Term0
Reference Manual

HEWLETT
PACKARD

Edition 1 November 1985

82865-90002

Notice

Portable Computer Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Printing History **Edition 1** November 1985 Mfg. No. 82865-90002

Contents

1 Introduction

Contents

1 Introduction

This document describes the feature set and programming method for HP level 0 terminals (Term0). The basic features defined here should be found in all HP Personal Computers. Application programs that use only these features can expect a high degree of portability across systems that support Term0. Term0 has been defined to allow common programmatic access to built-in or external terminals. It is a subset of the HP 2622 level 1 terminal.

The feature set supports both HP and ANSI escape sequences, as well as the HP ITF (Integrated Terminal Family) keyboard. (However, only HP escape sequences are supported on Windows/9000.) Chapters 2 through 9 describe these features of Term0. An application should choose HP or ANSI mode, put the terminal in that mode, and use only that type of escape sequence. (If you use Windows/9000, you can use only HP escape sequences.)

- In HP mode only HP escape sequences are recognized, the keys generate HP escape sequences, and display enhancements are field-oriented. A field-oriented display enhancement affects characters starting at the cursor position and continuing to the next display enhancement or to the end of the line. Font changes (SO/SI) are considered display enhancements.

- In ANSI mode, ANSI escape sequences and some HP parameterized escape sequences are recognized. The keys generate ANSI escape sequences, and display enhancements are stream-oriented. A stream-oriented display enhancement affects all subsequent characters, regardless of position, until the enhancement is turned off. The HP parameterized escape sequences recognized in ANSI mode are: ␛&f, ␛&j, ␛&k, ␛&s, ␛*d, and ␛*s. If an implementation adds parameterized escape sequences, they may be recognized in ANSI mode.

In addition to the function set above, various HP computers may provide other functions controlled by escape sequences. Chapters 10, 11, and 12 describe escape sequences specific to The Portable, the Integral Personal Computer, and Windows/9000. *These product-specific escape sequences are not a part of the Term0 definition.*

For your reference, this document includes appendices that describe character sets, a comparison of terminals, and a comparison of The Portable, the Integral PC, and Windows 9000.

2 Term0 Features

Contents

Chapter 2

2 Term0 Features

Software Requirements

A level 0 terminal should implement several software features to ensure its portability to other systems.

More specifically, a level 0 terminal:

- Transmits information to the system using character mode only; block line, block page, and format modes are not supported. Information is received from the host in character mode only.
- Supports complete screen editing functions: insert/delete character, insert/delete line, and clear line/display.
- Supports cursor sensing: absolute and relative modes.
- Supports cursor addressing and cursor control: absolute, screen relative, and cursor relative modes.
- Provides at least two pages of display memory (one page equals 24 lines by 80 characters). The actual size of the viewing area is dependent upon the size of the display.
- Supports one set of softkeys.
- Supports fixed tabs, with tabs every eight spaces recommended. Terminals can implement settable tabs and margins if desired.
- Provides display functions mode and caps mode.
- Supports ENQ/ACK or no software handshaking.
- Supports both transmit and receive pacing mechanisms; for example, XON/XOFF.
- Allows both the terminal status and the terminal ID to be acquired using escape sequences.

- Supports display enhancements. In HP mode, the display enhancements are field oriented. In ANSI mode, the display enhancements are stream oriented.
- Provides a user-definable (RETURN) key.
- Supports eight-bit character sets. The default character set is Roman8 in which the lower 128 characters (characters 0 through 127) are the ASCII character set and the upper 128 characters (characters 128 through 255) contain European characters.

Display Requirements

Although a single-line display is acceptable (barely), you should use a multiline display. An optional integrated printer or printer interface may be provided. The printer is treated as a peripheral by the terminal.

Communications Requirements

Use combinations of RS-232, 300-baud modem, and 1200-baud modem. (Both U.S. and European.)

3 Escape Sequence Syntax

Contents

Chapter 3

3 Escape Sequence Syntax

Character Sets

Term0 supports eight-bit character sets. The default character set is the Roman8 set; the lower 128 characters are the ASCII character set and the upper 128 characters contain European characters. Escape sequences are defined using the ASCII character set. ASCII refers to the seven-bit coded character set defined by ANSI X3.4 - 1977 American National Standard Code for Information Interchange. (The lower 128 characters are shown in the ASCII Character Set table in appendix A. The upper 128 characters of the Roman8 character set are shown in the Roman Extension Character Set table.)

Fill Characters

The following control codes are often used for fill or communications and will be ignored if received within an escape sequence. The numbers to the right of each control code indicate the position of the character in the ASCII Character Set table. (The control codes are shown in decimal.)

Fill Characters

Character	Code	Character	Code
NUL	0	DC1	17
SOH	1	DC2	18
STX	2	DC3	19
ETX	3	DC4	20
EOT	4	NAK	21
ENQ	5	SYN	22
ACK	6	ETB	23
DLE	15	DEL	127

Also, space (32), carriage return, CR (13), and line feed, LF (10) are ignored after the third character in a parameterized escape sequence. The exception is an alphanumeric parameter: Only DEL (127) and NULL (0) are ignored in an alphanumeric parameter.

Two-Character HP Escape Sequences

An HP two-character escape sequence has the form of ␛X where X is a character from the ASCII Character Set table within the range 48 through 126. If the character is between 33 and 47, it is a parameterized escape sequence and is interpreted as described in chapter 3.

If a character between 0 and 32 is received (with the exception of the fill characters already discussed), the sequence is illegal; both characters are ignored and the escape sequence is terminated. If a two-character escape sequence has valid syntax (that is, the second character is between 48 and 127), but the escape sequence is not recognized by the terminal, the terminal treats it as an illegal sequence, ignoring both the escape character and the second character, and terminates the escape sequence.

Parameterized Escape Sequences

Parameterized escape sequences have the form:

ᴱc x y NNNN... z1 NNNN... z2 NNNN... z3 ... Zn

where:

- *x* is a *parameterized character*. This is a character from the ASCII Character Set within the range 33 through 47. This character indicates that the escape sequence is parameterized.
- *y* is the *group character*. This is a character within the range 96 through 126. This character specifies the group or type of control performed.
- *NNNN* is the *value field*. This is a string of characters, each within the range 33 through 63. This string specifies a value. This string is optional and if not present, a value of zero is assumed. The numbers are specified in decimal form. Specified values that are combinations of characters other than numbers or spaces will cause the value field and associated parameter to be ignored.
- *z* is the *parameter character*. It can be within the range 96 through 126. This character specifies how to interpret the previous value field. If the value field is not present, this character is optional.
- *Zn* is the *terminating character*. This character is within the range 64 through 94 (an uppercase letter). It terminates the escape sequence. If there are no parameters (and therefore no values) specified, then the group or type specifying character is capitalized and becomes the terminator.

Alphanumeric Parameters

To specify alphanumeric parameters, include a length parameter in the parameterized escape sequence. The length parameter specifies the number of characters to be used as the alphanumeric parameter. Then the alphanumeric parameter follows the terminating character for the specified number of characters. Blanks are *not* ignored in the alphanumeric parameter. Control codes may be included in the alphanumeric string, except for DEL (127) and NULL (0).

Illegal Syntax

If an illegal character is received within a parameterized escape sequence, the processing of that escape sequence terminates. The illegal character is handled independent of the previous characters and is processed immediately as text. Spaces are considered illegal characters if received before the group specifier. When found after the group specifier, however, they are ignored. The fill control characters are always ignored.

Unrecognized Parameterized Escape Sequences

Any parameterized escape sequence which is syntactically correct, but not recognized, is ignored. The escape sequence mode is active until a terminating character is received. All characters received after the escape character are ignored until the terminating character is received. Thus, an unrecognized escape sequence will be ignored and will not be displayed on the screen.

4 Control Codes

Contents

4 Control Codes

The control codes in the following table are recognized by Term0 as instructions.

Implemented Control Codes

Control code	Name	Description
(CTRL) @	NULL	The null character is ignored and not printed.
(CTRL) G, (CTRL) g	Bell	When the bell control code is received, the terminal causes the speaker to generate a tone.
(CTRL) H, (CTRL) h	Back space	The terminal performs a backspace from the current cursor position.
(CTRL) I, (CTRL) i	Horizontal tab	Tab causes the cursor to move to the next horizontal tab stop. If there are no more tab stops on the current row, the cursor will move to the left margin of the next row (that is, perform a carriage return-line feed).

Implemented Control Codes (Continued)

Control code	Name	Description
CTRL J, CTRL j	Linefeed	A line feed moves the cursor to the next row. The cursor remains in the same column. If there are no more rows on the screen, then a roll up is performed. If there are no more rows in the buffer, then the top row is discarded, all subsequent rows are shifted up one line, and a blank line is placed at the bottom of the screen buffer.
CTRL M, CTRL m	Carriage return	Carriage return moves the cursor to the first column of the current row.
CTRL N, CTRL n	Select alternate character set	In HP mode: change characters to the alternate font, starting at the cursor position and continuing to the next display enhancement or to the end of the line. In ANSI mode: enter characters in the alternate font until a Shift In (SI) character is received. SO and SI are required only if the terminal has an alternate font.
CTRL O, CTRL o	Select base character set	This selects the base character set, which remains active until an SO character is received.

Implemented Control Codes (Continued)

Control code	Name	Description
CTRL Q, CTRL q	Resume output	A control-Q is used to resume output that has been suspended by the stop output character, control-S. While the output is not suspended, resume output characters are ignored.
CTRL S, CTRL s	Stop output	A control-S is used to temporarily suspend output. A control-Q resumes output. It is useful with terminal displays to prevent output from disappearing before it can be read. While output is suspended, stop output characters are ignored.
CTRL [	Escape	The escape character starts an escape sequence.

