WRQ reflection®

TERMINAL REFERENCE MANUAL FOR HP HOSTS

Windows® 2000
Windows NT® 4.0
Windows Me
Windows 98
Windows 95
Windows Terminal Server Ed.
Citrix® MetaFrame™

English

VERSION 9.0

Copyright

Reflection *Terminal Reference Manual for HP Hosts*
Version 9.0
September 2001

Trademarks

Customer Service

WRQ Corporate Headquarters
1500 Dexter Avenue North
Seattle, WA 98109 USA
+1.206.217.7100
+1.206.217.0293 FAX
800.872.2829

European Headquarters
The Netherlands
+31.70.375.11.00
+31.70.356.12.44 FAX

Asia Pacific Headquarters
Singapore
+65.336.3122
+65.336.5233 FAX

Technical Support in the USA

WWW: support.wrq.com
Anonymous FTP Server: ftp.wrq.com
Technical Support: 206.217.7000
For Partners of WRQ visit: www.wrq.com/bp/

Technical Support Outside the USA

Please contact your WRQ Reseller: visit www.wrq.com/bp/, or call WRQ for the name of the reseller nearest you. You can also send an e-mail to wrqbp@wrq.com.

Technical Documentation

Visit the following web site to download the PDF (Portable Document Format) version of this and other WRQ manuals: support.wrq.com/manuals/.

We welcome suggestions on how to improve our printed and online documentation.
Send your comments to docs@wrq.com.

At WRQ, we are committed to using products that conserve the world’s resources. Therefore, the printed version of this manual uses recycled, elemental chlorine-free paper with 20% post-consumer waste. Printed in the USA.

TABLE OF CONTENTS

SECTION 1
Introduction

SECTION 2
HP Escape Sequences and Character Sets

SECTION 3
Emulating a VT Terminal

SECTION 4
Programming Sequence Index

SECTION 1

Introduction

Overview of Reflection

Reflection establishes and maintains communications between your PC running Microsoft Windows 95, Windows 98, or Windows NT 4.0, and a host computer. With Reflection for HP with NS/VT, your PC *emulates,* or operates like, an HP 70098 or HP 2392A terminal. This allows you to communicate with the host just as if you were using a terminal, and maintain a host connection while running other Windows applications. Reflection can communicate with the HP 1000, 3000, 9000, or UNIX system (including ULTRIX).

If you need to work with a host application that sends Digital escape sequences, you can con-figure Reflection to emulate a VT-series terminal. This type of emulation is ideal for making online transactions with a DEC mainframe, and for dialing into public data networks, bulletin boards, or electronic mail systems. Operating Reflection for HP with NS/VT in VT-supported mode also allows you to run HP applications that have embedded VT control sequences, such as Oracle and UDMS.

This section provides an overview of Reflection’s features.

Reflection Features

Using Reflection for HP with NS/VT you can:

- Execute HP escape sequences that cause Reflection to perform certain actions, such as move the cursor, add a line of text, assign display enhancements, and change character sets. See the section starting on page 9.
- Emulate a VT-series terminal. See the section starting on page 79.
- Transfer files between your PC and HP 3000, VAX/VMS, and UNIX-based systems, either for use by a host program or so another PC user can have access to them. Lengthy transfers can continue while you work in another window.
- Use multiple copies of Reflection to access hosts over different data paths, via serial and network connections. Both wide and local area networks are supported.
- Configure Reflection's toolbar to transmit text strings to the host, execute single commands, run Reflection scripts and execute macros, or act as host-writable softkeys.
- Remap your keyboard. This lets you assign any keystroke to a Reflection or host function.
- Save information from the host to a disk file.
- Create Reflection macros using Reflection and Visual Basic for Applications.
- Save Reflection macros to settings files that customize a Reflection session.

Who Should Use This Manual

This manual is intended for those who need to control Reflection for HP with NS/VT programmatically; that is, by writing programs and sending escape sequences directly to Reflection. Normally, it is the host that controls Reflection (via escape sequences), and this process is invisible to the user.

About Reflection Macros and Reflection Basic in Version 7.0

The term Reflection Basic describes the programming language and programming tools that shipped with Reflection prior to version 7.0. Reflection continues to support this programming language as well as Visual Basic for Applications.

The Reflection Basic language includes Reflection-specific methods and properties that are part of Reflection's OLE automation support (such as **Connect** and **Caption**), and Basic language functions and statements (such as Dir$ and For... Next). The methods and properties you use in Reflection Basic scripts are identical to those used in newer Reflection macros created using Visual Basic.

Where To Go From Here

The Reflection *Terminal Reference Manual for HP Hosts* is organized into the following sections:

Section 1, "Introduction"
The section you're reading now provides an introduction to Reflection's features, and instructs you on what you should know before proceeding.

Section 2, "HP Escape Sequences and Character Sets"
Contains a comprehensive description of supported sequences when using Reflection for HP with NS/VT to emulate an HP-series terminal.

Section 3, "Emulating a VT Terminal"
Describes how to configure Reflection for HP with NS/VT to emulate a VT-series terminal (referred to as VT emulation mode). Then, a description follows of sequences supported by Reflection.

Section 4, "Programming Sequence Index"
A conclusive summary of all the sequences contained in sections 2 and 3, allowing you to quickly locate information and page references for specific sequences.

SECTION 2

HP Escape Sequences and Character Sets

HP Escape Sequences

An escape sequence is a command sent to a terminal to perform an operation. For example, escape sequences can make Reflection do such things as move the cursor, add a line of text, assign display enhancements, and change character sets. Typically, escape sequences are sent to Reflection from a host program, but they can also be entered from the keyboard.

Note: To create escape sequences in the terminal window, Reflection must be in local mode or be connected to a host which echoes your characters.

All escape sequences begin with the ASCII escape character `Esc` (decimal 27). You can create the escape character in one of two ways:

Within the Terminal Window	Press the Esc key (no characters are displayed).
From the Reflection Command Line	Type `Transmit Chr$(27) &` followed by the quoted sequence. (This syntax uses the ANSI character table to equate a numeric expression to a character. In this example, the integer 27 is the decimal equivalent of the escape character.)

For example, the sequence `Esc&w6f132X` sets the display to 132 columns. To enter this sequence do one of the following:

- In Reflection's terminal window, press Enter↵ and type `&w6f132X`.
- On the Reflection command line, execute the following:

  ```
  Display  Chr$(27) & "&w6f132X"
  ```

  See the Reflection Programming Reference online help (Rwinprog.hlp) for more information on either the Transmit or Display commands.

- You can also store escape sequences in a button on Reflection's toolbar or in remapped keys using keyboard remapping, as described in the Reflection documentation.

Note: Some escape sequences (such as the Esc&p sequences) will cause your keyboard to lock. The keyboard remains locked until the host sends a DC1 character (XON), indicating that it is ready to receive a status code regarding the success or failure of the sequence. This is normal behavior. If the keyboard remains locked, a soft reset (see page 20) will unlock it.

Sequence Conventions

When coding escape sequences, take care to distinguish between upper and lowercase characters; in general, escape sequences are case sensitive. Further, if an escape sequence consists of more than one character and ends with a letter, the final letter must be capitalized. Reflection does not process the escape sequence until it receives an uppercase letter. The control codes are represented by Esc, CR, LF, etc.

Primary Terminal Modes

This chapter refers to many different modes that Reflection can run in; the primary terminal modes are defined here.

Local mode	Data entered from the keyboard is displayed on the screen, but is not transmitted to the host. The escape sequence for entering local mode is Esc&kØR.
Remote mode	When you have an active host connection, data entered from the keyboard is transmitted to the host and data received from the host is displayed on the screen. The escape sequence for entering remote mode is Esc&k1R.
Character mode	Active in remote mode. Data is transmitted to the host, one character at a time. This is one of the two primary modes. The escape sequence for entering character mode is Esc&kØB.
Block mode	Active in remote mode. Data is transmitted to the host in blocks; selectable as one line per block, or as one page per block. This is one of the two primary modes. The escape sequence for entering block mode is Esc&k1B.

Line mode	Active in remote block mode. Data is transmitted to the host in blocks, each consisting of one line. The escape sequence for entering line mode is `ESC&sØD`.
Page mode	Active in remote block mode. Data is transmitted to the host in blocks, consisting of all data in display memory. The escape sequence for entering page mode is `ESC&s1D`.
Format mode	Controls input and formatting of data entered into data entry forms, and enables restriction of entering data to selected fields on the form. The escape sequence for entering format mode is `ESCW`.

Reflection-Specific Escape Sequences

When you use Reflection and a PC instead of a terminal, there are some escape sequences that are unique to Reflection and not available on a terminal. For example, because Reflection adds typeahead features to your terminal, there is an escape sequence to clear the typeahead buffer. Following is a description of Reflection-specific escape sequences.

`ESC&bR`	Reset data communications.
`ESC&oG<command>CR`	Run the specified Reflection commands.
`ESC&oX`	Clear contents of typeahead buffer.
`ESC.`	If you're using an XL or MPE/iX host, use this sequence to enable network typeahead for NS/VT.
`ESC,`	If you're using an XL or MPE/iX host, use this sequence to disable network typeahead for NS/VT.

Configuration Control

In general, the escape sequences that begin with Esc&k and Esc&s are for configuration control, and have corresponding options on the tabs of Reflection's Terminal Setup dialog box (on the Setup menu, click Terminal to see this). These can be any length, as long as the last character is a capital letter. This lets you specify a number of parameters in a single sequence; for example, Esc&k0a0D turns off auto linefeed and the bell.

Sequences are listed below ending with a capital letter as if they are being issued alone.

Esc&k0] Disable the Select key sequence. The Enter key on the keypad reverts to the same function as the Enter↵ key on the main keypad.

Esc&k1] Enable the Select key sequence. The key is treated the same as a function key with the Transmit attribute, and sends Esc&P when pressed. You can use the Select key to easily select a field in a form or menu.

Esc&k0A Auto linefeed off.

Esc&k1A Auto linefeed on. A LF character is automatically appended to each CR character generated from the keyboard.

Esc&k0B Block mode off.

Esc&k1B Block mode on. Reflection waits until you press Enter before trans mitting any characters to the host. This lets you type and edit a block of data (hence the name), then transmit the block all at once. The size of each block can vary from just a few characters to several pages of data. ASCII control codes such as CR and LF work locally, but are not usually transmitted with the block.

Esc&k0C Caps lock (TeleType) off.

Esc&k1C Caps lock (TeleType) on. The characters produced from the keyboard are limited to 37 Teletype-compatible codes:

- No lowercase alphabetic characters are generated. All alphabetic keys appear as if they are shifted.
- The ~ (tilde) and ` (grave accent) are disabled.
- The characters {, |, and } are converted to the characters [, \, and].

Esc&k0D Margin bell off.

Esc&k1D Margin bell on.

ESC&kØI Set 7 data bits and 1 parity bit.

ESC&k1i Set 8 data bits.

ESC&kØK Auto keyboard lock off.

ESC&k1K Auto keyboard lock on. When Reflection is connected to a packet switching network (using the X.25 protocol) via a controller, it is necessary to ensure that the packet sent is received and acted upon before another is sent from Reflection. To do this, the keyboard must automatically lock so that it can only be unlocked by the receiving host.

ESC&kØL Local echo off.

ESC&k1L Local echo on. Each character typed at the keyboard is immediately displayed on the screen. In remote mode, each character you type is transmitted to the host computer. Most host systems immediately send the character back to your terminal—that is, they *echo* the character—and the character you see is the one that the host sent, not the one you typed. Therefore, when local echo is on *and* you are in remote mode, each character is sent two places: directly to your display and to the host computer.

ESC&kØM Modify all mode off.

ESC&k1M Modify all mode on. When Reflection is in character mode and remote mode, you can change existing text on the screen and retransmit the modified lines to the host by pressing Enter↵. This can save a lot of retyping.

ESC&kØN SPOW (SPace OverWrite) latch off.

ESC&k1N SPOW (SPace OverWrite) latch on. See page 15 for the sequence that turns SPOW on.

ESC&kØP Caps mode off.

ESC&k1P Caps mode on. All unshifted alphabetic keys generate uppercase letters and all shifted alphabetic letters generate lowercase letters. This only effects the 26 alpha-betic keys.

ESC&kØR Remote mode off (that is, places Reflection into local mode). You can edit Reflection's display memory without sending data to the host computer. Char-acters entered from the keyboard are not transmitted to the host. If the PC is connected to the host while in local mode, Reflection discards any data received from the host.

Esc&k1R Remote mode on. Reflection sends the host everything typed from the keyboard, and the host then sends each character back. As Reflection receives characters, it displays them in Reflection's terminal window.

Esc&kØ[Smooth scroll off.

Esc&k1[Smooth scroll on. Lines scroll smoothly on the display, which is slow
enough

to be readable without having to pause the display.

Esc&qØL Configuration commands unlocked.

Esc&q1L Configuration commands locked.

Esc&sØA Transmit functions off.

Esc&s1A Transmit functions on. The table below indicates the set of non-ASCII
keys

and key combinations which will be transmitted to the host when Reflection is in character and remote modes:

Single keys	Key combinations	
NumLock	Alt+J	(clear display)
ScrollLock	Alt+K	(clear line)
Home	Alt+D	(delete line)
↑	Alt+I	(insert line)
PgUp	Alt+Y	(Reflection command window)
	Alt+L	(Reflection command line)
←	Ctrl+End	
→	Ctrl+↑	
End	Ctrl+↓	
↓	Ctrl+←	
PgDn	Ctrl+→	
Insert	Ctrl+PgUp	

Single keys	Key combinations
Delete	Ctrl+PgDn
CapsLock	Ctrl+Home

Esc&sØB SPOW off. The Spacebar overwrites and erases existing characters.

Esc&s1B SPOW on. Spaces entered from the keyboard—not spaces echoed from the host—move the cursor over existing characters, but do not overwrite them with spaces:

- The SPOW latch is turned on by a carriage return.
- The SPOW latch is turned off by a linefeed, tab, or home up.

See page 13 for the SPOW latch sequence.

Esc&sØC Inhibit end-of-line wrap off. Reflection automatically returns the cursor to the left margin in the next line when the cursor reaches either the right margin or the right screen edge.

Esc&s1C Inhibit end-of-line wrap on. The cursor is not automatically advanced when you reach the right margin. As you type additional characters, each one overwrites the character at the right margin until you explicitly move the cursor by pressing Enter↵ or using an arrow key.

Esc&sØD Line mode on.

Esc&s1D Page mode on.

Esc&sØG Inhibit handshake off; that is, DC1 handshaking is active.

Esc&s1G Inhibit handshake on; that is, DC1 handshaking is inhibited.

Esc&sØH Inhibit DC2 off; that is, the DC2 handshake is active.

Esc&s1H Inhibit DC2 on; that is, DC2 handshaking is inhibited.

Esc&sØZ Parity checking off. The parity bit on received characters is ignored.

Esc&s1Z Parity checking on. When parity is set to anything other than *8/None* or *7/None*, Reflection generates a parity bit for all outgoing characters. When a parity error is found, the character is replaced with an ASCII DEL character.

Display Control

Most HP terminals support 16 combinations of display enhancements, including blinking, inverse video, underline, and half bright intensity.

Display Enhancements

Esc&d<x> Begin display enhancements, where <x> is a character from the table below.

<x>	Display Enhancement
@	No enhancement
A	Blinking
B	Inverse video
C	Blinking, inverse video
D	Underline
E	Blinking, underline
F	Inverse video, underline
G	Blinking, inverse video, underline
H	Half bright
I	Half bright, blinking
J	Half bright, inverse video
K	Half bright, blinking, inverse video
L	Half bright, underline
M	Half bright, blinking, underline
N	Half bright, inverse video, underline
O	Half bright, blinking, inverse video, underline

For example, to start typing characters that are blinking in inverse video:

1. Press Esc.
2. Type &dC. The enhancements take effect immediately.
3. Type some text: it appears as blinking inverse video.
4. To set the text back to normal, press Esc and type &d@.

Esc&ds<x> Add a security enhancement to the display enhancements listed above for the sequence Esc&d<x>. With the security enhancement, characters do not appear on the display.

Screen Blanking

Esc&w12F Turn on display.

Esc&w13F Turn off display (display memory is not cleared); this excludes the function key labels. This is known as "screen blanking."

Display Control Operations

EscS Scroll display up one line.

EscT Scroll display down one line.

EscU Page down.

EscV Page up.

Terminal Control

Terminal control escape sequences let you or a program control terminal operations.

Esc@ Delay one second. Multiple use of this sequence in succession can be used to obtain virtually any desired time delay.

Esc[Start an unprotected field.

Esc] End an unprotected or transmit-only field.

Esc{ Start a transmit-only field.

Esc^ Primary status request. See page 37.

Esc~ Secondary status request. See page 40.

Esc` Sense cursor position, relative to the current display. When the host computer sends the escape sequence Esc` (decimal 96). Reflection responds with a screen relative cursor positioning escape sequence. See the information starting on page 25 for information on cursor positioning escape sequences.

Esc0 Home cursor and copies display memory to destination device(s).

EscE A hard reset does everything that a soft reset does, plus the following:

- Emits a beep.
- Unlocks the keyboard (if it was locked).
- Initializes the serial communications port to the last activated values (if you’re using the serial port) and clears the receive buffer.
- Clears typeahead and keyboard buffers.
- Transmits an XON if **Receive** (under **Pacing**) is set to **Xon/Xoff**.
- Sets all configuration parameters that were set with escape sequences to their last activated values.
- Clears display memory and homes the cursor.
- Turns off SMOOTH SCROLL, MEMORY LOCK, and DISPLAY FUNCTNS.
- Sets the margins and columns to 1 and 80.
- Clears all tab stops.

- Turns the following items off:

  –Format mode

  –Insert character mode

  –Caps Lock

- Turns off any active logging mode.
- Clears the printer buffer and sends a reset command to the printer.
- Displays the function key labels selected as the Function key set on the Function Keys tab in the Terminal Setup dialog box.
- Sets the user keys to their last activated values.

Hard reset does *not* change the following modes keys:

F2 MODIFY ALL
F3 BLOCK MODE
F4 REMOTE MODE
F8 AUTO LF

Note: Because macro execution is terminated, you should not use this escape sequence as part of a **Display** method in a Reflection macro.

`EscW` Turn on format mode.

`EscX` Turn off format mode.

`EscY` Turn on display functions. Reflection displays most of the ASCII control characters on the screen as if they were normal text, but it does not execute them. You can see exactly what characters are received from the host, and what characters are generated by the keyboard.

`EscZ` Turn off display functions.

`Esca` Sense cursor position, absolute. See page 25.

`Escb` Unlock the keyboard.

`Escc` Lock the keyboard.

`Escd` In block mode, transmit a block of text from memory.

`Escf` Disconnect (drops DTR). With a LAN connection, disconnect closes the current connection.

`Esc g` A soft reset does the following:

- Emits a beep.
- Unlocks the keyboard (if it was locked).
- Turns Display Functions off (if it was on).
- Initializes the serial communications port to the last activated values (if you're using the serial port) and clears the receive buffer.
- Clears typeahead and keyboard buffers.
- Transmits an XON if **Receive** (under **Pacing**) is set to **Xon/Xoff**.

If you are using the WRQ/Reflection file transfer protocol, you cannot perform a soft reset during a file transfer.

Note: The Reset cascading command (on the Connection menu) is dimmed when the Reflection **Quit** method is set to Yes.

`Esc l` Turn on memory lock mode. This fixes specified lines of data on the screen, letting you move a block of data within display memory.

When enabled, the *R* parameter in the cursor position escape sequences (see page 22) is ignored. The *Y* parameter in the sequences, however, is always accepted.

