WRQ reflection®

TERMINAL REFERENCE MANUAL FOR ADDS, ANSI, DG, VT, WYSE AND UNISYS HOSTS

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.

TABLE OF CONTENTS

SECTION 3
Reflection for ReGIS Graphics

SECTION 4
Tektronix Graphics

SECTION 8
Characters and Character Sets

SECTION 9
Control Function Index

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, your PC *emulates,* or operates like, a Digital VT terminal (other terminals are listed below). 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 for UNIX and Digital and Reflection for ReGIS Graphics can emulate the following Digital Equipment Corporation terminals:

- Reflection for UNIX and Digital emulates Digital's VT420, VT320, VT220, VT102, VT101, VT100, and VT52 text terminals. You can also emulate ANSI, ADDS, DG, Unisys A Series T27, and WYSE terminals.

- Reflection for ReGIS Graphics emulates the same terminals as Reflection for UNIX and Digital, and adds the ReGIS, Tektronix, and sixel graphics features of the Digital VT340,[*] VT330, VT241, and VT240 graphics terminals. Reflection for ReGIS Graphics also provides complete emulation for the Tektronix 4014 terminals.

- SCO-ANSI and BBS-ANSI emulation, including ANSI color support.

This section provides an overview of Reflection's features.

---

[*] Reflection for ReGIS Graphics does not support the dual session features or the Tektronix graphics emulation of the VT340.

Reflection Features

Using Reflection for UNIX and Digital or Reflection for ReGIS Graphics you can:

- Issue control functions that cause Reflection to perform certain actions, such as move the text cursor, add a line of text, assign character attributes, and change character sets. See the section starting on page 9.

- Run both text and graphic host applications using Reflection for ReGIS Graphics. If you have a color printer, you can even print graphics in color. You can issue ReGIS commands, as described in the section starting on page 113.

- Emulate a Tektronix terminal. Reflection for ReGIS Graphics supports all the 4010/4014 features available on VT terminals, including the Tektronix 4105 commands for line style and color. See the section starting on page 189.

- Transfer files between your PC and VAX/VMS or 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.

  Using the WRQ/Reflection protocol, you can "drag and drop" files between your local PC and the host in the File Transfer dialog box—you can even drag files from Windows Explorer directly to the host in this dialog box. You can also use wildcards to transfer groups of files. Click the Show Host Files button in this dialog box and you'll see a list of all files in your current host group and account; the display format of the host files is the same as the one used for local PC folders in Windows applications.

- 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 one or more commands or macros, run Reflection scripts, or act as host-writable softkeys. The toolbar displays above Reflection's terminal window.

- Remap your keyboard using a graphical keyboard interface. This lets you assign any keystroke to a Reflection or host function. You can also display a graphical terminal keyboard, and use it to click keys (such as **PF1**).

- Customize the colors of your Reflection terminal window. You can configure the color of the window background and all styles of text.

  In Reflection for ReGIS Graphics, you can also create custom colors, called *color mapping,* to modify the colors used for graphics and text drawing.

- 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 programmatically; that is, by writing programs and sending control functions directly to Reflection. Normally, it is the host that controls Reflection (via control functions), and this process is invisible to the user.

The information provided is specifically aimed at teaching you how to transmit control functions.

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

This *Terminal Reference Manual* 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, "Control Functions"

Contains a comprehensive description of supported sequences when using Reflection to emulate a VT-series terminal.

Section 3, "Reflection for ReGIS Graphics"

Reflection for ReGIS Graphics supports Digital's Remote Graphics Instruction Set, or ReGIS, and the sixel features of Digital's VT340, VT330, VT241, and VT240 graphics terminals. This section describes the emulation of graphics terminals, and provides ReGIS-specific control functions and commands.

Section 4, "Tektronix Graphics"

Describes how to configure Reflection for ReGIS Graphics for Tektronix emulation mode, explains the Tektronix display modes, and provides Tektronix-specific escape sequences.

Section 5, "DG Control Sequences"

Describes how to configure Reflection for ReGIS Graphics for Data General emulation mode, and provides escape sequences.

Section 6, "WYSE Escape Sequences"

Describes how to configure Reflection for ReGIS Graphics for WYSE emulation mode, and provides escape sequences.

Section 7, "Unisys A Series T27 Escape Sequences"

Describes how to configure Reflection for ReGIS Graphics for Unisys A Series T27 terminal emulation, and provides escape sequences.

Section 8, "Characters and Character Sets"

Shows figures of all the supported Digital character sets, and describes how to enter and transmit national characters.

Section 9, "Control Function Index"

A conclusive summary of Digital control functions contained in section 2, allowing you to quickly locate information and page references for specific sequences. These summaries are shown sorted by:

- Actual sequence (for example, $^{C}S_{I}?1_{J}$)

- Digital mnemonic (for example, DECCOLM)

- Functionality (for example, Keyboard)

SECTION 2

Control Functions

Introduction to Control Functions

Control functions cause Reflection to perform certain actions, such as move the text cursor, add a line of text, assign character attributes, and change character sets. Typically, the host sends control functions to Reflection to perform the desired actions. Some of them have equivalent options and buttons on one of the tabs of Reflection's Terminal Setup dialog box, as this section notes.

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

E_{S_C} Stands for the Escape character, and begins an escape sequence.

C_{S_I} Stands for the Control Sequence Introduction character; when Reflection receives this character, it recognizes the string that follows as a control sequence.

D_{C_S} Stands for the Device Control String character, and begins a device control sequence.

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 opening the Reflection command line (press Alt+L) and typing the sequence:

Sequence:	E_{S_C}	C_{S_I}	D_{C_S}
In the Reflection terminal window press:	Esc	Esc+[	Esc+P
In the Reflection command line, type:	Chr$(27)	Chr$(27)&"["	Chr$(27)&"P"

Reflection syntax 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 Reflection command line.

Entering Control Functions Locally

Page 9 shows the symbols and keystrokes you press to enter a control function. If you want to enter control functions locally, there are two ways to do so:

- Place Reflection into local mode by clearing the **Online** check box on the Emulation tab of the Terminal Setup dialog box. Then, type the control function from the keyboard.

  As an example, the control function to move the cursor forward is ^{CS}I<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 function, and performs its action.)

- Execute the Display method from the Reflection command line.

  Using the above example to move the cursor, open the Reflection command line (press [Alt]+[L]) and type the following in the command line, then press [Enter ↵]:

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

Not all control functions executed locally produce results you can see. For example, you must turn on event tracing to capture the results of some request sequences.

This chapter explains C0 and C1 control characters, defines single-character control functions, and provides the format for using multiple-character sequences.

The next chapter describes all the control functions that Reflection supports. Three summaries of control functions begin on page 271. They are shown sorted by:

- Actual sequence (for example, ^{CS}I?1J)

- Digital mnemonic (for example, DECCOLM)

- Functionality (for example, Keyboard)

A Word About Notation

The following notation is used throughout this section:

- Most control functions have a mnemonic identifier. You will never need to enter the mnemonic; it's simply a convenient "word" to help you remember the name of the control function. For example, DECCOLM is the Digital mnemonic for setting the number of display columns.

- Control functions are case sensitive, and must be typed exactly as shown.

- Where appropriate, a slash is used through a zero (Ø) to distinguish it from the uppercase letter O.

- Note the difference between a lowercase L (l) and the number 1 (1).

- References are occasionally given to a character's decimal value in the character set charts.

  For example, the space character is ASCII decimal 32 (see the figure on page 250). Control characters are always in the same positions in the charts, and can be located uniquely by their decimal value; the word "ASCII" is omitted.

- This guide always shows control functions in their 8-bit format. You can always use the 7-bit equivalent escape sequence to represent the 8-bit control, as shown in the table on page 14.

- Parameters you supply for a sequence are enclosed in angle brackets.

  For example, when using the ^{C}SI<n>C control function, replace <n> with the number of spaces you want the cursor to move. If <n> is omitted, the default is used. For most sequences, the default is 1 (when there is not a corresponding Ø sequence).

- Numeric parameters are represented by ASCII strings.

  For example, in the sequence ^{C}SI10;13H the numbers are the strings 10 and 13, not the decimal values 10 (^{L}F character) and 13 (^{C}R character).

  Numeric parameters are constrained to the range 0–9999, inclusive. Any numeric parameter with a value greater than 9999 is interpreted as 9999.

  The plus sign (ASCII decimal 43) and the minus sign (ASCII decimal 45) are not within the range of legal control characters. Therefore, signed numbers, for example "+10" or "-12," cause a control sequence or a device control string to be rejected. Do not include signs with numeric parameters.

Single-Character Control Functions

A single-character control function is made up of one control character (in contrast to multiple-character control functions, explained on page 16). There are two sets of single-character control functions available on the VT200, VT300, and VT400 terminals:

- C0 control characters have decimal values 0–31 in the charts on page 244. Reflection supports only the C0 characters shown in the table on the following page. Each C0 character is encoded in 7 data bits, with the high-order bit always 0. C0 codes, therefore, can be used in both 7-bit and 8-bit operating environments.

    L_F and C_R, for example, are single-character C0 control functions.

- C1 control characters have decimal values 128–159 in the character charts. Reflection supports only the C1 characters shown in the table on the following page. Each C1 character is encoded in 8 data bits, with the high-order bit always 1. C1 characters provide a few more functions than C0 characters, but can only be used directly in an 8-bit operating environment.

    D_{CS}, for example, is a single-character C1 control function.

    In 7-bit operating environments, C1 characters must be converted to a two-character escape sequence equivalent; the table on the following page provides the equivalents.

Recognized C0 (7-Bit) Control Characters

The following table lists the C0 control characters that Reflection recognizes. C0 controls can be used in both 7-bit and 8-bit environments. To represent C0 controls in a Reflection macro, use the Chr$(<n>) syntax, where <n> is the decimal value of the C0 control. See the "Decimal" column on the following page.

