around on the screen.

The dragged object should be plotted in Exclusive Or mode so as not to be destroy what was previously on the screen. However, if a -mode- command is encountered in the trapped display, the -mode- commands should be obeyed which means that the drag can destroy what was previously on the screen. Therefore, while dragging a trapped display, filled objects may not necessarily be in mode write or mode rewrite.

In this command, the five "Words" are interpreted as:

"Word1". Buffer ID to drag.
"Word2". Offset from X location.
"Word3". Offset from Y location.
"Word4". Initial X location for local pointer.
"Word5". Initial Y location for local pointer.

2.19.6. Stop drag

ESC q h
[1B 71 68 hex]

This command stops the drag.

2.19.7. Trap write

ESC q l ("Word")
[1B 71 6C hex]

This command specifies that a trapped buffer (with ID "Word1") should be written to local disk. The buffer is written to the file specified by the local filename buffer (see section 2.17).

2.19.8. Trap read

ESC q m ("Word")
[1B 71 6D hex]

This command specifies that a trapped buffer should be read from local disk and stored in a buffer with ID "Word1". The buffer is read from the file specified by the local filename buffer (see section 2.17).

2.19.9. Trap test

ESC q k
[1B 71 6B]

This command allows you to test and see if a trapped file exists. The filename must be previously loaded in the filename buffer (see section 2.17.2).

2.19.10. Echo Codes

The echo code returned from the Level 1 Program for the trap commands are:

0	ok
1	file operations not allowed
2	no file name in the filename buffer
3	can't open specified file
4	file is too short for trap file
5	file is not a trap file or not version compatible
6	trap buffer id out of range
7	l1pp general error
8	file too big for trap file
9	l1pp error (not enough memory)
10	file read error
11	trap buffer empty / no trap save in effect
12	can't create file
13	file write error
14	file name sumchk error

2.20. Local Text Editing

The following commands are used to support local text editing. (See also section 3.2 of this manual.)

2.20.1. Turn on Text Cursor

ESC x A
[1B 78 41 hex]

This command turns on the text cursor. The cursor is turned off by a full screen erase or by the ESC x B sequence.

2.20.2. Turn off Text Cursor

ESC x B
[1B 78 42 hex]

This command turns off the text cursor.

2.20.3. Set Right Margin

ESC x C ("Word")
[1B 78 43 hex]

This command sets the right margin for scroll deletion and scroll insertion to "Word" pixels (NovaNET pixels).

2.20.4. Scroll Deletion

ESC x D ("Character")
[1B 78 44 hex]

This command deletes "Character"-32 character positions and scrolls the line in from the right margin.

2.20.5. Scroll Insertion

ESC x E ("Character")
[1B 78 45 hex]

This command inserts "Character"-32 character positions and scrolls the line out to the right margin.

2.20.6. Scroll

ESC x F ("Character1" "Character2" "Character3")
[1B 78 46 hex]

This command scrolls "Character1"-32 characters on "Character2"-32 lines for "Character3"-70 lines up. Or, if "Character3"-70 is less than zero, it scrolls ("Character3-70) lines down.

2.20.7. Set Cursor Style

ESC x G ("Character")
[1B 78 47 hex]

This command sets the cursor style to "Character"-32.

0	horizontal underline at x/y
1	vertical line from x-1, y to x-1, y+15
2	block cursor (invert char at x/y)

2.20.8. Control Cursor Flashing

ESC x H
[1B 78 48 hex]

This command turns on the flashing of the text cursor. (This overrides the terminal configuration, but does not change it.)

ESC x I
[1B 78 49 hex]

This command turns off the flashing of the text cursor. (This overrides the terminal configuration, but does not change it.)

2.20.9. Keymapping

ESC x K

[1B 78 4B hex]

This command turns on text editing keymapping. (See also section 3.2 of this manual.)

ESC x M
[1B 78 4D hex]

This command turns off text editing keymapping.

When terminal text editing keymapping is turned on, the following keys are sent to the central system:

Key Name	for unshifted	for shifted
UP	ESC 0	ESC (space)
DOWN	ESC 1	ESC !
RIGHT	ESC 2	ESC "
LEFT	ESC 3	ESC #
INSERT	ESC 4	ESC $
DELETE	ESC 5	ESC %
HOME	ESC 6	ESC &
END	ESC 7	ESC '
PageUp	ESC 8	ESC (
PageDown	ESC 9	ESC)

In text editing keymapping mode, the following Control keys (on MS-DOS machines) or Option keys (on Macintosh machines) should be mapped to these NovaNET function keys:

Control or Option Key	NovaNET key
a	ANS
b	BACK
B	SHIFT-BACK
c	COPY
C	SHIFT-COPY
d	DATA
D	SHIFT-DATA
e	EDIT (not on Macintosh)
E	SHIFT-EDIT (not on Macintosh)
f	FONT
g	Divide symbol (÷)
G	Shifted divide symbol (∩)
h	HELP
H	SHIFT-HELP
I	LAB
L	SHIFT-LAB
m	MICRO
p	SUPER
P	SHIFT-SUPER (Locking SUPER)
q	SQUARE
Q	ACCESS
s	STOP
S	SHIFT-STOP
t	TERM
u	SUPER (not on Macintosh)
U	SHIFT-SUPER (not on Macintosh)
w	EDIT
W	SHIFT-EDIT
x	Times symbol (x)
X	Shifted Times symbol (∪)
y	SUB
Y	SHIFT-SUB (Locking SUB)

Sometimes it may be impossible to press Control or Option and a certain character. (For example, Option-e does not work on the Macintosh). If this situation, a different combination of keys should be pressed such as Option-w.)

2.21. Image

These commands are used to support the TUTOR -image- command. The -image- command is used to save parts of the screen as a bitmap. The command can also be used to place the saved bitmap back on the screen in any location. See lesson "aid" for more information about the -image- command. (See also section 3.2 of this manual.)

Format of word used to specify the location:

bits 0-7:	ID number 0 = all slots, 1..255 = slot 0 to 254.
bits 8-10:	Type 0 = disk, 1 = memory, 2 = clipboard
bit 11	Get Only, load palette flag
bits 11 - 13	Save Only, format: 0 = local, 1= pcx

2.21.1. Save Image

ESC j a ("Word1" "Word2" "Word3" "Word4" "Word5")
[1B 6A 61 hex]

This command saves a rectangle defined by first four words to the location specified by the fifth word. This command echoes the results.