`Esc m` Turn off memory lock mode.

`Esc z` Initiate terminal self-test (always successful).

`Esc &kØ\` Switch to HP terminal class. This sequence reactivates the last saved or activated values for HP terminal class.

`Esc &k1\` Switch to VT terminal class. This sequence reactivates the last saved or activated values for VT terminal class.

Editing Display Memory

ESCJ	Clear from cursor position to end of display memory.
ESCK	Clear from cursor position to end of line.
ESCL	Insert line.
ESCM	Delete line.
ESCN	Insert character with wraparound. This works the same as insert character except that characters forced beyond the right margin are not lost.
ESCO	Delete character with wraparound.
ESCP	Delete character.
ESCQ	Start insert character mode.
ESCR	End insert character mode.

Cursor Positioning

The escape sequences listed below control cursor positioning. Columns and rows are numbered from 0, starting at the top leftmost column and row.

ESCh	Home cursor.
ESCi	Perform backtab.
ESCA	Cursor up one line.
ESCB	Cursor down one line.
ESCC	Cursor right one character.
ESCD	Cursor left one character.
ESCF	Cursor home down.
ESCG	Move cursor to the left margin.
ESCH	Home cursor. This performs the same function as ESCh, except that it stops in initial transfer-only fields.
ESCI	Horizontal tab.
ESC&xØC	Clear send cursor position mode.

`EsC&x1C` Set send cursor position mode. When this mode is set and Enter or a transmit-only user key is pressed, the absolute cursor position escape sequence is added to the beginning of the transmitted block.

Cursor Positioning by Row and Column

The cursor can be placed anywhere in Reflection's display memory by an escape sequence issued from the host computer or from the keyboard. There are four ways to specify cursor position:

- Absolute
- Screen relative
- Display relative
- Cursor relative

In the cursor-positioning escape sequences, row and column numbers are always expressed relative to zero; the first row and column position is row 0, column 0 (at the top left of the display). The row and column numbers within the escape sequences are expressed as ASCII decimal digits. Spaces preceding or following a number are ignored.

The column and row commands can be switched, in which case the C would be in uppercase, and the *Y* or *R* in lowercase.

Note: When memory lock is on (see page 20), the R parameter in the cursor position escape sequences that follow is ignored. The Y parameter, however, is always accepted.

Absolute Cursor Positioning

`EsC&a<col>c<row>R`

Moves the cursor to <*col*> and <*row*>, relative to the the first row (row 0) of display memory.

If the new position is outside the bounds of the current screen, the display is scrolled in the direction required to bring the target position onto the screen.

Only the <*row*> portion of the escape sequence is absolute, since column numbers are always relative to column 0.

The maximum value of <*col*> is either 79 or the right margin column, whichever is greater. If the specified value of <*col*> is greater than the limit, the limit is used instead of the specified value. This prevents the cursor from being placed in a column that is not accessible by horizontal scrolling.

The maximum allowable <*row*> value is 32767. If the <*row*> value is larger than Reflection's display memory can hold, then rows are deleted off the top of display memory and added to the bottom until the cursor has been moved the number of rows indicated by the escape sequence.

The escape sequence can take any of the following forms:

```
Esc&a<col>c<row>R
Esc&a<row>r<col>C
Esc&a<row>R
```

For example, the following escape sequence moves the cursor to the 103rd column of the 1451st row of display memory:

```
Esc&a102c1450R
```

Screen Relative Positioning

```
Esc&a<col>x<row>Y
```

Moves the cursor to <*col*> and <*row*>, relative to the upper left-corner of the currently displayed screen.

If only a row is specified, the cursor is moved to the current column in the specified row. If only a column is specified, the cursor is moved to the specified column in the current row. If the value of either row or column is so large that it implies moving the cursor off the screen, the maximum on-screen value is used instead of the specified value. In other words, if you try to place the cursor in column 85 using this form, Reflection places it in column 79 (relative to zero).

The escape sequence can take any of the following forms:

```
Esc&a<col>x<row>Y
Esc&a<row>y<col>X or Esc&a<row>y<col>C
Esc&a<row>Y
Esc&a<col>X or Esc&a<col>C
```

For example, the following sequence places the cursor in the 12th screen row and the 10th column:

```
Esc&a11y9X
```

Display Relative Positioning

```
Esc&a+/-<col>c+/-<row>R
```

Moves the cursor *<col>* columns (left or right) and *<row>* rows (up or down), relative to the current cursor position. The direction may be either a “+” or “–” sign.

When positioning is specified relative to the current display memory position, the display rolls in whatever direction is required to bring the new position onto the screen window.

The following escape sequence moves the cursor up 2 rows and right 7 columns from its current position:

```
Esc&a-2r+07C
```

Cursor Relative Positioning

```
Esc&a+/-<col>x+/-<row>Y
```

Moves the cursor *<col>* columns (left or right) and *<row>* rows (up or down), relative to the current cursor position. Positioning is limited to screen size. The direction may be either a “+” or “–” sign. You can also use the sequence `Esc&a+/-<col>c+/-<row>Y`.

When positioning is relative to the current cursor position, the display rolls down if necessary to bring the specified row onto the top of the screen, but if the specified row is below the bottom of the screen, the cursor moves to the last row and the display does not roll.

Combining Cursor Positioning Methods

You can use any combination of screen-relative, absolute, and cursor-relative cursor positioning. For example, use the following escape sequence to move the cursor to the 15th column of absolute row 52 in display memory:

```
Esc&a14c51R
```

The following escape sequence moves the cursor to the character 9 columns to the right of the current cursor position in the 2nd row presently visible on the screen:

```
Esc&a+9c1Y
```

Cursor Sensing

The host computer can request and receive the current cursor position. The position can be expressed relative to the current screen or relative to the first row in display memory.

The cursor sensing response is a block transfer, so the rules of block transfers apply.

Screen-Relative Sensing

When the host computer sends the escape sequence `Esc`` (decimal 96), Reflection responds with a cursor positioning escape sequence of the form `Esc&a<col>c<row>Y`, where <*col*> and <*row*> are both three-digit decimal numbers. The maximum value of <*col*> is 131; the maximum value of <*row*> is 023.

Absolute Sensing

When the host computer sends the escape sequence `Esca`, Reflection responds with a cursor positioning escape sequence, which takes the form `Esc&a<col>c<row>R`, where <*col*> and <*row*> are both three-digit decimal numbers. The maximum value of <*row*> is 999; the maximum value of <*col*> is 131. Note that the actual row and column numbers may be larger than 999; in that case, Reflection reports only the three low-order digits.

`Esc*dQ`	Make text cursor visible.*
`Esc*dR`	Make text cursor invisible.*
`Esc*dØQ`	Underline cursor.*

* This sequence is not available when emulating an HP 2392A terminal.

`␛*d1Q`	Block cursor.*
`␛*dØE`	Normal video.*
`␛*d1E`	Inverse video.*
`␛&w6f80X`	Set 80 display columns.
`␛&w6f132X`	Set 132 display columns.

User Key Control

See page 30 for information on how to define user keys programmatically.

`␛p`	Default definition of user key F1.
`␛q`	Default definition of user key F2.
`␛r`	Default definition of user key F3.
`␛s`	Default definition of user key F4.
`␛t`	Default definition of user key F5.
`␛u`	Default definition of user key F6.
`␛v`	Default definition of user key F7.
`␛w`	Default definition of user key F8.

User Key Definition and Display

`␛j`	When executed locally, this displays the Function Keys tab of the Terminal Setup dialog box. When transmitted by the host, this displays the user key definition screen in HP terminal style.
`␛k`	Close the Terminal Setup dialog box.

Setting Margins and Tabs

`Esc1` Set a tab stop at the current cursor position.

`Esc2` Clear the tab stop at the current cursor position.

`Esc3` Clear all tab stops.

`Esc4` Set the left margin at the current cursor position.

`Esc5` Set the right margin at the current cursor position.

`Esc9` Reset the margins to their default values (left margin at 1 and right margin at 80).

Printer and Data Operations

The following escape sequences control printing and data transfer.

`Esc&kØS` Enable regular (80-column) printing.

`Esc&p<x>p<y>u14c` Enable regular (80-column) printing.

`Esc&k2S` Enable compressed (132-column) printing.

`Esc&p<x>p<y>u16c` Enable compressed (132-column) printing.

`Esc&p[<a>d]<b>D` Select one or more destination devices <*a*> and <*b*>. The display is device 3; an external device (typically a printer) is device 4.

`Esc&p[<a>d][<b>d]<Y>` Copy <*Y*> amount of data to one or more destination devices <*a*> and <*b*>. The display is device 3; an external device (typically a printer) is device 4. The current device(s) is used as the destination if none is specified. The amount of data, <*Y*>, can be:

M Copy from the cursor line to the end of display memory. Equivalent to COPY ALL from the device control keys.

F Copy from the cursor line to the end of the display screen. Equivalent to COPY PAGE from the device control keys.

B Copy the cursor line. Equivalent to COPY LINE from the device control keys.

`Esc&p<x>^`

Device status request. <x> can be either 4 or 6; Reflection returns its device status response. See page 46.

`Esc&p<x>p<y>u<z>C`

Perform action <*z*> on device <*y*>. <*y*> can be either 4 or 6 for a printer. <*z*> can be one of the following:

0	Generate a form feed
1	Skip <x> lines
2-10	Generate a form feed
11	Turn on log bottom (this locks the keyboard)
12	Turn on log top
13	Turn off either logging mode (this locks the keyboard)
14	Standard (80-column) printing enabled
16	Compressed (120-column) printing enabled

If the action is *skip line*, an <*x*> can specify the number of lines to skip. If the action is *record mode*, an <*x*> can define the character (decimal 0–127) that turns off record mode.

`Esc&p14C`

Enable regular (80-column) printing.

`Esc&p16C`

Enable compressed (132-column) printing.

`Esc&p<x>p2ØC`

Enable record mode. Data from the host is passed through to the “to” devices without appearing on the screen.

The value of <*x*> (the decimal equivalent of an ASCII character in the range 0–127) defines the character that turns off record mode. When record mode is switched off (either from the terminal or upon recognizing a terminator), Reflection transmits *S* (Success) or *F* (Failure).

`Esc&p[<a>d][<b>d]<x>W<data string>`

Transfer <*x*> number of bytes of the <*data string*> to the destination devices <*a*> and <*b*>. <*x*> is a decimal value from 1 to 256, and the transfer is in binary form (and the string can contain non-ASCII characters). The current device(s) is used as the destination if none is specified.

`Esc&p[<a>d][<b>d]W<data string>`

Transfer <*data string*> to the destination devices <*a*> and <*b*>. The string is transferred in ASCII form, and ends either with the 256th character or an ASCII linefeed character. The current device(s) is used as the destination if none is specified.

Function Keys and Message Operations

The following escape sequences control whether the function key labels appear at the bottom of Reflection's terminal window, and which set appears.

`Esc&f<parameters>` Define user keys (see "Defining User Keys Programmatically" on page 30 for detailed instructions).

`Esc&f<x>E` Trigger function key <*x*>. When received, Reflection acts exactly as if the function key <*x*> had been pressed. Only the current user function keys are triggered.

`Esc&f-1E` Transmit HP Enter to the host.

`Esc&j@` Enable the user keys but remove the on-screen labels.

`Esc&jA` Display the modes keys.

`Esc&jB` Display the user key labels.

`Esc&jC` Clear any status messages.

`Esc&j<n>D` Select additional features of function keys (not available with HP 2392A emulation):

<n>	**Bell rings**	**CR sent**	**Labels restored**
0	No	No	Yes
1	Yes	No	Yes
2	No	Yes	No
3	Yes	Yes	No

`EsC&j<x>L<string>` Replace the screen labels with a message on the bottom two lines of the screen, where <*x*> is a number from 0 to 160, and <*string*> is a message of <*x*> number of characters.

`EsC&jR` Enable the function key labels.

`EsC&jS` Disable the function key labels; this includes the modes and user key labels.

Defining User Keys Programmatically

The host system can send an escape sequence to Reflection to define a user key. The sequence can also be entered from the keyboard or from a Reflection macro.

This escape sequence takes the following form:

```
EsC&f<attr>a<key>k<label length>d<string length>l<label><string>
```

Up to 80 characters or 160 diacritical combinations can be loaded into a user key from the host.

The following table defines the parameters in the user key definition escape sequence. The parameters `<attr>`, `<key>`, `<label length>`, and `<string length>` may appear in any order. However, the letter that identifies the last parameter must be uppercase, and all the preceding identifiers must be lowercase.

User Key Escape Sequence Parameters

Parameter	Values	Default
`<attr>a`	0 (Normal) 1 (Local) 2 (Transmit) 3 (Command)	0
`<key>k`	Key number, 1–8	1
`<label length>d`	0–16	0
`<string length>l`	0–80 -1 erases the field definition	1
`<label>`	A string of exactly <label length> characters	
`<string>`	A string of exactly <string length> characters	

If a value of 0 is used for the <*label length*> or <*string length*>, then the current contents of the label or string are left unchanged. The <*label*>, if its length is not zero, must precede the <*string*>.

The following examples show two equivalent escape sequences that assign the string `HELLO MGR.DEV` to user key F8. The attribute is Transmit, and the label reads `LOG ON`. After entering the sequence, choose the **User** option from the **Function key set** list (on the Function Keys tab in the Terminal Setup dialog box) to display the new labels (even if the user keys are currently displayed, you'll need to select the **User** option to refresh the labels). The hyphens represent spaces and should not be entered. Spaces are required to properly position the label and string.

```
EsC&f2a8k16d13L---LOG-----ON---HELLO-MGR.DEV
EsC&f13l8k2a16D---LOG-----ON---HELLO-MGR.DEV
```

A Reflection macro can be used to quickly reload user keys. Following is a sample macro that loads three user keys, then displays and enables them (again, each hyphen represents a space for positioning):

```
Display & "^[&f1k2a16d20L---LOG-----ON---HELLO-MYNAME,MGR.DEV
Display & "^[&f2k2a8d8L---MAIL-READMAIL"
Display & "^[&f3k3a8d6L---DIR--DIR C:"
Display & "^[&f4k0a8d5L--LISTF-LISTF"
Display & "^[&jB"
```

Once a user key is defined, the host can display the user key definition screen, read the key definitions, then remove the screen. Enter the escape sequences to put Reflection into block mode, page mode, home up, and request a block transfer.

Reflection responds by sending one EsC&f... escape sequence for each of the eight user keys. This capability is used by some host programs to save the current user key definitions before changing them, so they can be reloaded with their original definitions when the program ends.

Forms Cache (HP 700/94 Only)

See “Using Forms Cache” on page 52 for more information about forms cache.

`ESc&p9u<form#>pF`	Display a form.
`ESc&p9u<form#>pL`	Purge a form from the buffer.
`ESc&p9u<form#>p<length>L<contents>`	Store a form of known length.
`ESc&p9u<form#>p<<contents>>L`	Store a form of unknown length. The innermost set of angled brackets is literal; be sure to enclose the <contents> parameter in angled brackets.
`ESc&p9u<name>n<form#>p<length>L<contents>`	Store a form specified by a form name of known length.
`ESc&p9u<name>n<form#>p<<contents>>L`	Store a form of unknown length, specified by a form name. The innermost set of angled brackets is literal; be sure to enclose the <*contents*> parameter in angled brackets.

Forms Buffer Status

`ESc&p9^`	Request the status of the forms buffer.
`ESc&p<form#>p9^`	Request the status of the specified forms buffer.
`ESc&p<<name>>n9^`	Request the status of the named forms buffer. The innermost set of angled brackets is literal; be sure to enclose the <name> parameter in angled brackets.
`ESc&p <>n9^`	Return a list, or directory, of the forms currently stored in cache memory. This is known as reading the forms cache directory.

Alternate Character Set Selection

`Esc)B` Select the line drawing character set as the alternate set (this set is used as an alternative to the base character set, which by default is Roman 8).

`Esc)@` Select the base character set as the alternate character set.

Restricted (HP 700/94) Escape Sequences

The following escape sequences require that the **HP 70094**, **HP 70098** option in the **Terminal type** group box (on the Terminal Type tab of the Terminal Setup dialog box) be selected.

`Esc{` Start a transmit-only field.

`Esc&kØZ` Turn modified data tags off.

`Esc&k1Z` Turn modified data tags on.

`Esc&q<m>teØ{ØA` Set Transmit Functions off (<*m*>= 4–7).

`Esc&q<m>teØ{1A` Set Transmit Functions on (<*m*> = 4–7).

`Esc&q<m>teØ{ØB` Set SPOW off (<*m*> = 4–7).

`Esc&q<m>teØ{1B` Set SPOW on (<*m*> = 4–7).

`Esc&q<m>teØ{ØC` Set Inhibit end-of-line wrap off (<*m*> = 4–7).

`Esc&q<m>teØ{1C` Set Inhibit end-of-line wrap on (<*m*> = 4–7).

`Esc&q<m>teØ{ØD` Set Line mode (<*m*> = 4–7).

`Esc&q<m>teØ{1D` Set Page mode (<*m*> = 4–7).

`Esc&q<m>teØ{ØG` Set Inhibit Handshake off (<*m*> = 4–7).

`Esc&q<m>teØ{1G` Set Inhibit Handshake on (<*m*> = 4–7).

`Esc&q<m>teØ{ØH` Set Inhibit DC2 off (<*m*> = 4–7).

`Esc&q<m>teØ{1H` Set Inhibit DC2 on (<*m*> = 4–7).

`Esc&q<m>teØ{ØN` Set Esc Xfer to printer off.

`Esc&q<m>teØ{1N` Set Esc Xfer to printer on.

`Esc&q<m>te1{ØA` Turn Auto Linefeed off (<*m*> = 4–7).

`Esc&q<m>te1{1A` Turn Auto Linefeed on (<*m*> = 4–7).

`Esc&q<m>te1{ØB`	Turn Block mode off (<*m*> = 4–7).
`Esc&q<m>te1{1B`	Turn Block mode on (<*m*> = 4–7).
`Esc&q<m>te1{ØC`	Turn CapsLock off (<*m*> = 4–7).
`Esc&q<m>te1{1C`	Turn CapsLock on (<*m*> = 4–7).
`Esc&q<m>te1{ØL`	Turn Local Echo off (<*m*> = 4–7).
`Esc&q<m>te1{1L`	Turn Local Echo on (<*m*> = 4–7).
`Esc&q<m>te1{ØM`	Turn Modify All mode off (<*m*> = 4–7).
`Esc&q<m>te1{1M`	Turn Modify All mode on (<*m*> = 4–7).
`Esc&q<m>te1{ØR`	Turn Remote mode off (<*m*> = 4–7).
`Esc&q<m>te1{1R`	Turn Remote mode on (<*m*> = 4–7).
`Esc&q8te1{<x>A`	Set Enter↵ (Return) definition's first character where <*x*> = 0–127.
`Esc&q8te1{<x>B`	Set Enter↵ (Return) definition's second character where <*x*> = 0 127.
`Esc&q8teØ{ØD`	Turn the bell off.
`Esc&q8teØ{1D`	Turn the bell on.
`Esc&q8te1{ØR`	Make Enter↵ (Return) key a simple carriage return, not equal to (keypad) Enter. The Reflection macro equivalent is `ReturnEqualsEnter=No`.
`Esc&q8te1{1R`	Make Enter↵ (Return) key equivalent to (keypad) Enter. The Reflection macro equivalent is `ReturnEqualsEnter=Yes`.
`Esc&q8te1{ØT`	Set TransmitTabAsSpaces off.
`Esc&q8te1{1T`	Set TransmitTabAsSpaces on.
`Esc&q<m>te2{<x>F`	Set field separator character where <*m*> = 4–7 and <*x*> = 0–127.