Name	Character	Decimal	Keystroke	Action
Null	$^N U_L$	00	^@	Ignored when received.
Enquiry	$^E N_Q$	05	^E	Transmits the answerback message.
Bell	$^B E_L$	07	^G	Sounds a bell if the **Warning bell** check box is selected on the Keyboard tab in the Terminal Setup dialog box.
Backspace	$^B S$	08	^H	Moves cursor left one position on the current line.
Horizontal tab	$^H T$	09	^I	Moves cursor to the next tab stop. If there are no more tab stops, the cursor moves to the right margin.
Linefeed	$^L F$	10	^J	Causes a linefeed or new line operation, depending on the LNM function setting
Vertical tab	$^V T$	11	^K	Same as linefeed.
Form feed	$^F F$	12	^L	Same as linefeed.
Carriage return	$^C R$	13	^M	Moves the cursor to the left margin of the current line.
Shift out (locking shift 1)	$^S O$	14	^N	Maps the G1 character set into GL. You designate G1 by using a Select Character Set (SCS) sequence.
Shift in (locking shift 0)	$^S I$	15	^O	Maps the G0 character set into GL. You designate G0 by using a Select Character Set (SCS) sequence.
Device control 1 (XON)	$^D C_1$	17	^Q	Causes Reflection to continue sending characters when the **Transmit** option in the Connection Setup dialog box is set to **Xon/Xoff**.
Device control 3 (XOFF)	$^D C_3$	19	^S	Causes Reflection to stop sending characters when the **Transmit** option in the Connection Setup dialog box is set to **Xon/Xoff**. Reflection cannot continue until it receives a DC1. Select Clear Communications on the Connection-Reset menu to reset this condition.
Cancel	$^C A_N$	24	^X	Cancels the sequence when received during an escape or control sequence.
Substitute	$^S U_B$	26	^Z	Same as cancel; displays a backwards question mark.
Escape	$^E S_C$	27	^[	Introduces an escape sequence, and cancels any escape or control sequence in progress.
Delete	$^D E_L$	127	(none)	Ignored when received; should not be used as a fill character.

Recognized C1 (8-Bit) Control Characters

In 8-bit environments, C1 controls can be sent directly. In a 7-bit environment, you can send an 8-bit C1 control character by converting it to an equivalent 7-bit escape sequence. The 8-bit controls are single character codes (such as ^{C}SI), whereas their 7-bit equivalents are two-character sequences (such as ^{E}SC[ ).

To form an equivalent 7-bit escape sequence from an 8-bit control character:

1.   Subtract the hexadecimal value 40 from the C1 control code's value.

2.   Precede the result with the ^{E}SC character.

For example, the ^{I}ND character (decimal 132) has a hexadecimal value of 84. To convert ^{I}ND to a 7-bit equivalent, first subtract hexadecimal 40:

84 hex - 40 hex = 44 hex

Hexadecimal 44 is the letter D (as seen in the figure on page 248). Therefore, to represent the ^{I}ND character in a 7-bit environment, you would use ^{E}SCD.

The following table lists the C1 control characters that Reflection recognizes.

Name	Character	Hex	7-Bit Equiv.	Action
Index	^{I}N$_D$	84	^{E}S$_C$ D	Moves cursor down one line in same column, and scrolls at bottom margin.
Next line	^{N}E$_L$	85	^{E}S$_C$ E	Moves cursor to first position of next line, and scrolls at bottom margin.
Horizontal tab set	^{H}S	88	^{E}S$_C$ H	Sets a tab stop at the cursor position.
Reverse index	^{R}I	8D	^{E}S$_C$ M	Moves cursor up one line in the same column, and scrolls at top margin.
Single shift 2	^{S}S$_2$	8E	^{E}S$_C$ N	Maps G2 into GL for the next character only.
Single shift 3	^{S}S$_3$	8F	^{E}S$_C$ O	Maps G3 into GL for the next character only.
Device control string	^{D}C$_S$	90	^{E}S$_C$ P	Introduces a device control string.
Control sequence	^{C}R	9B	^{E}S$_C$ [	Introduces a control sequence.
String	^{S}T	9C	^{E}S$_C$ \	Ends a device control string.

Sending C1 Control Characters to the Host

Whether Reflection sends 8-bit C1 characters directly, or as their 7-bit equivalents, depends on the type of data path and the operating level of Reflection.

To configure Reflection to send 8-bit C1 control characters in an 8-bit environment:

- Make sure you have selected a VT-series terminal that operates in an 8-bit environment. To do this, on the Setup menu, click Terminal and then click the Terminal Type tab and note the selection in the **Terminal type** group box.

Note: You cannot use 8-bit controls if Reflection is configured as a VT52 or VT100 terminal; you are limited to a 7-bit environment.

Now, you can change parity. To do this:

1. On the Connection menu, click Connection Setup.

    - Modem or Serial port connection: Click More Settings and, in the More Settings Connection Setup dialog box, change parity using the **Parity** list.

    - Network connection: As above.

2. Click OK.

By default, Reflection uses 7-bit controls, automatically converting 8-bit C1 characters to their 7-bit equivalents. You can force how Reflection converts control characters with the following sequences:

E_{SC}<SP>F   Switches Reflection from an 8-bit mode to a 7-bit mode. The <SP> indicates the space character (ASCII decimal 32).

E_{SC}<SP>G   Switches Reflection from a 7-bit mode to an 8-bit mode.

These sequences have the same effect as selecting a value from the **Terminal ID** list on the Emulation tab in the Terminal Setup dialog box. The operating level does not change; only the terminal control mode changes.

Note: The terminal mode used for sending C1 controls has no effect on Reflection's ability to receive and interpret C1 controls. As long as the data path allows 8 data bits, 8-bit controls are correctly interpreted (regardless of the terminal control mode).

Multiple-Character Control Functions

The C0 and C1 codes listed in the tables on page 13 and page 14 are single-character control functions, performing simple functions.

Multiple-character control functions, in contrast, can perform many more functions than the C0 and C1 controls, and are formed by a sequence of characters.

There are three types of multiple-character control functions, introduced by the following single-character C0 and C1 controls:

- Escape sequences, introduced by the ^{ES}C character
- Control sequences, introduced by the ^{CS}I character
- Device control strings, introduced by the ^{DC}S character

Multiple-character control functions include both control characters and normal ASCII text, such as letters, numbers, and punctuation. For example, the multiple-character device control string ^{DC}SØ!u%5ST assigns a user-preferred supplemental character set.

Escape Sequences

An escape sequence begins with the C0 character ^{ES}C (decimal 27). To enter the ^{ES}C character on the Reflection command line, type Chr$(27): for either 7 and 8 bit mode. If you're in the terminal window, you can simply press the Esc key. After receiving an ^{ES}C character, Reflection interprets the next characters as part of the sequence.

Control Sequences

A control sequence begins with the C1 control character ^{CS}I (decimal 155). To enter the ^{CS}I character on the Reflection command line, type Chr$(27)&"[" for 7-bit mode, or Chr$(155) for 8-bit mode. If you're in the terminal window, you can press Esc+[.

Control sequences usually include variable parameters. The format is:

^CS_I P...P I...I F

P...P	are zero or more *parameters*. You can have up to 16 parameters per sequence, using a semicolon (;) to separate each one.
I...I	are zero or more *intermediate* characters.
F	is the *final* character indicating the end of the control sequence.

For example, the following control sequence sets the scrolling region, where the top margin is at line 7 and the bottom margin is at line 18:

^CS_I7;18r

In this example, 7 and 18 are the parameters and r is the final character. This sequence does not have any intermediate characters.

Device Control Strings

A device control string begins with the C1 control character ^DC_S (decimal 144). To enter the ^CS_I character on the Reflection command line, type Chr$(27)&"P" for 7-bit mode, or Chr$(144) for 8-bit mode. If you're in the terminal window, you can press Esc + P.

Device control strings always include a data string. The format is:

^DC_S P...P I...I F <data string> ^ST

P...P	are zero or more *parameters*. You can have up to 16 parameters per sequence, using a semicolon (;) to separate each one.
I...I	are zero or more *intermediate* characters.
F	is the *final* character indicating the end of the device control string.
<data string>	is a *data string* of zero or more characters. Use a semicolon (;) to separate individual strings. The particular range of characters included in a data string is determined by the individual device control string. Any character except ^ST (decimal 156) can be included in a data string.

S_T	is the *string terminator*. Type Chr$(27)& "Ë for 7-bit mode, or Chr$(156) for the 8-bit equivalent. (You can also press Esc+\ in the terminal window.)

APC, OSC, and PM

The application program command (APC), operating system command (OSC), and privacy message (PM) are also C1 controls. However, VT terminals—and Reflection—ignore them. The controls have the same format as device control strings and end with an S_T. When Reflection receives an APC, OSC, or PM introducer, it discards all following characters until receiving a string terminator (so the whole sequence is discarded).

Interrupting Control Sequences

You can use the following C0 control characters to interrupt a control function or recover from an error:

E_{S_C}	Cancels a sequence in progress, and begins a new sequence.
C_{A_N}	Cancels a sequence in progress. Reflection interprets the characters that follow the C_{A_N} character as usual.
S_{U_B}	Same as C_{A_N}, except displays a backwards question mark.

Control Function Descriptions

This chapter describes all of the VT terminal control functions that Reflection supports. Control functions are grouped by categories, and include their mnemonic identifier. If a function is linked to a dialog box option, the menu and command, as well as the relevant dialog box tab, are documented.

Reflection-Specific Control Functions

When you use Reflection and a PC instead of a terminal, there are a few control functions unique to Reflection that are not available on the terminal. The following are these control functions explained.

Invoke a Reflection Command (WRQCMD)

To send a Reflection command remotely to Reflection, you must place the command within a string containing the following elements:

    ^{DC}S2000;<term>;<tag>{<commands><st>

Display Color ANSI Sequences

To send these keystrokes, enter the control sequences shown:

`Csi =<n>F`	Set the default normal foreground color to *<n>*; see the following table. This sequence also sets the current color to *<n>*.
`Csi =<n>H`	Set the reverse video foreground color to *<n>*; see the following table.
`Csi =<n>G`	Set the default normal background color to *<n>*; see the following table This sequence also sets the current color to *<n>*.
`Csi 2<c1>;<c2>m`	Set the default normal color for the foreground to *<c1>* and for the background to *<c2>*; see the following table.
`Csi 7<c1>;<c2>m`	Set the default reverse video color for the foreground to *<c1>* and for the background to *<c2>*; see the following table.
`Csi =ID`	Turn on background blink/bold. This sequence is equivalent to `Csi5m`.
`Csi =OD`	Turn off background blink/bold. This sequence is equivalent to `Csi25`.
`Csi =OE`	Turn off blink bit, allowing 16 colors to be displayed (the default). This sequence is equivalent to `Csi3;0m`. It is also equivalent to selecting the **Enable blink** check box on the Colors tab of the Display Setup dialog box.
`Csi ;Om`	Turn off blink bit, allowing 16 colors to be displayed (the default). This sequence is equivalent to `Csi=)E`. It is also equivalent to selecting the **Enable blink** check box on the Colors tab of the Display Setup dialog box.
`Csi =1E`	Turn on blink bit, allowing 8 colors to be displayed. This sequence is equivalent to `Csi3;1m`. It is also equivalent to selecting the **Enable blink** check box on the Colors tab of the Display Setup dialog box.

Csi =1m	Turn on blink bit, allowing 8 colors to be displayed. This sequence is equivalent to Csi=1E. It is also equivalent to selecting the **Enable blink** check box on the Colors tab of the Display Setup dialog box.