In this command, the "Words" are interpreted as:

"Word1". X location of first corner of rectangle
"Word2". Y location of first corner of rectangle
"Word3". X location of second corner of rectangle
"Word4". Y location of second corner of rectangle
"Word5". Location to store image

2.21.2. Get Image

ESC j b ("Word1" "Word2" .. "Word9")
[1B 6A 62 hex]

This command gets a subrectangle (first 4 "Words") of a specified image (9th "Word") and places it on the screen in a specified location ("Words" 5 - 8).

In this command, the "Words" are interpreted as:

"Word1". X location of first corner of source rectangle
"Word2". Y location of first corner of source rectangle
"Word3". X location of second corner of source rectangle
"Word4". Y location of second corner of source rectangle
"Word5". X location of first corner of destination rectangle
"Word6". Y location of first corner of destination rectangle
"Word7". X location of second corner of destination rectangle
"Word8". Y location of second corner of destination rectangle
"Word9". Image to get

2.21.3. Image Information

ESC j c ("Word")
[1B 6A 63 hex]

This command returns a pixel touch word with information about the X and Y resolution of the image specified by "Word".

2.21.4. Copy Image

ESC j d ("Word1" "Word2" .. "Word8")
[1B 6A 64 hex]

This command copies an image specified by "Words" 1 - 4 to a rectangle specified by "Words" 5 - 8.

In this command, the "Words" are interpreted as:

"Word1". X location of first corner of source rectangle
"Word2". Y location of first corner of source rectangle
"Word3". X location of second corner of source rectangle
"Word4". Y location of second corner of source rectangle
"Word5". X location of first corner of destination rectangle
"Word6". Y location of first corner of destination rectangle
"Word7". X location of second corner of destination rectangle
"Word8". Y location of second corner of destination rectangle

2.21.5. Delete Image

ESC j e ("Word")
[1B 6A 65 hex]

This command deletes an image specified by "Word" and echoes the result.

2.21.6. Save Entire Screen

ESC j 1 (1 as in number one)
[1B 6A 31 hex]

This command saves the entire screen.

2.21.7. Restore Entire Screen

ESC j 2
[1B 6A 32 hex]

This command restores the screen which had been saved with an ESC j 1.

2.21.8. Echo Key

ESC j f
[1B 6A 66 hex]

This command returns an echo key with the result from the last get/copy.

The echoes from the L1PP are:

0	success
1	file operations disable
2	no filename in buffer
3	can't open file
4	file not in format readable by -image- command
5	slot ID out of range
6	terminal has insufficient memory for operation
7	ID buffer is empty
8	filename sumchk error
9	program doesn't support this file format
10	subrectangle or coordinate out of range

2.22. Output

(See also section 3.2 in this manual.)

2.22.1. Inhibit output

ESC x b
[1B 78 62 hex]

This command inhibits the user from doing local output such as generating prints or storing information in files.

2.22.2. Allow output

ESC x c
[1B 78 63 hex]

This command allows the user to do local output such as generating prints or storing information in files. The default is to allow output.

2.23. Input

(See also section 3.2 in this manual.)

2.23.1. Inhibit input

ESC x d
[1B 78 64 hex]

This command inhibits local input such as automatic typing.

2.23.3. Allow input

ESC x e
[1B 78 65 hex]

This command allows local input such as automatic typing. The default is to allow input.

3. Upline Extensions

3.1. Pixel Resolution Pointing

When pixel resolution pointing has been turned on (by ESC q w), pointing information is transmitted upline in the following format:

ESC 1E(hex) ("Byte" "Word1" "Word2")
[1B 1E hex]

"Byte" has the seventh bit set and the lower 6 bits are used for a map of the mouse buttons pressed. ("Byte" is usually equal to 41 (hex)).

The two "Words" are the X and Y locations of the pointing device (x first).

If pixel resolution pointing has not been enabled by ESC q w (see section 2.14), pointing information should be transmitted according to "s0ascers" .

3.2. Report Terminal Characteristics

This command sends information about the terminal's capabilities as a pixel touch response. All of these capabilities are optional.

ESC 1E(hex) (60(hex) "Word1" "Word2")
[1B 1E 60 hex]

"Word1":

Bits	Information	Section in Manual
0-9	X resolution	
10	Trap/Stretch	(2.19)
11	User defined patterns/dash styles	(2.10.2, 2.10.5)
12	Local Program Execution	(2.18)
13	Xor mode	(2.1)
14	Image support	(2.21)
15	Local inhibit	(2.22, 2.23)
16	Text editing primitives	(2.20)
17	Text prints available	(2.8.2, 2.8.3)

"Word2":

Bits	Information	Section in Manual
0-11	Number of palettes/colors supported	
12	-palette- command supported	(2.5)
13	-color- command supported	
14	Kermit server	(4)
15	Text editing keymapping	(2.20.9)
16	unused, should be zero	
17	unused, should be zero	

3.3. Terminal Subtype Echo

The terminal should report subtype 116 to identify itself as a level 1 protocol terminal.

3.4. Operating System Echo

An additional echo has been added to the ESC Y sequence ("s0ascers" 3.2.3.1.4) to identify the operating system of the microcomputer.

ESC Y 54 (hex)
[1B 59 54 hex]

This echo returns:

0	MS-DOS®
1	Macintosh®
2	Unix™
3	Microsoft® Windows™
4	OS/2®

4. KERMIT Interface

(See also section 3.2 of this manual)

4.1. Escape Sequences

4.1.1. Set Upline transmission rate

ESC x x ("Word")
[1B 78 78 hex]

This command sets an upper limit of "Word" bytes per second on the upline transmission rate of the terminal while in KERMIT server mode. This command is received before entering KERMIT server mode.

4.1.2. Enter KERMIT Server Mode

ESC x z
[1B 78 7A hex]

This command causes the terminal to enter KERMIT server mode. Commands in the KERMIT protocol will follow.

4.1.3. Exit KERMIT Server Mode

ESC x y
[1B 78 79 hex]

This command causes to terminal to exit KERMIT server mode and return to normal "NovaNET" mode of operation.