Esc&q<m>te2{<x>L Allocate the forms buffer size where <*m*> = 4–7 and <*x*> = 0–255.

Esc&q<m>te2{<x>R Set block terminator character where <*m*> = 4–7 and <*x*> = 0–127.

Esc&q<m>te2{ØZ Set Transmit to All (<*m*>= 4–7).

Esc&q<m>te2{1Z Set Transmit to Modified (<*m*>= 4–7).

Terminal Capabilities Status

The escape sequences below are sent from the host to request information about the terminal's capabilities. See page 42 for more information about terminal status requests.

Esc*s-1^ Text capabilities. See page 42.

Esc*s-3^ Memory available for downloading. See page 43.

Esc*s-4^ Interface capabilities. See page 44.

HP 700 Series Sequences

The following sequences require that the **HP 70092**, **HP70096** or **HP 70094**, **HP70098** option in the **Terminal type** group box be selected (on the Terminal Type tab of the Terminal Setup dialog box).

Esc&fØB Store current configuration entries, including function key labels, tab stops, margins, and user-definable key selections.

The following items are not stored:

Connection Parameters	Modes Keys
Baud Rate	Line Modify
Parity	Modify All
ENQ/ACK	Block Mode
Check Parity	Remote Mode
Receive Pacing	Smooth Scroll
Transmit Pacing	Memory Lock

Esc&f1B Restore configuration saved with the above escape sequence.

`Esc&f211P<!<x>>`
Configure the Tab key as Tab, Enter↵, or keypad Enter. Because the PC keyboards have separate Tab, Enter↵, and keypad Enter keys, Reflection does not process this sequence; it is only for compatibility with the HP 700/92 and 700/94 terminal.

`Esc&fR` Configure the Tab key as Tab, and the Enter↵ key as Enter↵; this is a non-volatile memory selection (it cannot be changed with a Hard Reset). Because the PC keyboards have separate Tab, Enter↵, and keypad Enter keys, Reflection does not process this sequence; it is only for compatibility with the HP 700/92 and 700/94 terminal.

`Esc&f1m149P<!154>`
Configure Enter↵ as keypad Enter; the angle brackets are a literal part of this equence.

`Esc&f1m149P<!149>`
Reset the Enter↵ key as Enter↵; the angle brackets are a literal part of this sequence. The sequence Esc&f1m149P<> also sets this configuration.

HP Status Requests

A program running on the host computer can request status and terminal information from Reflection. There are seven types of status requests:

- Terminal status (primary and secondary)
- Terminal capabilities
- Downloadable character set (not available with HP 2392A emulation)
- Terminal identification
- Device status
- Cursor position sensing
- Command completion status

In response to a status request, Reflection transmits a block of data that includes an escape sequence, a series of data bytes, and a block terminator. The rules for block transfer handshaking and the terminator characters that Reflection appends to the end of any block transfer are explained in the Reflection documentation. Status responses are the same for all terminals that Reflection emulates, unless noted.

Most status request escape sequences will cause your keyboard to lock. The keyboard remains locked until the host sends a DC1 character (XON), indicating that it is ready to receive the completion code for the request. This is normal behavior. If the keyboard remains locked, a soft reset will unlock it. If typeahead is enabled, you can continue to type while the host is waiting for a completion code.

Terminal Status Requests

A terminal status request is composed of 14 bytes: bytes 0–6 describe the *primary terminal status* and bytes 7–13 describe the *secondary terminal status*. Terminal status includes information such as display memory size, configuration settings, and terminal errors, as well as other related information.

Primary Status Request

To request the primary status from Reflection, the host computer sends the escape sequence Esc^.

Reflection responds by sending an Esc\, seven status bytes (0 –6), and a terminator. The seven status bytes are all 8-bit binary, with the four high-order bits of each byte 0011. This ensures that all responses can be read as ASCII characters. The four low-order bits of each byte are described below, where the bits are numbered as follows:

```
8 7 6 5  4 3 2 1 - bit numbers
| | | |  | | | |
0 0 1 1  ? ? ? ? - bit values
```

Character	Binary Equivalent
0	0011 0000
1	0011 0001
2	0011 0010
3	0011 0011
4	0011 0100

Character	Binary Equivalent
5	0011 0101
6	0011 0110
7	0011 0111
8	0011 1000
9	0011 1001
:	0011 1010
;	0011 1011
<	0011 1100
=	0011 1101
>	0011 1110
?	0011 1111

Byte 0: Display memory response.

Bit 4: 1 = 8K bytes

Bit 3: 1 = 4K bytes

Bit 2: 1 = 2K bytes

Bit 1: 1 = 1K bytes

Byte 1: Configuration straps A–D.

Bit 4: Strap D, page/line mode
0 = line, 1 = page

Bit 3: Strap C, inhibit end-of-line wrap
0 = no, 1 = yes

Bit 2: Strap B, space overwrite latch
0 = no, 1 = yes

Bit 1: Strap A, transmit functions
0 = no, 1 = yes

Byte 2: Configuration straps E–H.

Bit 4: Strap H, inhibit DC2
0 = no, 1 = yes

Bit 3: Strap G, inhibit DC1 handshake
0 = no, 1 = yes

Bit 2: Always 0

Bit 1: Always 0

Byte 3: Latching keys.

Bit 4: Always 1, indicating this model of terminal can send a secondary status

Bit 3: Auto linefeed
0 = off, 1 = on

Bit 2: Block mode
0 = character mode, 1 = block mode

Bit 1: CapsLock status
0 = off, 1 = on

Byte 4: Pending transfer flags.

Bit 4: Secondary status pending
0 = no, 1 = yes

Bit 3: Enter ↵ pending
0 = no, 1 = yes

Bit 2: Function key pending
0 = no, 1 = yes

Bit 1: Cursor sense pending
0 = no, 1 = yes

Byte 5: Error flags.

Bit 4: Device error
0 = no error, 1 = error

Bit 3: Always 0

Bit 2: Self-test result
0 = Error or self-test not requested
1 = No error

Bit 1: Datacomm error status
0 = no error
1 = framing, overrun, or parity error has occurred since the last status request

Byte 6: Device transfer pending flags. This byte reports on the S, F, or U completion codes associated with the ESCc&p escape sequence.

Bit 4: Always 0

Bit 3: Always 0

Bit 2: Device operation status pending
0 = no, 1 = yes

Bit 1: Device status pending
0 = no, 1 = yes

Secondary Status Request

To request a secondary status response from Reflection, the host computer sends the escape sequence ESC~. Reflection responds with the escape sequence ESC| (the ASCII vertical bar character, decimal 124), seven status bytes (7–13), and a terminator. Like the primary status response, the secondary status bytes are all 8-bit binary, with the four high-order bits of each byte 0011. The four low-order bits of each byte are described below, where the bits are num-bered as follows:

```
8 7 6 5  4 3 2 1 - bit numbers
| | | |  | | | |
0 0 1 1  ? ? ? ? - bit values
```

Byte 7: Reports the available terminal memory, besides display memory, available for data buffers.

Bit 4: Always 0, 8K bytes

Bit 3: Always 0, 4K bytes

Bit 2: Always 0, 2K bytes

Bit 1: Always 0, 1K bytes

Byte 8: Terminal firmware configuration.

Bit 4: Always 0, non-programmable terminal

Bit 3: Always 1, terminal identifies self

Bit 2: Always 0, no APL firmware

Bit 1: External printer interface present 0 = no, 1 = yes

Byte 9: Configuration straps J–M. These straps do not apply to these terminals. Reflection returns a 0 in this byte.

Byte 10: Keyboard interface keys N–R. Reflection returns a 0 in this byte.

Byte 11: Configuration straps S–V. These straps do not apply to these terminals. Reflection returns a value of 0 in this byte.

Byte 12: Configuration straps W–Z. Straps W and Y do not apply to these terminals. Reflection returns a value of 0 in this byte.

Byte 13: Memory lock mode status.

Bit 4: Always 0

Bit 3: 0 = memory is not full
1 = memory is full

Bit 2: 0 = memory lock not on
1 = memory lock on

Bit 1: 0 = locked in row 0 (overflow protect)
1 = not locked in row 0

Terminal Status Requests

The host can request information about terminal capabilities, such as the amount of memory, with the escape sequence `Esc*s<x>^` where <x> is one of the following:

<x>	Request
–1	Text capabilities
–3	Amount of RAM memory available for downloading
endash 4	Interface capabilities

Reflection responds with a number of bytes. The first byte indicates the number of status bytesin the response (it does not include itself in the count); it is followed by the requested status information. The high-order bits of the count byte are 01. The high-order bits of each status byte are 001. The response is returned as ASCII characters, which correspond to the 8-bit pattern.

If <*x*> is less than –5 (–6 for example), no status bytes are returned.

Terminal Text Capabilities

Byte 0: Returns 2 as the string length. The HP 700/92 and 700/94 terminal returns 3 as the string length, and byte 3 below applies.

Byte 1: Forms characteristics.

Bit 5: Not used

Bit 4: Forms cache (700/94 only)
0 = no, 1 = yes

Bit 3: Modified data tags (700/94 only)
0 = no, 1 = yes

Bit 2: Not used
0 = no, 1 = yes

Bit 1: Not used

Byte 2: Display characteristics.

- Bit 5: Not used
- Bit 4: Not used
- Bit 3: Not used
- Bit 2: Color
0 = no, 1 = yes
- Bit 1: Security enhancement supported
0 = no, 1 = yes

Byte 3: Additional features (applicable only to the HP 700/92 and 700/94 terminal).

- Bit 5: Not used
- Bit 4: Not used
- Bit 3: Not used
- Bit 2: Multiple screen widths
0 = no, 1 = yes
- Bit 1: Store/restore configuration and function keys
0 = no, 1 = yes

Memory Available for Downloading Code

Byte 0: Returns 2 as the string length.

Byte 1: Least significant byte:

- Bit 5: 2^4
- Bit 4: 2^3
- Bit 3: 2^2
- Bit 2: 2^1
- Bit 1: 2^0

Byte 2: Most significant byte:

Bit 5: 2^9

Bit 4: 2^8

Bit 3: 2^7

Bit 2: 2^6

Bit 1: 2^5

To determine the amount of memory in kilobytes, convert the binary number represented by the five low-order bits of bytes 1 and 2 to a decimal number, and then multiply by 8K.

When the host requests this status information from Reflection, Reflection returns BSpSpCR, which indicates no memory is available for downloading.

Terminal Interface Capabilities

Byte 0: Returns 2 as the string length.

Byte 1: Interfaces present.

Bit 5: Not used

Bit 4: Programmable configuration (Esc&q)
0 = no, 1 = yes

Bit 3: Printer pass through
0 = no, 1 = yes

Bit 2: Serial printer port present
0 = no, 1 = yes

Bit 1: HP-IB interface
0 = no, 1 = yes

Byte 2: 8-bit parallel port.

Bit 5: Not used

Bit 4: Not used

Bit 3: 8-bit parallel printer port installed
0 = no, 1 = yes

Bit 2: Not used

Bit 1: Not used

Downloadable Character Sets

The host computer can check the downloadable character sets with the escape sequence `Esc*y^`. This escape sequence is specific to the HP 700/92 and 700/94 terminals. When the **Terminal type** group box option (on the Terminal Type tab of the Terminal Setup dialog box) is set to either **HP 70092**, **HP70096** or **HP 70094**, **HP70098**, Reflection responds with five status bytes. The first byte indicates the number of bytes to follow. The two high-order bits are always 01. Reflection's response to this request is always `D@@@`, indicating that downloadable characters are not available.

Byte 0: Returns 4 as the string length

Byte 1: Half shift/minimum character set number

Bit 6: Half shift supported
0 = no, 1 = yes

Bits 5–1: Number of minimum downloadable character set (1–3)

Byte 2: Maximum character set number

Bits 6–1: Number of maximum downloadable character set

Byte 3: Character width

Bits 6–1: Character width (in dots)

Byte 4: Character height

Bits 6–1: Character height (in dots)

Terminal Identification

The host computer can request the terminal ID with the escape sequence `ESc*s^` or `ESc*s1^`. Reflection responds with the numeric portion of the **Terminal type** option selected on the Terminal Type tab of the Terminal Setup dialog box. For example, the **HP 70092**, **HP70096** or **HP 70094**, **HP70098** option returns a value of 70094.

Device Status

The host computer can request the status of an external device, typically a printer, by issuing the escape sequence `ESc&p<device code>^`, where <*device code*> is either 4 or 6. Codes 4 and 6 both indicate external printers in Reflection, so there is no difference in which device code is used.

Reflection responds by sending the string `ESc\p<device code>`, followed by three bytes of status information, then a terminator. The four high-order bits of each of the three status bytes is 0011.

Byte 0:	Bit 4:	Always 0
	Bit 3:	Always 0
	Bit 2:	Printer error report 0 = no error, 1 = printer error
	Bit 1:	Status of the last print command (S or F) 0 = last command successful 1 = last command failed
Byte 1:	Bit 4:	Status of last command (U) 0 = last command was interrupted 1 = last command was performed
	Bit 3:	Always 0
	Bit 2:	Always 0
	Bit 1:	Printer status 0 = not busy, 1 = busy

Byte 2:	Bit 4:	Always 0
	Bit 3:	Always 0
	Bit 2:	Always 0
	Bit 1:	Printer connected 0 = no, 1 = yes

Command Completion Status

A program can determine whether a Copy printer operation completed successfully by checking the completion status code returned.

The print functions are controlled by the escape sequence Esc&p, and return completion status codes of S, F, or U. A value of S indicates successful completion, F indicates the operation failed, and U indicates the user interrupted the operation by pressing Enter↵. U can only be returned for the Reflection Copy method.

Bit 1 of byte 0 of the device status request (see page 46) tracks the success or failure of printer operations. If bit 1 is *0*, a completion status of S is returned. If bit 1 is *1*, a completion status of F is returned.

Bit 4 of byte 1 of the device status request tracks whether the last command was completed or interrupted. If bit 4 is set to 0, the last printer operation was interrupted, and a completion status of U is returned.

CHAPTER 3

HP Format Mode

Like the HP terminal, Reflection has a *format mode*. In format mode, the screen is divided into *unprotected* fields, in which data can be entered, and *protected* fields, in which data can't be entered. By using a combination of unprotected and protected fields, color settings, and the line drawing character set, you can create forms for "fill-in-the-blanks" data entry using an HP 3000 forms program such as HP VPLUS/3000.

Forms generation can be simplified by creating user keys with escape sequences for designing forms. Assign the *Normal* attribute to each key's definition. See the Reflection online help (R1win.hlp) for detailed information about setting up user keys.

Format Mode Fields

Typically with format mode, a form is created and then displayed on the screen. Next, format mode is enabled for entering data into the form. Data is entered only in the unprotected fields; if the cursor is in a protected area of the screen and a character is typed, the cursor automatically advances to the next unprotected field before the character appears.

After all data is entered into the form, it is sent to the host program designed to accept the data by pressing Enter↵ while in block, page, and format modes.

Format mode is enabled and disabled by using the Reflection **FormatMode** property, or by using the following escape sequences, either from the keyboard or from the host computer:

- Enable format mode: ESCW
- Disable format mode: ESCX

When format mode is enabled, a home up function is automatically performed; the cursor moves to the first column of the first unprotected field in display memory. All tab stops are cleared, and the margins are set to columns 1 and 80.

Tab stops are ignored in format mode. If there are no unprotected fields in display memory, the cursor moves to row 1, column 1.

Format mode affects keyboard functions as listed in the table below.

Function	Keystroke	Effect in Format Mode
Insert line	Alt+I	Disabled
Delete line	Alt+D	Disabled
Insert character	Insert	Works within unprotected field boundaries
Delete character	Delete	Works within unprotected field boundaries
Clear line	Alt+K	Clears to end of unprotected field
Clear display	Alt+J	Clears only unprotected fields from cursor to end of screen
Tab	Tab	Advances cursor to next unprotected field
Backtab	Shift+Tab	Moves cursor back to start of current field or to previous unprotected field
Home up	Ctrl+Home	Moves cursor to first column of first unprotected field in display memory

Designing Forms

Forms can be drawn on the screen manually through the keyboard, or by escape sequences and text received from the host computer. With the HP VPLUS/3000 forms handling system, forms are initially designed from the keyboard, then loaded into the program; when the application is run, the form descriptions are sent to the terminal to draw the forms.

The steps involved in drawing a form, either from the keyboard or from the host computer, are as follows:

1. Make sure that format mode is not on. The escape sequence EscX turns it off.
2. Home the cursor (EscH) and clear display memory (EscJ).

3. Create the form background; that is, the fields you want protected when the form is used to enter data.

   You can use ASCII characters, Roman 8 extension characters, line drawing characters, and display enhancements (a table of display enhancement escape sequences appears on page 16). To turn line drawing characters on, press Ctrl+N. To turn line drawing characters off, press Ctrl+O. The line drawing character set is shown on page 65. Use the escape sequence ESC&d<x> to select display enhancements.

4. After creating the protected fields, create the unprotected fields.

   Move the cursor to the row and column where you want the unprotected field to start. If you want the field to be enhanced, use the escape sequenceESC&d<x> to start the display enhancement.

5. Enter the escape sequence ESC[to start the unprotected field.

6. Move the cursor one column beyond the last column that is to be part of the unprotected field, and enter the escape sequence ESC] to terminate the field.

   If you want a field to extend through column 80 (that is, that wraps from the end of one screen row to the next), do not enter an *end of field* escape sequence; move the cursor to the first column of the following row, then enter an ESC[sequence. This sequence can be used to define a field that spans any number of screen rows.

7. To clear a *start of field* indicator, position the cursor at the first column of the field and press Delete.

8. To end a display enhancement, move the cursor one column beyond the last column that is to be enhanced, and enter the sequence ESC&d@.

9. When the form is complete, you transmit it to the host program using VPLUS/300, FORMIO, or using the Enter↵ key to send the form to a user-supplied receiving program.

When you use the security display enhancement in forms, data typed does not appear on the screen, but is transmitted with the form. This enhancement can be useful for letting users enter a password or other confidential information into a form.

Entering Data into Forms

In general, forms are created for use with specific host programs. For example, a form could be created for use in a host database program; when a user runs the program and enters data into the form, the data fields are transmitted to the host, and the database program interprets the fields as it was designed to do.

A form is divided into protected and unprotected fields only when used in format mode. When a user runs a host program such as the database example above, the program would transmit the form and enable format mode; the user can then enter data only in the unprotected fields. All other areas of the display are protected and cannot be changed.

In format mode, the Tab key moves the cursor from one unprotected field to the next; Shift+Tab moves the cursor to the previous unprotected field. The arrow keys can also be used to move around the screen, but in format mode, data can be entered only in the unprotected fields. If you type in a protected field, the cursor moves to the next unprotected field and the typed characters appear. As you type in an unprotected field and reach the end, the cursor automatically advances to the next unprotected field.

After entering all data into a form, you press the Enter↵ key while in block mode, with page mode and format mode also enabled, to transmit the contents of the form to the host. A host program must be ready to accept the data. Typically, if you're working in a host program that uses format mode, the host enables the proper modes for you.

Using Forms Cache

Forms cache can be used to reduce the amount of data transmission from the host computer when running applications that use forms. In a typical format mode application, most of the data sent merely defines the forms to be displayed on the screen. When a form is displayed more than once, a good deal of time is wasted by redefining the same form each time.