<n>	Color
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	White
8	Grey
9	Light blue
10	Light green
11	Light cyan
12	Light red
13	Light magenta
14	Yellow
15	Light white

Defining Softkey

^{ES}C&f<attr><key><label length><string length><label><string>

This escape sequence lets you define a Reflection Softkey. If you're using Reflection for ReGIS Graphics, this sequence is supported by both VT and Tektronix modes. The parameters *<attr>, <key>, <label length>,* and *<string length>* may appear in any order. However, the letter that identifies the last parameter must be in uppercase, and all the preceding identifiers must be in lowercase.

Softkey Escape Sequence Parameters

Variable	Parameters
<attr>	0a NORMAL (default)
	1a LOCAL
	2a NORMAL; appends Cr to *<string>*
	3a COMMAND
<key>	Key number, *1k–8k* (Default is *1k*)
<label length>	Ød–8d (Default is Ød)
<string length>	ØL–8ØL (Default is *1L*); *-1 erases* the field definition
<label>	A string of exactly *<label length>* characters
<string>	A string of exactly *<string length>* characters

If a parameter is omitted, the default value is used. If a value of zero 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 *<string>*.

The following example shows an escape sequence that assigns the string EDIT/EDT MYFILE to ⌷F3⌷. The attribute is NORMAL with a carriage return (*<attr>* is 2a), and the label reads EDIT. The *<label length>* is 4, and the text that appears in the label is EDIT. (The *<label length>* and text must match, otherwise the definition may begin in the label.)

^{ES}C&f2a3k4d15LEDITEDIT/EDT MYFILE

Operating Levels

You can configure Reflection to emulate terminals from the VT52, VT100, VT200, VT300, and VT400 series using the DECSCL control functions described here. You may need to change operating levels if the host application you use requires a terminal from a specific series.

The DECSCL functions in this section are linked to the **Terminal type** options on the Terminal Type tab of the Terminal Setup dialog box.

You can select from five levels of operation:

- VT52 mode, set using DECANM (page 24).

  In VT52 mode, no DECSCL sequences are recognized; see page 109 for a list of control functions specific to the VT52 terminal.

- Level 1 for VT100 operation (VT102 emulation).

- Level 2 for VT200 operation.

- Level 3 for VT300 operation.

- Level 4 for VT400 operation.

Select an Operating Level (DECSCL)

By default, Reflection emulates a VT400 level terminal.

`Csi 61"p`	Select Level 1 operating level. In Level 1, certain functions are not available, as listed on page 25.
`Csi 62;1"p`	Select Level 2 operating level, 7-bit controls.

`Csi 62"p`
`Csi 62;Ø"p`
`Csi 62;2"p`
	Select Level 2 operating level, 8-bit controls.

`Csi 63;1"p`	Select Level 3 operating level, 7-bit controls.

`Csi 63"p`
`Csi 63;Ø"p`
`Csi 63;2"p`
	Select Level 3 operating level, 8-bit controls.

`Csi 64;1"p`	Select Level 4 operating level, 7-bit controls.

`Csi 64"p`	Select Level 4 operating level, 8-bit controls.
`Csi 64;Ø"p`	
`Csi 64;2"p`	

When you change the operating level, Reflection performs a soft reset (DECSTR); see page 106 for more information.

Select 7-Bit C1 Control Characters (S7C1T)

${}^E\!S_C$`<SP>F`   Send all C1 control characters as two-character, 7-bit equivalents. `<SP>` is the space character (ASCII decimal 32). This sequence changes the type of controls sent for the current operating level to 7-bit controls; the operating level does not change.

Select 8-Bit C1 Control Characters (S8C1T)

${}^E\!S_C$`<SP>G`   Send all C1 control characters as single 8-bit characters. `<SP>` is the space character (ASCII decimal 32). This sequence changes the type of controls sent for the current operating level to 8-bit controls; the operating level does not change.

Select VT52 Mode (DECANM)

`Csi ?2l`   Select VT52 emulation mode. In this mode, only VT52 control functions are recognized (see page 109 for a list).

Exit VT52 Mode

${}^E\!S_C$`<`   Exit VT52 emulation mode and enter VT100 mode.

VT100 Mode—Restrictions and Ignored Control Functions

When you set up Reflection to emulate a VT100 series terminal, certain restrictions apply:

- Reflection sends only 7-bit characters. All C1 control characters are sent as 7-bit equivalents; the table on page 14 provides the equivalents.

- The VT function and editing keys do not function, with the exception of the following three keys:

    – VT [F11] sends the ᴱsᴄ character

    – VT [F12] sends the ᴮs character

    – VT [F13] sends the ᴸꜰ character

    These VT function keys are specific to VT100 mode, and do not perform the same functions in higher operating levels.

- The 8th bit of all received characters is set to Ø.

- Keystrokes that send characters from the DEC character set (page 252) or the ISO Latin-1 supplemental graphic character set (page 253) result in errors.

- A limited number of character sets are available; see page 73. Soft character sets are not available.

- The following control functions are ignored:

Mnemonic	Description
DECCIR	Cursor information report
DECCTR	Color table report
DECDLD	Downloadable character set
DECRPM	Report mode
DECRPSS	Report selection or setting
DECRQM	Request mode
DECRQPSR	Request presentation state
DECRQSS	Request selection or setting
DECRQTSR	Request terminal state
DECRSPS	Restore presentation state

Mnemonic	Description
DECRSTS	Restore terminal state
DECSASD	Select active status display
DECSCA	Select character protection attribute
DECSED	Selective erase in display
DECSEL	Selective erase in line
DECSSDT	Select status line type
DECSTR	Soft terminal reset
DECTABSR	Tabulation stop report
DECTSR	Terminal state report
DECUDK	User-defined keys
DSR	UDK and keyboard language
ECH	Erase character
ICH	Insert character
LS1R	Locking shift 1 right
LS2	Locking shift 2
LS2R	Locking shift 2 right
LS3	Locking shift 3
LS3R	Locking shift 3 right
S7C1T	Send 7-bit C1 controls
S8C1T	Send 8-bit C1 controls

Character and Line Attributes

You can use control functions to select visual attributes for characters and lines. Visual char-acter attributes change the way characters appear on the screen without changing the actual characters.

Character Attributes

You can set the following character attributes:

- Bold
- Underline
- Blinking
- Inverse video
- Invisible

Select Graphic Rendition (SGR)

Csi <n>;...<n>m

This lets you specify one or more character attributes at a 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.

All Terminal Modes:		VT200 Mode and Higher:	
Clear all attributes	Ø	Bold off	22
Bold	1	Underline off	24
Underline	4	Blinking off	25
Blinking	5	Inverse video off	27
Inverse video	7	Invisible off	28
Invisible	8		

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.

For example, use Csi 5;7m to display blinking text in inverse video; use Csi 1m to display text as bold.

Line Attributes

Line attributes affect the way a line of characters appears on the screen.

Single-Width, Single-Height Line (DECSWL)

^ES_C#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 (DECDWL)

^ES_C#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 big, so that it can move one character at a time. Cursor positioning sequences (such as CUP on page 35) 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.

Double-Width, Double-Height Line (DECDHL)

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

^ES_C#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 big, so that it can move one character at a time. Cursor positioning sequences (such as CUP, shown on page 35) 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 would address columns 1–80, as usual).

Page Memory and Selecting Lines: VT420

The VT420 terminal has off-screen memory for storing data entered from the keyboard or the host application; up to 144 lines can be stored. These 144 lines are called page memory. By default, the terminal (and Reflection) uses 6 pages of 24 lines each of page memory.

Select Lines Per Page (DECSLPP)

This function sets the number of lines for each page in page memory.

Control Function	Lines Per Page	Set Number of Pages To
Csi 24t	24	6
Csi 25t	25	5
Csi 36t	36	4
Csi 48t	48	3
Csi 72t	72	2
Csi 144t	144	1

Rectangular Area Operations: VT420

These control functions allow you to manipulate rectangular areas of text within page memory. You can:

- Copy them from one area in page memory to another.

- Erase them.

- Fill them with a character of your choice.

- Change or reverse their visual character attributes.

Copy Rectangular Area (DECCRA)

`Csi <n1>;<n2>;<n3>;<n4>;<n5>;<n6>;<n7>;<n8>$v`

Copy a rectangular area of display memory from one part of page memory to another. Characters and their attributes remain unchanged. The parameters <n1> through <n5> describe the area to be copied:

<n1>	Top line
<n2>	Left column
<n3>	Bottom line
<n4>	Right column
<n5>	Page number

The parameters *<n6>* through *<n8>* describe where the area should be copied:

<n6>	Top line
<n7>	Left column
<n8>	Page number

Erase Rectangular Area (DECERA)

`Csi <n1>;<n2>;<n3>;<n4>$z`

Erase the characters (and their visual attributes) in the specified rectangular area and replace each one with a space (decimal 32). Line attributes (for example, the attributes that specify double-wide, double-high characters) are not erased. The areas to erase are:

<n1>	Top line	*<n3>*	Bottom line
<n2>	Left column	*<n4>*	Right column

Fill Rectangular Area (DECFRA)

```
Csi <n1>;<n2>;<n3>;<n4>;<n5>$x
```

Fill an area in display memory with a specified character. The fill character takes on the visual attributes set by the last SGR control function, not the attributes of the characters that it replaces. Current line attributes (for example, the attributes that specify double-wide, double-high characters) remain unchanged. The parameters are:

<n1>   Decimal code of fill character

<n2>   Top line

<n3>   Left column

<n4>   Bottom line

<n5>   Right column

Selective Erase of Rectangular Area (DECSERA)

```
Csi <n1>;<n2>;<n3>;<n4>${
```

Erase all erasable characters from a specified rectangular area in page memory; a space character replaces erased character positions. The DECSERA control function does not change:

- Visual attributes set by the select graphic rendition (SGR) function.
- Protection attributes set by DECSCA.
- Line attributes.

The parameters are:

<n1>	Top line	*<n3>*	Bottom line
<n2>	Left column	*<n4>*	Right column

Select Attribute Change Extent (DECSACE)

Csi <n>*x

Select which character positions within a rectangular area can have their attributes changed or reversed. The DECSACE control function affects the rectangular area control functions for changing and reversing attributes (DECCARA and DECRARA).

<n>   The <n> parameter specifies what character positions are affected. *0* or *1* indicates a stream of character positions that begins in the first position specified in the DECCARA or DECRARA control function, and ends with the second one that is specified. A 2 indicates a rectangular area of character positions; the DECCARA and DECRARA control functions specify the upper left and lower right corners of the area.