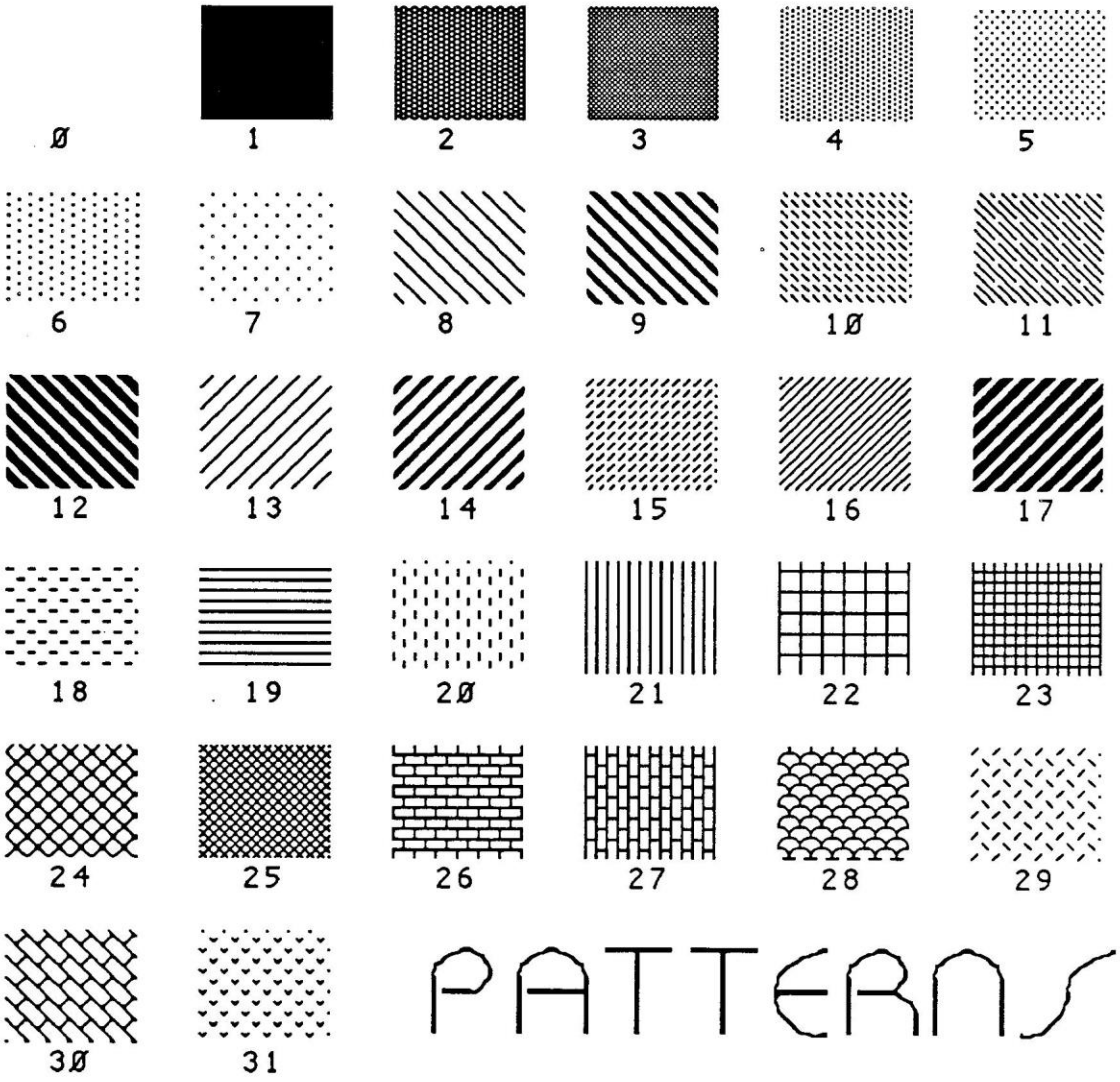
4.2. KERMIT Server Mode

In this mode the terminal, except for recognizing the exit KERMIT server mode command, should act as a normal KERMIT server.

At the minimum, the server should support the send initiate, receive initiate, and generic (logout) commands.

For a complete description of the KERMIT protocol, see "KERMIT. A File Transfer Protocol", by Frank da Cruz.

5. Appendix A: Style Patterns



Each 8 by 8 pattern is detailed left to right, top to bottom:

```
Pattern 0       Pattern 1       Pattern 2       Pattern 3

--------        11111111        111-111-        1-1-1-1-
--------        11111111        1-111-11        -1-1-1-1
--------        11111111        111-111-        1-1-1-1-
--------        11111111        1-111-11        -1-1-1-1
--------        11111111        111-111-        1-1-1-1-
--------        11111111        1-111-11        -1-1-1-1
--------        11111111        111-111-        1-1-1-1-
--------        11111111        1-111-11        -1-1-1-1

Pattern 4       Pattern 5       Pattern 6       Pattern 7

1---1---        --1---1-        ----1---        --------
--1---1-        --------        --------        --------
1---1---        1---1---        1-------        ----1---
--1---1-        --------        --------        --------
1---1---        --1---1-        ----1---        --------
--1---1-        --------        --------        --------
1---1---        1---1---        1-------        1-------
--1---1-        --------        --------        --------

Pattern 8       Pattern 9       Pattern 10      Pattern 11

1-------        111-----        1---1---        -1---1--
-1------        -111----        -1---1--        --1---1-
--1-----        --111---        --1---1-        ---1---1
---1----        ---111--        --------        1---1---
----1---        ----111-        1---1---        -1---1--
-----1--        -----111        -1---1--        --1---1-
------1-        1-----11        --1---1-        ---1---1
-------1        11-----1        --------        1---1---

Pattern 12      Pattern 13      Pattern 14      Pattern 15

--11111-        ---1----        ---111--        --1---1-
---11111        --1-----        --111---        -1---1--
1---1111        -1------        -111----        1---1---
11---111        1-------        111-----        --------
111---11        -------1        11-----1        --1---1-
1111---1        ------1-        1-----11        -1---1--
11111---        -----1--        -----111        1---1---
-11111--        ----1---        ----111-        --------
```

Pattern 16

```
---1---1
--1---1-
-1---1--
1---1---
---1---1
--1---1-
-1---1--
1---1---
```

Pattern 17

```
-11111--
11111---
1111---1
111---11
11---111
1---1111
---11111
--11111-
```

Pattern 18

```
111-----
--------
--------
--------
----111-
--------
--------
--------
```

Pattern 19

```
11111111
--------
--------
--------
11111111
--------
--------
--------
```

Pattern 20

```
1-------
1-------
1-------
--------
----1---
----1---
----1---
--------
```

Pattern 21

```
1---1---
1---1---
1---1---
1---1---
1---1---
1---1---
1---1---
1---1---
```

Pattern 22

```
11111111
1-------
1-------
1-------
1-------
1-------
1-------
1-------
```

Pattern 23

```
11111111
1---1---
1---1---
1---1---
11111111
1---1---
1---1---
1---1---
```

Pattern 24

```
--1--1--
---11---
---11---
--1--1--
-1----1-
1------1
1------1
-1----1-
```

Pattern 25

```
-1---1--
1-1-1-1-
---1---1
1-1-1-1-
-1---1--
1-1-1-1-
---1---1
1-1-1-1-
```

Pattern 26

```
----1---
----1---
11111111
1-------
1-------
1-------
11111111
----1---
```