With the forms cache feature available with VPLUS/3000 or other HP 3000 software, you can send a number of forms definitions to Reflection for local storage and cause a given form to be displayed by sending a single short escape sequence.

To use forms cache, **HP70094**, **HP70098** must be selected as the **Terminal type** option on the Terminal Type tab of the Terminal Setup dialog box.

Allocating the Forms Buffer

`Esc&q<m>te2{<x>L`

Allocates a forms buffer in memory, where <*m*> = 4–7 and <*x*> is the number of 256-byte blocks to be allocated.

Before any forms can be stored in the PC, a memory buffer must be allocated. The forms buffer is related to the amount of memory allocated for display memory—if you intend to use forms buffer applications, you may need to raise the value for forms buffer size using the Reflection **FormsBufferSize** property.

If the requested number of blocks cannot be allocated, the size of the forms buffer is increased to use as much memory as is available. The maximum number of blocks that can be requested is 255. A value of Ø deletes the forms buffer.

Any attempt to change the forms buffer size causes memory to be re-allocated, clearing the contents of display memory. The forms buffer size increases the memory allocated to run Reflection.

To determine if the requested allocation was successful, the host program must issue a forms buffer status request.

Storing a Form

The escape sequences below store a form in the forms buffer.

To store a form:

```
Esc&p9u<form#>p<length>L<contents>
```

To store a form of unknown length:

```
Esc&p9u<form#>p<<contents>>L
```

To store a form using a name:

```
Esc&p9u<<name>>n<form#>p<length>L<contents>
```

To store a form of unknown length using a name:

```
Esc&p9u<<name>>n<form#>p<<contents>>L
```

<form#>

The form number can be any number from 1–255. If the specified form number already exists in the buffer, it is overwritten with the new form's contents.

<length>

A decimal number specifying the size of the form contents in bytes. Form size is optional.

<contents>

The contents of the form. If <length> is included in the sequence, the contents can have any characters except NUL, DEL, ENQ, and DC1. If <length> is not included, all characters following the "<" are stored until a ">" is encountered.

Use additional "<" and ">"characters before and after <contents>. These are required if <length> is not specified, for example, <<contents>>.

<<name>>

A user-selected name for the form. The form name must be enclosed in angle brackets (<>). Names for forms are optional. Any symbol except a control character or an escape sequence is allowed, including spaces.

If the name contains an embedded bracket ("<" or ">"), the bracket in the name must have a a second "<" in front of it.

Reading the Forms Buffer Status

A program on the host can tell Reflection to send the status of the forms buffer, which can include the following:

- The number of 256-byte blocks of memory in the forms buffer.
- The number of unused 256-byte blocks remaining in the forms buffer.

The status request escape sequence can specify a form number or the name of a form.

Requests the total size of the forms buffer:

```
Esc&p9^
```

Requests the number of unused blocks and the presence or absence of the specified form number:

```
Esc&p<form#>p9^
```

Requests the number of unused blocks and the presence or absence of the named form:

ESC&p<<name>>n9^

The format of the status response is:

ESC\p9<xyz>

The string <xyz> is three ASCII characters for which the four high-order bits are 0011 and the four low-order bits are as defined:

- The four low-order bits of the first byte, <x>, contain the most significant four bits of the number of blocks (either total or unused) in the forms buffer.
- The four low-order bits of the second byte, <y>, contain the least significant four bits of the number of 256-byte blocks.
- If a non-zero form number is specified in the status request, the four low-order bits of the third byte, <z>, indicate the presence (0001) or absence (0000) of the form in the buffer.

In the examples below, 50 blocks with 256 characters each have been allocated for the forms buffer and form numbers 1 through 15 have been stored, leaving 30 blocks unused:

Status Request	**Status Response**
ESC&p9^	ESC\p9320CR
ESC&p0p9^	ESC\p9320CR
ESC&p7p9^	ESC\p91>1CR
ESC&p19p9^	ESC\p91>0CR

In this example, a form named RECEIPT is present, and 7 blocks are available:

Status Request	**Status Response**
ESC&p9^	ESC\p9320CR
ESC&p0p9^	ESC\p9320CR
ESC&p7p9^	ESC\p91>1CR
ESC&p19p9^	ESC\p91>0CR

Reading the Forms Cache Directory

The forms cache directory allows an application program to determine exactly which forms are currently stored in forms cache. The application can easily determine if any or all of its associated forms have already been downloaded to cache memory. This is useful when several different programs in an application share one or more common forms, or when an application is stopped and restarted.

To read the forms cache directory, an extension of the device status request escape sequence is used:

```
EsC&p <>9^
```

The terminal returns a list of the forms currently stored in cache memory, including form numbers and names (if defined). The format of the list is as follows:

```
EsCp9<fnum1><<fname1>>...<terminator>
```

The form number (`<fnum n>`) and form name (`<<fname n>>`) of each form are returned. If a form has not been assigned a name, the "<" and ">" characters appear in the list, indicating a null form name. Form numbers and names are returned in the same sequence that they were defined.

In the example below, form 3 was defined with the name ORDERS and form 1 was subsequently defined with no name:

Status Request	Status Response
`EsC&p<>9^`	`EsC\p93<ORDERS>1<>CR`

Displaying a Form

To display a form that has previously been stored in the forms buffer, send the following escape sequence to Reflection:

```
EsC&p9u<form#>pF
```

A completion code of S (Success) or F (Failure) is returned to the host by Reflection after the appropriate handshaking sequence.

Purging a Form

The following escape sequences purge a form from the forms buffer:

```
EsC&p9u<form#>pL
```

or

```
EsC&p9u<form#>p0L
```

If the form number is found, the amount of space it used in the buffer is released.

Modified Data Tags

`EsC&kØZ` Modified data tags are used for data entry in format mode, which enable transmitting only those fields on the form which have been modified. To use modified data tags, **HP70094**, **HP70098** must be selected as the **Terminal type** option on the Terminal Type tab of the Terminal Setup dialog box.

You can set modified data tags using the following escape sequences:

`EsC&kØZ`	Turns modified data tags off.
`EsC&k1Z`	Turns modified data tags on.
`EsC&q<m>te2{ØZ`	Sets Transmit to All (<*m*>= 4–7).

You can also set this value in the Advanced HP Option dialog box (click Advanced on the Emulation tab of the Terminal Setup dialog box to see this). Or, you can execute the following Reflection Basic property from the command line:

```
Transmit  {ALL | MODIFIED}
```

When modified data tags are enabled, trailing blanks are eliminated from the contents of unprotected fields. In block/page mode, only modified fields are sent.

HP Character Sets and National Characters

Reflection has a base character set (Roman 8) and a secondary character set (HP Line Drawing). The secondary set can be used as an “alternate” character set. The *active* character set is selected from the base or the alternate set. The active set determines both what you see in Reflection’s terminal window when you type from the keyboard, and also controls the display of information received from the host.

The Roman 8 character set is a combination of:

- The USASCII character set (codes 0–127), shown on page 61.
- The national characters in the Roman 8 Extension character set (codes 161–254), shown on page 63.

The Roman 8 character set becomes active when either of the following occurs:

- You start Reflection.
- The cursor moves to a new line.
- Reflection receives a SI (s in) control code, which was either transmitted by a host application or sent directly from you (by pressing Ctrl+O).

The alternate HP Line Drawing character set is loaded when:

- Reflection receives the SO control code (or you sent it directly by pressing Ctrl+N), and remains in effect until either of the following occur:
  - Reflection receives an SI control code.
  - The cursor moves to a new line.
  - A display enhancement (underline, inverse video, and so on) is encountered.

The figure on page 65 shows the HP line drawing characters.

Escape sequence equivalents:

Esc)@ (select base Roman 8 character set)
Esc)B (select HP Line Drawing character set)

The Host Character Set vs. Windows Characters

Because Reflection is designed to be used as an HP terminal, it uses the ASCII character set for most common characters (such as letters and numbers) and, by default, the Roman 8 character set for national and special characters. Windows, on the other hand, use the ANSI character set which is a similar collection of characters. The character values in the range 32–126 are identical for both ASCII and ANSI. With the default host character set loaded, the differences occur in the upper range of values.

Even when identical characters exist in both character sets, they are often represented by different numbers. For example, in the Roman 8 character set (page 63), the y with a diaeresis (ÿ) is represented by decimal 239, while in ANSI it is represented by decimal 255. When you use the Alt key method (explained on page 69) to enter national characters, and the ANSI value is different from that in the host character set, Reflection displays the equivalent of the ANSI character if it can be found in the host character set. If there is no equivalent (such as the ANSI decimal value 247), Reflection displays a blank.

Since Reflection uses the host character set when transmitting and receiving characters, you will only be concerned with entering national characters when:

- You need to enter special characters in a host application, explained on page 69.
- Writing and reading ASCII disk files, explained on page 71.
- Entering foreign characters in a file name when transferring files to a UNIX host, explained in the Reflection online help (R1win.hlp).
- Copying information from the terminal window—the text is automatically converted from the terminal character set to the ANSI character set when placed on the Windows Clipboard. (When a character has no equivalent in one or the other character set, Windows displays a symbol such as | , +, –, or _.)

ASCII Character Set

The ASCII character set consists of characters with decimal values 0–127; this set is identical to the Windows character set (ANSI) codes 32–126. The characters 0–31 are control codes; they appear on your screen with the symbols shown on page 62 only when display functions mode is turned on (F7 from the modes keys).

Decimal →
Hex →

32	20	(space)	48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
45	2D	—	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

ASCII Control Codes 0–32

ASCII Decimal	Keystroke	Display	Definition
0	Ctrl+@	NU	Null
1	Ctrl+A	SH	Start of header
2	Ctrl+B	SX	Start of text
3	Ctrl+C	EX	End of text
4	Ctrl+D	ET	End of transmission
5	Ctrl+E	EQ	Enquiry
6	Ctrl+F	AK	Acknowledge
7	Ctrl+G	BL	Bell
8	Ctrl+H	BS	Backspace
9	Ctrl+I	HT	Horizontal tab
10	Ctrl+J	LF	Linefeed
11	Ctrl+K	VT	Vertical tab
12	Ctrl+L	FF	Form feed
13	Ctrl+M	CR	Carriage return
14	Ctrl+N	SO	Shift out
15	Ctrl+O	SI	Shift in
16	Ctrl+P	DL	Data link escape
17	Ctrl+Q	D1	Device control 1 (XON)
18	Ctrl+R	D2	Device control 2
19	Ctrl+S	D3	Device control 3 (XOFF)
20	Ctrl+T	D4	Device control 4
21	Ctrl+U	NK	Negative acknowledge
22	Ctrl+V	SY	Synchronous idle
23	Ctrl+W	EB	End of transmission block
24	Ctrl+X	CN	Cancel
25	Ctrl+Y	EM	End of medium
26	Ctrl+Z	SB	Substitute
27	Ctrl+[	EC	Escape
28	Ctrl+\	FS	Field separator
29	Ctrl+]	GS	Group separator
30	Ctrl+Shift+6	RS	Record separator
31	Ctrl+Shift+–	US	Unit separator
32	Ctrl+Shift+2		Space (blank)

Roman 8 Extension Character Set

The Roman 8 Extension character set includes special and multinational characters used in non-USASCII national languages.

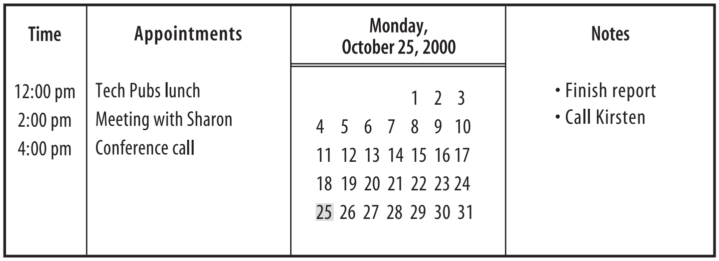
Decimal → Hex →	160 A0	176 B0 ‾	192 C0 â	208 D0 Å	224 E0 Á	240 F0 Þ
	161 A1 À	177 B1 Ý	193 C1 ê	209 D1 î	225 E1 Ã	241 F1 þ
	162 A2 Â	178 B2 ý	194 C2 ô	210 D2 Ø	226 E2 ã	242 F2 •
	163 A3 È	179 B3 °	195 C3 û	211 D3 Æ	227 E3 Đ	243 F3 µ
	164 A4 Ê	180 B4 Ç	196 C4 á	212 D4 å	228 E4 ð	244 F4 ¶
	165 A5 Ë	181 B5 ç	197 C5 é	213 D5 í	229 E5 Í	245 F5 3/4
	166 A6 Î	182 B6 Ñ	198 C6 ó	214 D6 ø	230 E6 Ì	246 F6 –
	167 A7 Ï	183 B7 ñ	199 C7 ú	215 D7 æ	231 E7 Ó	247 F7 1/4
	168 A8 ´	184 B8 ¡	200 C8 à	216 D8 Ä	232 E8 Ò	248 F8 1/2
	169 A9 `	185 B9 ¿	201 C9 è	217 D9 ì	233 E9 Õ	249 F9 ª
	170 AA ^	186 BA ¤	202 CA ò	218 DA Ö	234 EA õ	250 FA º
	171 AB ¨	187 BB £	203 CB ù	219 DB Ü	235 EB Š	251 FB «
	172 AC ~	188 BC ¥	204 CC ä	220 DC É	236 EC š	252 FC ▮
	173 AD Ù	189 BD §	205 CD ë	221 DD ï	237 ED Ú	253 FD »
	174 AE Û	190 BE ƒ	206 CE ö	222 DE ß	238 EE Ÿ	254 FE ±
	175 AF ₤	191 BF ¢	207 CF ü	223 DF Ô	239 EF ÿ	255 FF ◊

HP Line Drawing Character Set

Reflection generates all the characters in the HP line drawing character set, which is an alternate character set comprising simple line and graphic elements. Often, the line drawing set is used with format mode to create forms, tables, and other images. Each letter key generates only one line drawing character, whether Shift is used or not. Most of the numeric and punctuation keys can generate two line drawing characters; one when the key alone is pressed, and another when the key and Shift are pressed together.

Press Ctrl+N to access the line drawing character set; press Ctrl+O to return to normal characters. Reflection reverts to the normal character set when the cursor moves to a new row.

The following figure shows the line drawing characters used to create a sample form:



Time	Appointments	Monday, October 25, 2000	Notes
12:00 pm 2:00 pm 4:00 pm	Tech Pubs lunch Meeting with Sharon Conference call	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	• Finish report • Call Kirsten

```
Q;;;;;;#;;;;;;;;;;;;;;;;;;;;#;;;;;;;;;;;;;;;;;;;;;#;;;;;;;;;;;;;;;;;;;;W
:      .                    .                     .                    :
:      .                    .                     .                    :
:      .                    @999999999999999999999{                    :
:      .                    .                     .                    :
:      .                    .                     .                    :
:      .                    .                     .                    :
:      .                    .                     .                    :
:      .                    .                     .                    :
A;;;;;;$;;;;;;;;;;;;;;;;;;;;$;;;;;;;;;;;;;;;;;;;;;$;;;;;;;;;;;;;;;;;;;;S
```

In the following figure, the line drawing character associated with a key is printed directly on the sample keyboard's key.

Decimal	ASCII	Line Drawing	Decimal	ASCII	Line Drawing	Decimal	ASCII	Line Drawing
32			48	0	╋	64	@/`	╞
33	!	┝	49	1	┣	65	A/a	┗
34	"	┥	50	2	┫	66	B/b	╂
35	#	┯	51	3	┳	67	C/c	█
36	$	┷	52	4	┻	68	D/d	▌
37	%	╟	53	5	├	69	E/e	▎
38	&	╢	54	6	┤	70	F/f	└
39	'	╤	55	7	┬	71	G/g	┘
40	(	╧	56	8	┴	72	H/h	┐
41	)	║	57	9	═	73	I/i	┤
42	*	╂	58	:	┃	74	J/j	┬
43	+	┿	59	;	━	75	K/k	╥
44	,	─	60	<	╫	76	L/l	╦
45	-	└	61	=	┌	77	M/m	╇
46	.	│	62	>	╪	78	N/n	╈
47	/	┼	63	?	╬	79	O/o	╡

Decimal	ASCII	Line Drawing
80	P/p	┘
81	Q/q	┏
82	R/r	┌
83	S/s	┛
84	T/t	┐
85	U/u	┤
86	V/v	╉
87	W/w	┓
88	X/x	█
89	Y/y	┐
90	Z/z	▄
91	[	╡
92	\	╌
93	]	╨
94	^	-
95	_	╥
123	{	╡
124	|	╌
125	}	╨
126	~	-

ANSI Character Set

Reflection can send and receive ASCII, Roman 8, and HP line drawing characters. Windows, however, uses the ANSI character set. This does not present a problem for the characters in the range 32–126 (since both ANSI and ASCII have the same values in this range), but characters above 126 must be converted to ANSI before they can be displayed by other Windows applications. This is explained in the steps on page 69.

Decimal →
Hex →

128 80 ■	144 90 ■	160 A0	176 B0 °	192 C0 À	208 D0 Ð	224 E0 à	240 F0 ð
129 81 ■	145 91 '	161 A1 ¡	177 B1 ±	193 C1 Á	209 D1 Ñ	225 E1 á	241 F1 ñ
130 82 ■	146 92 '	162 A2 ¢	178 B2 ²	194 C2 Â	210 D2 Ò	226 E2 â	242 F2 ò
131 83 ■	147 93 ■	163 A3 £	179 B3 ³	195 C3 Ã	211 D3 Ó	227 E3 ã	243 F3 ó
132 84 ■	148 94 ■	164 A4 ¤	180 B4 ´	196 C4 Ä	212 D4 Ô	228 E4 ä	244 F4 ô
133 85 ■	149 95 ■	165 A5 ¥	181 B5 µ	197 C5 Å	213 D5 Õ	229 E5 å	245 F5 õ
134 86 ■	150 96 ■	166 A6 ¦	182 B6 ¶	198 C6 Æ	214 D6 Ö	230 E6 æ	246 F6 ö
135 87 ■	151 97 ■	167 A7 §	183 B7 ▪	199 C7 Ç	215 D7 ×	231 E7 ç	247 F7 ÷
136 88 ■	152 98 ■	168 A8 "	184 B8 ¸	200 C8 È	216 D8 Ø	232 E8 è	248 F8 ø
137 89 ■	153 99 ■	169 A9 ©	185 B9 ¹	201 C9 É	217 D9 Ù	233 E9 é	249 F9 ù
138 8A ■	154 9A ■	170 AA ª	186 BA º	202 CA Ê	218 DA Ú	234 EA ê	250 FA ú
139 8B ■	155 9B ■	171 AB «	187 BB »	203 CB Ë	219 DB Û	235 EB ë	251 FB û
140 8C ■	156 9C ■	172 AC ¬	188 BC 1/4	204 CC Ì	220 DC Ü	236 EC ì	252 FC ü
141 8D ■	157 9D ■	173 AD –	189 BD 1/2	205 CD Í	221 DD Ý	237 ED í	253 FD ý
142 8E ■	158 9E ■	174 AE ®	190 BE 3/4	206 CE Î	222 DE Þ	238 EE î	254 FE þ
143 8F ■	159 9F ■	175 AF —	191 BF ¿	207 CF Ï	223 DF ß	239 EF ï	255 FF ÿ

■ Indicates that this character is not printable by other Windows applications.

IBM PC Extended Character Set (Code Page 437)

Windows applications can display only those characters from the IBM PC extended character set (ECS) that also appear in the ANSI character set (for example, the character ¢ appears in both sets). You can enter any common character, even if the ECS code is different from the ANSI character set code—this is explained in the steps on page 70.