5 HP Two-Character Escape Sequences

Contents

5 HP Two-Character Escape Sequences

Two-Character Escape Sequences

Sequence	Name	Description
EcA	Cursor Up	Moves the cursor up one row. If the cursor is at the top of the screen, the data will roll down. If the cursor is at the top of the buffer, the escape sequence will be ignored.
EcB	Cursor Down	Moves the cursor down one row. If the cursor is at the bottom of the screen, the data will roll up. If the cursor is at the bottom of the buffer, the escape sequence will be ignored.
EcC	Cursor Right	Moves the cursor right one column. If the cursor is at the right of the screen, the data will roll left. If the cursor is at the right of the buffer, the escape sequence will be ignored.
EcD	Cursor Left	Moves the cursor left one column. If the cursor is at the left of the screen the data will roll right. If the cursor is at the left of the buffer, the escape sequence will be ignored.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛F	Home Down	Moves the cursor to the first column of the row just below the last row of text in the buffer. If the last row is not on the screen, the information will roll up until there is one blank row below the last row of text.
␛H, ␛h	Home Up	Moves the cursor to the first column of the top row in the buffer. If the top row of text is not on the screen, the information will roll down until the top row of text is on the top row of the screen.
␛I	Tab	␛I is the same as the control code (CTRL) I.
␛J	Clear Display	Clears the terminal buffer from the current cursor position to the end of the buffer. The corresponding information is removed from the screen.
␛K	Clear Line	Clears all characters from the cursor position to the right end of the row that the cursor is on.
␛L	Insert Line	Inserts a blank line above the row that the cursor is on. The cursor is moved to the first column of the inserted row. A row will be lost off the bottom or the top of the buffer.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛M	Delete Line	Deletes the row that the cursor is on. All rows below that row are moved up one row. A new row is added to the bottom of the buffer. The cursor moves to the first column of the row below the deleted row.
␛P	Delete Character	Deletes the character at the current cursor position. All characters to the right of the cursor move to the left one position.
␛Q	Insert Character	Enables insert character mode. When insert mode is active, the new characters will be placed before the character that the cursor was under. The old characters to the right of the cursor are pushed to the right. Any characters pushed past the right edge of the buffer are lost.
␛R	Leave Insert Mode	Disables insert character mode.
␛S	Roll Up	Moves the information on the screen up one row. If the bottom of the screen is at the bottom of the buffer, the escape sequence is ignored.
␛T	Roll Down	Moves the information on the screen down one row. If the top of the screen is at the top of the buffer, the escape sequence is ignored.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛U	Next Page	Rolls the information in the display up until a new page of information is visible. A *page* of information is the number of rows on the screen. The cursor is moved to the first column on the first row.
␛V	Previous Page	Causes the information in the display to roll down until a new page of information is visible. A *page* of information is defined to be the number of rows on the screen. The cursor is moved to the first column on the first row.
␛Y	Display Functions On	Turns the display functions mode on. While in the display functions mode, all control codes and escape sequences are printed and are not executed. The exceptions are carriage return, which is displayed and then executed with a line feed, and ␛Z, which is displayed and then executed. With Windows/9000, a control character is displayed only if the current font supports the display of special characters. For example, some fonts have a display character to represent a (CTRL)(G), while others do not.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛Z	Display Functions Off	Turns the display functions mode off. The default mode is display function mode off.
␛^	Primary Terminal Status	Returns ten bytes that indicate the primary terminal status: two bytes of preceding characters, seven bytes of information about the status of the terminal and one terminating character. The first two bytes are ␛\ followed by bytes 0 thru 6 of information and then terminated by a CR. Byte 0 indicates the size of display memory in the terminal. Byte 1 gives the status of configuration straps A-D and byte 2 gives the status of configuration straps E-H. Byte 3 gives the state of the latching keys. Byte 4 gives the status of the transfer pending flags. Byte 5 indicates the error flags and byte 6 gives the state of the device transfer pending flags. (Refer to chapter 9 for more information about the definitions of individual bits.) Straps A and C are the only required straps.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛ '	Sense Cursor Position (Relative)	Sends a screen-relative cursor addressing sequence. The cursor position is calculated relative to the top left corner of the screen. The escape sequence that is returned is of the form ␛&a *column number* x *row number* Y. Both *column number* and *row number* are represented by three decimal digits.
␛a	Sense Cursor Position (Absolute)	Returns an absolute cursor addressing sequence. The cursor position is calculated as absolute in the buffer. The escape sequence that is returned is of the form ␛&a *column* number c *row number* R. Both *column number* and *row number* are three decimal digits.
␛d	Enter Line	This causes the line containing the cursor to be entered. The characters on the line from the cursor position and to the right are returned. If you wish to read the entire line, first place the cursor in column 0.
␛h	Home Up	␛h is the same as ␛H.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛i	Back Tab	Moves the cursor back to the previous tab stop. There is an unclearable tab stop at the beginning of each row. If the cursor is at the beginning of the row, then the cursor will wrap to the last tab stop of the previous row. If the cursor is originally at the top left of the screen, then the screen will roll down. If the cursor is at the top left corner of the buffer, the escape sequence will be ignored.
␛p	Default Application Softkey 1 Value	␛p is the default value for application softkey 1. If nothing else is assigned to the softkey, ␛p is returned.
␛q	Default Application Softkey 2 Value	␛q is the default value for application softkey 2. If nothing else is assigned to the softkey, ␛q is returned.
␛r	Default Application Softkey 3 Value	␛r is the default value for application softkey 3. If nothing else is assigned to the softkey, ␛r is returned.
␛s	Default Application Softkey 4 Value	␛s is the default value for application softkey 4. If nothing else is assigned to the softkey, ␛s is returned.
␛t	Default Application Softkey 5 Value	␛t is the default value for application softkey 5. If nothing else is assigned to the softkey, ␛t is returned.

Two-Character Escape Sequences (Continued)

Sequence	Name	Description
␛u	Default Application Softkey 6 Value	␛u is the default value for application softkey 6. If nothing else is assigned to the softkey, ␛u is returned.
␛v	Default Application Softkey 7 Value	␛v is the default value for application softkey 7. If nothing else is assigned to the softkey, ␛v is returned.
␛w	Default Application Softkey 8 Value	␛w is the default value for application softkey 8. If nothing else is assigned to the softkey, ␛w is returned.

6 HP Parameterized Escape Sequences

Contents

Chapter 6

6 HP Parameterized Escape Sequences

Cursor Addressing

Term0 supports three cursor addressing modes:

- Relative addressing.
- Absolute addressing.
- Cursor relative addressing.

In relative addressing, the position of the cursor is determined according to its location on the screen. In absolute addressing, the cursor position is determined according to its location in display memory. Using cursor relative addressing, parameters specify how the cursor should be moved relative to its current position on the screen.

Relative Addressing. The following escape sequences move the cursor to any character position in the screen. (The screen is typically 24 rows by 80 columns, but might be different.)

Ec&a *column number* x *row number* Y
Ec&a *row number* y *column number* X
Ec&a *column number* X
Ec&a *row number* Y

where:

- *column number* is a decimal number specifying the screen column to which you wish to move the cursor. Zero specifies the leftmost column. If *column number* is greater than the number of columns in the screen, the cursor will move to the rightmost column of the screen.

- *row number* is a decimal number specifying the screen row to which you wish to move the cursor. Zero specifies the top row of the screen. If *row number* is greater than the number of rows in the screen, the cursor will move to the bottom of the screen.

If you specify only a *column number,* the cursor remains in the current row. Similarly, if you specify only a *row number,* the cursor remains in the current column.

For example, the escape sequence

```
␛&a44x15Y
```

moves the cursor to column 44 and row 15 on the screen.

Absolute Addressing. The following escape sequences move the cursor to a position in the display buffer. The screen will be moved as necessary so that it always contains the cursor.

␛&a *column number* c *row number* R
␛&a *row number* r *column number* C
␛&a *column number* C
␛&a *row number* R

where:

- *column number* is a decimal number specifying the column coordinate (within the display buffer) of the character at which you want the cursor positioned. Zero specifies the leftmost column. If *column number* is greater then the number of columns in the buffer (that is, greater than 79), the cursor moves to the rightmost column of the buffer.
- *row number* is a decimal number specifying the row coordinate (within the display buffer) of the character at which you want the cursor positioned. Zero specifies the top row of the buffer. If *row number* is greater than the number of rows in the buffer (47 rows), the cursor will move to the bottom row of the buffer.

If you specify only a *column number,* the cursor remains in the current row. Similarly, if you specify only a *row number,* the cursor remains in the current column.

For example, the escape sequence

ᴱc&a4c3R

Moves the cursor to column 4 and row 3 in the display buffer.

Cursor Relative Addressing. The following escape sequences enable you to specify the cursor position relative to the current location of the cursor. The screen will be moved as necessary so that it always contains the cursor.

ᴱc&a± *column number* c± *row number* R
ᴱc&a± *row number* r ± *column number* C
ᴱc&a± *column number* C
ᴱc&a± *row number* R

where:

- *column number* is a decimal number specifying the relative column to which you wish to move the cursor. A positive number specifies how many columns to the right you wish to move the cursor; a negative number specifies how many columns to the left. If the resultant column would be outside the buffer (less than 0 or greater than 79), the cursor is moved to the edge of the buffer (column 0 or 79).
- *row number* is a decimal number specifying the relative row to which you wish to move the cursor. A positive number specifies how many rows down you wish to move the cursor; a negative number specifies how many rows upward. If the resultant row would be outside the buffer (less than 0 or greater than 47), the cursor is moved to the edge of the buffer (row 0 or 47).

If you specify only a *column number,* the cursor remains in the current row. Similarly, if you specify only a *row number,* the cursor remains in the current column.

For example, the escape sequence

```
␛&a+3c-4R
```

moves the cursor right 3 columns and up 4 rows.

Display Enchancements

Term0 supports four display enhancements:

- Inverse video.
- Underline.
- Half bright (or bold).
- Blinking.

All implementations of Term0 must have inverse video and underline. Since hardware constraints might prevent the implementation of half-bright and blinking, half-bright can be mapped onto inverse video and blinking can be mapped onto underline.

The display enhancement is field oriented. It applies to all characters from the cursor position to the right until either the start of a display enhancement or the end of the line is reached.

To turn all display enhancements off, use `␛&d@`.

To turn a combination of display enhancements on, use ᴱc&d *char* where *char* is a letter from the table below. For example, ᴱc&dE turns on underline and blinking. (An "x" under a letter indicates that the corresponding enhancement is turned on when that letter terminates the escape sequence.)

Display Enhancement Settings

	char															
	@	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
Half-Bright									X	X	X	X	X	X	X	X
Underline					X	X	X	X					X	X	X	X
Inverse Video			X	X			X	X			X	X			X	X
Blinking		X		X		X		X		X		X		X		X
Turn Off Enhancements	X															

Softkey Definition

The following escape sequence sets up the softkeys.