Change Attributes in Rectangular Area (DECCARA)

Csi <n1>;<n2>;<n3>;<n4>;<n5>...<nn>$r

Change the visual attributes for characters in a specified area of display memory—the characters themselves remain unchanged. The DECSACE control function is used to determine whether all or just some of the character positions are affected. The parameters are:

<n1>   Top line

<n2>   Left column

<n3>   Bottom line

<n4>   Right column

<n5>...*<nn>*   Visual character attributes

Reverse Attributes in Rectangular Area (DECRARA)

`Csi <n1>;<n2>;<n3>;<n4>;<n5>...<nn>$t`

Reverse the visual attributes for characters in a specified area of display memory—the characters themselves remain unchanged. The DECSACE control function is used to determine whether all or just some of the character positions are affected. Reversing a visual attribute means toggling it, for example, if bold is on, it is toggled off, if underlining is off, it is toggled off. The parameters are:

<n1>	Top line
<n2>	Left column
<n3>	Bottom line
<n4>	Right column
<n5>...*<nn>*	Visual character attributes

Request Memory Checksum (DECRQCRA)

`Csi <n1>;<n2>;<n3>;<n4>;<n5>;<n6>*y`

Request a memory checksum of a rectangular area on a specified page. Reflection returns a checksum report (DECCKSR) in response. DECRQCRA also works on the status line.

<n1>	A numeric label you give to identify the checksum request (the report—DECCKSR—returns this number).
<n2>	The number of the page on which the rectangular area is located. If *<n2>* is 0 or omitted, Reflection ignores the remaining parameters and reports a checksum for all pages in page memory. If *<n2>* is more than the number of pages, Reflection does a checksum on the last page.
<n3>–*<n6>*	The parameters *<n3>* through *<n6>* are the top, right, bottom, and left borders of the rectangular area. *<n3>* must be less than or equal to *<n5>*; *<n4>* and *<n6>* are column numbers: *<n6>* must be less than or equal to *<n4>*. Without these parameters, Reflection returns a checksum of page *<n2>*.

<n3> (default = 1)

<n6>   <n4> (default = current page width)

<n5> (default = current page length)
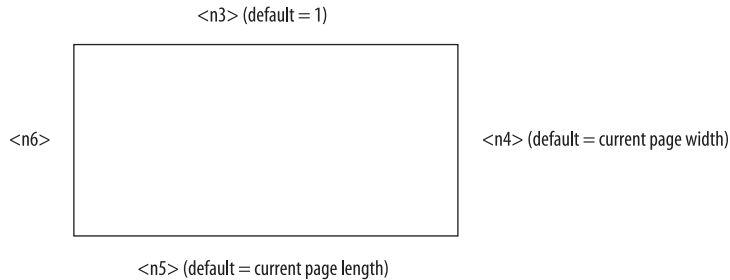
The coordinates are affected by the setting of origin mode (DECOM).

Request Memory Checksum of Macros (DECCKSR)

Csi ?63;<n>n

DSR request for a memory checksum report of current text macro definitions. *<n>* is an optional numeric parameter that provides a label to identify the checksum request.

DECCKSR Response

$^{D}C_S$<n>!~D...D^{S}T

Response to memory checksum request (DECCKSR). *<n>* is a label indicating which DSR request the report is for. *D...D* is the data string: four hexadecimal digits indicating the checksum.

Cursor Movement

The cursor movement functions let you make the cursor visible or invisible, and move it around on the screen. See page 37 for VT420-specific cursor movement functions.

Text Cursor Enable Mode (DECTCEM)

Csi ?25h   Make the text cursor visible.

Csi ?25l   Make the text cursor invisible.

This control function is linked to the **Visible** check box in the **Cursor** group box on the Screen tab of the Display Setup dialog box.

Cursor Position (CUP)

Csi <r>;<c>H   Move the cursor to row <r> and column <c>. Normally row 1, column 1 is the upper-left corner of the screen. However, if top and bottom margins have been defined (with DECSTBM on page 42) and origin mode is set (with DECOM on page 43), then the top row of the scrolling region is row 1. The cursor is never positioned beyond the bottom margin or right margin. If the screen width is greater than 80 columns, the display is scrolled horizontally, if necessary, to bring the cursor into view. The default is Csi 1;1H.

Horizontal and Vertical Position (HVP)

Csi <r>;<c>f   Move the horizontal and vertical position of the cursor. This works the same as the CUP sequence above. This sequence is supported for compatibility with older programs: new applications should use CUP.

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 Backtab (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.

Index (IND)

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

The 7-bit equivalent of I_{ND} is $E_{S}C_{D}$.

Next Line (NEL)

$N_{E_{L}}$ Move the cursor to column 1 of the next line. If the cursor is at the bottom margin, the display scrolls up one line.

The 7-bit equivalent of $N_{E_{L}}$ is $E_{S}C_{E}$.

Reverse Index (RI)

R_{I} 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.

The 7-bit equivalent of R_{I} is $E_{S}C_{M}$.

Cursor Movement: VT420

Cursor movement is tied to the page memory you have allocated when emulating a VT420 terminal. See page 29 for more information on page memory.

In all of the following examples, the sequence is ignored if page memory is configured for a single page.

Next Page (NP)

Csi <n>U   Move the cursor forward to the home position on one of the following pages in page memory. ⟨*n*⟩ is the number of pages to move forward. If ⟨*n*⟩ is 0 or 1, the cursor moves to the next page in page memory. If ⟨*n*⟩ is greater than the number of pages, the cursor moves to the home position on the last page.

Previous Page (PP)

Csi <n>V   Move the cursor forward to the home position on one of the following pages in page memory. ⟨*n*⟩ is the number of pages to move forward. If ⟨*n*⟩ is 0 or 1, the cursor moves to the next page in page memory. If ⟨*n*⟩ is greater than the number of pages, the cursor moves to the home position on the last page.

Page Position Absolute (PPA)

Csi <n><SP>P Move the cursor backward to the home position on one of the preceding pages in page memory. ⟨*n*⟩ is the number of pages to move backward. If ⟨*n*⟩ is 0 or 1, the cursor moves to the preceding page in page memory. If ⟨*n*⟩ is greater than the number of pages, the cursor moves to the home position on the first page.

Page Position Relative (PPR)

Csi <n><SP>Q Move the cursor to the corresponding row and column on a page in page memory. ⟨*n*⟩ is the number of the page to which the cursor should move. If ⟨*n*⟩ is greater than the number of pages in memory, the cursor stops on the last page. If ⟨*n*⟩ is 0 or 1, the cursor stops on the first page.

Page Position Backward (PPB)

Csi <n><SP>R   Move the cursor forward to the corresponding row and
               column on a following page in page memory. If there is only
               one page, this control function is ignored. *<n>* is the number of
               pages to move the cursor forward. If *<n>* is greater than the
               number of following pages in page memory, the cursor stops on
               the last page.

Horizontal Cursor Coupling (DECHCCM)

This control function determines whether the user window pans with the cursor
when the cursor moves past the right or left border of the user window. DECHCCM
is only useful when the width of the current user window is narrower than the page.
The cursor must stay on the current page.

Csi ?60h   Couple the cursor to the display. By default the cursor is coupled to
              the display for horizontal movement. When the cursor moves past
              the right or left border of the user window, the window pans to
              keep the cursor in view. If the cursor moves past the left border of
              the display, the user window pans to the left and new columns
              appear at the right border of the window.

Csi 60l   Uncouple the cursor from the display. If the cursor is uncoupled
              from the display and is moved past the right or left border of the
              user window, the cursor disappears from view.

Vertical Cursor Coupling (DECVCCM)

This control function determines whether the user window pans with the cursor when the cursor moves past the top or bottom border of the user window. DECVCCM is only useful when the height of the current user window is smaller than the page. The cursor must stay on the current page.

`Csi ?61h`	Couple the cursor to the display. By default the cursor is coupled to the display for vertical movement. When the cursor moves past the top or bottom border of the user window, the window pans to keep the cursor in view. If the cursor moves past the top of the display, the user window pans up and new lines appear at the top of the screen.
`Csi 61l`	Uncouple the cursor from the display. If the cursor is uncoupled from the display and is moved past the top or bottom border of the user window, the cursor disappears from view.

Page Cursor Coupling (DECPCCM)

This control function determines if a new page appears in the display when the cursor moves to a new page. DECPCCM is only useful with a multiple-page format.

`Csi ?64h`	Couple the cursor to the display. By default the cursor is coupled to the display when the cursor moves to a new page. The new page appears in the display to keep the cursor in view.
`Csi 64l`	Uncouple the cursor. If the cursor is uncoupled from the display and is moved to a new page, the cursor disappears from view.

Screen Display

Screen display control functions affect how the screen looks, whether to echo characters typed from the keyboard, and if a status line should display.

Normal/Inverse Video (DECSCNM)

Csi ?5h   Set screen to inverse video.

Csi ?5l   Set screen to normal video.

This control function is linked to the **Inverse video** check box on the Colors tab of the Display Setup dialog box.

Scrolling Mode (DECSCLM)

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.

This control function is linked to the **Jump** and **Smooth** options in the **Scrolling** group box on the Screen tab of the Display Setup dialog box.

Pan Down (SU)

Csi <n>S   Move the user window down a specified number of lines in page memory. *<n>* is the number of lines to move the user window down in page memory: *<n>* new lines appear at the bottom of the display and *<n>* scroll off the top of the display. You cannot pan past the bottom margin of the current page.

Pan Up (SD)

Csi <n>T   Move the user window up a specified number of lines in page
            memory. ⟨*n*⟩ is the number of lines to move the user window up in
            page memory: ⟨*n*⟩ new lines appear at the top of the display and
            ⟨*n*⟩ scroll off the bottom of the display. You cannot pan past the
            top margin of the current page.

Local Echo (SRM)—Send/Receive Mode

Csi 12h   Turn off local echo. Reflection sends characters only to the host. If
           the host echoes characters, they appear on the display.

Csi 12l   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.

This control function is linked to the **Local echo** check box in the **Keyboard modes**
group box on the Keyboard tab of the Terminal Setup dialog box.

Select Status Line Type (DECSSDT)

The line at the bottom of the display is reserved for the status line. The position of
the status line depends on the number of display lines set. For example, if there are
50 display lines, the status line is line 51.

Csi Ø$˜   Do not display the status line.

Csi 1$~   Select **Indicator** as the status line type. The last line of the display
           is shown in inverse video and displays the cursor position (row/
           column) and printer status.

Csi 2$~   Select a status line that is **host writable**. The status line then acts
           as a one-line dis-play that the host can write to. Most control
           functions that affect the main display also affect the status line. If
           you change from an indicator to a host-writable status line, the
           new host-writable status line is empty.

This control function is linked to the **Status line** list on the Advanced Option dialog
box. Click Advanced on the Emulation tab of the Terminal Setup dialog box to see
this.

Select Active Status Display (DECSASD)

Csi Ø$}   Select the main display (the top 24 lines) as the active display. The main display is the top 24–144 lines, depending on how many lines Reflection is configured to display. You define the number of rows using the **Rows** box in the **Dimensions** group box on the Screen tab of the Display Setup dialog box.

Csi 1$}   Select the status line as the active display.

Select 80 or 132 Columns (DECCOLM)

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.