Pattern 27

```
1---1---
1---1---
1---1111
1---1---
1---1---
1---1---
11111---
1---1---
```

Pattern 28

```
1-------
111---11
---1-1--
----1---
----1---
--11111-
-1-----1
1-------
```

Pattern 29

```
-----1--
------1-
--------
--1-----
-1------
1-------
--------
----1---
```

Pattern 30

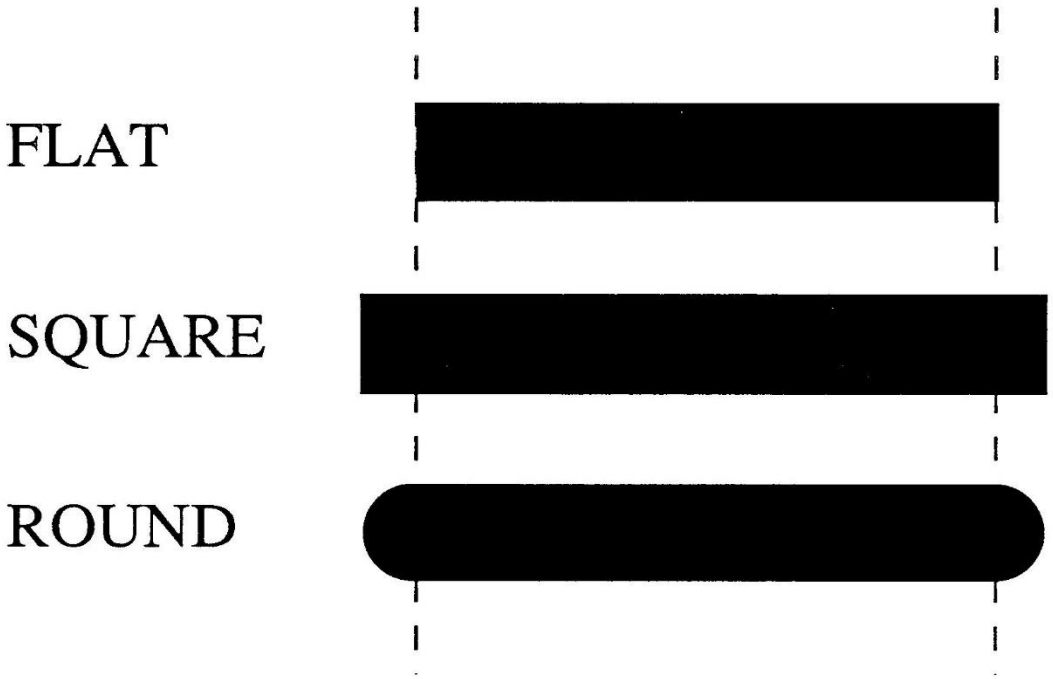
```
-----1--
------1-
-------1
1------1
-1----1-
--1--1--
---11---
----1---
```

Pattern 31

```
1-1-----
-1------
--------
--------
----1-1-
-----1--
--------
--------
```

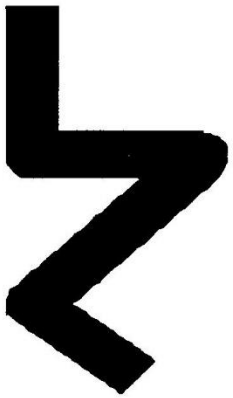
6. Appendix B: Line Cap Styles

The same line drawn with flat, square, and round Cap style:



7. Appendix C: Line Join Style

Sample of round joints:



8. Appendix D: Complex Filled Areas

The following drawings illustrate the same lines drawn with the "fill object flag" set and not set. Notice that the beginning and ending points are joined when the "fill object flag" is set. The code which generated these drawings is also given.

```
*
*.        draw figure 1
*
style     graphics;option,fill
rorigin   226,88
rdraw     -96,302;-96,222;32,222;32,286;-72,182;24,182
          -32,278;0,150
*
* end figure 1
*
* draw figure 2
*
style     graphics;clear
rorigin   226,88
rdraw     -96,152;-96,72;32,72;32,136;-72,32;24,32;-32,128
          0,0
*
* end figure 2
```

figure 1

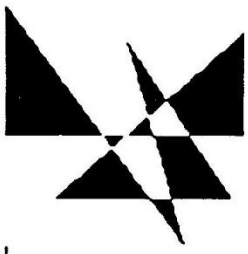
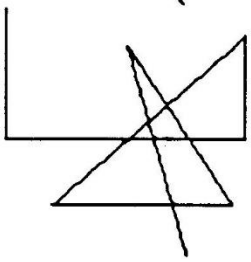


figure 2



9. Appendix F: Functional Summary of Commands and Modes

Screen Modes

ESC d	[1B 64 hex]	Inverse Screen Mode
ESC e	[1B 65 hex]	Write Screen Mode
ESC f	[1B 66 hex]	Erase Screen Mode
ESC g	[1B 67 hex]	Rewrite Screen Mode
ESC h	[1B 68 hex]	Exclusive OR Screen Mode

Word Coordinate Mode

ESC 4	[1B 34 hex]	Enter "word" coordinate mode
ESC 5	[1B 35 hex]	Return to standard coordinate mode

Centering

ESC q x ("Word")	[1B 71 78 hex]	Add "Word" to each x coordinate received
ESC q y ("Word")	[1B 71 79 hex]	Add "Word" to each y coordinate received

Windowing

ESC q z (C1 ,C2)	[1B 71 7A hex]	Specify clipping rectangle

Palette Support

ESC n ("Slot" "Color")	[1B 6E hex]	Load Palette Slot
ESC 6	[1B 36 hex]	Palette Clear
ESC l ("Slot")	[1B 6C hex]	Set Foreground color
ESC m ("Slot")	[1B 6D hex]	Set Background color

Graphics Styles

ESC i A ("Byte")	[1B 69 41 hex]	Set the current pattern to "Byte"
ESC i B ("Word")	[1B 69 42 hex]	Set thickness to "Word" pixels
ESC i C ("Byte")	[1B 69 43 hex]	Set the cap style for lines
ESC i D ("Byte")	[1B 69 44 hex]	Set dash style to "Byte"
ESC i E ("Byte")	[1B 69 45 hex]	Set the join style for Polylines
ESC i F ("Byte")	[1B 69 46 hex]	Set or clear filled object flag
ESC i G (B, W1..W4)	[1B 69 47 hex]	Load user defined pattern
ESC i H (B1, B2, W1..Wn)	[1B 69 48 hex]	Load user defined dash style
ESC 7	[1B 37 hex]	Return all graphic styles to defaults