ᴱc&f *attribute* a *key* k *label length* d *string length* l *label string*

where:

- *attribute* is 0, 1 or 2. 0a indicates a normal softkey. The softkey string is treated just as if it was typed from the keyboard. 1a indicates a local softkey. Local softkeys are handled only in the terminal and are not sent to the system. 2a indicates a transmit softkey. Transmit softkeys are only sent to the system and are not echoed to the screen. Normal softkeys (0a) are the only type required for Term0.

- *key* is 1 thru 8 for the softkeys, 0 for the Return key or −1 for the Enter key.
- *label length* indicates the number of characters in the label that is used for the menu entry. It can be −1 thru 16, where −1 clears the contents of the string to a null string, and 0 leaves the string unchanged. If a number larger than 16 is given, that number of characters will be read for the label, but only 16 characters will be kept and used. The label will be read in and ignored if the key being defined is the (Return) or (Enter) key.
- *string length* indicates the number of characters in the definition string. It can be −1 thru 80, where −1 clears the contents of the string to a null string, and 0 leaves the string unchanged. If a number larger than 80 is given, that number of characters will be read for the string, but only 80 characters will be kept and used. Only eight characters are kept and used if the key being defined is the (Return) key or the (Enter) key.
- *label* is a string of characters of length *label length*. This string is used as the menu entry for the designated softkey. If it exists, it must follow the terminating character.
- *string* is a string of characters of length *string length*. This string is used as the definition of the softkey. If is exists, it must follow *label* if *label* exists. If *label* does not exist, *string* must follow the terminating character.

Example. Assign `LOG-ON` as the label and `HELLO USER.ACCOUNT` as the definition for softkey 5. The key is to have the normal attribute.

```
Ec&f0a5k6d18LLOG-ONHELLO USER.ACCOUNT
```

The parameters can be in any order, but the last one, excluding *label* and *string*, must be capitalized because it is the terminating character. If the *attribute* parameter is absent, 0a (normal softkey) is assumed. If the *key* parameter is absent, 1k (softkey 1) is assumed. If *label length* is absent, no characters will be read for *label* , and the the label for that softkey will remain unchanged. If not previously defined, the label will be blank. If *string length* is absent, no characters will be read for *string*, and the definition of that softkey will remain unchanged. If softkeys 1 thru 8 aren't previously defined, they will have the default definitions of of Ecp thru Ecw. The (Return) key has a default definition of CR, and the (Enter) key has a default definition of CR Ecd NL.

Menus

The following escape sequences control the menu.

Turn Off Menu

Ec&j@

This escape sequence turns off the menu. If the menu is already off, nothing happens.

Turn On Application Menu

Ec&jB

This escape sequence activates the application softkeys and turns on the application menu.

Caps Mode

Ec&k *n* P

0 disables the caps mode; 1 enables the caps mode. The absence of 0 or 1 disables caps mode.

ANSI/HP Compatible Mode

Ec&k *n* \

This escape sequence changes the mode of the terminal emulator. Ec&k0\ changes the mode to HP mode, which is the default mode. Ec&k1\ changes the mode to ANSI mode. The mode of the terminal determines which type of escape sequences it recognizes, which are returned when a key is pressed, and what type of display enhancements are selected.

Note

Remember that Windows/9000 doesn't support ANSI escape sequences. Therefore, this escape sequence should not be used on Windows/9000.

In HP mode, HP escape sequences are returned when a function key is pressed, and display enhancements are field-oriented. Field-oriented display enhancements affect characters starting at the cursor position and continuing to the next display enhancement or to the end of the line. In ANSI mode, ANSI escape sequences are returned when a function key is pressed, and display enhancements are stream-oriented. Stream-oriented display enhancements affect all characters then entered, regardless of position, until another enhancement is sent or display enhancement is turned off.

Straps

Strap A—Transmit Functions. The following escape sequence changes the A strap:

Ec&s *n* A

Strap A specifies whether escape sequences are sent to the terminal or to the host. A "1" sets the strap and a "0" clears it. If the strap is set, escape sequences generated by keys are transmitted to the host. If the strap is clear, escape sequences generated by keys are executed locally and are not transmitted to the host. The default setting for the strap is clear.

Strap C—Inhibit End-of-Line Wrap

ᴱc&s *n* C

Strap C specified whether the cursor wraps to the next line when the end of the line is reached or stays in the last column. A "1" sets the strap and a "0" clears it. When strap C is set, the cursor will not wrap around to the next line of the screen. Instead, when the cursor reaches the far-right column, it stays there until a carriage return or a cursor-movement function is performed. Any typed characters overwrite whatever is in the last column on the screen.

When the strap is clear, a carriage return/line feed is automatically performed when the cursor reaches the last column of the screen. In this way, the cursor wraps from the last column to the first column of the next line.

The default setting for strap C is "0" (clear).

Cursor Control

Cursor On

ᴱc*dQ

This escape sequence turns the cursor on. If the cursor is already on, nothing happens.

Cursor Off

ᴱc*dR

This escape sequence turns the cursor off. If the cursor is already off, nothing happens.

Terminal ID

The following escape sequence returns the terminal ID.

␛*s^

The terminal responds to this escape sequence by returning a character string that identifies the terminal, followed by a carriage return. This string has an arbitrary length and contents which are defined by the terminal.

7 ANSI Escape Sequences

Contents

7 ANSI Escape Sequences

An ANSI escape sequence begins with ᴱᶜ[, is followed by one or more optional parameters, and ends with a character, normally an alpha character. Multiple parameters are separated by a ";". Multiple subfunctions may be selected by separating the parameters with semicolons. The default value is used when a value of zero or no value is specified.

Note HP Windows/9000 does not support ANSI escape sequences.

HP and ANSI escape sequences number rows and columns differently. HP numbers the rows and columns starting with zero. ANSI escape sequences expect the first row and column to be one. Thus the home position is (0,0) for HP escape sequences and (1,1) for ANSI escape sequences.

CPR–Cursor Position Report

ᴱᶜ[*n* ; *n* R

The CPR sequence is the sequence returned to indicate the current cursor position in the buffer. This sequence is output upon receipt of a Device Status Report (DSR). This sequence has two parameter values: the first specifies the row and the second specifies the column of the current cursor position.

CUB–Cursor Backward

␛[*n* D

This escape sequence moves the cursor to the left the specified number of character positions. The distance moved is determined by the parameter. If the parameter value is zero or one, the cursor is moved one column backward. Otherwise the cursor is moved *n* columns backward. The default value of the parameter is 1. This sequence is ignored if the cursor is already in the leftmost column of the buffer. The row position is unchanged.

CUD–Cursor Down

␛[*n* B

This escape sequence moves the cursor down without altering its column position. The number of rows moved is determined by the parameter. If the parameter value is zero or one, the cursor is moved down one row. Otherwise the cursor is moved down *n* rows. The default value of the parameter is 1. This sequence is ignored if the cursor is already on the bottom row of the buffer.

CUF–Cursor Forward

␛[*n* C

This escape sequence moves the cursor forward without changing the row position. The distance moved is determined by the parameter. A parameter value of zero or one moves the cursor to the right one column. Otherwise the the cursor moves right *n* columns. The default value of the parameter is 1. This sequence is ignored if the cursor is already in the rightmost column of the buffer.

CUP–Cursor Position

␛[*n* ; *n* H

This escape sequence moves the cursor to the position on the screen specified by the two parameters. The first parameter specifies the row and the second parameter specifies the column. A parameter of zero or one moves the cursor to the first row or column. If no parameters are specified, the cursor moves to the home position (1, 1).

CUU–Cursor Up

␛[*n* A

This escape sequence move the cursor up without altering the column position. The number of rows moved is determined by the parameter. A parameter value of zero or one moves the cursor up one row. Otherwise the cursor moves up *n* rows. If no parameter is specified, the cursor moves up one row. This sequence is ignored if the cursor is already on the top row of the buffer.

DCH–Delete Character

␛[*n* P

This escape sequence deletes one or more characters from the cursor position to the right of the cursor position. Characters to the right of the cursor that aren't deleted are shifted left until the leftmost of those characters is at the cursor position. A parameter value of zero or one indicates one character is deleted. Otherwise *n* characters are deleted. The default value of the parameter is one. If *n* is greater than the number of characters from the cursor position to the right, only the characters at the cursor position and to the right are deleted.

DL–Delete Line

```
EC[ n M
```

This escape sequence deletes the specified number of rows, beginning with the row containing the cursor. Rows following the cursor row that are not deleted are shifted up until the topmost of those rows is at the cursor position. The deleted rows are blanked and appended to the end of the memory.

A parameter value of zero or one deletes the row the cursor is on and shifts all following rows up one row toward the cursor row. A parameter value of *n* deletes the contents of the *n* rows including and following the cursor row and shifts all adjacent rows *n* rows toward the cursor row.

If you specify an *n* that is greater than the number of rows following the row the cursor is in (the current row), then all rows following the current row, as well as the current row, will be deleted.

DSR–Device Status Report

```
EC[6n
```

This escape sequence requests the current cursor position in the buffer. The terminal emulator will return a Cursor Position Report (CPR) sequence followed by a carriage return in response to a DSR.

ED–Erase Display

```
EC[ n J
```

This escape sequence erases some or all of the characters in the display according to the parameter. If the parameter is zero, the display is erased from the cursor position to the end of the display memory. The cursor is not moved. If the parameter is 1, the display is erased from the beginning of the display memory to the cursor, inclusive. The cursor is not moved. If the parameter is 2, the entire display memory is erased and the cursor goes to the home position. The default value of the parameter is 0.

EL–Erase Line

␛[*n* K

This escape sequence erases some or all of the characters in the row containing the cursor. The cursor isn't moved if the parameter is 0 or 1. If the parameter is zero, the characters at the cursor and to the right are erased. If the parameter is one, the characters at the cursor and to the left are erased. If the parameter is two, the entire row is erased and the cursor moves to the first column. The default value of the parameter is 0.

HVP–Horizontal and Vertical Position

␛[*n* ; *n* f

This escape sequence moves the cursor to the position on the screen specified by the parameters. The first parameter specifies the row and the second specifies the column. A parameter value of zero or one moves the cursor to the first row or column in the display. If no parameters are specified, the cursor moves to the home position (1, 1); this is the same as CUP.

IL–Insert Line

␛[*n* L

This escape sequence inserts one or more blank rows before the row containing the cursor by shifting the contents of the cursor row and all following rows down. The rows at the bottom of display memory are deleted. A parameter value of zero or one causes the contents of the cursor row and the following rows to move down one row. A parameter value of *n* causes the contents of the cursor row and the following rows to move down *n* rows. *n* blank rows are added starting from the cursor row. The cursor is placed at the left margin of the cursor row (the top blank line). If no parameter is specified, one blank line is inserted.