Decimal →
Hex →

128 80 Ç	144 90 É	160 A0 á	176 B0 ■	192 C0 ■	208 D0 ■	224 E0 ■	240 F0 ■
129 81 ü	145 91 æ	161 A1 í	177 B1 ■	193 C1 ■	209 D1 ■	225 E1 ß	241 F1 ±
130 82 é	146 92 Æ	162 A2 ó	178 B2 ■	194 C2 ■	210 D2 ■	226 E2 ■	242 F2 ■
131 83 â	147 93 ô	163 A3 ú	179 B3 ■	195 C3 ■	211 D3 ■	227 E3 ¶	243 F3 ■
132 84 ä	148 94 ö	164 A4 ñ	180 B4 ■	196 C4 ■	212 D4 ■	228 E4 ■	244 F4 ■
133 85 à	149 95 ò	165 A5 Ñ	181 B5 ■	197 C5 ■	213 D5 ■	229 E5 ■	245 F5 ■
134 86 å	150 96 û	166 A6 ª	182 B6 ■	198 C6 ■	214 D6 ■	230 E6 µ	246 F6 ■
135 87 ç	151 97 ù	167 A7 º	183 B7 ■	199 C7 ■	215 D7 ■	231 E7 ■	247 F7 ■
136 88 ê	152 98 ÿ	168 A8 ¿	184 B8 ■	200 C8 ■	216 D8 ■	232 E8 ■	248 F8 °
137 89 ë	153 99 Ö	169 A9 —	185 B9 ■	201 C9 ■	217 D9 ■	233 E9 ■	249 F9 ■
138 8A è	154 9A Ü	170 AA ¬	186 BA ■	202 CA ■	218 DA ■	234 EA ■	250 FA ■
139 8B ï	155 9B ¢	171 AB 1/2	187 BB ■	203 CB ■	219 DB ■	235 EB ■	251 FB ■
140 8C î	156 9C £	172 AC 1/4	188 BC ■	204 CC ■	220 DC ■	236 EC ■	252 FC ■
141 8D ì	157 9D ¥	173 AD ¡	189 BD ■	205 CD ■	221 DD ■	237 ED ■	253 FD 2
142 8E Ä	158 9E ■	174 AE «	190 BE ■	206 CE ■	222 DE ■	238 EE ■	254 FE "
143 8F Å	159 9F ■	175 AF »	191 BF ■	207 CF ■	223 DF ■	239 EF ■	255 FF ■

■ Indicates that this character is not printable by other Windows applications.

Entering National Characters

The entry of a national character depends on the character, your keyboard, and whether you are operating in 7- or 8-bit mode.

Some PCs have keyboards and keyboard drivers available for languages other than English. If the character you want to type is on your keyboard, simply press the key to produce the character. Reflection does nothing to alter your keyboard's characters no matter what keyboard or language you have selected.

Entering Characters Using the Alt Key Method

You can enter national characters using standard Windows conventions; for this reason Reflection has no equivalent for the HP *extend* sequence.*

Using the Alt key method, you provide a decimal value from the ANSI character set to enter a character in Reflection's terminal window.

1. Log on to the host as you would normally do.
2. Make sure the **National Replacement Set** list is set to its default value of **None** (click Advanced on the Emulation tab of the Terminal Setup dialog box to see this).
3. Find the ANSI decimal value for the character in the figure shown on page 67.
4. Make sure NumLock is off.
5. Hold down Alt, type `0`, then type the 3-digit decimal code on your numeric keypad.
6. Release Alt.

For example, if you hold down Alt and press 0, 1, 9, and 6 on the numeric keypad, an Ä appears when you release the Alt key.

Using the Reflection Display Method

You can also enter characters in Reflection's terminal window using the Reflection Display method. With this method, you provide a decimal value from the host's character set; therefore, you can produce characters unique to the host that do not also appear in the ANSI character set.

* The extend key turns on an extend sequence; this gives access to the Roman 8 character set.

1. Press Alt+L to open the Reflection command line.
2. Find the decimal value for the character from the Roman 8 Extension character set (page 63).
3. Use the **Display** method syntax as follows, and press Enter↵:

   ```
   Display String [, Options]
   ```

For example, with the Roman 8 character set loaded, the following displays the £ symbol in Reflection's terminal window:

```
Display & "^(187)"
```

Entering IBM PC Extended Characters

Windows applications can display only those characters from the IBM PC extended character set (ECS) that also appear in the ANSI character set (for example, the character ¢ appears in both sets). You can enter any common character, even if the ECS code is different from the ANSI character set code: Windows automatically performs the conversion for you.

To enter a character in Reflection's terminal window from the IBM extended character set:

1. Log on to the host as you would normally do.
2. Make sure the **National Replacement Set** list is set to its default value of **None** (click Advanced on the Emulation tab of the Terminal Setup dialog box to see this).
3. Find the ECS decimal value for the character in the figure shown on page 68.
4. Hold down Alt and type the 3-digit decimal code on your numeric keypad (do *not* type a Ø before the code, as you would to enter a national character using the procedure on page 69).
5. Release Alt.

For example, if you are using code page 437 (United States), hold down Alt and press 1, 5, and 5 on the numeric keypad. Windows converts 155 to its ANSI equivalent 162 (as if you had typed Alt+0+1+6+2), and a ¢ symbol appears.

Translation of Characters

Reflection translates between the Windows and the host character sets, so using characters that exist in both sets preserves the most characters.

There are times when translation should not occur at all in order to completely preserve what is being read from or stored to disk. In this case, execute the following in the Reflection command line:

```
TranslateCharacters = Value
```

This command causes Reflection to assume that characters are already in the host character set, and nothing is translated. There are two basic issues for deciding if Reflection should translate between the host and Windows character set:

- The source of the characters.
- The destination for the characters.

The following table lists the source and destination for disk files. YES and NO answer the question “Should translation be disabled?”

	Destination or Use for File	
Source of File Read from Disk	**HP 3000 Host**	**Windows Application**
Host file	YES	NO
Saved from display memory	n/a	NO
Windows application file	NO	YES

The **TranslateCharacters** property also affects:

- ASCII file transfers when reading from or writing to disk.
- Printing. You may want to select the **Disable printer translation** check box in the Print Setup dialog box (the **Bypass Windows print driver** check box must be selected to use this option).

Retaining ASCII Files for the DOS Environment

When you receive an ASCII file under Windows, it is received as an ANSI file. If you plan on using the file in DOS, and therefore want the characters to be IBM PC extended characters (see the figure on page 68), execute the following in the Reflection command line:

```
TextFileCharacterSet = Value
```

Note: When the Reflection TranslateCharacters value equals YES, the **TextFileCharacterSet** property has no effect.

Translation to DOS Problems

If you have **TextFileCharacterSet** set to *DOS* (or WindowsCharacterSet) and characters are not being translated as you would expect them to be, the code page that Windows is set up to recognize may be different from the code page defined at the DOS level. This difference in code pages can come about in two ways:

- You incorrectly answered a question when you ran the Windows Setup program (for example, you gave the wrong response for your language or keyboard type).
- You changed the DOS code page since installing Windows.

To see what code page DOS is using, type `CHCP` at the DOS prompt (you can do this from a DOS session under Windows). To see the Windows code page, execute the following in the Reflection command line:

```
Value = CodePage
```

The result you see (for example, 437) should be the same code page as that set up for DOS. If the two are different, then the translation of characters will be incorrect. To fix the Windows code page, you must re-run the Windows Setup program.

Transmitting National Characters

Transmission of Roman 8 characters depends on the setting of the **Parity** list in the Connection Setup dialog box.

8-Bit Operations

When parity is set to **8/None**, Reflection is operating in 8-bit mode. Each character consists of eight data bits with no parity bit (all eight bits of each byte are used for data).

When the high-order bit is used, the character is interpreted as an extended Roman 8 character. Bit eight in U.S. ASCII characters is always set to *zero*, and to *one* in Roman 8. This allows all Roman 8 characters to be sent and received.

7-Bit Operations

When parity is set to a value other than **8/None** or **7/None**, Reflection is operating in 7-bit mode—that is, seven data bits and one parity bit. The 8th bit of each byte is used for parity and is not available for use as data.

In 7-bit operations, some characters in the U.S. ASCII character set are replaced by certain Roman 8 characters as shown in the table on following page. (The Roman 8 set is a combination of the ASCII set and the Roman Extension set.) This only happens when the **National Replacement Set** list in the Advanced HP Options dialog box is configured to a value other than None. This is often called *ISO-7 mode* because the International Standards Organization has defined which USASCII (7-bit) characters are replaced.

National Replacement Set	Characters													
ASCII Decimal	**35**	**39**	**60**	**62**	**64**	**91**	**92**	**93**	**94**	**96**	**123**	**124**	**125**	**126**
U.S. ASCII	#	´	<	>	@	[	\	]	^	`	{	|	}	~
British (U.K.)	£	´	<	>	@	[	\	]	^	`	{	|	}	~
French	£	´	<	>	à	°	ç	§	^ **	`	é	ù	è	¨ **
German	£	´	<	>	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
Italian	£	´	<	>	§	°	ç	é	^ *	ù	à	ò	è	ì
Norwegian	#	´ *	<	>	@	Æ	Ø	Å	^	` *	æ	ø	å	¨
Danish	§	´ *	<	>	@	Æ	Ø	Å	^	` *	æ	ø	å	¨
Swedish	#	´	<	>	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Dutch	#	´	<	>	@	ç	\	§	^ *	` *	ƒ	|	´ *	¨ *
Swiss-German	£	´ *	é	è	à	°	ç	§	^ *	` *	ä	ö	ü	¨ *
Spanish-European	#	´	<	>	@	¡	Ñ	¿	°	` **	´	ñ	ç	¨ **
Spanish-Latin	#	´	<	>	@	¡	Ñ	¿	^	` **	´	ñ	ç	¨ **
Finnish	#	´	<	>	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Canadian-English	#	´	<	>	@	[	ç	]	^ *	` *	é	Ç	É	¨ *
Flemish (Belgian)	£	´	<	>	à	°	ç	§	^ **	`	é	ù	è	¨ **
Canadian-French	#	´	<	>	@	[	ç	]	^ *	` *	é	Ç	É	¨ *
Swiss-French	£	´ *	é	è	à	°	ç	§	^ *	` *	ä	ö	ü	¨ *
Line Drawing	┳	╤	╫	╪	╞	╡	--	╨	-	╞	╡	--	╨	-

* In 8-bit operations only, this diacritical mark is mute.
** In 7-bit and 8-bit operations, this diacritical mark is mute.

In 7-bit operations, characters from the Roman 8 set cannot be directly sent to or received from the host. The Roman 8 characters are coded in eight bits, whereas U.S. ASCII characters are coded in seven bits. Reflection automatically substitutes the corresponding U.S. ASCII character code for the Roman 8 character when receiving such characters, and vice-versa during transmission.

What Roman 8 characters replace U.S. ASCII characters depends on which set is selected in the **National Replacement Set** list. Diacritical marks labeled as “mute” do not appear immediately when you press the key that generates the mark; they appear only after pressing the character you want to accent.

The line-drawing character set (page 65) is affected by this since the elements correspond to U.S. ASCII characters, and some of these are not available if they are replaced by national characters. Therefore, when operating in 7-bit mode using national keyboards, the replacement character should be used to access the line-drawing element.

SECTION 3

Emulating a VT Terminal

Configuring Reflection to Emulate a VT Terminal

By default, Reflection for HP with NS/VT provides complete emulation of the HP 70092, 70094, and 2392A terminals. If you need to work with a host application that must understand Digital escape sequences, you can configure Reflection to emulate the VT52, VT102, and VT200 level terminals.

Operating Reflection in VT emulation mode also allows you to run HP applications that have embedded VT control sequences, such as Oracle and UDMS.

This section describes how to configure Reflection for VT terminal emulation, then explains what changes are made to your keyboard map. Figures of the character sets supported in VT emulation mode as shown starting on page 123.

Starting Reflection for VT Emulation

To emulate a VT terminal, you need only select the terminal type and configure the connection. Then, save the connection and any other changes to a settings file shortcut. To do this:

1. On the Setup menu, click Terminal to open the Terminal Setup dialog box.
2. On the Terminal Type tab, select a VT terminal as the **Terminal type**.
3. Click OK.
4. On the Connection menu, click Connection Setup and select the appropriate options.
5. Click Connect, follow the prompts and log in as you usually do.
6. Click Save As, and in the Save Settings dialog box, click Shortcut.
7. Select a location for the shortcut in the **Place the shortcut** box.
8. Click OK, then give the settings file a name in the File name box.
9. Click OK. The shortcut is created, and when you click it, will return you to the settings you had during this Reflection session.

While in VT emulation mode, Reflection recognizes and executes the control codes and escape sequences for the level of VT terminal you selected on the Terminal Type tab in the Terminal Setup dialog box (VT52 escape sequences are ignored when you are emulating a VT102 or VT220 level terminal).

Emulating the VT Terminal Keyboard

In VT emulation mode, Reflection maps the following keys on your PC keyboard, which were previously emulating an HP terminal, to perform the indicated VT terminal function. ("Cp" stands for cursor pad and "Kp" stands for numeric keypad.)

VT Key	Default PC Keystroke
VtF6	Shift+Ctrl+6
VtF7	Shift+Ctrl+7
VtF8	Shift+Ctrl+8
VtF9	Shift+Ctrl+9
VtF10	Shift+Ctrl+0
VtF11	Shift+Ctrl+Q
VtF12	Shift+Ctrl+W
VtF13	Shift+Ctrl+E
VtF14	Shift+Ctrl+R
VtF15	Shift+Ctrl+T
VtF16	Shift+Ctrl+Y
VtF17	Shift+Ctrl+U
VtF18	Shift+Ctrl+I
VtF19	Shift+Ctrl+O
VtF20	Shift+Ctrl+P
Find	Cp Home
Insert Here	Cp Insert

VT Key	Default PC Keystroke
Remove	Cp Delete
Select	Cp End
Previous Screen	Cp PgUp
Next Screen	Cp PgDn
PF1	NumLock
PF2–PF4	Kp /, Kp *, and Kp –
Vt0–Vt9	Kp 0 – Kp 9
Decimal	Kp .
Minus	Kp +
Comma	Alt+Kp +

In addition, the following keystrokes perform these functions during VT emulation:

Alt+J	Clears the display from the cursor to the bottom of display memory.
Alt+K	Clears the line the cursor is on.
Ctrl+PgUp	Scrolls back one screen in display memory.
Ctrl+PgDn	Scrolls forward one screen in display memory (this adds lines to display memory if you're already at the bottom of display memory).
Ctrl+Home	Moves to the top of display memory.
Ctrl+End	Moves to the bottom of display memory.
Alt+I	Inserts a line at the cursor position.
Alt+D	Deletes a line at the cursor position.

Default Keyboard Mapping

Reflection maintains a separate keyboard map for VT emulation mode (On the Setup menu, click Keyboard Map to see this). Changes you make to the keyboard map while in HP mode (see “Customizing the Keyboard” in the Reflection *User Guide*) will not be available when you switch to VT emulation mode: you should create and save the mapping profile again for this and future VT emulation under Reflection.

You can restore the default map for the current terminal type by clicking Defaults in the Keyboard Map Setup dialog box. Restoring defaults while in VT emulation mode will have no effect on any changes you may have made to your keyboard map while emulating an HP terminal, and vice versa.

VT Keyboard Map Precautions

When emulating an HP terminal, you can use the mouse or press the function keys to activate F1–F8. In VT emulation mode, you must use the mouse to activate HP function keys; F5–F8 perform VT functions, and F1–F4 perform no function (since during HP emulation, these function keys send HP user key definitions). You can always remap these keystrokes to different functions, as explained in the Reflection online help (R1win.hlp).

VT Emulation Mode Control Sequences

Control sequences cause Reflection to perform certain actions, such as move the text cursor, add a line of text, assign character attributes, and change character sets. This chapter describes the control sequences supported by Reflection while in VT emulation mode. To emulate a VT terminal in Reflection for HP with NS/VT, see page 79.

Most VT-supported sequences have mnemonics associated with them; for example, SD is a VT-specified sequence that represents Scroll Down. The mnemonic is shown in parentheses. Mnemonics that begin with HP represent HP private sequences–for example, HPCOLM which sets 80 or 132 columns.

Typically, the host sends control sequences to Reflection to perform the desired actions. Entering sequences manually is described on page 84.

Notation Used in this Chapter

There are three symbols used in this section that describe a sequence:

`ESC` stands for the Escape character, which begins an escape sequence. Escape and control sequences are usually sent to Reflection from the host, but you can also enter them from the keyboard. VT102 and VT220 modes support HP escape sequences (in the chapter starting on page 9) that begin with the following prefixes:

`ESC&f  ESC&j  ESC&k  ESC&p  ESC&s  ESC&w`

`CSI` stands for the Control Sequence Introduction character, which begins a control sequence. The 7-bit equivalent is `ESC[`. Control sequences can include variable parameters. For example, `CSI7;18r` sets the scrolling region, where the top margin is at line 7 and the bottom is at line 18.

`DCS` stands for the Device Control String character, which begins a device control sequence. The 7-bit equivalent is `ESCP`.

How to Enter Control Sequences

To use the functions in this section, you must know how to represent these characters either by using keystrokes within Reflection's terminal window, or by executing a command on the Reflection command line by typing the sequence:

Sequence:	Esc	Csi	Dcs
In terminal window, press:	Esc	Esc+[	Esc+P
In command line, type:	`Chr$(27)`	`Chr$(27)& "["`	`Chr$(27)&"P"`

Reflection uses the ANSI character table to equate a numeric expression to a character. In the example above, the integer 27 corresponds to the escape character. See the Reflection Programming Reference online help (Rwinprog.hlp) for more information on the Chr$ function, built-in variables, and sending commands from the command line or incorporating them into Reflection macros.

If you want to enter control sequences locally, there are two ways to do so (you must be emulating a VT terminal before executing these sequences; see page 79):

- Place Reflection into local mode (press Alt+M to bring up the modes keys, and use the mouse to click[*] on the REMOTE MODE label to remove the asterisk). Then, type the rest of the control sequence from the keyboard.

  As an example, the control sequence to move the cursor forward is `CSI<n>C`. To move the cursor position forward seven characters, press Esc and type `[7C`. (What you type does not display—Reflection recognizes the sequence as a control sequence, and performs its action.)

- Execute the Reflection Display method in the Reflection command line.

  Using the above example to move the cursor, press Alt+L to open the Reflection command line, type the following, then press Enter↵:

  ```
  Display Chr (27) & "[7C"
  ```

[*] Pressing a function key, in this instance F4, has no effect in VT emulation (see page 82).

Reflection Private Sequences

Reflection offers its own set of control sequences, explained below.

Invoke a Reflection Command

```
DCS1234;Ps{<cmd>ST
```

Invoke Reflection <*command*> to be executed as if it had been entered from the command line or a maco, and return the completion code to the host (the command should not be terminated by a carriage return).

The *Ps* parameter specifies the completion code to return, as shown in the table below.

<Ps>	**<Completion Code>**
0	(Default) Execute Reflection command and return a completion code.
1	Execute Reflection command and always return a completion code.
2	Execute Reflection command and never return a completion code.

Reset Communications

```
ESC&bR
```

Reset data communications.

Selecting a Terminal Class and Operating Level

You can configure Reflection to emulate a VT-series terminal using the control sequences described here.

Select a Terminal Class (HPANM)

ESC&kØ\	Switch to HP terminal class and reactivate the last saved or last activated HP configuration values.
ESC&k1\	Switch to VT terminal class and reactivate the last saved or last activated VT configuration values.