Graphics Modes

ESC q A (W1..W4)	[1B 71 41 hex]	Elliptical Arc mode
ESC q B (W1..W3)	[1B 71 42 hex]	Circular Arc mode
ESC q C ("Word")	[1B 71 43 hex]	Circle Mode
ESC q D (W1, W2)	[1B 71 44 hex]	Ellipses
ESC q K (C1, C2, W)	[1B 71 4B hex]	Box mode

Stretch Commands

ESC q L (W1..W4)	[1B 71 4C hex]	Stretch Endline
ESC q M (W1..W6)	[1B 71 4D hex]	Stretch Midline
ESC q N ("Word")	[1B 71 4E hex]	Stretch Circle
ESC q O (W1, W2)	[1B 71 4F hex]	Stretch Ellipse in X direction
ESC q P (W1, W2)	[1B 71 50 hex]	Stretch Ellipse in Y direction
ESC q Q (W1..W5)	[1B 71 51 hex]	Stretch Box Corner
ESC q R (W1..W5)	[1B 71 52 hex]	Stretch Thick
ESC q S (W1..W4)	[1B 71 53 hex]	Stretch Fill
ESC q Z	[1B 71 5A hex]	Stop Stretch

Text Styles

ESC i a ("Word")	[1B 69 61 hex]	Logical OR of "Word" with old style bits
ESC i b ("Word")	[1B 69 62 hex]	Exclusive OR of "Word" with old style bits
ESC i c ("Word")	[1B 69 63 hex]	Set style to "Word"
ESC i d ("Word")	[1B 69 64 hex]	Logical AND the complement of "Word" with old style bits

ESC i e	[1B 69 65 hex]	Clear bit 0 (text style)
ESC i f	[1B 69 66 hex]	Set bit 0
ESC i g	[1B 69 67 hex]	Clear bit 1
ESC i h	[1B 69 68 hex]	Set bit 1
ESC i i	[1B 69 69 hex]	Clear bit 2
ESC i j	[1B 69 6A hex]	Set bit 2
ESC i k	[1B 69 6B hex]	Clear bit 3
ESC i l	[1B 69 6C hex]	Set bit 3
ESC i m	[1B 69 6D hex]	Clear bit 4
ESC in	[1B 69 6E hex]	Set bit 4
ESC i o	[1B 69 6F hex]	Clear bit 5
ESC i p	[1B 69 70 hex]	Set bit 5
ESC i q	[1B 69 71 hex]	Clear bit 6
ESC i r	[1B 69 72 hex]	Set bit 6
ESC i s	[1B 69 73 hex]	Clear bit 7
ESC i t	[1B 69 74 hex]	Set bit 7
ESC i u	[1B 69 75 hex]	Clear bit 8
ESC i v	[1B 69 76 hex]	Set bit 8
ESC i w	[1B 69 77 hex]	Clear bit 9
ESC i x	[1B 69 78 hex]	Set bit 9
ESC i y	[1B 69 79 hex]	Clear bit 10
ESC i z	[1B 69 7A hex]	Set bit 10

Filename Buffer

ESC q r	[1B 71 72 hex]	Clear Filename
ESC q s (B1..Bn)	[1B 71 73 hex]	Load filename
ESC q i ("Word")	[1B 71 69 hex]	Filename sumchk
ESC x a	[1B 78 61 hex]	Run Filename
ESC /	[1B 2F]	Successful local activity
ESC ?	[1B 3F]	Failed local activity

Trap

ESC q p (W1, B1)	[1B 71 70 hex]	Trap save
ESC q q	[1B 71 71 hex]	Trap end
ESC q n (W1..W3)	[1B 71 6E hex]	Trap display
ESC q o ("Word")	[1B 71 6F hex]	Trap delete
ESC q j (W1.. W5)	[1B 71 6A hex]	Trap drag
ESC q h	[1B 71 68 hex]	Stop drag
ESC q l ("Word")	[1B 71 6C hex]	Trap write
ESC q m ("Word")	[1B 71 6D hex]	Trap read
ESC q k	[1B 71 6B hex]	Trap test

Local Text Editing

ESC x A	[1B 78 41 hex]	Turn on Text Cursor
ESC x B	[1B 78 42 hex]	Turn off Text Cursor
ESC x C ("Word")	[1B 78 43 hex]	Set Right Margin
ESC x D ("Char")	[1B 78 44 hex]	Scroll Deletion
ESC x E ("Char")	[1B 78 45 hex]	Scroll Insertion
ESC x F (C1..C3)	[1B 78 46 hex]	Scroll
ESC x G ("Char")	[1B 78 47 hex]	Set Cursor Style
ESC x H	[1B 78 48 hex]	Text Cursor should Flash
ESC x I	[1B 78 49 hex]	Text Cursor should not Flash
ESC x K	[1B 78 4B hex]	Turns on text editing keymapping
ESC x M	[1B 78 4D hex]	Turns off text editing keymapping

Image

ESC j a (W1..W5)	[1B 6A 61 hex]	Save Image
ESC j b (W1..W9)	[1B 6A 62 hex]	Get Image
ESC j c ("Word")	[1B 6A 63 hex]	Image Information
ESC j d (W1..W8)	[1B 6A 64 hex]	Copy Image
ESC j e ("Word")	[1B 6A 65 hex]	Delete Image
ESC j f	[1B 6A 66 hex]	Echo for Image
ESC j 1	[1B 6A 31 hex]	Save Entire Screen (1 as in "one")
ESC j 2	[1B 6A 32 hex]	Restore Entire Screen

Output

ESC v A	[1B 76 41 hex]	Print Screen Graphically
ESC v B	[1B 76 42 hex]	Send entire screen to text output file
ESC v C(W1..W4)	[1B 76 43 hex]	Send partial screen to text output file
ESC x b	[1B 78 62 hex]	Inhibit Output
ESC x c	[1B 78 63 hex]	Allow Output

Input

SOH	[01 hex]	Abort Data in Input Stream
ESC x d	[1B 78 64 hex]	Inhibit Input
ESC x e	[1B 78 65 hex]	Allow Input

Other Downline Extensions

ESC q t	[1B 71 74 hex]	Alert Terminal
ESC q u	[1B 71 75 hex]	External data mode
ESC q v	[1B 71 76 hex]	Report Terminal Characteristics
ESC q w	[1B 71 77 hex]	Extension select
SOH (1)	[01 hex]	Abort Data in input stream
ESC 8 ("Word")	[1B 38 hex]	Delay "Word" milliseconds
ESC s	[1B 73 hex]	Load Character by Slot Mode
ESC v A	[1B 76 41 hex]	Print Screen (graphically)
ESC v B	[1B 76 42 hex]	Output Entire screen to text output file
ESC v C(W1..W2)	[1B 76 43 hex]	Output Partial screen to text output file