IRM–Insert/Replace Mode

Ec[4h (Insert character mode)
Ec[4l (Replace character mode)

When Insert mode is set, a received character is inserted before the current cursor position. All characters to the right of the cursor on the same row shifted to the right one position for each character entered. The characters shifted off the right are lost. When this mode is cleared, the received character replaces the character displayed at the current cursor location.

NP–Next Page

Ec[*n* U

This escape sequence causes a subsequent page of a multiple-page display memory to be displayed according to the parameter. The number of rows in a page is defined to be the number of rows in the screen. If the parameter value is zero or one, then the next page is displayed. If the parameter value is *n*, then the *n*th page after the current page is displayed. The cursor is moved to the first column and the first row in the screen (the selected page). If no parameter is specified, the next page is displayed.

PP–Previous Page

Ec[*n* V

This escape sequence causes a preceding page of a multiple-page display memory to be displayed. (The number of rows in a page is defined to be the number of rows in the screen.) If the parameter value is zero or one, then the preceding page is displayed. If the parameter value is *n*, then the *n*th page preceding the current page is displayed. The cursor is moved to the first column and the first row in the screen (the selected page). If no parameter is specified, then the previous page is selected.

RCP–Restore Cursor Position

`␛[u`

This escape sequence restores the cursor to the position it had in the buffer when the terminal emulator received an SCP sequence.

SCP–Save Cursor Position

`␛[s`

This escape sequence causes the current cursor position in the buffer to be saved. This cursor position can be restored with the RCP sequence.

SGR–Set Graphics Rendition

␛[*n* ; ... ; *n* M

This escape sequence changes the character attributes or font. The attributes and fonts stay selected until changed by another SGR sequence.

All implementations of Term0 are required to support inverse video and underline. If the hardware doesn't support half bright (or bold) or blinking, these two attributes can be mapped to other enhancements. Half bright or bold maps to inverse video and blinking maps to underline.

Graphics Rendition Parameters

Parameter	Description
0	All attributes off.
1	Bold or half bright on.
4	Underline on.
5	Blinking on.
7	Inverse video on.
10	Load base font.
11	Load alternate font.

The definition of the base and alternate fonts can be changed by using the HP change font escape sequence (if the capability is provided). Using SGR to load the fonts is equivalent to using SI/SO. The default value for the parameter is zero.

Wrap at End of Line

```
␛[=7h
␛[=7l
␛[?7h
␛[?7l
```

␛[=7h and ␛[?7h cause the cursor to wrap at the end of the line when typing. This sets strap C to 0. ␛[=7l and ␛[?7l cause the cursor to not wrap at the end of the line, but remains in the last column when typing. Subsequent characters overwrite the characters in the last column. This sets strap C to 1.

Unimplemented Escape Sequences. The following escape sequences should be accepted without causing an error even though no action is performed.

`␛[=`*n*`h` where *n* = 0 – 6.
`␛[=`*n*`l` where *n* = 0 – 6.
`␛[`*n*`m` where *n* = 8, 30 – 37, 40 – 47.

8 Keys

Contents

8 Keys

This chapter lists the keys on the keyboard and what action the terminal emulator will perform upon receiving the key. In the case of function keys, if strap A is set, the listed escape sequence will be returned.

Key Descriptions

Key	Description
Reset	Reset (system key).
CTRL Reset	System reset (system key).
Break	System dependent (system key).
Stop	System dependent (system key).
f1 through f16	If the user menu is active, the terminal will get the definition of the softkey as was previously defined and will place it in the data stream as the attributes indicated. Otherwise, response is system dependent.
Menu	Turns the display of the menu on and off (system key).
User/System	Controls which menu is displayed and makes the corresponding softkeys active. (System key.)
↖	This key generates an Ech in HP mode, or an Ec[H in ANSI mode, which moves the cursor to home up.
Shift ↖	This key generates an EcF, which moves the cursor to home down.
Clear line	This key generates an EcK in HP mode, or an Ec[K in ANSI mode, which clears a line.

Key Descriptions (Continued)

Key	Description
(Shift)(Clear line)	This key generates ␛G ␛K on HP mode and ␛[2K in ANSI mode, which clears the entire line.
(Clear display)	This key generates an ␛J in HP mode, or an ␛[J in ANSI mode, which clears the display from cursor position to end of display.
(Shift)(Clear display)	This key generates ␛h ␛J in HP mode and ␛[2J in ANSI mode, which clears the entire display.
(Back space)	This key generates a control-H, which back-spaces the cursor.
(Insert line)	This key generates an ␛L in HP mode, or an ␛[L in ANSI mode, which inserts a line.
(Delete line)	This key generates an ␛M in HP mode, or an ␛[M in ANSI mode, which deletes a line.
(Tab◀)	This key generates ␛i in HP mode, or an ␛[Z in ANSI mode, which tabs the cursor backward. (␛[Z isn't a required ANSI escape sequence.)
(Tab▶)	This key generates an control-I, which tabs the cursor forward.
(Insert char)	If insert mode is off, this key generates an ␛Q in HP mode, or an ␛[4h in ANSI mode, which sets insert-character mode. If insert mode is on, this key generates an ␛R in HP mode, or an ␛[4L in ANSI mode, which exits insert-character mode.
(Delete char)	This key generates an ␛P in HP mode, or an ␛[P in ANSI mode, which deletes a character.
(Caps)	This key is trapped by the keyboard driver. It toggles the Caps mode of the keyboard. When Caps mode is on, capital letters are unshifted.
(CTRL)	This is the control key. It is a modifier key used by the keyboard driver.

Key Descriptions (Continued)

Key	Description
Return	This key generates a carriage return by default. It can be redefined, using an escape sequence, to generate up to eight characters.
Prev	This key generates an ␛V in HP mode, or an ␛[V in ANSI mode, which selects the previous page.
Shift	This is a modifier key used by the keyboard driver.
Select	Selection key. (System dependent.)
Next	This key generates an ␛U in HP mode, or an ␛[U in ANSI mode, which selects the next page.
Print	This is the system dependent print key. (System key.)
Enter	This key generates a carriage return/␛d/new line. It can be redefined using an escape sequence. The line the cursor is on is entered.
Extend char	This is a modifier key used by the keyboard driver to allow access to additional character keys.
▲	This key generates an ␛A in HP mode, or an ␛[A in ANSI mode, which moves the cursor up.
◄	This key generates an ␛D in HP mode, or an ␛[D in ANSI mode, which moves the cursor left.
▼	This key generates an ␛B in HP mode, or an ␛[B in ANSI mode, which moves the cursor down.
►	This key generates an ␛C in HP mode, or an ␛[C in ANSI mode, which moves the cursor right.

Key Descriptions (Continued)

Key	Description
(▲)(◀)	These keys generate an ␛A - ␛D in HP mode, or an ␛[A - ␛[D in ANSI mode, which moves the cursor up and left.*
(▲)(▶)	These keys generate an ␛A - ␛C in HP mode, or an ␛[A - ␛[C in ANSI mode, which moves the cursor up and right.*
(▼)(◀)	These keys generate an ␛B - ␛D in HP mode, or an ␛[B - ␛[D in ANSI mode, which moves the cursor down and left.*
(▼)(▶)	These keys generate an ␛B - ␛C in HP mode, or an ␛[B - ␛[C in ANSI mode, which moves the cursor down and right.*
(Shift)(▲)	This key generates an ␛S in HP mode, or an ␛[S in ANSI mode, which rolls the screen up.
(Shift)(◀)	This key generates ␛&r1L in HP mode, or ␛[(space) @ in ANSI mode, which rolls the screen left. These are not required Term0 escape sequences.
(Shift)(▼)	This key generates ␛T in HP mode, or ␛[T in ANSI mode, which rolls the screen down.
(Shift)(▶)	This key generates ␛&r1R in HP mode, or ␛[(space) A in ANSI mode, which rolls the screen right. These are not required Term0 escape sequences.
(Shift)(▲)(◀)	These keys generate ␛&r1u1L in HP mode, or ␛[S - ␛[(space) @ in ANSI mode, which rolls the screen up and to the left.*
(Shift)(▲)(▶)	These keys generate ␛&r1u1R in HP mode, or ␛[S - ␛[(space) A in ANSI mode, which rolls the screen up and to the right.*
(Shift)(▼)(◀)	These keys generate ␛&r1d1L in HP mode, or an ␛[T - ␛[(space) @ in ANSI mode, which rolls the screen down and to the left.*
(Shift)(▼)(▶)	These keys generate ␛&r1dlR in HP mode, or ␛[T - ␛[(space) A in ANSI mode, which rolls the screen down and to the right.*

Key Descriptions (Continued)

Key	Description
(CTRL)(▲)	Graphics cursor up, system dependent (system key).
(CTRL)(◄)	Graphics cursor left, system dependent (system key).
(CTRL)(▼)	Graphics cursor down, system dependent (system key).
(CTRL)(►)	Graphics cursor right, system dependent (system key).
(CTRL)(▲)(◄)	Graphics cursor up and left, system dependent (system key).*
(CTRL)(▲)(►)	Graphics cursor up and right, system dependent (system key).*
(CTRL)(▼)(◄)	Graphics cursor down and left, system dependent (system key).*
(CTRL)(▼)(◄)	Graphics cursor down and right, system dependent (system key).*
Character Keys	The remaining keys are character keys. They input characters but otherwise have no special functions. However, the following character keys when pressed simultaneously with the (CTRL) key have special functions.
(CTRL)(G)	This code causes the speaker to emit a tone.
(CTRL)(H)	This code causes the cursor to backspace.
(CTRL)(I)	This code tabs the cursor right.
(CTRL)(J)	This code executes a line feed.
(CTRL)(M)	This code executes a carriage return.
(CTRL)(N)	This code selects the alternate character set.
(CTRL)(O)	This code selects the base character set.
(CTRL)(Q)	This code resumes output that has been suspended.
(CTRL)(S)	This code suspends output.

* These keystroke sequences are not supported on Windows/9000.