When you switch between 80 and 132 columns, the display clears and the cursor moves to the upper-left corner. However, changing this setting does not clear data from the status line (DECSSDT).

This control function is linked to the **Columns** option in the **Dimensions** group box on the Screen tab of the Display Setup dialog box.

Scrolling Region (DECSTBM)

Csi <t>;<b>r   Set a scrolling region with a top margin at row <*t*> and a bottom margin at row <*b*>. The scrolling region is the area between these margins that moves during vertical scrolling. The <*b*> parameter must be greater than <*t*>. Rows are counted from row 1 at the top of the display. After DECSTBM is set, the cursor moves to column 1, line 1 of the display.

Omitting both parameters (that is, entering Csi r) sets the margins to full screen. Lines that scroll off the top of the display are saved in display memory.

Origin Mode (DECOM)

Origin mode determines if the cursor is allowed to move outside the margins set by the DECSTBM function, or if movement is restricted to within the margins. This allows cursor addressing relative to the scrolling margins or the complete display. When you start Reflection or do a soft reset, origin mode is reset.

Csi ?6h   Set origin mode, making the home cursor position the upper-left corner of the display within the margins. The first row in the scrolling region is now row 1, and the cursor cannot move outside the margins.

Csi ?6l   Reset origin mode, making the home cursor position the upper-left corner of the screen. The first row on the screen is now row 1, and the cursor can move outside the margins.

Display Controls (no mnemonic)

Csi 3h   Display C0 and C1 control characters (such as ^{E}C and ^{L}F) instead of interpreting them.

Csi 3l   Set control codes to function normally; they are interpreted and not displayed. If control codes are being displayed, the Csi 3l function is first displayed, then interpreted.

This control function is linked to the **Interpret** and **Display** options in the **Control characters** group box on the Screen tab of the Display Setup dialog box.

Editing

You can insert and delete lines in the scrolling region—the area on the screen between the top and bottom margins. (Use the DECSTBM control function on page 42 to set the scrolling region.)

Insert/Replace Mode (IRM)

Csi 4h   Insert mode. Characters are inserted at the cursor position, and all characters to the right of the cursor move one column to the right.

Csi 4l   Replace mode. Characters replace characters at the cursor position.

Delete Line (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 function is available only in VT200 mode and higher.

Erasing Text

The erase text control functions can affect data inside or outside the scrolling region, they are not restricted by margins.

Erase in Display (ED)

Csi ØJ   Erase from the cursor to the end of the screen.

Csi 1J   Erase from the top of the display to the cursor.

Csi 2J   Erase all of the display.

The ED functions retain display memory.

Erase in Line (EL)

Csi ØK   Erase from the cursor position to the end of the line.

Csi 1K   Erase from the beginning of the line to the cursor position.

Csi 2K   Erase the entire line the cursor is on.

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 function is available only in VT200
              mode and above.

Selectively Erasing Text—VT200 Mode and Above

With selective erase, you only can erase characters defined as erasable. These control functions do not affect visual character attributes set by the SGR function (see page 27).

Select Character Protection Attribute (DECSCA)

Csi Ø"q or   Characters following either of these control functions may be
Csi 2"q   erased by the selective erase control functions (DECSED and
              DECSEL).

Csi 1"q   Protect characters from erasure by the selective erase control
              functions (DECSED or DECSEL). The characters can still be
              erased by a normal ECH erase sequence.

Selective Erase in Display (DECSED)

Csi ?ØJ   Erase unprotected characters from the cursor position to the end
              of the display.

Csi ?1J   Erase unprotected characters from the top of the display to the
              cursor position.

Csi ?2J   Erase unprotected characters from the entire display.

DECSED erases only those characters in the display that are defined as erasable by
the DECSCA function; text in display memory is retained.

Selective Erase in Line (DECSEL)

Csi ?ØK   Erase unprotected characters from the cursor position to the end
              of the line.

Csi ?1K   Erase unprotected characters from the beginning of the line to the
              cursor position.

Csi ?2K   Erase unprotected characters from the entire line.

DECSEL erases only those characters in the current line that are defined as erasable
by the DECSCA function.

Keyboard

Keyboard control functions affect actions that lock and unlock the keyboard, cause keystrokes to repeat automatically, and perform other keyboard-related activities.

Keyboard Action Mode (KAM)

Csi 2h   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.

Csi 2l   Unlock the keyboard.

Linefeed/New Line Mode (LNM)

Csi 2Øh   Turn on new line mode. When you press Enter↵, Reflection sends both a carriage return and a linefeed. When Reflection receives a linefeed, form feed, or vertical tab, it moves the cursor to the first column of the next line.

Csi 2Øl   Turn off new line mode. The Enter↵ key sends only a Cr character. A received L_F, F_F, or V_T character moves the cursor down one line in the current column.

This control function is linked to the **New line** check box on the Emulation tab of the Terminal Setup dialog box.

Autorepeat Mode (DECARM)

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

Csi ?8l   Turn off autorepeat.

This control function is linked to the **Auto repeat** check box in the **Keyboard modes** group box on the Keyboard tab of the Terminal Setup dialog box.

Autowrap Mode (DECAWM)

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.

This control function is linked to the **Autowrap** check box on the Emulation tab of the Terminal Setup dialog box.

Cursor Keys Mode (DECCKM)

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 49.

This control function is linked to the **Cursor keys** options in the **Terminal keys** group box on the Keyboard tab of the Terminal Setup dialog box.

Keypad Mode (DECKPAM and DECKPNM)

$E_{S_{C=}}$ Numeric keypad keys send application-specific sequences (DECKPAM).

$E_{S_{C>}}$ Numeric keypad keys send numeric characters (DECKPNM), 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 50.

This control function is linked to the **Keypad** options in the **Terminal keys** group box on the Keyboard tab of the Terminal Setup dialog box.

Numeric Keypad Mode (DECNKM)

Csi ?66h   Set the numeric keypad to application mode.

Csi ?66l   Set the numeric keypad keys to normal mode.

This control function works like the DECKPAM and DECKPNM functions above; it is provided mainly for use with the request and report modes (DECRQM on page 100 and DECRPM on page 102).

Backarrow Key Function (DECBKM)

Csi ?67h   Set the function of the ⎵Backspace⎵ key to backspace (decimal 8).

Csi ?67l   Set the function of the ⎵Backspace⎵ key to delete (ASCII decimal 127).

This control function is linked to the **VT backspace sends** options in the **Terminal keys** group box on the Keyboard tab of the Terminal Setup dialog box.

Keyboard Codes

The following tables list the control functions sent by the arrow keys, numeric keypad, and editing and function keys. For the arrow keys and the numeric keypad, the codes sent in ANSI mode apply only to VT100–VT400 modes; the VT52 terminal is not compatible with ANSI mode.

Codes Sent by the Arrow Keys

The following table shows codes sent by the arrow keys, depending on the setting of the DECCKM function (see page 48):

	ANSI Mode		**VT52 Mode**
Key	**Cursor**	**Application**	**Cursor or Application**
↑	Csi A	$^{S}S_{3A}$	$^{E}S_{CA}$
↓	Csi B	$^{S}S_{3B}$	$^{E}S_{CB}$
→	Csi C	$^{S}S_{3C}$	$^{E}S_{CC}$
←	Csi D	$^{S}S_{3D}$	$^{E}S_{CD}$

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 functions DECKPAM and DECKPNM (see 48):

	ANSI 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 Cr LF	SS3M	Cr or Cr LF	ESC?M
PF1	SS3P	SS3P	ESCP	ESCP
PF2	SS3Q	SS3Q	ESCQ	ESCQ
PF3	SS3R	SS3R	ESCR	ESCR
PF4	SS3S	SS3S	ESCS	ESCS

a. The LNM function determines the characters sent by the Enter key (either a Cr or a CrLF).

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 either VT200, VT300, or VT400 mode (three of the keys are recognized in VT52 and VT100 mode, as noted).

VT Keystroke	Generated Code	
Find	Csi 1~	
Insert Here	Csi 2~	
Remove	Csi 3~	
Select	Csi 4~	
Prev Screen	Csi 5~	
Next Screen	Csi 6~	
F6	Csi 17~	
F7	Csi 18~	
F8	Csi 19~	
F9	Csi 20~	
F10	Csi 21~	
F11 (ESC)	Csi 23~	(ᴱsᴄ in VT52/VT100 modes)
F12 (BS)	Csi 24~	(ᴮs in VT52/VT100 modes)
F13 (LF)	Csi 25~	(ᴸF in VT52/VT100 modes)
F14	Csi 26~	
F15 (Help)	Csi 28~	
F16 (Do)	Csi 29~	
F17	Csi 31~	
F18	Csi 32~	
F19	Csi 33~	
F20	Csi 34~	

User-Defined Keys

The Digital keyboard has a row of 20 function keys along the top, labeled F1 through F20. Using the DECUDK device control string, you can define the codes sent by 15 of these function keys (F6–F20; F1–F5 are reserved for terminal functions). These keys are called user-defined keys, or UDKs. UDKs are available only in VT200 mode and above.

Once created, you activate a UDK by pressing Shift+<*key*>, where <*key*> is the function key you defined.

A total of 256 bytes is available for all UDKs. When all 256 bytes are used, you cannot define any more keys until you clear some space.

To clear space:

- Use DECUDK to redefine one or more UDKs.

- Use DECUDK to clear existing UDKs.

- Clear all UDKs with a power-up or RIS operation (see page 106).

Keyboard ANSI Sequences (WRQANSI)

These sequences are supported when you are using Reflection to emulate a BBS-ANSI or SCO-ANSI terminal. To send these keystrokes, enter the control sequences shown:

Sequence	ANSI Function Key
Csi [M	F1
Csi [N	F2
Csi [O	F3
Csi [P	F4
Csi [Q	F5
Csi [R	F6
Csi [S	F7
Csi [T	F8
Csi [U	F9
Csi [V	F10
Csi [W	F11

Sequence	ANSI Function Key
Csi [X	F12
Csi [H	Home
Csi [F	End
Csi [L	Insert
Csi [I	PgDn
Csi [G	PgUp
Csi [E	Center
Csi [Y	Shift + F1
Csi [Z	Shift + F2
Csi [a	Shift + F3
Csi [b	Shift + F4
Csi [c	Shift + F5
Csi [d	Shift + F6
Csi [e	Shift + F7
Csi [f	Shift + F8
Csi [g	Shift + F9
Csi [h	Shift + F10
Csi [i	Shift + F11
Csi [j	Shift + F12
Csi [k	Ctrl + F1
Csi [l	Ctrl + F2
Csi [m	Ctrl + F3
Csi [n	Ctrl + F4

Sequence	ANSI Function Key
Csi [o	Ctrl + F5
Csi [p	Ctrl + F6
Csi [q	Ctrl + F7
Csi [r	Ctrl + F8
Csi [s	Ctrl + F9
Csi [t	Ctrl + F10
Csi [u	Ctrl + F11
Csi [v	Ctrl + F12
Csi [w	Ctrl + Shift + F1
Csi [x	Ctrl + Shift + F2
Csi [y	Ctrl + Shift + F3
Csi [z	Ctrl + Shift + F4
Csi [@	Ctrl + Shift + F5
Csi [[	Ctrl + Shift + F6
Csi [\\	Ctrl + Shift + F7
Csi []	Ctrl + Shift + F8
Csi [^	Ctrl + Shift + F9
Csi [_	Ctrl + Shift + F10
Csi ['	Ctrl + Shift + F11
Csi [{	Ctrl + Shift + F12

User-Defined Keys (DECUDK)

^DCS<C>;<L>|<def1>;<def2>;...;<defn>^ST

The control string that loads a user-defined key.