Upline Extensions

ESC 1E(hex) (B,W1,W2)	[1B 1E hex]	Pixel Resolution Pointing
ESC 1E (h) (60 (h) W1,W2)	[1B 1E 60 hex]	Report Terminal Characteristics
ESC Y 54(hex)	[1B 59 54 hex]	Operating System Echo

Kermit Server Mode

ESC x x ("Word")	[1B 78 78 hex]	Set Upline transmission rate
ESC x z	[1B 78 7A hex]	Enter Kermit Server Mode
ESC x y	[1B 78 79 hex]	Exit Kermit Server Mode

10. Appendix G: Alphabetical Summary of Commands and Modes

SOH	[01 hex]	Abort Data in Input Stream
ESC 1E(hex) (B, W1, W2)	[1B 1E hex]	Pixel Resolution Pointing
ESC 1E (h) (60 (h),W1,W2)	[1B 1E 60 hex]	Report Terminal Characteristics
ESC 4	[1B 34 hex]	Enter "word" coordinate mode
ESC 5	[1B 35 hex]	Return to standard coordinate mode
ESC 6	[1B 36 hex]	Palette Clear
ESC 7	[1B 37 hex]	Return all graphic styles to defaults
ESC 8 ("Word")	[1B 38 hex]	Delay "Word" milliseconds
ESC /	[1B 2F]	Successful local activity
ESC ?	[1B 3F]	Failed local activity
ESC Y 54(hex)	[1B 59 54 hex]	Operating System Echo
ESC d	[1B 64 hex]	Inverse Screen Mode
ESC e	[1B 65 hex]	Write Screen Mode
ESC f	[1B 66 hex]	Erase Screen Mode
ESC g	[1B 67 hex]	Rewrite Screen Mode
ESC h	[1B 68 hex]	Exclusive OR Screen Mode
ESC i a ("Word")	[1B 69 61 hex]	Logical OR of "Word" with old style bits
ESC i b ("Word")	[1B 69 62 hex]	Exclusive OR of "Word" with old style bits
ESC i c ("Word")	[1B 69 63 hex]	Set style to "Word"
ESC i d ("Word")	[1B 69 64 hex]	Logical AND the complement of "Word" with old style bits
ESC i e	[1B 69 65 hex]	Clear bit 0 (text style)
ESC i f	[1B 69 66 hex]	Set bit 0
ESC i g	[1B 69 67 hex]	Clear bit 1
ESC i h	[1B 69 68 hex]	Set bit 1
ESC i i	[1B 69 69 hex]	Clear bit 2
ESC i j	[1B 69 6A hex]	Set bit 2
ESC i k	[1B 69 6B hex]	Clear bit 3
ESC i l	[1B 69 6C hex]	Set bit 3
ESC i m	[1B 69 6D hex]	Clear bit 4
ESC i n	[1B 69 6E hex]	Set bit 4
ESC i o	[1B 69 6F hex]	Clear bit 5
ESC i p	[1B 69 70 hex]	Set bit 5
ESC i q	[1B 69 71 hex]	Clear bit 6
ESC i r	[1B 69 72 hex]	Set bit 6
ESC i s	[1B 69 73 hex]	Clear bit 7

ESC i t	[1B 69 74 hex]	Set bit 7
ESC i u	[1B 69 75 hex]	Clear bit 8
ESC i v	[1B 69 76 hex]	Set bit 8
ESC i w	[1B 69 77 hex]	Clear bit 9
ESC i x	[1B 69 78 hex]	Set bit 9
ESC i y	[1B 69 79 hex]	Clear bit 10
ESC i z	[1B 69 7A hex]	Set bit 10
ESC i A ("Byte")	[1B 69 41 hex]	Set the current pattern to "Byte"
ESC i B ("Word")	[1B 69 42 hex]	Set thickness to "Word" pixels
ESC i C ("Byte")	[1B 69 43 hex]	Set the cap style for lines
ESC i D ("Byte")	[1B 69 44 hex]	Set dash style to "Byte"
ESC i E ("Byte")	[1B 69 45 hex]	Set the join style for Polylines
ESC i F ("Byte")	[1B 69 46 hex]	Set or clear filled object flag
ESC i G (B, W1..W4)	[1B 69 47 hex]	Load user defined pattern
ESC i H (B1, B2, W1..Wn)	[1B 69 48 hex]	Load user defined dash style
ESC j a (W1..W5)	[1B 6A 61 hex]	Save Image
ESC j b (W1..W9)	[1B 6A 62 hex]	Get Image
ESC j c ("Word")	[1B 6A 63 hex]	Image Information
ESC j d (W1..W8)	[1B 6A 64 hex]	Copy Image
ESC j e ("Word")	[1B 6A 65 hex]	Delete Image
ESC j f	[1B 6A 66 hex]	Echo for Image
ESC j 1	[1B 6A 31 hex]	Save Entire Screen (1 as in "one")
ESC j 2	[1B 6A 32 hex]	Restore Entire Screen
ESC l ("Slot")	[1B 6C hex]	Set Foreground color (l as in the letter)
ESC m ("Slot")	[1B 6D hex]	Set Background color
ESC n ("Slot" "Color")	[1B 6E hex]	Load Palette Slot

ESC q h	[1B 71 68 hex]	Stop drag
ESC q i ("Word")	[1B 71 69 hex]	Filename sumchk
ESC q j ("Word1" .. "Word5")	[1B 71 6A hex]	Trap drag
ESC q k	[1B 71 6B hex]	Trap test
ESC q l ("Word")	[1B 71 6C hex]	Trap write
ESC q m ("Word")	[1B 71 6D hex]	Trap read
ESC q n ("Word1" .. "Word3")	[1B 71 6E hex]	Trap display
ESC q o ("Word")	[1B 71 6F hex]	Trap delete
ESC q p ("Word1" "Byte1")	[1B 71 70 hex]	Trap save
ESC q q	[1B 71 71 hex]	Trap end
ESC q r	[1B 71 72 hex]	Clear Filename
ESC q s ("Byte1" .. "Byten")	[1B 71 73 hex]	Load filename
ESC q t	[1B 71 74 hex]	Alert Terminal
ESC q u	[1B 71 75 hex]	External data mode
ESC q v	[1B 71 76 hex]	Report Terminal Characteristics
ESC q w	[1B 71 77 hex]	Extension select
ESC q x ("Word")	[1B 71 78 hex]	Add "Word" to each x coordinate received
ESC q y ("Word")	[1B 71 79 hex]	Add "Word" to each y coordinate received
ESC q z (C1, C2)	[1B 71 7A hex]	Specify clipping rectangle
ESC q A (W1, W4)	[1B 71 41 hex]	Elliptical Arc mode
ESC q B (W1, W3)	[1B 71 42 hex]	Circular Arc mode
ESC q C ("Word")	[1B 71 43 hex]	Circle Mode
ESC q D (W1, W2)	[1B 71 44 hex]	Ellipses
ESC q K (C1, C2, W)	[1B 71 4B hex]	Box mode
ESC q L (W..W4)	[1B 71 4C hex]	Stretch Endline
ESC q M (W1..W6)	[1B 71 4D hex]	Stretch Midline
ESC q N ("Word")	[1B 71 4E hex]	Stretch Circle
ESC q O (W1, W2)	[1B 71 4F hex]	Stretch Ellipse in X direction
ESC q P (W1, W2)	[1B 71 50 hex]	Stretch Ellipse in Y direction
ESC q Q (W1..W5)	[1B 71 51 hex]	Stretch Box Corner
ESC q R (W1..W5)	[1B 71 52 hex]	Stretch Thick
ESC q S (W1..W4)	[1B 71 53 hex]	Stretch Fill
ESC q Z	[1B 71 5A hex]	Stop Stretch