9 Primary Terminal Status

Contents

Chapter 9

9 Primary Terminal Status

Byte 0: Display Memory Size

The lower four bits of this byte specify the amount of display memory available in the terminal. For example, the byte 00110110 specifies 6K bytes of memory. (The most significant four bits are always set as shown.)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	8K bytes	4K bytes	2K bytes	1K bytes

Byte 1: Configuration Straps A through D

Straps A through D affect how the terminal transmits particular keystrokes, and how certain local functions behave.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	Strap D: Page/Line	Strap C: Inhibit End-Of-Line Wrap-around	Strap B: Space Overwrite Latch	Strap A: Function Key Trans-mission

Bit 0: Strap A (Function Key Transmission). When this bit is set, the escape sequence generated by any key is transmitted to the host. When this bit is clear, the escape sequence generated by any key is executed locally, but is not transmitted to the computer.

Bit 1: Strap B (Space Overwrite Latch). When this bit is set, spaces typed will overwrite existing characters. When this bit is clear, spaces typed cause the cursor to move forward but not overwrite existing characters.

Bit 2: Strap C (Inhibit End-of-Line Wraparound). This bit, when set, causes the cursor to remain in the last column rather than wrap around to the first column of the next line. When clear, this bit causes the cursor to wrap around to the first column of the next line when it is in the last column and a character is typed.

Bit 3: Strap D (Page/Line Transmit). When set, this causes the terminal to transmit a block-mode page at a time to the host. When this bit is clear, the terminal transmits a line at a time.

Straps A and C are the only required straps.

Byte 2: Configuration Straps

These strap settings affect how the terminal performs a block transfer.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	Strap H: Inhibit DC2	Strap G: DC2 Hand-shake	Strap F (not used)	Strap E (not used)

The settings of straps G and H affect the terminal's response to a block transfer request according to the table below.

Block Transfer Action

Switch G	Switch H	Block Operation
0	0	Data transfers use DC1/DC2 handshake. Other transfers are triggered by receipt of a DC1 character.
0	1	Data is sent when the host receives a DC2 character. All other transfers are triggered by the receipt of a DC1 character.
1	0	All block transfers require a DC1/DC2 handshake.
1	1	No DC1/DC2 handshake is required for any block transfer.

Byte 3: Latching Keys

This byte defines the action of the latching keys.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	Secondary Status	Automatic Linefeed Key	Block Mode Key	Caps Lock Key

Bit 0: Caps Lock Key. When set, this bit indicates that the terminal generates uppercase characters only. When clear, the terminal generates uppercase and lowercase characters.

Bit 1: Block Mode Key. When this bit is set, the terminal operates in block mode. When this bit is clear, the terminal operates in character mode.

Bit 2: Automatic Linefeed Key. When this bit is set, a linefeed occurs whenever a carriage return occurs. When this bit is clear, a linefeed does not automatically occur.

Bit 3. Secondary Status. When set, this bit indicates that the terminal sends its secondary status when requested. When this bit is clear, the status is not sent.

Byte 4: Transfer Pending Flags

These flags indicate that terminal information is available.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	Secondary Status Pending	(ENTER) Key Pending	Function Key Pending	Cursor Sense Pending

Bit 0: Cursor Sense Pending. When set, this bit indicates that a cursor sense is pending. When clear, no cursor sense is pending.

Bit 1: Function Key Pending. When set, this bit indicates that a function key has been pressed. When clear, the function key does not require attention.

Bit 2: (ENTER) Key Pending. When set, this bit indicates that the (ENTER) key on the terminal has been pressed. When clear, this bit indicates the condition no longer exists.

Bit 3: Secondary Status Pending. When set, this bit indicates that the terminal's secondary status is pending. When this bit is clear, no secondary status is pending.

Byte 5: Error Flags

This byte indicates the status of three datacomm error conditions.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	Device Error	0	Self-Test	Data Comm

Bit 0: Data Comm. When set, this bit indicates that one or more of the following errors has occurred: parity, buffer overflow, and framing overrun.

Bit 1: Self-Test. When set, this bit indicates that a self-test error did not occur. When clear, an error has occurred.

Bit 3: Device Error. This bit is set when an external device indicates an error. When this bit is clear, no device error is indicated.

Byte 6: Device Transfer Pending Flags

This byte indicates when device status information is available.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	1	1	0	0	Device Operation Status Pending	Device Status Pending

10 Escape Sequences Specific to The Portable

Contents

10 Escape Sequences Specific to The Portable

The Portable implements most of Term0 features. However, some variations are implemented on the computer. This chapter describes how The Portable differs from Term0.

Display Enhancements. For the ␛&d *enhancements*, The Portable implements blinking and inverse attributes for each character, and it will also implement the inverse video blinking escape sequences. However, escape sequences that aren't understood will be ignored and not mapped to an existing characteristic. (Refer to the table under "Display Enhancements" in chapter 6 for combinations that include inverse video and blinking.) Escape sequences are interpreted according to the ANSI stream standard, not the HP field-oriented method.

Tabs and Margins. The Portable allows only fixed-space tabs of eight spaces, forward and backward.

Cursor Movement. All cursor movement sequences in The Portable adheres to the ANSI standard. When the cursor reaches a screen boundary, it will not wrap. (This is the Term0 standard but conflicts with the HP-2622 definition.)

Font Selection. The Portable does not allow the Shift In (^N) and Shift Out (^O) of base and alternate fonts.

Select Screen Type. The Portable supports the following ANSI escape sequences:

- Ec[=2 h (80 by 25 mode).
- Ec[=3 h (80 by 25 mode).
- Ec[=4 h (Graphics mode).
- Ec[=8 h (80 by 16 mode).

Multiple Keys and Diagonal Scrolling. The Portable cannot detect multiple key presses, so only motion in the indicated directions is possible. Pressing multiple keys is an illegal keystroke and will result in nothing happening.

11 Integral PC Escape Sequences

Contents

Chapter 11

11 Integral PC Escape Sequences

This chapter describes escape sequences in the Integral PC that are not a part of Term0.

HP Escape Sequences

Set Tab Stop (Ec1). Sets a tab stop at the current cursor column.

Clear Tab Stop (Ec2). Clears the tab stop from the current cursor column.

Clear All Tab Stops (Ec3). Clears all of the tab stops.

Set Left Margin (Ec4). Sets the left margin at the current cursor column. If you try to set the left margin to the right of the right margin, the terminal beeps to indicate an error and ignores the escape sequence.

Set Right Margin (Ec5). Sets the right margin at the current cursor column. If you try to set the right margin to the left of the left margin, the terminal will beep to indicate an error and will ignore the escape sequence.

Clear Margins (Ec9). Clears the margins to the default of the left margin at column 0 and the right margin at column 79 to give the full 80 character display.

Primary Terminal Status(Ec^). The request for the primary terminal status causes seven bytes of information about the status of the terminal to be returned. The terminal first sends Ec\ then sends bytes 0 through 6 of the primary terminal status, and then a carriage return.

These bytes have the following values and meanings:

Primary Terminal Status Bytes

Byte Number	Decimal Value	Description
0	52	There are 4K bytes of display memory—2 pages of 24 rows by 80 columns.
1	48	No straps are set.
	49	Strap A is set.
	52	Strap C is set.
	53	Straps A and C are both set.
2	60	Straps G and H are clear.
3	56	The locking keys are not implemented and the secondary status is not sent.
4	48	No transfers are pending. (This is related to block mode transfers, which are not implemented.)
5	48	There are no errors.
6	48	There are no pending device transfers.

Secondary Terminal Status (Ec~). The request for secondary terminal status causes ten bytes to be sent. The terminal sends Ec | followed by bytes 0 through 6 and a carriage return.

Secondary Terminal Status Bytes

Byte Number	Decimal Value	Description
0	48	The terminal has no additional memory.
1	52	Terminal is non-programmable; terminal identifies itself; there is no APL firmware; there is no I/O firmware.
2	48	Straps J through M are not implemented.
3	48	Straps N through R are not implemented.
4	48	Straps S through V are not implemented.
5	48	Straps W through Z are not implemented.
6	48	Memory lock is not implemented.

Terminal ID (Ec*s^). The terminal responds by sending an identification string. This is an arbitrary string that is defined by the terminal.

Enable Keyboard (Ecb). Allows the keys typed to be accepted.

Disable Keyboard (Ecc). This causes the keys that are typed, with the exception of the signals (such as (Reset), (Stop) and (Break)), to be ignored. The terminal should beep if typing is attempted with the keyboard disabled. The keyboard remains disabled until it is explicitly enabled, a reset is performed, or the power is turned off.

Softkey Definition. The Integral PC has sixteen softkeys rather than eight. The key number can be 1 through 16 for the function keys, 0 for the (Return) key, and −1 for the (Enter) key. Initially, keys 1 through 8 are defined to be Ecp through Ecw and the shifted keys 9 through 16 are undefined. When an unshifted softkey (key 1 through 8) is defined and no shifted softkeys are defined, 16 characters are used for the label, using both lines in the menu to display the label. If any shifted softkey is defined, only the first eight characters of the label are kept and used in the menu. The shifted softkey label is on the top line, the unshifted softkey label is on the bottom line, and a line between the labels indicates that they are for separate keys. If the shifted function key is not defined, it is mapped to the corresponding unshifted function key.

Invoke Softkeys

Ec&f *key* E

The parameter *key* is the number of the key to be invoked. −1 invokes the (Enter) key, 0 invokes the (Return) key, and 1 through 16 invoke the function keys. If *key* isn't in the range of −1 through 16, the escape sequence is ignored.

Menus. In addition to turning off the menu and turning on the user menu, in the Integral PC you can turn on the system menu.

Ec&jA

This escape sequence turns on the system menu and makes the system softkeys active.

Roll. The following escape sequence rolls the screen in all four directions.

```
ᴱc&r nd n l n r n U
```

The parameters are interpreted as follows:

- *n*d is the number of rows to roll the screen down.
- *n*l is the number of rows to roll the screen to the left.
- *n*r is the number of rows to roll the screen to the right.
- *n*u is the number of rows to roll the screen up.

The parameters can be in any order, and any number of them (up to four) can be present. The last parameter should be capitalized to serve as the terminating character. You can roll only until you reach a buffer boundary; any further rolls in that direction will be ignored.

Select Character Font. Each character set has a unique identification string assigned to it. Also, a character set can have different-sized fonts of the same characters. The following escape sequences select the character set to be either the base character set or the alternate character set. The size information of the font is associated with the terminal since each terminal can have only one size of font.