<C>   The clear parameter. Ø clears all UDK definitions before loading new values. 1 clears one key at a time, as each new UDK value is loaded.

<L>   The lock parameter. Ø locks the UDK definitions against future redefinition. 1 does not unlock the keys, it simply does not lock them.

                Once you lock UDKs, the only way to unlock them is to clear the **User-defined keys locked** check box. Click Advanced on the Emulation tab of the Terminal Setup dialog box to do this.

| The final character, identifying the control string as DECUDK.

<defn>   The definition for each UDK being defined, in the format of:

                <key identifier>/<key definition>

You include the definition(s) between the final character (|) and the ^ST. Each key consists of a key identifier (listed in the following table below) and a key definition (explained follwing the table), separated by a forward slash. A semicolon separates UDK definitions.

Key	Identifier	Key	Identifier
F6	17	F14	26
F7	18	F15 (Help)	28
F8	19	F16 (Do)	29
F9	20	F17	31
F10	21	F18	32
F11	23	F19	33
F12	24	F20	34
F13	25		

The <*key definition*> is a string of hex pairs that represent the ASCII hex codes for the characters in the key definitions. See page 248 for an ASCII character chart showing hex codes.

For example, the hex coding for DIR is as follows:

    44 = D
    49 = I
    52 = R
    ØD = Cr

Example of Defining UDKs

As an example, a valid UDK definition of VT [F11] to generate a directory listing on the VAX is:

    $^DCSØ;1|23/444952ØD^ST$

As another example, the following sequence clears all previous UDK definitions, loads [F6] with the string EDIT and [F8] with the string COPYCr, and does not lock the new definitions:

    $^DCSØ;1|17/45444954;19/434F5059ØD^ST$

Printer Control

The printer control functions select the printer mode, the screen region to print, and other printing options.

Printer Extent Mode (DECPEX)

Csi ?19h   When using the print screen function, print the entire screen in a print-screen operation.

Csi ?19l   When using the print screen function, print just the scrolling region (data inside the top and bottom margins) during a print-screen operation. The scrolling region is defined by the DECSTBM function (page 42).

Form Feed After Printing (DECPFF)

Csi ?18h   Send a form feed to the printer after printing a file, the screen, or display memory.

Csi ?18l   Do not send a form feed to the printer after printing.

This control function is linked to the **Auto formfeed** check box in the Print Setup dialog box.

Character Control by Inch and Row (DECSHORP)

Csi 0w   Print 10 characters per inch, 80 characters per row.

Csi 1w   Print 10 characters per inch, 80 characters per row.

Csi 2w   Print 12 characters per inch, 96 characters per row.

Csi 4w   Print 16.5 characters per inch, 132 characters per row.

Line Control (DECVERP)

Csi 0z   Print 6 lines per inch, 63 lines per page.

Csi 1z   Print 6 lines per inch, 63 lines per page.

Csi 2z   Print 8 lines per inch, 84 lines per page.

Csi 3z   Print 12 lines per inch, 125 lines per page.

Csi 4z   Print 2 lines per inch, 21 lines per page.

Csi 5z   Print 3 lines per inch, 32 lines per page.

Csi 6z   Print 4 lines per inch, 42 lines per page.

Media Copy (MC)

`Csi ?5i`   Turn on auto print mode (also called *log to printer*). Data received to the display, either from datacomm or from the keyboard, is sent to a local printer one line at a time.

Lines must be delimited by a linefeed (`LF`), form feed (`FF`), or vertical tab (`VT`) to be recognized as lines. The delimiting character is also sent to the printer.

`Csi ?4i`   Turn off auto print mode, and return to normal print mode.

This control function is linked to the **Logging on** check box in the Logging dialog box.

Printer Controller Mode (MC)

`Csi 5i`   Turn on printer controller mode (also called *passthru* mode). All characters, data received from datacomm, and control sequences are sent directly to the printer without appearing on the display. Everything but the `Csi 5i` and `Csi 4i` control functions are passed to the printer.

`Csi 4i`   Turn off printer controller mode, and return to normal print mode.

Print Screen (MC)

`Csi i` or   If the printer extent mode (DECPEX) is set to the *scrolling region*,
`Csi Øi`   only the scrolling region is printed; otherwise, the entire display is printed (not including display memory).

Print Line (MC)

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

Macros for the VT420

The following macros are supported when Reflection emulates a VT420 terminal:

Define Macro (DECDMAC)

^DC_S<n1>;<n2>;<n3>!zD...DSt

You can define macros that are interpreted as normal input when emulating the VT420 terminal. Any macro definitions that do not fit into the memory allotted are ignored. An RIS or DECSR clears all macro definitions. A soft reset (DECSTR) has no effect.

<n1>	Macro ID number (0-63). If you give a macro an ID number that already exists, the old macro definition is deleted and the new one is used. If *<n1>* exceeds 63, Reflection ignores this control function. Everything but the `Csi 5i` and `Csi 4i` control functions are passed to the printer.
<n2>	Determine how a new macro definition is treated: If *<n2>* is 0 or omitted, an old macro with the same macro ID number is deleted. If *<n2>* is 1, DECDMAC deletes all current macro definitions before defining this macro. If *<n2>* is a number that is not 0 or 1, the macro is ignored.
<n3>	Select the encoding format for the text of the macro definition: If *<n3>* is 0 or omitted, ASCII text is used in the macro. If *<n3>* is 1, hex pairs are used (each pair of characters in the macro is the hex value for a single ASCII character. If *<n3>* is a number that is not 0 or 1, the macro is ignored. The string of text and control functions to be performed when the macro is invoked. If hex pairs are used, you can also use a repeat introducer (!) in D...D. The repeat introducer lets you repeat any hex pair in the definition string any number of times. Embed the repeat sequences in D...D.

Invoke Macro (DECIMAC)

Csi <n>*z   Invoke a stored macro. *<n>* is the macro ID number used in DECDMAC. If *<n>* is not associated with a particular macro, Reflection ignores this control function. If a macro definition includes control functions, these functions remain in effect after the macro is invoked.

Request Macro Report (DECMSR)

Csi <n>*}   DSR request for a macro space report.

Request Space Report (no mnemonic)

Csi <n>*}   *<n>* is the number of bytes divided by 16 (rounded down).

Reflection for ReGIS Graphics Control Functions

The control functions in this section are applicable only if you're using Reflection for ReGIS Graphics for Windows.

ReGIS Graphics

The following table shows the device control strings and their 7-bit equivalents for entering and exiting ReGIS mode:

8-Bit Device Control String	7-Bit Equivalent Escape Sequence	Meaning
^{D}Csp or ^{D}Cs0p	^{E}ScPp or ^{E}ScP0p	Enter ReGIS at the same point in the command as when last exited. ReGIS commands are not displayed.
^{D}Cs1p	^{E}ScP1p	ReGIS begins at a new command. ReGIS commands are not displayed.
^{D}Cs2p	^{E}ScP2p	Enters ReGIS at the same point in the command as when last exited. ReGIS commands are displayed.
^{D}Cs3p	^{E}ScP3p	ReGIS begins at a new command. ReGIS commands are displayed.
^{S}T	^{E}Sc\	Exit ReGIS mode.

Graphics Mouse Buttons

In Reflection for ReGIS Graphics, the PC mouse buttons can send control sequences to the host that correspond to a combination of the Digital left, middle, and right mouse buttons. The default codes sent by each button are shown on page 115.

With the DECLBD function, the three mouse button functions can be configured to send codes other than the defaults, and different functions can be sent when the button is pressed and released. Up to six characters can be defined for either a button press or release.

Programming Locator Device Buttons (DECLBD)

^{D}C$_S$<C>$w<def1>;<def2>;...;<defn>^{S}T

The device control string that defines the mouse buttons.

<C>   The clear parameter. Ø clears all button definitions before defining new ones. 1 clears one button at a time.

$w   The intermediate and final characters that identify the string as DECLBD.

<defn>   The definition for each mouse button. The definition has the format:

            <button>/<down code>/ <up code>

You include the definitions between the final character (w) and the string terminator (^{S}T). The *<button>* is the button on the Digital mouse you want to define:

Digital Button	<button>
Left	1
Middle	2
Right	3

The *<down code>* and *<up code>* are the codes that the button sends when pressed and released. A forward slash separates the button number, the down code, and the up code; a semicolon (;) separates each button definition. If you only have a two-button mouse, you can hold down the Alt key and click to simulate the middle mouse button.

In the DECLBD sequence, the <*down code*> and <*up code*> are strings of hex pairs, with each hex pair representing a single ASCII character. Up to six hex pairs (six characters) can be specified for each down code and each up code. (See the ASCII character chart on page 250 for the hex values for each character.)

For example, if you want to clear all button definitions, and define the left mouse button to send the default PF3 key code (^{SS}3R) when pressed, and a carriage return character when released, you would use the following coding:

    ^DCSØ$w1/8F52/ØD^ST

The Ø following the ^DCS clears all current definitions, and the `1` following the `w` selects the left button. `8F52` are the hex pairs for ^{SS}3R, and the `0D` is the hex pair for the carriage return.

Sixel Graphics

This section contains information about the sixel data format and sixel control functions. Unless you use sixels to create graphics images pixel by pixel, you would typically not need to know this information.

A *sixel* is a vertical column of six display pixels, using six bits of an 8-bit character code. Each pixel, the smallest display unit on the screen, can be set either on (1) or off (Ø). Sixels are typically used to create and print graphics images, and design character sets and fonts. When you print a ReGIS graphics screen, for example, the screen is recorded as a series of sixels, which are then sent to a printer as a bitmap.

The Graphics tab of the Terminal Setup dialog box in Reflection for ReGIS Graphics allows you to control sixel printing options.

If you use sixels to design character sets and fonts, you can then download them as a soft character set; see 78 for information about defining a soft character set.

Sixel Data Format

^{D}CSPn1;Pn2;Pn3;q s...s^{S}T

The sixel data format. The semicolons (;) separate parameters in the control string.