ESC s	[1B 73 hex]	Load Character by Slot Mode
ESC v A	[1B 76 41 hex]	Print Screen (graphically)
ESC v B	[1B 76 42 hex]	Output entire screen to text output file
ESC v C(W1..W4)	[1B 76 43 hex]	Output partial screen to text output file
ESC x a	[1B 78 61 hex]	Run Filename
ESC x b	[1B 78 62 hex]	Inhibit Output
ESC x c	[1B 78 63 hex]	Allow Output
ESC x d	[1B 78 64 hex]	Inhibit Input
ESC x e	[1B 78 65 hex]	Allow Input
ESC x x ("Word")	[1B 78 78 hex]	Set Upline transmission rate
ESC x y	[1B 78 79 hex]	Exit Kermit Server Mode
ESC x z	[1B 78 7A hex]	Enter Kermit Server Mode
ESC x A	[1B 78 41 hex]	Turn on Text Cursor
ESC x B	[1B 78 42 hex]	Turn off Text Cursor
ESC x C ("Word")	[1B 78 43 hex]	Set Right Margin
ESC x D ("Char")	[1B 78 44 hex]	Scroll Deletion
ESC x E ("Char")	[1B 78 45 hex]	Scroll Insertion
ESC x F ("Char1" .. "Char3")	[1B 78 46 hex]	Scroll
ESC x G ("Char")	[1B 78 47 hex]	Set Cursor Style
ESC x H	[1B 78 48 hex]	Text Cursor Should Flash
ESC x I	[1B 78 49 hex]	Text Cursor Should not Flash
ESC x K	[1B 78 4B hex]	Turns on text editing keymapping
ESC x M	[1B 78 4D hex]	Turns off text editing keymapping
ESC Y 54 (hex)	[1B 59 54 hex]	Echo identifies operating system

11. Appendix H: Level 0 Protocol (Summary)

For more information on Level 0 protocol, see the documentor file "s0ascers" which is available for on-line inspection on PLATO and NovaNET systems.

11.1. Downline Data Formats for Level 0 Protocol

11.1.1. Control Codes

Control Character		Description
BS	[08 hex]	Backspace (Move one character width back.)
HT	[09 hex]	Tab. (Move one character width forward.)
LF	[0A hex]	Linefeed. (Move one character height down.)
VT	[0B hex]	Vertical tab. (Move one character height up.)
FF	[0C hex]	Form feed. (Set x,y position to top of page.)
CR	[0D hex]	Carriage return. (Set x,y position to margin on next line.)
EM	[19 hex]	Use block write/erase mode. (mode 4)
ESC	[1B hex]	Terminal should check if next character is valid escape sequence.
FS	[1C hex]	Use point-plot mode. (mode 0)
GS	[1D hex]	Use draw line mode. (mode 1)
US	[1F hex]	Use alpha mode. (mode 3)

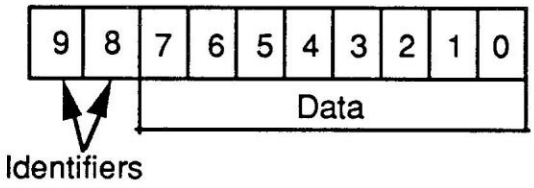
11.1.2. Escape Sequences

Escape Sequence		Description
ESC SOH	[01 hex]	Use Tektronix emulation mode in CDC IST III graphics and in Viking 721-30 residents.
ESC STX	[1B 02 hex]	Use NovaNET (PLATO) operation.
ESC ETX	[1B 03 hex]	Use TTY operation.
ESC FF	[1B 0C hex]	Clear screen without resetting x,y.
ESC DC1	[1B 11 hex]	Use inverse video screen mode.
ESC DC2	[1B 12 hex]	Use mode write.
ESC DC3	[1B 13 hex]	Use mode erase.
ESC DC4	[1B 14 hex]	Use mode rewrite.
ESC ESC	[1B 1B hex]	Ignore first ESC. For example, ESC ESC SOH should be treated as ESC SOH.
ESC 2	[1B 32 hex]	Load Coordinate command.
ESC @	[1B 40 hex]	Superscript
ESC A	[1B 41 hex]	Subscript
ESC B	[1B 42 hex]	Use character set M0.
ESC C	[1B 43 hes]	Use character set M1.
ESC D	[1B 44 hex]	Use character set M2.
ESC E	[1B 45 hes]	Use character set M3.
ESC F	[1B 46 hex]	Use character set M4.
ESC G	[1B 47 hes]	Use character set M5.
ESC H	[1B 48 hex]	Use character set M6.
ESC I	[1B 49 hes]	Use character set M7.
ESC J	[1B 4A hex]	Select horizontal.
ESC K	[1B 4B hex]	Select vertical.
ESC L	[1B 4C hex]	Select forward.
ESC M	[1B 4D hex]	Select reverse.
ESC N	[1B 4E hex]	Use size 0.
ESC O	[1B 4F hex]	Use size 2.
ESC P	[1B 50 hex]	Load memory (mode 2 data with character convert).
ESC Q	[1B 51 hex]	SSF command.
ESC R	[1B 52 hex]	External data.
ESC S	[1B 53 hex]	Load memory (mode 2 data).
ESC T	[1B 54 hex]	Use mode 5.
ESC U	[1B 55 hex]	Use mode 6.
ESC V	[1B 56 hex]	Use mode 7.
ESC W	[1B 57 hex]	Load Memory Address command.
ESC Y	[1B 59 hex]	Load Echo command.
ESC Z	[1B 5A hex]	Set margin.
ESC a	[1B 61 hex]	Set foreground color.
ESC b	[1B 62 hex]	Set background color.
ESC c	[1B 63 hex]	Paint

11.2. Upline Data Formats for Level 0 Protocol

The resident program sends different types of data to the central computer. The data sent upline to the central computer is referred to as "keys". This data can include a key that the user presses, an echo response, touch input, external device data, and unsolicited status.