With the ID string of the character set and the size information, an ID number is constructed for the font. Given this ID number, the system pool of fonts is searched. If the font is not in the system pool, the escape sequence is ignored and no change is made. If the font is in the pool, it becomes either the base font or the alternate font. The escape sequences change the base font and the alternate font independently. SI and SO are used to switch between the base and the alternate fonts.

Change Base Font

Ec(*ID string*

The *ID string* of the character set consists of a number (optional) and a letter. It selects the character set indicated by the ID to be the base character set.

Some of the character sets which the Integral PC implements are:

Integral PC Character Sets

Identify String	Description
8U	The Roman 8 character set. This is the default character set.
8K	The Kana 8 character set. This is the Katakana set for use in Japan.
8M	Math/Special Symbols 8. This character set conatians the Greek alphabet and other math symbols.
8L	Line Draw 8. This is the line drawing character set used for drawing forms.

Change Alternate Font

Ec) *ID string*

ANSI Escape Sequences

CBT–Cursor Backward Tabulation

␛[*n* Z

This escape sequence moves the cursor backwards to the preceding tab stop indicated by the parameter value. A parameter value of zero or one indicates the preceding tab stop, while a larger value (*n*) indicates the *n*th preceding tab stop. The default parameter value is one.

SD–Scroll Down

␛[*n* T

This escape sequence causes the entire contents of the screen to be moved down a specified number of lines. The bottom lines are removed from view and new lines are moved into the top lines. The number of lines to scroll is determined by the value of the parameter. If the parameter value is zero or one, one line is scrolled. If the parameter value is *n*, *n* lines are scrolled. If no parameter is specified, one line is scrolled.

SL–Scroll Left

␛[*n* (space) @

This escape sequence causes the entire contents of the screen to be moved to the left a specified number of columns. The leftmost columns are removed from view and new columns are moved into the rightmost columns. The number of columns to scroll is determined by the value of the parameter. If the parameter is zero or one, one column is scrolled. If the parameter value is *n*, *n* columns are scrolled. If no parameter is specified, one column is scrolled.

SR–Scroll Right

ᴱc[*n* (space) A

This escape sequence causes the entire contents of the screen to be moved to the right a specified number of columns. The rightmost columns are removed from view and new columns are moved into the leftmost columns. The number of columns to scroll is determined by the parameter. If the parameter value is zero or one, one column is scrolled. If the parameter value is *n*, *n* columns are scrolled. If no parameter is specified, one column is scrolled.

SU–Scroll Up

ᴱc[*n* S

This escape sequence causes the entire contents of the screen to be moved up a specified number of lines. The top lines are removed from view and new lines are moved into the bottom lines. The number of lines to scroll is determined by the parameter. If the parameter value is zero or one, one line is scrolled. If the parameter value is *n*, *n* lines are scrolled. If no parameter is specified, one line is scrolled.

12 Windows/9000 Escape Sequences

Contents

12 Windows/9000 Escape Sequences

This chapter describes escape sequences in Windows/9000 that are not a part of Term0.

HP Escape Sequences

Set Tab Stop (Ec1). Sets a tab stop at the current cursor column.

Clear Tab Stop (Ec2). Clears the tab stop from the current cursor column.

Clear All Tab Stops (Ec3). Clears all of the tab stops.

Set Left Margin (Ec4). Sets the left margin at the current cursor column. If you try to set the left margin to the right of the right margin, the terminal beeps to indicate an error and ignores the escape sequence.

Set Right Margin (Ec5). Sets the right margin at the current cursor column. If you try to set the right margin to the left of the left margin, the terminal will beep to indicate an error and will ignore the escape sequence.

Clear Margins (Ec9). Clears the margins to the default of the left margin at column 0 and the right margin at column 79 to give the full 80 character display.

Primary Terminal Status (␛^–). The request for the primary terminal status causes seven bytes of information about the status of the terminal to be returned. The terminal first sends ␛\ then sends bytes 0 through 6 of the primary terminal status, and then a carriage return.

These bytes have the following values and meanings:

Primary Terminal Status Bytes

Byte Number	Decimal Value	Description
0	N≥48	There are (N−48) kilobytes of display memory—pages × rows × columns.
1	48 49 52 53	No straps are set. Strap A is set. Strap C is set. Straps A and C are both set.
2	60	Straps G and H are clear.
3	56	The locking keys are not implemented and the secondary status is not sent.
4	48	No transfers are pending. (This is related to block mode transfers, which are not implemented.)
5	48	There are no errors.
6	48	There are no pending device transfers.

Terminal ID (␛*s^). The terminal responds by sending an identification string. This is an arbitrary string that is defined by the terminal.

Enable Keyboard (␛b). Allows the keys typed to be accepted.

Disable Keyboard (␛c). This causes the keys that are typed, with the exception of the keys Reset, Stop and Break, to be ignored. The terminal should beep if typing is attempted with the keyboard disabled. The keyboard remains disabled until it is explicitly enabled, or a reset is performed.

Softkey Definition. The Windows/9000 windows each have sixteen softkeys rather than eight. The key number can be 1 through 16 for the function keys, 0 for the (Return) key, and −1 for the (Enter) key. The default case is for keys 1 through 8 to have the default definitions of Ecp through Ecw and for the shifted keys 9 through 16 to be undefined. When an unshifted softkey (key 1 through 8) is defined and no shifted softkeys are defined, 16 characters are used for the label, using both lines in the menu to display the label. If any shifted softkey is defined, only the first eight characters of the label are kept and used in the menu. The shifted softkey label is on the top line, the unshifted softkey label is on the bottom line, and a line between the labels indicates that they are for separate keys. If the shifted function key is not defined, it is mapped to the corresponding unshifted function key.

Windows/9000 implements only *normal attribute* windows (0a).

Invoke Softkeys

Ec&f *key* E

where *key* is the number of the key to be invoked. −1 invokes the (Enter) key, 0 invokes the (Return) key, and 1 through 16 invoke the function keys. If *key* isn't in the range of −1 through 16, the escape sequence is ignored.

Menus. In addition to turning off the menu and turning on the user menu, with Windows/9000 you can turn on the Term0 menu.

Ec&jA

turns on the Term0 menu, activating the Term0 function keys. From this menu you can select Display Functions mode.

Roll. The following escape sequence rolls the screen in all four directions.

ᴱc&r *n* d *n* l *n* r *n* U

where:

- *n*d is the number of rows to roll the screen down.
- *n*l is the number of rows to roll the screen to the left.
- *n*r is the number of rows to roll the screen to the right.
- *n*u is the number of rows to roll the screen up.

The parameters can be in any order, and any number of them (up to four) can be present. The last parameter should be capitalized to serve as the terminating character. You can roll only until you reach a buffer boundary; any further rolls in that direction will be ignored.

Select Character Font. Each character set has a unique identification string assigned to it. Also, a character set can have different-sized fonts of the same characters. The following escape sequences select the character set to be either the base character set or the alternate character set. The size information of the font is associated with the Term0 window since each window can have only one size of font.

With the ID string of the character set and the size information, an ID number is constructed for the font. Given this ID number, the system pool of fonts is searched. If the font is not in the system pool, the escape sequence is ignored and no change is made. If the font is in the pool, it becomes either the base font or the alternate font. The escape sequences change the base font and the alternate font independently. SI and SO are used to switch between the base and the alternate fonts.

Change Base Font

Ec< *font ID*

This escape sequence selects the new base font for the terminal. Windows/9000 stores fonts in special files. The last two characters of the file name indicate the type of font. (The table below describes these two-character codes.) The last two characters of the escape sequence are the *font id.* To activate a font using this escape sequence you must use the two-character font ID.

Some of the character sets which the Windows/9000 implements are:

Windows/9000 Font IDs

Font ID	Description
8U	The Roman 8 character set. This is the default character set.
8K	The Kana 8 character set. This is the Katakana set for use in Japan.
8M	Math/Special Symbols 8. This character set conatians the Greek alphabet and other math symbols.
0L	Line Draw 8. This is the line drawing character set used for drawing forms.

Change Alternate Font

Ec> *Font ID*

This escape sequence selects the alternate font specified by *Font ID*.

Color Selection Escape Sequences

Each Term0 window has eight color pairs that can be used as character enhancements. Each color pair consists of a foreground and a background color. The following table defines the eight *default* color pairs.

Default Term0 Color Pairs

Color Pair Number	Foreground Color	Background Color
0	White	Black
1	Red	Black
2	Green	Black
3	Yellow	Black
4	Blue	Black
5	Magenta	Black
6	Cyan	Black
7	Black	Yellow

When no other color enhancement is selected, the default color pair is "0". Thus, when a window is created, text is displayed as white characters on a black background. Note that in Term0 windows, the half-bright enhancement is displayed as color pair 3 and blinking is ignored.

Color Escape Sequence Format

Color escape sequences have the following form:

$^{E}{}_{C}$&v *parameters*

Parameters can be one or more of the commands in the table below.

Color Escape Sequence Parameters

Parameter	**Description**
`0m`	Set RGB as the color method (default).
n`a`	Red color value for foreground.
n`b`	Green color value for foreground.
n`c`	Blue color value for foreground.
n`x`	Red color value for background.
n`y`	Green color value for background.
n`z`	Blue color value for background.
color-pair`i`	Color pair to be initialized.
color-pair`s`	Color pair to be selected.
color-pair`^`	Color pair definition status request.

The following rules apply to color escape sequences:

- *n* is either 0 or 1.
- *color-pair* is within the range 0 through 7.
- The "^" character terminates the escape sequence.
- "I" and "S" terminate the sequence, but "i" and "s" do not.
- The "0m" is optional.
- It doesn't make sense to use "s" or "S" more than once in a sequence.
- You can initialize ("i" or "I") more than one color pair in an escape sequence.
- You cannot initialize more than one color pair to the same colors without respecifying those colors every time.
- In specifying RGB colors, you can omit any parameter whose value is zero, or you can show the 0 parameter for completeness. Note that this is true only for the a, b, c, x, y, and z parameters.

Selecting a Color Pair

The escape sequence

Ec&v *color-pair* S

sets the foreground and background colors to those specified by *color-pair.* These colors remain in effect until a different color pair is selected, but only on the current line of text. If the cursor is moved to a different line of text, color pair 0 will be used again. (The color pairs behave much the same as other HP "field-oriented" escape sequences, such as inverse video and underlining.)

Redefining a Color Pair

You can redefine any color pair in the color pair table. Each color is defined from different combinations of red, green, and blue (RGB color values). By specifying different combinations of red, green, and blue, you can redefine the foreground and background colors for a specific color pair in the Term0 color pair table.