Select an Operating Level (HPANM)

Reflection emulates VT102 and VT52 terminals fully, and provides partial emulation of VT220 terminals. This flexibility allows you to emulate any of these terminals when an application you need to use is terminal-specific.

CSI61"p	Select VT102 emulation.
CSI62;1"p	Select VT200 emulation with 7-bit controls. All 8-bit controls are converted to their 7-bit equivalents before transmitting.
CSI62;Ø"p or CSI62;2"p	Select VT220 emulation with 8-bit controls.
CSI?2l	Select VT52 emulation (only VT52 control sequences are recognized).

When you change the operating level, Reflection performs a soft reset (HPSTR); see page 118.

Display Enhancements

You can use control sequences to select visual attributes for characters and lines. Visual character attributes change the way characters appear on the screen without changing the actual characters.

Select Graphic Rendition (SGR)

`CSI<n>;...<n>m`

Select a display enhancement. You can specify one or more character attributes at the same time. The *<n>* is a number representing a visual attribute listed in the following table. To select more than one attribute, separate each one with a semicolon. The default is Ø, which clears all attributes.

Clear all attributes	Ø		
Bold	1	Bold off	22
Underline	4	Underline off	24
Blinking	5	Blinking off	25
Inverse video	7	Inverse video off	27
Invisible	8	Invisible off	28

For example, use `CSI5;7m` to display blinking text in inverse video; use `CSI1m` to display text as bold. After selecting an attribute, Reflection applies that attribute to all new characters received. If you move characters by scrolling, the attributes move with the characters.

Line Attributes

The control sequences explained below set the height and width of lines.

Double-Width, Double-Height Line (HPDHL)

Esc#3 Select the top half of the line containing the cursor as double-width, double-height.

Esc#4 Select the bottom half of the line containing the cursor as double-width, double-height.

Only the top or bottom half of the characters appear on the line; for best results, make sure the characters used to generate the top half of the line match the characters used to generate the bottom half.

When the cursor is on a double-width line, the cursor becomes twice as wide so that it can move one character at a time. Cursor positioning sequences (such as CUP described on page 89) also use the enlarged characters; therefore, on an 80-column display, a CUP sequence can address columns 1–40 on a double-width line (a single-width line addresses columns 1–80).

Single-Width, Single-Height Line (HPSWL)

Esc#5 Select single-width, single-height line for the line containing the cursor. This line attribute is the default for all new lines on the screen.

Double-Width, Single-Height Line (HPDWL)

Esc#6 Select double-width, single-height line for the line containing the cursor. Any characters that extend beyond the right edge of the double-width, single-height line are lost, even if the line is returned to single-width.

When the cursor is on a double-width line, the cursor becomes twice as wide so that it can move one character at a time. Cursor positioning sequences (such as CUP described on page 89) also use the enlarged character cells; therefore, on an 80-column display, a CUP sequence can address columns 1–40 on a double-width line (a single-width line addresses columns 1–80).

Cursor Movement

The cursor movement sequences let you make the cursor visible or invisible, and move it around on the screen.

Text Cursor Enable Mode (HPTCEM)

`CSI?25h` Make the text cursor visible.

`CSI?25l` Make the text cursor invisible.

Cursor Position (CUP)

`CSI<r>;<c>H` Move the cursor to row <*r*> and column <*c*>. The default is `CSI1;1H`.

Horizontal and Vertical Position (HVP)

`CSI<r>;<c>f` The horizontal and vertical position sequences work the same as CUP (described above). However, Digital Equipment Corporation suggests that CUP be used instead to insure feature compatibility.

Cursor Forward (CUF)

`CSI<n>C` Move the cursor forward (right) <*n*> columns. If the cursor is at the right margin, it does not move.

Cursor Backward (CUB)

`CSI<n>D` Move the cursor backward (left) <*n*> columns. If the cursor is at the left margin, it does not move.

Cursor Up (CUU)

CSI<n>A

Move the cursor up <*n*> lines in the same column. If the cursor is within the scrolling region, it stops at the top margin. If the cursor is above the top margin of the scrolling region, it stops at the top of the screen.

Cursor Down (CUD)

CSI<n>B

Move the cursor down <*n*> lines in the same column. If the cursor is within the scrolling region, it stops at the bottom margin. If the cursor is below the bottom margin of the scrolling region, it stops at the bottom of the screen.

Cursor Back Tabulation (CBT)

CSI<n>Z

Move the cursor backward along the active line. The cursor moves to the <*n*>*th* preceding tab position. The cursor stops at column 1 if the <*n*>*th* tab stop is not found.

Horizontal Position Absolute (HPA)

CSI<n>G or
CSI<n>`

Move the cursor to column <*n*> without changing lines.

Horizontal Position Relative (HPR)

CSI<n>a

Move the cursor to column <*n*> without changing lines.

Vertical Position Absolute (VPA)

CSI<n>d

Move the cursor to line <*n*> without changing the column.

Vertical Position Relative (VPR)

CSI<n>e

Move the cursor down <*n*> lines without changing the column.

Index (IND)

ESCD or IND

Move the cursor down one row (*index*). If the cursor is at the bottom margin, the display scrolls up one line.

Next Line (NEL)

ESCE or
CSI<n>E or
NEL

Move the cursor to column 1 of the next line. If the cursor is at the bottom margin, the display scrolls up one line.

Cursor Next Line (CNL)

CSI<n>E

Move the cursor to the first position of the next line <n> lines down. If line <*n*> is below the last line, a scroll up is performed.

Previous Line (CPL)

CSI<n>F

Move the cursor to the first position of the line <*n*> lines up. If line <*n*> is above the first line, a scroll down is performed.

Reverse Index (RI)

ESCM or RI

Move the cursor up one row (*reverse index)*. If the cursor is at the top margin, the display scrolls down one line and a blank line is inserted.

Screen Display

Screen display control sequences affect how the screen looks, and whether to echo characters typed from the keyboard.

Screen Reverse/Normal (HPSCNM)

`CSI?5h` Set screen to inverse video.

`CSI?5l` Set screen to normal video.

Scrolling Mode (HPSCLM)

`CSI?4h` Set smooth scrolling, limiting the speed at which new lines appear on the screen (this produces a slower scroll).

`CSI?4l` Set jump scrolling. Reflection adds lines to the display as fast as it receives them from the host.

Local Echo: Send/Receive Mode (SRM)

`CSI12h` Turn off local echo. Reflection sends characters only to the host. If the host echoes characters, then they appear on the display.

`CSI12l` Turn on local echo. Characters entered from the keyboard are sent to both the host and the display. If the host echoes characters, the characters appear twice on your display.

Moving the Screen

The following control sequences move the position of the screen window in display memory. The sequences are disabled if top and bottom margins are set with HPSTBM (see page 94).

Home Up (HU)

`CSI>Øs` Move to the top of display memory (home up).

Home Down (HD)

`CSI>1s` Move to the bottom of display memory (home down).

Next Page (NP)

`CSI<n>U` Move forward <*n*> pages.

Previous Page (PP)

`CSI<n>V` Move backward <*n*> pages.

Scroll Up (SU)

`CSI<n>S` Scroll the display up <*n*> lines.

Scroll Down (SD)

`CSI<n>T` Scroll the display down <*n*> lines.

Controlling Display Format

The following control sequences allow you to change the width of the display, the height of the scrolling region, and specify whether or not the cursor can move outside the scrolling margins.

Setting 80 or 132 Columns (HPCOLM)

`CSI?3h` Set the left margin to 1 and the right margin to 132.

`CSI?3l` Set the left margin to 1 and the right margin to 80.

If you change the HPCOLM setting, the display clears and the cursor moves to the upper-left corner.

Set Scrolling Region (HPSTBM)

The scrolling region is the area between the top and bottom margins. This is the area that moves during vertical scrolling. When the margins are set, the cursor moves to the home position as determined by the origin mode. Rows are counted from 1 at the top.

`CSI<t>;<b>r` Set the top margin to row <*t*> and the bottom margin to row <*b*>. The <*t*> parameter must be at least two less than <*b*>.

Origin Mode (HPOM)

`CSI?6h` Set the first row in the scrolling region as the home position. The cursor cannot move outside the margins.

`CSI?6l` Set the upper-left corner of the screen as the home position. The cursor can move outside the margins.

The HPOM sequences allow cursor addressing relative to the scrolling margins or the complete display. Starting Reflection resets origin mode.

Editing

You can insert and delete lines in the scrolling region—the area on the screen between the top and bottom margins. (Use the HPSTBM control sequence above to set the scrolling region.)

Insert/Replace Mode (IRM)

`CSI4h` Insert mode. Characters are inserted at the cursor position, and all characters to the right of the cursor move one column to the right.

`CSI4l` Replace mode. Characters replace characters at the cursor position.

Delete Lines (DL)

`CSI<n>M` Delete <*n*> lines at the cursor position and shift lower lines up. Blank lines with normal character attributes are added at the bottom of the scrolling region.

Insert Line (IL)

`CSI<n>L` Insert <*n*> blank lines at the cursor position.

Delete Character (DCH)

`CSI<n>P` Delete <*n*> characters, starting with the cursor position and then deleting characters to the right. As characters are deleted, characters to the right of the cursor move left.

Insert Character (ICH)

`CSI<n>@` Insert <*n*> space characters before the cursor position on the current line only. This sequence is available only in VT220 mode.

Erasing Text

The following control sequences can affect data inside or outside the scrolling region: they are not restricted by margins.

Erase in Display (ED)

`CSIØJ` Erase from the cursor to the end of the screen.

`CSI1J` Erase from the top of the display to the cursor.

`CSI2J` Erase all of the display.

The ED sequences retain display memory.

Erase in Line (EL)

`CSIØK` Erase from the cursor position to the end of the line.

`CSI1K` Erase from the beginning of the line to the cursor position.

`CSI2K` Erase the entire line the cursor is on.

The EL sequences erase characters on the line the cursor is on, clearing all character attributes from the erased character positions.

Erase Character (ECH)

`CSI<n>X` Erase <*n*> characters from the cursor position to the right (without moving the cursor). A value of Ø or 1 erases one character. ECH clears character attributes from erased cursor positions. This sequence is available only in VT220 mode.

Selective Erase, VT220 Mode

With selective erase, you can only erase characters that are defined as erasable.

Select Character Attributes (HPSCA)

`CSIØ"q` or `CSI2"q` Characters following either of these control sequences will be erased by a selective erase.

`CSI1"q` Protects characters from erasure by the selective erase control sequences (see below). The characters can still be erased by a normal erase sequence.

These sequences define characters as erasable or not erasable. It does not affect visual character attributes set by display enhancements (see page 87).

The selective erase control sequences cannot erase characters defined as not erasable.

Selective Erase in Display (HPSED)

`CSI?ØJ` Erase unprotected characters from the cursor through the end of the screen. Display memory above the top of the screen is retained.

`CSI?1J` Erase unprotected characters from the top of the screen to the cursor. Display memory above the top of the screen is retained.

`CSI?2J` Erase unprotected characters from the entire screen. Display memory above the top of the screen is retained.

Selective Erase in Line (HPSEL)

`CSI?ØK` Erase unprotected characters from the cursor through the end of the line.

`CSI?1K` Erase unprotected characters from the beginning of the line through the cursor.

`CSI?2K` Erase unprotected characters from the complete line.

Keyboard

Keyboard control sequences enable the keyboard modes listed below.

Keyboard Action Mode (KAM)

`CSI2h` Lock the keyboard so it cannot send characters to the host. The padlock icon appears to the right of the Row/Column indicators in Reflection's status bar while the keyboard is locked, and the PC ignores all keystrokes that send characters to the host. Typing while the keyboard is locked causes Reflection to beep.

`CSI2l` Unlock the keyboard.

Linefeed/New Line Mode (LNM)

`CSI20h` Turn on new line mode. When you press Enter↵, Reflection sends both a carriage return and a linefeed. When Reflection receives a LF, FF, or VT character, it moves the cursor to the first column of the next line.

`CSI20l` Turn off new line mode. The Enter↵ key sends only a CR character. A received LF, FF, or VT character moves the cursor down one line in the current column.

Autorepeat Mode (HPARM)

`CSI?8h` Turn on autorepeat. When a key is held down, it repeatedly sends a character until released.

`CSI?8l` Turn off autorepeat.

Autowrap Mode (HPAWM)

`CSI?7h` Enable autowrap. Received characters automatically wrap to the next line when the cursor reaches the right margin of the display.

`CSI?7l` Disable autowrap. When the cursor reaches the right margin, it does not move. Additional characters overwrite the character at the right margin until the cursor is moved explicitly.

Cursor Key Mode (HPCKM)

`CSI?1h` Cursor keys send special application sequences.

`CSI?1l` Cursor keys send normal cursor positioning sequences.

The codes sent by the cursor keys are listed on page 115.

Keypad Mode (HPKPAM and HPKPNM)

`ESC=` Numeric keypad keys send application-specific sequences (HPKPAM).

`ESC>` Numeric keypad keys send numeric characters (HPKPNM), and the top row of numeric keys (NumLock, /, *, and -) send the PF1 through PF4 codes.

The codes sent by the keys on the keypad are listed on page 117.

Display Controls

The following control sequences determine whether control characters are executed or displayed.

Control Representation Mode (CRM)

CSI3h Display most of the ASCII control characters on the screen as if they were normal text, but do not execute them. However, when you press Enter↵, a carriage return and linefeed are executed in addition to a CR character being displayed. Also, LF, VT, and FF display the appropriate symbol and also cause a LF and CR to be performed.

CSI3l Control characters are executed and not displayed.

Control Characters

Control characters govern cursor movement, data communications operations, and other terminal actions. They include C0 controls with decimal values 0–31, and C1 controls with decimal values 128–159.

Normally, control characters do not display. However, if CRM is set (see above), control characters appear as a two-letter mnemonic. The following table shows how to enter control characters; Reflection recognizes only those codes denoted by an asterisk (*).

CO Controls (Decimal 0-31)

Decimal Value	Keystroke	Display	Definition
0	Ctrl+Shift+2	NU	Null (Ignored)
1	Ctrl+A	SH	Start of header
2	Ctrl+B	SX	Start of text
3	Ctrl+C	EX	End of text
4	Ctrl+D	ET	End of transmission
5*	Ctrl+E	EQ	Enquiry
6	Ctrl+F	AK	Acknowledge

Decimal Value	Keystroke	Display	Definition
7*	Ctrl+G	BL	Bell
8*	Ctrl+H	BS	Backspace
9*	Ctrl+I	HT	Horizontal tab
10*	Ctrl+J	LF	Linefeed
11*	Ctrl+K	VT	Vertical tab
12*	Ctrl+L	FF	Form feed
13*	Ctrl+M	CR	Carriage return
14*	Ctrl+N	SO	Shift out
15*	Ctrl+O	SI	Shift in
16	Ctrl+P	DL	Data link escape
17*	Ctrl+Q	D1	Device control 1 (XON)
18	Ctrl+R	D2	Device control 2
19*	Ctrl+S	D3	Device control 3 (XOFF)
20	Ctrl+T	D4	Device control 4
21	Ctrl+U	NK	Negative acknowledge
22	Ctrl+V	SY	Synchronous idle
23	Ctrl+W	EB	End of transmission block
24*	Ctrl+X	CN	Cancel
25	Ctrl+Y	EM	End of medium
26	Ctrl+Z	⸮	Substitute
27*	Ctrl+[	EC	Escape
28	Ctrl+E	FS	Field separator

Decimal Value	Keystroke	Display	Definition
29	Ctrl+]	GS	Group separator
30	Ctrl+Shift+6	RS	Record separator
31	Ctrl+Shift+-	US	Unit separator

C1 Controls (Decimal 128-159)

Decimal Value	7-bit Equivalent	Display	Definition
128		80	Ignored
129		81	Ignored
130		82	Ignored
131		83	Ignored
132*	ESC D	IN	Index
133*	ESC E	NEL	Next line
134		SS	Start selected area
135		ES	End selected area
136*	ESC H	HS	Horizontal tab set
137		HJ	Horizontal tab with justification
138		VS	Vertical tab set
139		PD	Partial line down
140		PU	Partial line up
141*	ESC M	RI	Reverse index
142*	ESC N	S2	Single shift 2
143*	ESC O	S3	Single shift 3
144*	ESC P	DC	Device control string
145		P1	Private use 1
146		P2	Private use 2

Decimal Value	7-bit Equivalent	Display	Definition
147		SE	Ignored
148		CC	Cancel character
149		MW	Message waiting
150		SP	Start protected area
151		EP	End protected area
152		98	Ignored
153		99	Ignored
154		9A	Ignored
155*	ESC [	CS	Control sequence introducer
156*	ESC \	ST	String terminator
157		OS	Operating system command
158		PM	Privacy message
159		AP	Application program command

Printer and Disk Control

If a printer is configured, the following control sequences send data to the printer. If a disk file has been selected as the output device, data is sent to both the printer and disk file.

Print Form Feed (HPPFF)

`CSI?18h` A form feed is sent to the printer after a print-screen operation.

`CSI?18l` No form feed is sent to the printer after a print-screen operation.

Printer Extent Mode (HPPEX)

`CSI?19h` Print the entire screen in a print-screen operation.

`CSI?19l` Print just the scrolling region (data inside the top and bottom margins) during a print-screen operation.

Log to Printer (MC)

`CSI?5i` Turn on logging to the printer. Data received into display memory is sent to the printer one line at a time. Lines must be delimited by `LF`, `FF`, `IND`, or `VT` to be recognized as lines. The delimiting character is also sent to the printer.

`CSI?4i` Turn off logging to the printer.

Pass Through to Printer (MC)

`CSI5i` Turn on pass-through mode (also called *printer controller mode*), allowing characters to be sent directly to the printer. Data received from datacomm is passed through to the printer without appear-ing on the screen. All codes except `NUL`, `DC1`, `DC3`, `CSI5i`, and `CSI4i` are passed to the printer.

`CSI4i` Turn off pass-through mode and return to normal (that is, no logging to printer).

Print Screen (MC)

`CSIi` If the print extent mode (HPPEX) is set to the scrolling region, only the scrolling region is printed. If the **Auto formfeed** check box is selected in the Print Setup dialog box (this is the default setting), a `FF` is sent to the printer after the print-screen operation.

Print Cursor Line (MC)

`CSI?1i` Print the line that currently contains the cursor.

Requests and Reports

Reflection and the host exchange device attribute (DA) sequences to provide the host with basic operating information. The host uses this information to make the best use of Reflection's features. It also allows the host to determine the cause of any communication errors.

There are two types of DA exchanges between the host and Reflection: primary and secondary.

Primary DA Request (DA)

`CSIc` The host can query Reflection for *primary device attributes*—the service code, conformance level, and basic attributes—using this control sequence. Reflection replies according to the setting of the **Terminal ID** list in the Advanced VT Options dialog box.

The reply can be one of the following:

Terminal ID	Reflection Reply
VT100	`ESC[?1;2c`
VT101	`ESC[?1;0c`
VT102	`ESC[?6c`
VT220	`ESC[?62;1;2;6;7;8;9c`

Secondary DA Request (DA)

`CSI>c` or `CSIØc` Host request for secondary device attributes: Reflection's identification code and hardware options.

`CSI>1;11;Øc` Reflection response to the above request.

Operating Status (DSR)

CSI5n Host request for Reflection’s operating status.

CSIØn Reflection response to the above request when its operating status is normal.

Cursor Position (CPR)

CSI6n Host cursor position request.

CSI<r>;<c>R Reflection response to the above request with the row/column cursor position.

Printer Status (DSR)

CSI?15n Host request for the current printer status.