Pn1   The pixel aspect ratio parameter. Values for *Pn1* are listed in the following table. An aspect ratio of 2:1 means that each dot drawn is 2 pixels high by 1 pixel wide. This parameter is ignored by level 1 printers; they always use the default ratio of 2:1.

Parameter	Aspect Ratio
(omitted)	2:1 (default)
Ø, 1	2:1
2	5:1
3, 4	3:1
5, 6	2:1
7, 8, 9	1:1

Pn2   Select how the background color is drawn. current color. current color.

Parameter	Meaning
Ø or 2	Pixels that are set to Ø (off) are set to the current background color (the default).
1	Pixels that are set to Ø (off) remain at their current color.

Pn3   The horizontal grid size parameter. The VT300 has a fixed grid size, and this parameter is ignored.

q   The final character. This identifies the control string as a sixel command.

s...s   The string of sixel data. Each character must be in the range ? (hex 3F) to ~ (hex 7E). Each character represents six vertical pixels whose binary value equals the character code minus hex 3F. The least significant bit is at the top. The data string can also contain sixel control functions; these are listed as follows.

Sixel Graphics Levels

When sixel data is sent to the host, the format of the data string sent depends on your selection in the **Graphics level** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box:

- When set to **Level 1**, Reflection sends sixel data using a 7-bit format ($^{E_S}C_P$ and $^{E_S}C\backslash$ are the 7-bit equivalents for the $^{D}C_S$ and ^{S}T characters):

  $^{E_S}C_P$ 1q s...s$^{E_S}C\backslash$

- When set to **Level 2**, Reflection sends sixel data using this format:

  $^{E_S}C_P$ Pn1;Pn2;Pn3;q"Pan;Pad;Ph;Pv s...s$^{E_S}C\backslash$

  The parameters *Pn1*, *Pn2*, and *Pn3* are as explained above. The parameters *Pan*, *Pad*, *Ph*, and *Pv* are the raster attributes introduced by the"character, as explained in the next section.

- When set to **LA210**, Reflection uses the same sixel data format as **Level 1** for compressed printing.

  For expanded and rotated printing, the following format is used:

  $^{E_S}C_P$ 9q s...s$^{E_S}C\backslash$

  The parameter 9 specifies a 1:1 aspect ratio.

Sixel Control Functions

In the data string that is part of the sixel device control string above, control functions can be included to repeat graphics, change raster attributes, and select colors.

!Pn<character>	Graphics repeat introducer. The ! character is a graphics repeat introducer, causing <*character*> to be repeated *Pn* times. The character to repeat must be in the range ? (hex 3F) to ˜ (hex 7E).
"Pan;Pad;Ph;Pv	Raster attributes. The " character sets sixel raster attributes. *Pan* is the pixel aspect ratio numerator, and *Pad* is the pixel aspect ratio denominator. *Ph* and *Pv* define the size of the image in horizontal and vertical pixels (this does not limit the size of the sixel image, however).

`#Pc;Pu;Px;Py;Pz`	Color introducer. The # character is the color introducer. When only the first parameter is specified, this function selects color map entry *Pc*. When the remaining parameters are included, the function loads color map entry *Pc* with HLS or RGB colors. *Pu* indicates the color coordinate system: *1* indicates HLS values; *2* indicates RGB values. *Px* specifies either a hue or red intensity, depending on the coordinate system, *Py* specifies a lightness or green intensity, and *Pz* specifies a saturation or blue intensity.
`$`	Graphics carriage return. The $ character is a graphics carriage return, returning the active position to the left margin of the same sixel line. This is used to overprint a line of sixels.
`-`	Graphics new line. The – character returns the active position to the left margin of the next sixel line.

Sixel Scrolling

When sixel scrolling is enabled, the sixel image begins at the top left of the active text position. A sixel image will scroll the display when the image reaches the bottom margin of the display (the image may also scroll off the top of the display). A graphics new line character (–) is sent immediately after the sixel dump, and the text cursor is set at the same position as the sixel cursor upon exiting sixel mode.

With sixel scrolling disabled, the sixel image begins at the top left of the display. When the image reaches the bottom margin, the display does not scroll, and additional sixel commands are ignored. Upon exiting sixel mode, the text cursor is set at the same position as when sixel mode was entered.

Enable/Disable Sixel Scrolling (DECSDM)

`Csi ?8Øl`   Enable sixel scrolling.

`Csi ?8Øh`   Disable sixel scrolling.

This control function is linked to the **Jump** and **Smooth** options in the **Scrolling** group box on the Screen tab of the Display Setup dialog box.

Graphics Printing

The following control functions are for printing graphics.

Expanded/Compressed Print Mode (DECGEPM)

Csi ?43h   Select an expanded image for a graphics print screen.

Csi ?43l   Select a compressed image for a graphics print screen.

This control function is linked to the **Print mode** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box.

Print Rotated/Compressed Mode (DECGRPM)

Csi ?47h   Send a graphics image to the printer rotated 90 degrees.

Csi ?47l   Send a compressed image to the printer. This performs the same function as the Csi ?43l sequence described previously.

This control function is linked to the **Print mode** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box.

Print Color/Black and White Mode (DECGPCM)

Csi ?44h   Send a color image to the printer for a graphics print screen.

Csi ?44l   Send a black and white image to the printer for a graphics print screen.

This control function is linked to the **Color printing** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box.

Print Color Syntax (DECGPCS)

Csi ?45h   Select the RGB color format for a graphics print screen.

Csi ?45l   Select the HLS color format for a graphics print screen.

This control function is linked to the **Color specification** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box.

Print Background (DECGPBM)

Csi ?46h   Send the entire graphics image, including the background, to the printer for a graphics print screen.

Csi ?46l   Send the graphics image, except for the background, to the printer for a graphics print screen.

This applies only when the DECGPCM function is set to send a color image.

This control function is linked to the **Print background** check box in the **Printing** group box on the Graphics tab of the Terminal Setup dialog box.

Send Graphics To Host/Printer (MC)

Csi ?2i   Send graphics images to the host when a ReGIS hard copy command is issued, or when graphics are printed using Reflection's printing commands. The image is sent to the host as a stream of sixel data. (The hard copy command is an option of the screen command.)

Csi ?i or
Csi ?Øi   Send graphics images to the printer when a ReGIS hard copy command is issued, or when graphics are printed using Reflection's printing commands.

The ReGIS hard copy command is an option of the print screen command, described on page 135.

This control function is linked to the **Destination** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box.

Selecting and Mapping Character Sets

Reflection supports two types of VT terminal character sets:

- *Hard* character sets, which are built into Reflection

- *Soft* character sets, which you can create and download to Reflection

Character sets are selected by control functions, and by the Ctrl+N and Ctrl+O keystrokes. As an example of loading and using a different character set, perform the following steps:

1. Clear the **Online** check box on the Emulation tab of the Terminal Setup dialog box; this puts Reflection in local mode.

2. Press Esc, then type)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. The escape sequence ᴱSC)Ø selects the Digital Special Graphic character set, and the Ctrl+N keystroke maps the character set for use (this is explained later in this chapter).

5. Now press Ctrl+O, and type abcd again. The characters on the screen are the normal characters. The codes stored in display memory and transmitted to the host are the same in both cases; it is only the representation of the code on the screen that is different.

Character Set Support

Characters are divided into the following character sets:

- ASCII

- Digital Supplemental Graphic

- ISO Latin-1 supplemental graphic

- National replacement sets (12)

- Digital Special Graphic

- Digital Technical

- Dynamically redefinable character set (DRCS)

All of the above character sets—except the dynamically redefinable character set—are *hard* character sets; that is, those built in to Reflection. The DRCS characters are soft characters that can be downloaded.

The Digital Supplemental Graphic and ISO Latin-1 supplemental graphic character sets are composed of symbols and characters for the English language and many Western European languages. For example, the £ symbol and the Ä character are in the supplemental character sets.

Together, the ASCII character set and the Digital Supplemental Graphic set comprise the *Digital Multinational* character set. The ASCII character set and the ISO Latin-1 supplemental graphic set comprise the *ISO Latin Alphabet Nr 1* character set. The two sets have only a few differences, as seen in the charts on 252 and 253.

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. For example, the Ä character is contained in the Finnish, German, and Swedish NRC sets. NRC sets are used primarily in 7-bit operating environments that cannot access supplemental characters.

The *Digital Special Graphic* and *Digital Technical* character sets contain characters, line-drawing elements, and mathematical symbols; see 254 and 255 for charts of these two sets.

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 C_R, L_F, E_SC, and D_CS; 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 on 68; 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 Digital 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, because 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, defining all 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

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.

When you start Reflection, it places the following character sets in the in-use table:

• ASCII in GL

• Digital Supplemental Graphic in GR

You can have Reflection place the ISO Latin-1 supplemental graphic set into GR by selecting it as the *user-preferred supplemental set* (UPSS). Use DECAUPSS on 84 to do this, or select **ISO Latin-1** from the **Host character set** list on the Emulation tab of the Terminal Setup dialog box.

There are two steps to selecting a character set:

1. Designate the set.

    Character sets can be designated as G0, G1, G2, or G3. These are *logical* designations. You can designate up to four character sets, and switch among them by selectively mapping them to the in-use table.

2. Map the set.

    After designating a character set as G0, G1, G2, or G3, you map it into either GL or GR, which places it in the in-use table. You can then display and send any character from that set.

Designating a Character Set

You designate a character set as G0 through G3 using a select character set (SCS) escape sequence. A 96-character set cannot be designated as G0 (the ISO Latin-1 supplemental character set is the only built-in 96-character set). The following section describes the codes used to select each available character set.

Select Character Set (SCS)

^{E}S$_C$<Gn><set>

This sequence designates a character set. The character you enter as *<Gn>* to specify G0, G1, G2, or G3 depends on whether you are selecting a 94- or 96-character set:

94-Character Set Table	<Gn>	96-Character Set Table	<Gn>
G0	(	G0	(not available)
G1	)	G1	- (hyphen)
G2	*	G2	. (period)
G3	+	G3	/ (forward slash)

You must use a 94-character set SCS sequence to designate a 94-character set, and a 96-character set SCS sequence to designate a 96-character set.

The <*set*> indicates the character set selected. The character sets that can be designated as the <*set*> are specified by a one or two character code, as follows:

Character Set Name	The <set> Specifier
ASCII	B
Digital Special Graphics	Ø
Digital Supplemental Graphics	%5
Digital Technical	>
DRCS	Name of downloaded soft character set
ISO Latin-1	A
User-preferred supplemental	<
ASCII	B
Digital Special Graphic (line drawing)	Ø
Digital Supplemental Graphic (multinational)	<
Digital Technical	1

For example, in VT420 mode the following sequence designates the ISO Latin-1 supplemental graphic set (a 96-character set) as the G3 logical set:

   ^{E}S$_C$/A

Mapping a Character Set

After designating a character set as G0, G1, G2, or G3, you must map the set into GL or GR of the in-use table. The values in GL and GR are used to translate all incoming codes and how you see them on the display.