To redefine the foreground and background colors for a color pair, use the following escape sequence:

Ec&v*n*a*n*b*n*c*n*x*n*y*n*z *color-pair*I

Where *a*, *b*, and *c* define the foreground colors and *x*, *y*, and *z* define the background colors. *n* can be either 0 or 1. If *n* is 1, then the corresponding color is turned on; if *n* is 0, the corresponding color is turned off. The table below shows the colors produced by different combinations of red, green, and blue.

RGB Color Definition Values

R	G	B	Color Produced
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

Example. For color pair 3, set the foreground color to blue and the background color to yellow.

```
Ec&v0a0b1c1x1y0z3I
```

If any parameter value is 0, it can be omitted. The escape sequence will have the same effect. So in the example above, you could use:

```
Ec&v1c1x1y3I
```

Getting the Color-Pair Status

To obtain the values of the current foreground and background colors for a color pair, use an escape sequence of the form:

Ec&v *color-pair* ^

If *color-pair* isn't specified, then information is returned for all color pairs. Information is returned in an escape sequence normally from the standard input. The format of this escape sequence is:

```
Ec&vnanbncnxnynznnI
```

If the corresponding value for an RGB parameter is 1, then *n* is returned as 1.00 for that parameter. However, if the RGB value is not set, then the associated parameter is not even returned in the string. *nn* represents a two-digit number in the range 00 through 07; it corresponds to the color pair for which status information is returned.

For example, if color pair 6 of the Term0 color pair table hasn't yet been modified, then the escape sequence

```
Ec&v6^
```

returns

```
Ec&v0m1.00b1.00c06I
```

Note that none of the x, y, or z parameters were returned. This is because the default background color is black (RGB values are all 0). Also, the "a" parameter was not returned because red is not used in making cyan.

The returned string is useful because it is also a valid escape sequence which can be used to set the color pair for a window. For example, you could use this escape sequence to get a window's default color pair 0. Then any windows spawned from the window can have their color pair 0 set to that of the "parent" window—all associated windows have the same default (color pair 0) foreground and background colors. To change the "child" window's color pair 0 in this manner, you would have to write the returned string from the "parent" to the opened file descriptor of the new "child" window.

A Roman8 Character Set

The Roman8 character set consists of the standard U.S. ASCII character set and the Roman Extension character set. Each character in the set is assigned a character code with a decimal value from 0 through 255. The system uses these codes for identifying characters. Characters are normally produced from the keyboard by pressing the corresponding keys. (Refer to appendix B for keyboard mapping.)

The first half of the Roman8 character set (decimal values 0 through 127) is identical to the standard character set used on many other computer systems. The second half (decimal values 128 through 255) contains special characters, including those used by other languages.

US ASCII Character Set

ASCII Char.	Character Code			
	Dec	Binary	Oct	Hex
NUL	0	00000000	000	00
SOH	1	00000001	001	01
STX	2	00000010	002	02
ETX	3	00000011	003	03
EOT	4	00000100	004	04
ENQ	5	00000101	005	05
ACK	6	00000110	006	06
BEL	7	00000111	007	07
BS	8	00001000	010	08
HT	9	00001001	011	09
LF	10	00001010	012	0A
VT	11	00001011	013	0B
FF	12	00001100	014	0C
CR	13	00001101	015	0D
SO	14	00001110	016	0E
SI	15	00001111	017	0F
DLE	16	00010000	020	10
DC1	17	00010001	021	11
DC2	18	00010010	022	12
DC3	19	00010011	023	13
DC4	20	00010100	024	14
NAK	21	00010101	025	15
SYNC	22	00010110	026	16
ETB	23	00010111	027	17
CAN	24	00011000	030	18
EM	25	00011001	031	19
SUB	26	00011010	032	1A
ESC	27	00011011	033	1B
FS	28	00011100	034	1C
GS	29	00011101	035	1D
RS	30	00011110	036	1E
US	31	00011111	037	1F

ASCII Char.	Character Code			
	Dec	Binary	Oct	Hex
space	32	00100000	040	20
!	33	00100001	041	21
"	34	00100010	042	22
#	35	00100011	043	23
$	36	00100100	044	24
%	37	00100101	045	25
&	38	00100110	046	26
'	39	00100111	047	27
(	40	00101000	050	28
)	41	00101001	051	29
*	42	00101010	052	2A
+	43	00101011	053	2B
,	44	00101100	054	2C
-	45	00101101	055	2D
.	46	00101110	056	2E
/	47	00101111	057	2F
0	48	00110000	060	30
1	49	00110001	061	31
2	50	00110010	062	32
3	51	00110011	063	33
4	52	00110100	064	34
5	53	00110101	065	35
6	54	00110110	066	36
7	55	00110111	067	37
8	56	00111000	070	38
9	57	00111001	071	39
:	58	00111010	072	3A
;	59	00111011	073	3B
<	60	00111100	074	3C
=	61	00111101	075	3D
>	62	00111110	076	3E
?	63	00111111	077	3F

US ASCII Character Set (Continued)

ASCII Char.	Character Code			
	Dec	Binary	Oct	Hex
@	64	01000000	100	40
A	65	01000001	101	41
B	66	01000010	102	42
C	67	01000011	103	43
D	68	01000100	104	44
E	69	01000101	105	45
F	70	01000110	106	46
G	71	01000111	107	47
H	72	01001000	110	48
I	73	01001001	111	49
J	74	01001010	112	4A
K	75	01001011	113	4B
L	76	01001100	114	4C
M	77	01001101	115	4D
N	78	01001110	116	4E
O	79	01001111	117	4F
P	80	01010000	120	50
Q	81	01010001	121	51
R	82	01010010	122	52
S	83	01010011	123	53
T	84	01010100	124	54
U	85	01010101	125	55
V	86	01010110	126	56
W	87	01010111	127	57
X	88	01011000	130	58
Y	89	01011001	131	59
Z	90	01011010	132	5A
[	91	01011011	133	5B
\	92	01011100	134	5C
]	93	01011101	135	5D
^	94	01011110	136	5E
_	95	01011111	137	5F

ASCII Char.	Character Code			
	Dec	Binary	Oct	Hex
`	96	01100000	140	60
a	97	01100001	141	61
b	98	01100010	142	62
c	99	01100011	143	63
d	100	01100100	144	64
e	101	01100101	145	65
f	102	01100110	146	66
g	103	01100111	147	67
h	104	01101000	150	68
i	105	01101001	151	69
j	106	01101010	152	6A
k	107	01101011	153	6B
l	108	01101100	154	6C
m	109	01101101	155	6D
n	110	01101110	156	6E
o	111	01101111	157	6F
p	112	01110000	160	70
q	113	01110001	161	71
r	114	01110010	162	72
s	115	01110011	163	73
t	116	01110100	164	74
u	117	01110101	165	75
v	118	01110110	166	76
w	119	01110111	167	77
x	120	01111000	170	78
y	121	01111001	171	79
z	122	01111010	172	7A
{	123	01111011	173	7B
|	124	01111100	174	7C
}	125	01111101	175	7D
~	126	01111110	176	7E
DEL	127	01111111	177	7F

Roman Extension Character Set

Char.	Character Code			
	Dec	Binary	Oct	Hex
	128	10000000	200	80
	129	10000001	201	81
	130	10000010	202	82
	131	10000011	203	83
	132	10000100	204	84
	133	10000101	205	85
	134	10000110	206	86
	135	10000111	207	87
	136	10001000	210	88
	137	10001001	211	89
	138	10001010	212	8A
	139	10001011	213	8B
	140	10001100	214	8C
	141	10001101	215	8D
	142	10001110	216	8E
	143	10001111	217	8F
	144	10010000	220	90
	145	10010001	221	91
	146	10010010	222	92
	147	10010011	223	93
	148	10010100	224	94
	149	10010101	225	95
	150	10010110	226	96
	151	10010111	227	97
	152	10011000	230	98
	153	10011001	231	99
	154	10011010	232	9A
	155	10011011	233	9B
	156	10011100	234	9C
	157	10011101	235	9D
	158	10011110	236	9E
	159	10011111	237	9F

Char.	Character Code			
	Dec	Binary	Oct	Hex
space	160	10100000	240	A0
À	161	10100001	241	A1
Â	162	10100010	242	A2
È	163	10100011	243	A3
Ê	164	10100100	244	A4
Ë	165	10100101	245	A5
Î	166	10100110	246	A6
Ï	167	10100111	247	A7
´	168	10101000	250	A8
`	169	10101001	251	A9
ˆ	170	10101010	252	AA
¨	171	10101011	253	AB
˜	172	10101100	254	AC
Ù	173	10101101	255	AD
Û	174	10101110	256	AE
₤	175	10101111	257	AF
¯	176	10110000	260	B0
Ý	177	10110001	261	B1
ý	178	10110010	262	B2
°	179	10110011	263	B3
Ç	180	10110100	264	B4
ç	181	10110101	265	B5
Ñ	182	10110110	266	B6
ñ	183	10110111	267	B7
¡	184	10111000	270	B8
¿	185	10111001	271	B9
¤	186	10111010	272	BA
£	187	10111011	273	BB
¥	188	10111100	274	BC
§	189	10111101	275	BD
ƒ	190	10111110	276	BE
¢	191	10111111	277	BF

Roman Extension Character Set (Continued)

Char.	Character Code			
	Dec	Binary	Oct	Hex
â	192	11000000	300	C0
ê	193	11000001	301	C1
ô	194	11000010	302	C2
û	195	11000011	303	C3
á	196	11000100	304	C4
é	197	11000101	305	C5
ó	198	11000110	306	C6
ú	199	11000111	307	C7
à	200	11001000	310	C8
è	201	11001001	311	C9
ò	202	11001010	312	CA
ù	203	11001011	313	CB
ä	204	11001100	314	CC
ë	205	11001101	315	CD
ö	206	11001110	316	CE
ü	207	11001111	317	CF
Å	208	11010000	320	D0
î	209	11010001	321	D1
Ø	210	11010010	322	D2
Æ	211	11010011	323	D3
å	212	11010100	324	D4
í	213	11010101	325	D5
ø	214	11010110	326	D6
æ	215	11010111	327	D7
Ä	216	11011000	330	D8
ì	217	11011001	331	D9
Ö	218	11011010	332	DA
Ü	219	11011011	333	DB
É	220	11011100	334	DC
ï	221	11011101	335	DD
ß	222	11011110	336	DE
Ô	223	11011111	337	DF