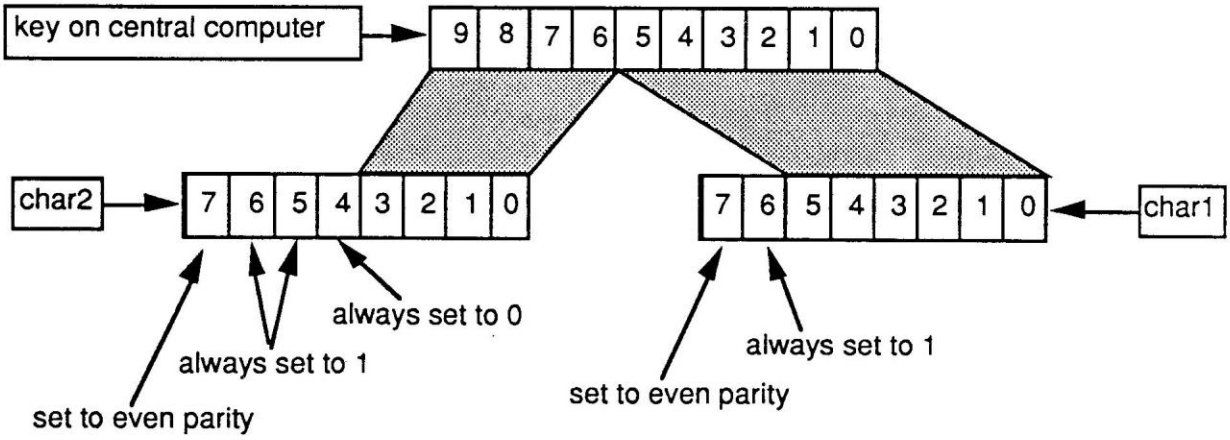
On the central computer, keys are 10 bits long. The bottom 8 bits are the data and the top two bits are an identifier.



The values for the identifier (bits 8 and 9) are:

bit 9	bit 8	Description
0	0	Echo response
0	1	Touch panel data
1	0	External data
1	1	Unsolicited status

Three characters are used to send the 10 bits to the central computer in an escape sequence of ESC char1 char2. The format for the characters "char1" and "char2" are:

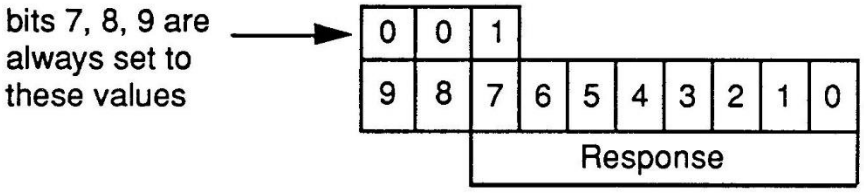


11.2.1. Keyboard data

Keyboard data gets sent to the central computer as individual bytes.

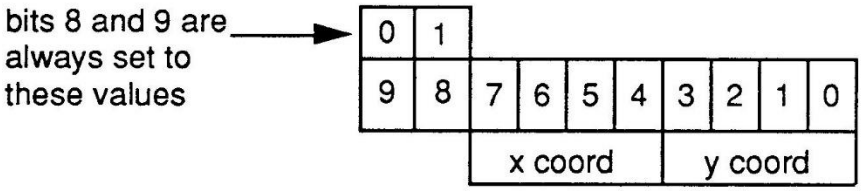
11.2.2. Echo responses

The ten bit key for echo response is formatted as:



11.2.3. Touch Panel Data

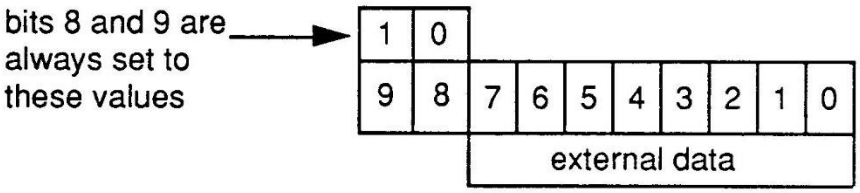
The ten bit key for touch panel data is formatted as:



The lower left hand corner of the screen is at touch location 0,0 and the upper right hand corner of the screen is at touch location 15,15. Other input devices with greater resolution must be scaled down to fit in the 0 to 15 range.

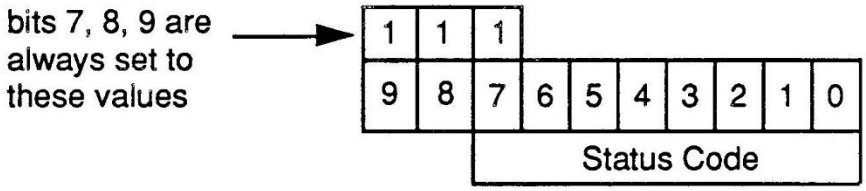
11.2.4. External Data

The ten bit key for external data is formatted as:



11.2.5. Unsolicited Status

The protocol program can send an unsolicited status word to the central computer when certain events occur at the terminal. The ten bit key for unsolicited status is formatted as:



The values of the status code are:

Status Code	Description
00	Unassigned
01	Unassigned
02	Terminal Master Reset
03	Unassigned
04	Unassigned
05	Reserved
06 to 3F	Unassigned
40 to 7E	Reserved
7F	Central System Backout Request

12. Appendix I: ASCII Code Chart

From: Stifle, Jack A Packet Data Network for the Delivery of PLATO Service, 1987

b7	0	0	0	0	1	1	1	1
b6	0	0	1	1	0	0	1	1
b5	0	1	0	1	0	1	0	1
	0	1	2	3	4	5	6	7
b4 b3 b2 b1								
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	A	LF	SUB	*	:	J	Z	j	z
1 0 1 1	B	VT	ESC	+	;	K	[	k	{
1 1 0 0	C	FF	FS	,	<	L	\	l	|
1 1 0 1	D	CR	GS	-	=	M	]	m	}
1 1 1 0	E	SO	RS	.	>	N	^	n	~
1 1 1 1	F	SI	US	/	?	O	_	o	DEL

Index