CSI?13n Reflection response to the above request, indicating that the host cannot print to a PC printer (there is no an active printer).

CSI?1Øn Reflection response to the above request, indicating that the host can print to a PC printer and the printer is ready.

Selecting and Mapping Character Sets

Character sets are selected by control sequences and by the Ctrl+N and Ctrl+O keystrokes.

As an example of loading and using a different character set, do the following:

1. Place Reflection into local mode (press Alt+M to bring up the modes keys, and use the mouse to click* on the REMOTE MODE label to remove the asterisk).
2. Type `ESC)Ø`.
3. Press Ctrl+N.
4. Type `abcd`.

Notice the results: the characters on the screen are not the usual display representation for these keys.

Now press Ctrl+O and type `abcd` again. The characters on the screen are the normal letters.

Character Set Support

During VT emulation, Reflection supports six character sets:

- ASCII
- DEC Supplemental Graphic
- DEC Special Graphic
- DEC national replacement sets
- HP line drawing
- Downloadable character sets

The DEC Supplemental Graphic character set is composed of symbols and characters for the English language and many Western European languages.

* Pressing a function key, in this instance F4, has no effect in VT emulation (see page 82).

Together, the ASCII character set and the DEC Supplemental Graphic set comprise the DEC Multinational character set. See the character set summaries starting on page 123. National replacement character (NRC) sets contain most of the same characters as the ASCII set, plus characters from the supplemental graphic sets used by specific national languages. NRC sets are used primarily in 7-bit operating environments that can't access supplemental characters. See page 128 for a figure of replacement characters.

Character Set Storage

Each character set contains 94 or 96 displayable characters. Reflection can store two character sets at a time, and a character may appear in more than one set.

To store two character sets at once, Reflection uses two tables, each containing 128 spaces for characters. The first 32 spaces in each table are reserved for control characters, such as CR, LF, ESC, and DCS; these spaces are called *C0* (control zero) and *C1* (control one). The remaining 96 spaces are for the graphical characters—letters, punctuation marks, and other symbols—from the sets listed above; these spaces are called *GL* (graphic left) and *GR* (graphic right).

The C0 and GL characters make up one table, and the C1 and GR characters make up the other table. Each table contains no more than 128 characters; therefore, an entire character set can be encoded in seven data bits.

Any character set can be stored in either the GL or GR table. The combination of a specific GL and GR character set (such as ASCII and DEC Supplemental Graphic) is called the *in-use table*.

When a host application displays a character on the screen, it selects the character by number from the in-use table. The characters in C0 and GL are selected by a 7-bit code, with the high bit of each 8-bit character code set to 0. The characters in C1 and GR are selected by an 8-bit code; the high bit of the 8-bit character code is set to 1.

In 8-bit operating environments, all characters in both tables can be accessed, by either a 7-bit or an 8-bit code. In 7-bit operating environments, however, only characters in C0 and GL can be accessed, since characters from this table are selected by 7-bit codes.

This restriction on 7-bit codes also applies to the VT52 and VT100 emulation modes. Both of these terminals send only 7-bit characters, and cannot send or display characters stored in C1 and GR.

Selecting a Character Set

Character sets are selected and mapped into Reflection's in-use table, which defines the characters Reflection can display. The in-use table is divided into four parts, based on the location of the characters in the character code table:

C0 control character set	Decimal 0 through 31
GL (left graphic) set	Decimal 32 through 127
C1 control character set	Decimal 128 through 159
GR (right graphic) set	Decimal 160 through 255

The C1 and GR character sets can be accessed only in 8-bit operating environments. They cannot be accessed in VT100 mode, for example.

When you start Reflection, it places the following default character sets in the in-use table (if the terminal type is VT102 or lower, GL and GR are both ASCII):

- ASCII in GL
- DEC Supplemental Graphic in GR

There are two steps to selecting a different character set:

1. Designate the set as G0, G1, G2, or G3. You can designate up to four character sets and have them ready for mapping in the in-use table.
2. Map the set. After mapping the set in the in-use table, you can display and send any character from that set using 8-bit code tables.

Designate Character Set (HPDCS)

You designate a character set as G0 through G3 by using the following control sequence:

`EsC<Gn><set>` Designate an available character set.

The values for <*Gn*> and <*set*> are:

<Gn>	Meaning
(	Designate G0 as set
)	Designate G1 as set
*	Designate G2 as set (only available in VT220 mode)
+	Designate G3 as set (only available in VT220 mode)

<set>	Character set
B	ASCII
A	British
9 or Q	Canadian
4	Dutch
5 or C	Finnish
R	French
K	German
3	HP line drawing set
Y	Italian
` or E or 6	Norwegian/Danish
%6	Portuguese
Z	Spanish

<set>	Character set
Ø	Special Graphic (line drawing)
<	Supplemental Graphic (multinational)
7 or H	Swedish
=	Swiss
<name>	Downloadable character set

Mapping Character Sets

After you designate a character set as G0, G1, G2, or G3, you must map the set into the left graphic set (GL) or right graphic set (GR) of the in-use table. The current values of GL are used to translate all incoming codes in the range 32 to 127 (characters with no high bit set), and the GR character set translates codes 160–255 (characters with the high bit set). *Locking shifts* and *single shifts* are used to map character sets.

Locking Shifts

When you use a locking shift, the character set remains in GL or GR until you use another locking shift.

The following locking shift sequences are available in all emulation modes:

Locking Shift	Keystroke	Control Code	Sequence
LS0 (locking shift 0)	Ctrl+O	SI	Map G0 into GL
LS1 (locking shift 1)	Ctrl+N	SO	Map G1 into GL

The following locking shift sequences are available in VT220 emulation mode:

Locking Shift	Control Code	Sequence
LS1R (locking shift 1, right)	ESC~	Map G1 into GR
LS2 (locking shift 2)	ESCn	Map G2 into GL
LS2R (locking shift 2, right)	ESC}	Map G2 into GR
LS3 (locking shift 3)	ESCo	Map G3 into GL
LS3R (locking shift 3, right)	ESC|	Map G3 into GR

Single Shifts

When you want to display just the next character using a different character set, use a single shift. A single shift maps the G2 or G3 set into GL for one displayable character. GL then automatically reverts to its previous mapping. Reflection has two single-shift control sequences:

Single Shift	8-Bit Character	7-Bit Equivalent	Sequence
Single shift 2	SS2	ESCN	Map G2 for next character into GL
Single shift 3	SS3	ESCO	Map G3 for next character into GL

National Replacement Characters (HPNRCM)

Only one national replacement character set can be used at a time. This feature is used in a 7-bit environment.

`CSI?42h` Reflection uses 7-bit characters from the specified national replacement set.

`CSI?42l` Reflection uses 7- and 8-bit characters.

Downloadable Character Set

You can design and download a soft character set, also called a Dynamically Redefinable Character Set (DRCS), from the host to Reflection. This feature works only in VT220 mode.

After designing and downloading a soft character set, you designate and map it the same way you designate and map hard character sets. See page 110.

The sequence that loads a soft character set is:

```
DCSPfn;Pcn;Pe;Pcmw;Pw;Pt{ Dscs UUUUUUUU/LLLLLLLL;...ST
```

You can load only one soft character set at a time, but you can load two renditions of the same set: an 80-column rendition and a 132-column rendition. In fact, you *should* load both column settings of your soft character set, so that Reflection can select the appropriate rendition based on the column width of the display and the number of display rows.

`Pfn` Font buffer to load. There is only one DRCS buffer and valid values are Ø and 1, which both refer to the same buffer.

`Pcn` Position in the ASCII table in which to load the first soft character. For a 94-character set, a value of Ø or 1 loads the first soft character as decimal 33 of the table. For a 96-character set, a value of Ø loads the first soft character as decimal 32 of the table. The VT200-series terminals allow only a 94-character set, and the valid range for this parameter is 1–94.

`Pe` Erase control. If an 80-column font is specified (`Pw` is Ø or 1), a `Pe` value of Ø or 2 erases the entire character set (all widths and renditions). If a 132-column font is specified (`Pw` is 2), a `Pe` value of Ø or 2 erases the 132-column font only. A `Pe` value of 1 erases only characters being replaced.

`Pcmw` Character matrix width. If this parameter is omitted, Reflection uses the default width for the current terminal type.

Ø	default width for 80/132 columns
1	not used
4	7 ✩ 10 pixel cell

`P` Font width (columns per line). The values Ø (the default) and 1 select 80 columns per line; 2 selects 132 columns.

`Pt` Text or full-cell font. With a text font, Reflection left-justifies the character within the cell, and leaves the rightmost columns blank. A full-cell font can fill the entire cell with pixels. Ø (the default) and *1* define a text font. *2* defines the font as full-cell.

`{` The final character of the string, marking the end of the parameter list and indicating that the string is a HPDLD function.

`Dscs` Soft character set name. The format is *IF*, where *I* can be Ø, 1, or 2 intermediate characters from the range decimal 32 to 47 in the ASCII chart, and *F* is a final character in the range decimal 48 to 126. For example, `%$R` could define a soft character set that is currently unused. The recommended default for a soft character set name is `<SP>@`.

`UUUUUUUU/LLLLLLLL`

The ASCII characters that represent the sixel patterns defining each character. Reflection receives the code for each soft character as a series of sixels; each sixel is a 6-bit binary code that represents a vertical column of 6 pixels on screen. Each bit in a sixel corresponds to a pixel on the screen. Sixel bit patterns are separated with a semicolon. Your character set can have from 1 to 94 patterns. *U...U* represents the sixels for the top half of the soft character. The forward slash (/) advances the sixel pattern to the bottom half of the character, and *L...L* represents the sixels in the bottom half.

After loading your soft character set, you designate the set as G0, G1, G2, or G3 using the HPDCS sequence on page 110. You can then map it into GL or GR of the in-use table using the sequence on page 111.

Keyboard Codes

The following tables list the control sequences sent by the arrow keys, editing and function keys, and numeric keypad.

Codes Sent by the Arrow Keys

The following table shows codes sent by the arrow keys, depending on the setting of the HPCKM sequence (see page 99). The codes sent in VT emulation mode apply only to VT102 and VT220 modes; the VT52 terminal is not compatible with VT emulation mode.

	VT emulation mode		VT52 Mode
Key	**Cursor**	**Application**	**Cursor or Application**
↑	CSIA	SS3A	ESCA
↓	CSIB	SS3B	ESCB
→	CSIC	SS3C	ESCC
←	CSID	SS3D	ESCD

Codes Sent by the Editing and VT Function Keys

The following table shows the codes sent by the six editing keys and the function keys along the top of the keyboard in VT220 mode (three of the keys are recognized in VT52 and VT102 mode, as noted).

VT Keystroke	Generated Code
Find	CSI1~
Insert Here	CSI2~
Remove	CSI3~
Select	CSI4~
Prev Screen	CSI5~
Next Screen	CSI6~
F6	CSI17~
F7	CSI18~

VT Keystroke	Generated Code
F8	CSI19~
F9	CSI20~
F10	CSI21~
F11 (ESC)	CSI23~ (ESC in VT52VT100 modes)
F12 (BS)	CSI24~ (BS in VT52VT100 modes)
F13 (LF)	CSI25~ (LF in VT52VT100 modes)
F14	CSI26~
F15 (Help)	CSI28~
F16 (Do)	CSI29~
F17	CSI31~
F18	CSI32~
F19	CSI33~
F20	CSI34~

Codes Sent by the Numeric Keypad

The following table shows the codes sent by the numeric keypad keys, depending on the setting of the keypad mode sequences HPKPAM and HPKPNM (see page 99). The codes sent in VT emulation mode apply only to VT102 and VT220 modes; the VT52 terminal is not compatible with VT emulation mode.

	VT Emulation Mode		VT52 Mode	
Key	**Numeric**	**Application**	**Numeric**	**Application**
Ø	Ø	SS3p	Ø	ESC?p
1	1	SS3q	1	ESC?q
2	2	SS3r	2	ESC?r
3	3	SS3s	3	ESC?s
4	4	SS3t	4	ESC?t
5	5	SS3u	5	ESC?u
6	6	SS3v	6	ESC?v
7	7	SS3w	7	ESC?w
8	8	SS3x	8	ESC?x
9	9	SS3y	9	ESC?y
–	(minus)	SS3m	-	ESC?m
,	(comma)	SS3l	,	ESC?l
.	(period)	SS3n	.	ESC?n
Enter ↵[a]	CR or CRLF	SS3M	CR or CRLF	ESC?M
PF1	SS3P	SS3P	ESCP	ESCP
PF2	SS3Q	SS3Q	ESCQ	ESCQ
PF3	SS3R	SS3R	ESCR	ESCR
PF4	SS3S	SS3S	ESCS	ESCS

[a] Same as the Return key (either a CR or a CRLF)

Resetting Reflection

Use the following sequences to reset Reflection.

Reset to Initial State (RIS)

ESCc Reset Reflection to its last saved setup (also called a *hard reset*).

Soft Terminal Reset (HPSTR)

CSI!p Set the following default settings (only available during VT220 emulation):

Feature	**Reset value**
Character set	Defaults
Character attribute	Normal
Cursor key mode	Normal
Cursor type	Underline
End-of-line wrap	Off
Insert mode	Off
Keyboard	Unlocked
Keypad mode	Normal
Origin mode	Off
Top/Bottom margin	Whole screen
Saved cursor state	Defaults
Selective erase	Normal

Save Cursor State (HPSC)

ESC7 Save the following cursor state settings in Reflection's memory:

- Cursor position
- Character attributes set by SGR
- Character sets in GL and GR
- Whether autowrap is set
- State of origin mode (HPOM)
- Selective erase attributes
- Any SS2 or SS3 functions sent

Restore Cursor State (HPRC)

ESC8 Restore the cursor state saved with HPSC. If no previous HPSC was performed, this sequence will:

- Home the cursor.
- Reset origin mode (HPOM).
- Turn off all character attributes.
- Map the ASCII character set into GL and the DEC Supplemental Graphic set into GR.

Tab Clear (TBC)

CSI3g Clear all tab stops.

CSIg Clear only the tab stop at the cursor.

VT52 Control Sequences

The control sequences available when you select VT52 as your terminal type are limited to those described below.

VT Emulation Mode

`CSI?2l`	Select VT52 emulation.
`ESC<`	Exit VT52 mode, and enter VT100 mode.

Character Sets

`ESCF`	Select the graphics character set.
`ESCG`	Select the ASCII character set.

Controller Mode

`ESCW`	Start controller mode.
`ESCX`	Stop controller mode.

Cursor Positioning

`ESCA`	Move the cursor up.
`ESCB`	Move the cursor down.
`ESCC`	Move the cursor right.
`ESCD`	Move the cursor left.
`ESCH`	Home the cursor (move the cursor to the top of the display).
`ESCY<r><c>`	Move the cursor to row <*r*>, column <*c*>. The parameters <*r*> and <*c*> are equal to the character whose binary value is the desired row or column value plus 31. For example, row 2 and column 4 would be `ESCY!#`.

Erasing

ESCJ Erase from the cursor to the end of the screen.

ESCK Erase from the cursor to the end of the line.

Identification Request

ESCZ Host issued identification request.

ESC/Z Reflection's reply to host issued identification request, as a VT100 emulating a VT52 terminal.

Keypad Mode

ESC= Numeric keypad keys send special application escape sequences.

ESC> Numeric keypad keys send the normal numeric values. The + or Enter ↵ key acts as Enter ↵, and PrtScrn sends a comma. See page 117 for the codes sent by the numeric keypad.

Log to Printer (Auto Print)

ESC^ Turn on log to printer.

ESC_ Turn off log to printer.

Printing

ESCV Print the cursor line.

ESC] Print the screen.

Reverse Linefeed

ESCI Reverse linefeed; the cursor moves up one row in the current column.

Character Sets Used During VT Emulation

The following character sets are available when running Reflection for HP in VT emulation mode (explained on page 79):

- ASCII character set
- DEC supplemental character set
- DEC special graphic character set
- The national replacement set
- Display control character set

The HP line drawing set (page 65) is also available.

ASCII Character Set

Decimal →
Hex →

32 20	(space)	48 30	0	64 40	@	80 50	P	96 60	`	112 70	p
33 21	!	49 31	1	65 41	A	81 51	Q	97 61	a	113 71	q
34 22	"	50 32	2	66 42	B	82 52	R	98 62	b	114 72	r
35 23	#	51 33	3	67 43	C	83 53	S	99 63	c	115 73	s
36 24	$	52 34	4	68 44	D	84 54	T	100 64	d	116 74	t
37 25	%	53 35	5	69 45	E	85 55	U	101 65	e	117 75	u
38 26	&	54 36	6	70 46	F	86 56	V	102 66	f	118 76	v
39 27	’	55 37	7	71 47	G	87 57	W	103 67	g	119 77	w
40 28	(	56 38	8	72 48	H	88 58	X	104 68	h	120 78	x
41 29	)	57 39	9	73 49	I	89 59	Y	105 69	i	121 79	y
42 2A	*	58 3A	:	74 4A	J	90 5A	Z	106 6A	j	122 7A	z
43 2B	+	59 3B	;	75 4B	K	91 5B	[	107 6B	k	123 7B	{
44 2C	,	60 3C	<	76 4C	L	92 5C	\	108 6C	l	124 7C	|
45 2D	—	61 3D	=	77 4D	M	93 5D	]	109 6D	m	125 7D	}
46 2E	.	62 3E	>	78 4E	N	94 5E	^	110 6E	n	126 7E	~
47 2F	/	63 3F	?	79 4F	O	95 5F	_	111 6F	o	127 7F	DEL

DEC Supplemental Graphic Character Set

Decimal →
Hex →

160	A0		176	B0	°	192	C0	À	208	D0	⸮	224	E0	à	240	F0	⸮
161	A1	¡	177	B1	±	193	C1	Á	209	D1	Ñ	225	E1	á	241	F1	ñ
162	A2	¢	178	B2	2	194	C2	Â	210	D2	Ò	226	E2	â	242	F2	ò
163	A3	£	179	B3	3	195	C3	Ã	211	D3	Ó	227	E3	ã	243	F3	ó
164	A4	⸮	180	B4	⸮	196	C4	Ä	212	D4	Ô	228	E4	ä	244	F4	ô
165	A5	¥	181	B5	µ	197	C5	Å	213	D5	Õ	229	E5	å	245	F5	õ
166	A6	⸮	182	B6	¶	198	C6	Æ	214	D6	Ö	230	E6	æ	246	F6	ö
167	A7	§	183	B7	·	199	C7	Ç	215	D7	Œ	231	E7	ç	247	F7	œ
168	A8	¤	184	B8	⸮	200	C8	È	216	D8	Ø	232	E8	è	248	F8	ø
169	A9	©	185	B9	1	201	C9	É	217	D9	Ù	233	E9	é	249	F9	ù
170	AA	ª	186	BA	º	202	CA	Ê	218	DA	Ú	234	EA	ê	250	FA	ú
171	AB	«	187	BB	»	203	CB	Ë	219	DB	Û	235	EB	ë	251	FB	û
172	AC	⸮	188	BC	1/4	204	CC	Ì	220	DC	Ü	236	EC	ì	252	FC	ü
173	AD	⸮	189	BD	1/2	205	CD	Í	221	DD	Ÿ	237	ED	í	253	FD	ÿ
174	AE	⸮	190	BE	⸮	206	CE	Î	222	DE	⸮	238	EE	î	254	FE	⸮
175	AF	⸮	191	BF	¿	207	CF	Ï	223	DF	ß	239	EF	ï	255	FF	

▨ Not part of the character set.