Char.	Character Code			
	Dec	Binary	Oct	Hex
Á	224	11100000	340	E0
Ã	225	11100001	341	E1
ã	226	11100010	342	E2
Đ	227	11100011	343	E3
đ	228	11100100	344	E4
Í	229	11100101	345	E5
Ì	230	11100110	346	E6
Ó	231	11100111	347	E7
Ò	232	11101000	350	E8
Õ	233	11101001	351	E9
õ	234	11101010	352	EA
Š	235	11101011	353	EB
š	236	11101100	354	EC
Ú	237	11101101	355	ED
Ÿ	238	11101110	356	EE
ÿ	239	11101111	357	EF
Þ	240	11110000	360	F0
þ	241	11110001	361	F1
▪	242	11110010	362	F2
µ	243	11110011	363	F3
¶	244	11110100	364	F4
¾	245	11110101	365	F5
—	246	11110110	366	F6
¼	247	11110111	367	F7
½	248	11111000	370	F8
ª	249	11111001	371	F9
º	250	11111010	372	FA
«	251	11111011	373	FB
■	252	11111100	374	FC
»	253	11111101	375	FD
±	254	11111110	376	FE
▒	255	11111111	377	FF

B Level 0/Level 1 Terminal Comparison

Feature	2622 Term1	Term0	2621
Character Mode	X	X	X
Block Mode			
Line	X		X
Page	X		
Block Transfer Straps			
G	X		X
H	X		X
D	X		
Modify Mode			
Line	X		X
All	X		
Format Mode	X		
Automatic Linefeed Mode	X		X
Memory Lock/Overflow Protect	X		
Display Functions Mode	X	X	X
Caps Mode	X	X	X
Caps Lock Mode (Teletype)	X		X

Feature	2622 Term1	Term0	2621
Cursor Control			
Absolute	X	X	X
Screen Relative	X	X	X
Screen Absolute	X	X	X
Screen Editing	X	X	X
User-Defined Tabs and Margins	X		X
Remote Cursor Sensing	X	X*	
Display Enhancements	4	2	1
ENQ/ACK Handshaking	X	X	X
Transmit Pacing (XON/XOFF)	X	X	
Receive Pacing (XON/XOFF)	X	X	X
Responds To:			
Terminal ID Request	X	X*	
Primary Status Request	X	X*	X
Secondary Status Request	X		
Definable Block Terminator	X		
Miscellaneous Straps			
A	X	X	
B	X		X
C	X	X	X
User-Definable Return Key	2 Char.	8 Char.	2 Char.
Multiple Handshakes for Block Transfers			

Feature	2622 Term1	Term0	2621
DC1	X		X
DC1/DC2/DC1	X		X
None	X	X	X
Definable Softkeys	8	8	8 Predefined
Softkey Types	Local, Normal, Transmit	Normal	Transmit

* The requested information is sent immediately. Block handshaking does not occur.

C Comparison of Escape Sequences

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
Control Codes					
CTRL G	Ring Bell.	X	X	X	X
CTRL H	Backspace.	X	X	X	X
CTRL I	Tab.	X	X	X	X
CTRL J	Linefeed.	X	X	X	X
CTRL M	Carriage Return.	X	X	X	X
CTRL N	Select alternate character set.	X	X		X
CTRL O	Select base character set.	X	X		X
CTRL Q	Resume output.	X	X	X	X
CTRL S	Stop output.	X	X	X	X
HP Escape Sequences					
Ec1	Set tab stop.		X		X
Ec2	Clear tab stop.		X		X
Ec3	Clear all tab stops.		X		X
Ec4	Set left margin.		X		X
Ec5	Set right margin.		X		X
Ec9	Clear margins.		X		X
EcA	Cursor up.	X	X	X	X
EcB	Cursor down.	X	X	X	X
EcC	Cursor right.	X	X	X	X
EcD	Cursor left.	X	X	X	X

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
␛F	Cursor home down.	X	X	X	X
␛H	Cursor home up.	X	X	X	X
␛I	Tab.	X	X	X	X
␛J	Clear display.	X	X	X	X
␛K	Clear line.	X	X	X	X
␛L	Insert line.	X	X	X	X
␛M	Delete line.	X	X	X	X
␛P	Delete character.	X	X	X	X
␛Q	Insert character mode on.	X	X	X	X
␛R	Insert character mode off.	X	X	X	X
␛S	Roll up.	X	X	X	X
␛T	Roll down.	X	X	X	X
␛U	Next page.	X	X	X	X
␛V	Previous page.	X	X	X	X
␛Y	Display functions on.	X	X	X	X
␛Z	Display functions off.	X	X	X	X
␛^	Primary terminal status.	X	X	X	X
␛~	Secondary terminal status.		X		
␛'	Sense cursor position (relative).	X	X	X	X
␛a	Sense cursor position (absolute).	X	X	X	X
␛b	Enable keyboard.		X		X
␛c	Disable keyboard.		X		X
␛d	Enter line.	X	X	X	X
␛h	Home up.	X	X	X	X
␛i	Back tab.	X	X	X	X
␛p	Softkey 1 default value.	X	X	X	X
␛q	Softkey 2 default value.	X	X	X	X
␛r	Softkey 3 default value.	X	X	X	X
␛s	Softkey 4 default value.	X	X	X	X

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
Ec t	Softkey 5 default value.	X	X	X	X
Ec u	Softkey 6 default value.	X	X	X	X
Ec v	Softkey 7 default value.	X	X	X	X
Ec w	Softkey 8 default value.	X	X	X	X
Ec&a *n* x *n* Y	Screen Relative cursor addressing.	X	X	X	X
Ec&a *n* c *n* R	Absolute cursor addressing.	X	X	X	X
Ec&a± *n* c± *n* R	Cursor relative addressing.	X	X	X	X
Ec&d@	Turn off attributes.	X	X	X	X
Ec&dA	Blinking.	X	*	X	‡
Ec&dB	Inverse video.	X	X	X	X
Ec&dC	Blinking and inverse video.	X	*	X	‡
Ec&dD	Underline.	X	X	X	X
Ec&dE	Blinking and underline.	X	*	X	‡
Ec&dF	Inverse video and underline.	X	X		X
Ec&dG	Blinking, inverse video, and underline.	X	*		‡
Ec&dH	Half-bright.	X	*		‡
Ec&dI	Blinking and half-bright.	X	*		‡
Ec&dJ	Inverse video and half-bright.	X	*		‡
Ec&dK	Blinking, inverse, and half-bright.	X	*		‡
Ec&dL	Underline and half-bright.	X	*		‡
Ec&dM	Blinking, underline, and half-bright.	X	*		‡
Ec&dN	Inverse, underline, and half-bright.	X	*		‡
Ec&dO	Blinking, inverse, underline, and half-bright.	X	*		‡

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
Ec&f *n* a *n* k *n* 1 *n* D *label string*	Define softkey.	X	X	X	X
Ec&f *n* E	Invoke softkey.		X		X
Ec&j@	Turn off menu.	X	X	X	X
Ec&jA	Turn on system menu.		X	X	X
Ec&jB	Turn on application menu.	X	X	X	X
Ec&k *n* O	Keycode mode.			X	
Ec&k *n* P	Caps mode.	X	X	X	X
Ec&k *n* \	Change ANSI/HP mode.	X	X	X	
Ec&r *n* d *n* 1 *n* r *n* U	Roll.		X		X
Ec&s *n* A	Strap A: Transmit function keys.	X	X	X	X
Ec&s *n* C	Strap C: Inhibit word wrap.	X	X		X
Ec*dQ	Cursor on.	X	X	X	X
Ec*dR	Cursor off.	X	X	X	X
Ec*s^	Send terminal ID.	X	X	X	X
Ec(*n* A	Change base font.		X		X
Ec) *n* A	Change alternate font.		X		X
ANSI Escape Sequences					
Ec<	Enter ANSI mode.		X		
Ec[*n* Z	Backtab		X		
Ec[*n* ; *n* R	Cursor position report.	X	X	X	
Ec[*n* D	Cursor left.	X	X	X	
Ec[*n* B	Cursor down.	X	X	X	
Ec[*n* C	Cursor right.	X	X	X	

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
Ec[*n* ; *n* H	Position cursor.	X	X	X	
Ec[*n* A	Cursor up.	X	X	X	
Ec[*n* P	Delete character.	X	X	X	
Ec[*n* M	Delete line.	X	X	X	
Ec[n	Request Cursor Position	X	X	X	
Ec[*n* J	Clear display.	X	X	X	
Ec[*n* K	Clear line.	X	X	X	
Ec[*n* ; *n* f	Cursor position.	X	X	X	
Ec[*n* L	Insert line.	X	X	X	
Ec[4h	Insert character mode on.	X	X	X	
Ec[4l	Insert character mode off.	X	X	X	
Ec[= *n* h	Select screen type.			X†	
Ec[=7h	Wrap at end of line.	X	X	X	
Ec[?7h	Wrap at end of line.	X	X	X	
Ec[= *n* l	Deactivate screen type.			X†	
Ec[=7l	Reset wrap at end of line.	X	X	X	
Ec[?7l	Reset wrap at end of line.	X	X	X	
Ec[*n* U	Next page.	X	X	X	
Ec[*n* V	Previous page.	X	X	X	
Ec[u	Restore cursor position.	X	X	X	
Ec[s	Save cursor position.	X	X	X	
Ec[*n*T	Roll down.		X		
Ec[0m	Turn off attributes.	X	X	X	
Ec[1m	Bold on.	X	*		
Ec[4m	Underline on.	X	X		
Ec[5m	Blink on.	X	*	X	

Sequence	Description	Term0	Integral PC	The Portable	Windows/ 9000
Ec[7m	Inverse video on.	X	X	X	
Ec[10m	Load base font.	X	X	X	
Ec[11m	Load alternate font.	X	X	X	
Ec[12m	Load third font.			X	
Ec[*n* @	Roll left.		X		
Ec[*n* A	Roll right.		X		
Ec[*n*S	Roll up.		X		

* The Integral Personal Computer maps bold and half-bright enhancements to inverse video. It also maps blinking to underline.

† The Portable supports screen types 2 (80 by 25, black and white), 3 (80 by 20, color), 4 (graphics mode), and 8 (80 by 16).

‡ Windows 9000 ignores the blinking enhancement and maps half-bright to color pair 3.