DEC Special Graphic Character Set

Decimal →
Hex →

160 A0		176 B0	0	192 C0	@	208 D0	P	224 E0	◆	240 F0	⎺ SCAN 3
161 A1	!	177 B1	1	193 C1	A	209 D1	Q	225 E1	▒	241 F1	─ SCAN 5
162 A2	"	178 B2	2	194 C2	B	210 D2	R	226 E2	$\mathrm{H_T}$	242 F2	⎽ SCAN 7
163 A3	#	179 B3	3	195 C3	C	211 D3	S	227 E3	$\mathrm{F_F}$	243 F3	⎽ SCAN 9
164 A4	$	180 B4	4	196 C4	D	212 D4	T	228 E4	$\mathrm{C_R}$	244 F4	├
165 A5	%	181 B5	5	197 C5	E	213 D5	U	229 E5	$\mathrm{L_F}$	245 F5	┤
166 A6	&	182 B6	6	198 C6	F	214 D6	V	230 E6	°	246 F6	┴
167 A7	'	183 B7	7	199 C7	G	215 D7	W	231 E7	±	247 F7	┬
168 A8	(	184 B8	8	200 C8	H	216 D8	X	232 E8	$\mathrm{N_L}$	248 F8	│
169 A9	)	185 B9	9	201 C9	I	217 D9	Y	233 E9	$\mathrm{V_T}$	249 F9	≤
170 AA	*	186 BA	:	202 CA	J	218 DA	Z	234 EA	┘	250 FA	≥
171 AB	+	187 BB	;	203 CB	K	219 DB	[	235 EB	┐	251 FB	π
172 AC	,	188 BC	<	204 CC	L	220 DC	\	236 EC	┌	252 FC	≠
173 AD	—	189 BD	=	205 CD	M	221 DD	]	237 ED	└	253 FD	£
174 AE	.	190 BE	>	206 CE	N	222 DE	^	238 EE	┼	254 FE	·
175 AF	/	191 BF	?	207 CF	O	223 DF	(space)	239 EF	⎺ SCAN 1	255 FF	

▨ Not part of the character set.

National Characters in 7-Bit Operation

When configured as a VT220 terminal, Reflection uses the Supplemental Graphic Character Set shown on page 125 for national characters not included in the ASCII character set. Access to this character set requires an 8-bit data path.

When only 7 data bits are available, Reflection is operating in 7-bit mode; the 8th bit of each byte is used for parity and is not available for use as data. Since Supplemental Graphics characters use the 8th bit, they cannot be directly sent or received in 7-bit operations.

This problem is partially solved on some host systems by using a replacement set for a few national characters.

When emulating a VT-series terminal, the *national replacement characters* allow access to a limited number of national characters. Use the procedure below only if you are sure your host recognizes national replacement characters.

If you use a national replacement set, some ASCII characters that have been replaced by national characters cannot be used. Only multinational characters seen in the next figure can be used.

To use the 7-bit national replacement characters shown in the following figure, configure Reflection as follows:

1. On the Setup menu, click Terminal to open the Terminal Setup dialog box.
2. Click the Terminal Type tab and note that a VT-series terminal is selected.
3. Click the Emulation tab, then click Advanced to open the Advanced VT Options dialog box.
4. With a VT terminal type set in the **Terminal ID** list, select one of the languages (they are listed on the next figure) from the **National Replacement Set** list.
5. Select the **Use NRC (7-bit) Set** check box (this is unavailable if **None** is selected).
6. Click OK to close the Advanced VT Terminal Items dialog box.
7. Click OK to close the Terminal Setup dialog box.

National Replacement Set	Characters													
ASCII Decimal	**35**	**39**	**60**	**62**	**64**	**91**	**92**	**93**	**94**	**96**	**123**	**124**	**125**	**126**
U.S. ASCII	#	´	<	>	@	[	\	]	^	`	{	|	}	~
British (U.K.)	£	´	<	>	@	[	\	]	^	`	{	|	}	~
French	£	´	<	>	à	°	ç	§	^ **	`	é	ù	è	¨ **
German	£	´	<	>	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
Italian	£	´	<	>	§	°	ç	é	^ *	ù	à	ò	è	ì
Norwegian	#	´ *	<	>	@	Æ	Ø	Å	^	` *	æ	ø	å	¨
Danish	§	´ *	<	>	@	Æ	Ø	Å	^	` *	æ	ø	å	¨
Swedish	#	´	<	>	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Dutch	#	´	<	>	@	ç	\	§	^ *	` *	ƒ	|	′ *	¨ *
Swiss-German	£	´ *	é	è	à	°	ç	§	^ *	` *	ä	ö	ü	¨ *
Spanish-European	#	´	<	>	@	¡	Ñ	¿	°	` **	′	ñ	ç	¨ **
Spanish-Latin	#	´	<	>	@	¡	Ñ	¿	^	` **	′	ñ	ç	¨ **
Finnish	#	´	<	>	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Canadian-English	#	´	<	>	@	[	ç	]	^ *	` *	é	Ç	É	¨ *
Flemish (Belgian)	£	´	<	>	à	°	ç	§	^ **	`	é	ù	è	¨ **
Canadian-French	#	´	<	>	@	[	ç	]	^ *	` *	é	Ç	É	¨ *
Swiss-French	£	´ *	é	è	à	°	ç	§	^ *	` *	ä	ö	ü	¨ *
Line Drawing	┳	╤	╫	╪	╞	╡	- -	╨	-	╞	╡	- -	╨	-

* In 8-bit operations only, this diacritical mark is mute.

** In 7-bit and 8-bit operations, this diacritical mark is mute.

Display Control Character Set

The display control character set is available when the CRM sequence is enabled (see page 100).

Decimal →
Hex →

00 00 NU	16 10 DL	32 20 (space)	48 30 0	64 40 @	80 50 P	96 60 `	112 70 p
01 01 SH	17 11 D1	33 21 !	49 31 1	65 41 A	81 51 Q	97 61 a	113 71 q
02 02 SX	18 12 D2	34 22 "	50 32 2	66 42 B	82 52 R	98 62 b	114 72 r
03 03 EX	19 13 D3	35 23 #	51 33 3	67 43 C	83 53 S	99 63 c	115 73 s
04 04 ET	20 14 D4	36 24 $	52 34 4	68 44 D	84 54 T	100 64 d	116 74 t
05 05 EQ	21 15 NK	37 25 %	53 35 5	69 45 E	85 55 U	101 65 e	117 75 u
06 06 AK	22 16 SY	38 26 &	54 36 6	70 46 F	86 56 V	102 66 f	118 76 v
07 07 BL	23 17 EB	39 27 '	55 37 7	71 47 G	87 57 W	103 67 g	119 77 w
08 08 BS	24 18 CN	40 28 (	56 38 8	72 48 H	88 58 X	104 68 h	120 78 x
09 09 HT	25 19 EM	41 29)	57 39 9	73 49 I	89 59 Y	105 69 i	121 79 y
10 0A LF	26 1A ⸮	42 2A *	58 3A :	74 4A J	90 5A Z	106 6A j	122 7A z
11 0B VT	27 1B EC	43 2B +	59 3B ;	75 4B K	91 5B [	107 6B k	123 7B {
12 0C FF	28 1C FS	44 2C ,	60 3C <	76 4C L	92 5C \	108 6C l	124 7C |
13 0D CR	29 1D GS	45 2D -	61 3D =	77 4D M	93 5D]	109 6D m	125 7D }
14 0E SO	30 1E RS	46 2E .	62 3E >	78 4E N	94 5E ^	110 6E n	126 7E ~
15 0F SI	31 1F US	47 2F /	63 3F ?	79 4F O	95 5F _	111 6F o	127 7F DT
C0 CODES		GL CODES ASCII GRAPHIC					

Decimal → Hex →	128 80 8_0	144 90 D_C	160 A0 A_0	176 B0 °	192 C0 À	208 D0 Đ	224 E0 à	240 F0 ð
	129 81 8_1	145 91 P_1	161 A1 ¡	177 B1 ±	193 C1 Á	209 D1 Ñ	225 E1 á	241 F1 ñ
	130 82 8_2	146 92 P_2	162 A2 ¢	178 B2 ^{2}	194 C2 Â	210 D2 Ò	226 E2 â	242 F2 ò
	131 83 8_3	147 93 S_E	163 A3 £	179 B3 ^{3}	195 C3 Ã	211 D3 Ó	227 E3 ã	243 F3 ó
	132 84 I_N	148 94 C_C	164 A4 ¤	180 B4 ´	196 C4 Ä	212 D4 Ô	228 E4 ä	244 F4 ô
	133 85 N_L	149 95 M_W	165 A5 ¥	181 B5 µ	197 C5 Å	213 D5 Õ	229 E5 å	245 F5 õ
	134 86 S_S	150 96 S_P	166 A6 ¦	182 B6 ¶	198 C6 Æ	214 D6 Ö	230 E6 æ	246 F6 ö
	135 87 E_S	151 97 E_P	167 A7 §	183 B7 ·	199 C7 Ç	215 D7 ×	231 E7 ç	247 F7 ÷
	136 88 H_S	152 98 9_8	168 A8 "	184 B8 ¸	200 C8 È	216 D8 Ø	232 E8 è	248 F8 ø
	137 89 H_J	153 99 9_9	169 A9 ©	185 B9 ^{1}	201 C9 É	217 D9 Ù	233 E9 é	249 F9 ù
	138 8A V_S	154 9A 9_A	170 AA ª	186 BA º	202 CA Ê	218 DA Ú	234 EA ê	250 FA ú
	139 8B P_D	155 9B C_S	171 AB «	187 BB »	203 CB Ë	219 DB Û	235 EB ë	251 FB û
	140 8C P_U	156 9C S_T	172 AC ¬	188 BC 1/4	204 CC Ì	220 DC Ü	236 EC ì	252 FC ü
	141 8D R_I	157 9D O_S	173 AD –	189 BD 1/2	205 CD Í	221 DD Ý	237 ED í	253 FD ý
	142 8E S_2	158 9E P_M	174 AE ®	190 BE 3/4	206 CE Î	222 DE Þ	238 EE î	254 FE þ
	143 8F S_3	159 9F A_P	175 AF —	191 BF ¿	207 CF Ï	223 DF ß	239 EF ï	255 FF ÿ
	C1 CODES		GR CODES ISO LATIN-1 SUPPLEMENTAL					

SECTION 4

Programming Sequence Index

CHAPTER 8

HP Escape Sequence Index

Escape Sequence	Function	Page
Esc&ds<x>	Display enhancement with security	17
Esc&f...	Define user key	30
Esc&f-1E	Transmit HP Enter	29
Esc&f<x>E	Trigger function key	29
Esc&fØB	Store setup	35
Esc&f1B	Restore setup	35
Esc&f211P...	Configure Tab key	36
Esc&fR	Configure Tab key	36
Esc&f1m149P<!149>	Configure Enter ↵ key as Enter ↵	36
Esc&f1m149P<!154>	Configure Enter ↵ key as keypad Enter	36
Esc&j@	Enable user keys, no labels	29
Esc&jA	Display modes keys	29
Esc&jB	Display and enable user keys	29
Esc&jC	Remove message, replace labels	29
Esc&j<n>D	Select function key features	29
Esc&j<x>L...	Remove function key labels and display message	30
Esc&jR	Enable function key labels	30
Esc&jS	Disable function key labels	30
Esc&kØA	Auto linefeed off	12
Esc&k1A	Auto linefeed on	12
Esc&kØB	Block mode off	12
Esc&k1B	Block mode on	12
Esc&kØC	Caps lock off	12
Esc&k1C	Caps lock on	12
Esc&kØD	Margin bell off	12
Esc&k1D	Margin bell on	12
Esc&kØI	Set previous parity	13

Escape Sequence	Function	Page
ESC&k1I	No parity	13
ESC&kØK	Auto keyboard lock off	13
ESC&k1K	Auto keyboard lock on	13
ESC&kØL	Local echo off	13
ESC&k1L	Local echo on	13
ESC&kØM	Modify all mode off	13
ESC&k1M	Modify all mode on	13
ESC&kØN	SPOW latch off	13
ESC&k1N	SPOW latch on	13
ESC&kØP	Caps mode (TeleType) off	13
ESC&k1P	Caps mode (TeleType) on	13
ESC&kØR	Remote mode off	13
ESC&k1R	Remote mode on	14
ESC&kØS	Enable 80 column printing	27
ESC&kØZ	Modified data tags off	57
ESC&k1Z	Modified data tags on	57
ESC&kØ\	HP terminal mode	20
ESC&k1\	VT terminal mode	20
ESC&kØ[	Smooth scroll off	14
ESC&k1[	Smooth scroll on	14
ESC&kØ]	Select key off	12
ESC&k1]	Select key on	12
ESC&k2S	Enable 120 column printing	27
ESC&oF<cmd>CR	Reflection command without completion code	11
ESC&oX	Clear typeahead	11
ESC&p[<a>d]<b>D	Select destination device	27
ESC&p[<a>d][<b>d]<Y>	Copy data to destination device	27
ESC&p<x>^	Device status request	28

Escape Sequence	Function	Page
`Esc&p<x>p<y>u<z>C`	Perform action on device	28
`Esc&p<x>p<y>u14c`	Print standard (80 column)	27
`Esc&p<x>p<y>u16c`	Print standard (132 column)	27
`Esc&p<x>p2ØC`	Enable record mode	28
`Esc&p[<a>d][<b>d]<x>W<data string>`	Transfer bytes of string to destination device	28
`Esc&p[<a>d][<b>d]W<data string>`	Transfer string to destination device	28
`Esc&p9^`	Forms buffer status request	32
`Esc&p <>n9^`	Read forms cache directory	32
`Esc&p<<name>>n9^`	Forms buffer status request (by name)	32
`Esc&p9u<f#>p<<con>>L`	Store form, length unknown	32
`Esc&p9u<<name>>n<f#>p<<con>>L`	Store form using name, length unknown	32
`Esc&p9u<f#>p<len>L<con>`	Store form	32
`Esc&p9u<name>n<f#>p<len>L<con>`	Store form, using name	32
`Esc&p9u<form#>pF`	Display form	32
`Esc&p9u<form#>pL`	Purge form	32
`Esc&p<form#>p9^`	Forms buffer status request (number)	32
`Esc&qØL`	Configuration unlocked	14
`Esc&q1L`	Configuration locked	14
`Esc&q8teØ{ØD`	Turn bell off	34
`Esc&q8teØ{1D`	Turn bell on	34
`Esc&q8te1{<x>A`	Return definition 1st character	34
`Esc&q8te1{<x>B`	Return definition 2nd character	34

Escape Sequence	Function	Page
Esc&q<m>te1{ØR	Remote mode off	34
Esc&q<m>te1{1R	Remote mode on	34
Esc&q<m>te2{<x>F	Field separator character	34
Esc&q<m>te2{<x>L	Set forms buffer size	35
Esc&q<m>te2{<x>R	Set block terminator character	35
Esc&q<m>te2{ØZ	Transmit All	35
Esc&q<m>te2{1Z	Transmit Modified	35
Esc&sØA	Transmit functions off	14
Esc&s1A	Transmit functions on	14
Esc&sØB	SPOW off	15
Esc&s1B	SPOW on	15
Esc&sØC	End-of-line wrap off	15
Esc&s1C	End-of-line wrap on	15
Esc&sØD	Line mode on	15
Esc&s1D	Page mode on	15
Esc&sØG	Inhibit handshake off	15
Esc&s1G	Inhibit handshake on	15
Esc&sØH	Inhibit DC2 off	15
Esc&s1H	Inhibit DC2 on	15
Esc&sØN	Escape xfer to printer off	–
Esc&s1N	Escape xfer to printer on	–
Esc&sØZ	Parity checking off	15
Esc&s1Z	Parity checking on	15
Esc&w6f8ØX	Set 80 column mode	26
Esc&w6f132X	Set 132 column mode	26
Esc&w12F	Turn on display	17
Esc&w13F	Turn off display	17
Esc&xØC	Clear send cursor position mode	21

Escape Sequence	Function	Page
Esci	Backtab	21
EscJ	Clear display	21
Escj	Display user key screen	26
EscK	Clear line	21
Esck	Remove user key screen	26
EscL	Insert line	21
Escl	Memory lock on	20
EscM	Delete line	21
Escm	Memory lock off	20
EscN	Insert with wraparound	21
EscO	Delete with wraparound	21
EscP	Delete character	21
Escp	F1 default	26
EscQ	Insert character mode on	21
Escq	F2 default	26
EscR	Insert character mode off	21
Escr	F3 default	26
EscS	Scroll up	17
Escs	F4 default	26
EscT	Scroll down	17
Esct	F5 default	26
EscU	Page down	17
Escu	F6 default	26
EscV	Page up	17
Escv	F7 default	26
EscW	Format mode on	19
Escw	F8 default	26
EscX	Format mode off	19

Escape Sequence	Function	Page
EscY	Display functions on	19
EscZ	Display functions off	19
Escz	Terminal self-test	20

CHAPTER 9

VT (ANSI) Control Sequence Index

When Reflection is configured for VT terminal emulation (explained on page 79), the following control sequences are supported.

Sequence	Description	Page
CSI<n>G or Csi<n>‘	Horizontal position absolute	90
CSI<r>;<c>H	Cursor position	89
CSI<n>L	Insert <*n*> lines	95
CSI<n>M	Delete <*n*> lines	95
CSI<n>P	Delete <*n*> characters	95
CSI<t>;<b>r	Top/bottom margin	94
CSI<row>;<col>R	Cursor position response	106
CSI<n>S	Scroll up <*n*> lines	93
CSI<n>T	Scroll down <*n*> lines	93
CSI<n>U	Page forward <*n*> pages	93
CSI<n>V	Page backward <*n*> pages	93
CSI<n>X	Erase character(s)	96
CSI<n>Z	Cursor backtab	90
CSIØJ	Erase from cursor to end of screen	96
CSIØ"q	Select erasable characters through end of screen	97
CSIØK	Erase from cursor through end of line	96
CSIØn	Response no malfunction	106
CSI1"q	Select protected characters	97
CSI12h	Local echo off	92
CSI12l	Local echo on	92
CSI1J	Erase from top of screen to cursor	96
CSI1K	Erase from start of line through cursor	96
CSI2"q	Select erasable characters	97
CSI2Øh	Auto linefeed on	98
CSI2Øl	Auto linefeed off	98
CSI2h	Keyboard locked	98
CSI2J	Erase complete screen	96
CSI2K	Erase entire line	96

Sequence	Description	Page
ESC_	Log to printer off (VT52)	121
ESC}	Map G2 into GR	112
ESC~	Map G1 into GR	112
ESC#3	Double-width, double-height line (top half)	88
ESC#4	Double-width, double-height line (bottom half)	88
ESC#5	Single-width, single-height line	88
ESC#6	Double-width, single-height line	88
ESC&bR	Connection reset	85
ESC7	Save cursor state	119
ESC8	Restore cursor state	119
ESCA	Cursor up (VT52)	120
ESCB	Cursor down (VT52)	120
ESCc	Hard reset	118
ESCC	Cursor right (VT52)	120
ESCD	Cursor left (VT52)	120
ESCD	Index	91
ESCE or CSI<n>E	Next line	91
ESCF	Character set graphics (VT52)	120
ESCG	Character set ASCII (VT52)	120
ESCH	Home cursor (VT52)	120
ESCI	Reverse linefeed (VT52)	121
ESCJ	Erase to end of screen (VT52)	121
ESCK	Erase to end of line (VT52)	121
ESCM	Reverse index	91
ESCn	Lock shift G2	112
ESCN	Single shift G2	112
ESCo	Lock shift G3	112
ESCO	Single shift G3	112