The GL character set translates codes 32–127 (characters with no high bit set), and the GR character set translates codes 160–255 (characters with the high bit set). Characters 0–31 and 128–159 are C0 and C1 control codes, respectively.

Locking-shifts and *single-shifts* are used to map character sets. These are explained next.

Locking Shifts (LS)

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

.ItalicT VT200 Mode and Above:

Locking Shift	Keystroke	Control Code	Function
LS0 (locking shift 0)	`Ctrl`+`O`	S_I	Map G0 into GL
LS1 (locking shift 1)	`Ctrl`+`N`	S_O	Map G1 into GL
LS1R (locking shift 1, right)		E_{SC}~	Map G1 into GR
LS2 (locking shift 2)		E_{SC}n	Map G2 into GL
LS2R (locking shift 2, right)		E_{SC}}	Map G2 into GR
LS3 (locking shift 3)		E_{SC}o	Map G3 into GL
LS3R (locking shift 3, right)		E_{SC}|	Map G3 into GR

In the following example, the Digital Special Graphic character set is designated as G2, then G2 is mapped into GR:

E_{SC}*Ø
E_{SC}}

Single Shifts (SS)

When you want to display just the next character using a different character set, use a single shift. A single shift maps the character set designated as G2 or G3 into GL for one displayable character. GL then automatically reverts to its previous mapping. Single shifting is like pressing the `Shift` key to generate a different character for a single keystroke, then releasing `Shift`.

Reflection has two single-shift escape sequences available:

Single Shift	8-Bit Character	7-Bit Equivalent	Function
Single shift 2	S_{S2}	E_{SC}N	Maps G2 for next character into GL
Single shift 3	S_{S3}	E_{SC}0	Maps G3 for next character into GL

The following figure summarizes how to designate and map characters sets in VT200 mode and higher:

Character Sets

Character Set	SCS Specifier
ASCII	B
DEC Supplemental Graphic	% 5
ISO Latin-1 Supplemental (96 characters)	A
DEC Special Graphic	0
DEC Technical	>
User-Preferred Supplemental	<
DRCS	name of set
NRC sets	see table in section below

1 Designate character set as one of four logical sets (G0, G1, G2, G3).
Refer to chart at right for <specifier> character.

For logical set:	Type:
G0 (94-character set)	ESC(<specifier>
G1 (94-character set)	ESC) <specifier>
G1 (96-character set)	ESC- <specifier>
G2 (94-character set)	ESC* <specifier>
G2 (96-character set)	ESC. <specifier>
G3 (94-character set)	ESC+ <specifier>
G3 (96-character set)	ESC/ <specifier>

Logical Sets

G0	G1	G2	G3

2 Map logical set to in-use table (GL and GR).

For locking shift:	Type:
LS0 (SI)	Control-O
LS1 (SO)	Control-N
LS1R	ESC~
LS2	ESCn
LS2R	ESC}
LS3	ESCo
LS3R	ESC C

For single shift:	Type:
SS2	ESCN
SS3	ESCO

In-Use Table

GL	GR

National Replacement Character Sets

The national replacement character sets are 94-character sets for use primarily in 7-bit operating environments. Each NRC set replaces some ASCII characters with characters from a specific European language. The NRC sets are useful in environments that cannot access the 8-bit ISO Latin-1 or Digital Supplemental Graphic character sets to generate national characters. The chart on page 267 shows which characters are replaced for each language.

Only one NRC set can be active at a time. To use an NRC set:

1. On the Setup menu, click Terminal.

2. On the Emulation tab, click Advanced to open the Advanced Options dialog box.

3. Select the NRC set you want active from the **National Replacement Set** list.

4. Enable the selected NRC set by doing one of the following:

    • Issue the DECNRCM control sequence given on page 78, or select the **Use NRC (7-Bit) Set** check box (this option is not available if **None** is selected).

    • Configure Reflection with a 7-bit data path for parity. See page 15 for information on changing parity.

When you perform these steps, the selected NRC set is automatically designated and mapped into GL.

Once an NRC set is selected and enabled, if the character you want is on your keyboard, just press the key to create it. For example, if you have enabled the French character set and you have a French keyboard, press the ç, key to create the ç character.

If the national character you want is not on your keyboard, you can press the key shown in the chart on 267 to produce the character. For example, to create the ç character on a U.S. keyboard, press the \ key. If the selected NRC set is **French**, the ç character replaces the backslash character in the ASCII character set.

If the national character you want is not on your keyboard, you can use a compose sequence to create it; see 259 for more information.

If, for some reason, the host changes the character set stored in GL and the selected NRC set is no longer mapped there, you can perform the following two steps to remap the NRC set:

1. Designate the character set as G0, G1, G2, or G3, as explained on 72. You must use one of the 94-character set designators to select an NRC set. The NRC set specifiers are listed in the following table.

2. Map the designated NRC set into GL, using the mapping codes on 73. You can map only the set selected in the **National Replacement Set** list, explained on 76.

    In a 7-bit operating environment, you should always map the selected NRC set into GL because GR is inaccessible in 7-bit modes.

With an 8-bit data path, using an NRC set is optional; when a character set other than **None** is selected in the **National Replacement Set** list box, you *must* enable the set either with the **Use NRC (7-Bit) Set** check box or the DECNRCM control sequence.

With a 7-bit data path and a NRC set other than **None**, the set is enabled automatically and the **Use NRC (7-Bit) Set** check box is automatically selected.

The following table lists the codes used to designate each NRC set with the SCS function (on 72). When more than one designator is listed, it's recommended that you use the first one.

Designate NRC Sets

NRC Set Name	The <Set> Specifier
British	A
Canadian	9 or Q
Dutch	4
Finnish	5 or C
French	R
German	K
Italian	Y
Norwegian/Danish	` or E or 6
Portuguese	%6
Spanish	Z

NRC Set Name	The <Set> Specifier
Swedish	7 or H
Swiss	=

For example, after **British** is selected in the **National Replacement Set** list, the British NRC set is designated as G1, then mapped into GL, using the following:

   ^{E}S$_C$)A

Then, press Ctrl+N to transmit the control code So to load this character set.

Use National Replacement Characters (DECNRCM)

^{CS}I?42h   Reflection uses 7-bit characters from the selected national replacement set.

^{CS}I?42l   Reflection uses 7- and 8-bit characters from the Digital Multinational or ISO Latin-1 character sets.

This control function is linked to the **Use NRC (7-Bit) Set** check box.

Soft Character Sets

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 VT200 mode and higher.

After designing and downloading a soft character set, you designate and map it the same way you designate and map hard character sets (see 72).

Designing a Soft Character Set

Reflection displays each character as a series of dots using pixels, the smallest displayable units on the screen. Each pixel can be either off or on. Each character is confined to a space called the *character cell*. The size of the default character depends on the type of terminal Reflection is emulating.

When designing your own characters, map out character cells on grid paper. Each box on the grid paper can represent one pixel. You can design characters for an 80-column or 132-column font. Reflection truncates characters that are larger than the cell size.

The tables following list the default dimensions for designing soft characters for the VT420, VT330, VT320, and VT220 terminals: VT340 and VT240 only apply if you're using Reflection for ReGIS Graphics. Characters can be defined as either *full cell*—occupying the entire character cell—or *text cell*—occupying a slightly smaller space to account for spacing between text characters.

Note: When emulating a VT420 terminal, you can load up to six variations of character wide and height. Each character's height is controlled by the number of rows you display in the terminal window; the width remains identical for the columns (80 or 132) you select.

	80-Column Font	132-Column Font
VT420 Character Dimension	**Number of Pixels**	**Number of Pixels**
Full cell width✥height (24 rows)	10✥16	6✥16
Text cell width	8	4
Full cell width✥height (36 rows)	10✥10	6✥10
Full cell width✥height (48 rows)	10✥8	6✥8
	80-Column Font	**132-Column Font**
VT330/VT340 Character Dimension	**Number of Pixels**	**Number of Pixels**
Full cell width✥height	10✥20	6✥20
Text cell width	9	5
	80-Column Font	**132-Column Font**
VT320 Character Dimension	**Number of Pixels**	**Number of Pixels**
Full cell width✥height	15✥12	9✥12
Text cell width	12	7
	80-Column Font	**132-Column Font**
V220-VT240 Character Dimension	**Number of Pixels**	**Number of Pixels**
Full cell width✥height	8✥10	6✥10
Text cell width	7	5

Soft Character Sets and VT420 Emulation

Character Width	Number of Pixels
For 80 columns	10
For 132 columns	6

Character Height	Number of Pixels
For 24 rows	16
For 36 rows	10
For 48 rows	8

Converting Binary Sixel Values to ASCII Characters

After you've designed a character, you must convert it to a hexadecimal value. Every place in your character where you want a pixel turned on you give a binary value of *1*, and every place where you want a pixel turned off you give a binary value of Ø.

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.

Perform the following steps below to convert a character from a binary grid to its sixel representation to its hexadecimal values:

1. After designing a character, divide the DRCS character cell vertically into groups of 6-pixel-high sixels. The number of groups depends on the cell height. The first group contains the sixels for the top 6-pixel section of the character cell; the second group contains the sixels for the next section of the character cell, and so on. For the VT330 and VT340, there can be as many as four groups of sixels (when loading a 20-pixel-high character, the fourth group contains only two pixels). The least significant bit is at the top, and the most significant bit is at the bottom. Reflection receives the sixel columns in order (from 1 to the maximum character cell width).

2. Convert the binary value of each sixel to its hexadecimal value. Sixel codes must be in the range ? (hex 3F) to ˜ (hex 7E), so you must add an offset of hex 3F to the hexadecimal value of each sixel. For example, the binary value ØØØØØ1 has a hex value of 01. Add hex 3F and the result is hex 40.

3. Convert the hexadecimal value to its ASCII character equivalent. You can find the ASCII equivalents for hex values in the chart on 250. For example, the character equivalent of hex 40 is @.

The complete conversion of a binary character pattern to its ASCII character equivalent results in a string of up to 40 ASCII characters for the VT330 and VT340 terminals—up to 10 characters for each of the four sections of the character cell.

Once you have made the binary conversion to ASCII characters, you can download your DRCS characters to Reflection, using the DECDLD device control string.

Downloading a Soft Character Set (DECDLD)

^{DC}S Pfn;Pcn;Pe;Pcmw;Pw;Pt;Pcmh;Pcss{ Dscs UUUUUUUU/LLLLLLLL;...^{S}T

The device control string that loads a soft character set.

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 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 terminals allow only a 94-character set, and the valid range for this parameter is 1–94. For the VT420 and VT340 (VT340 is only applicable for Reflection for ReGIS Graphics), the valid range is 0–95.
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 (see tables on 79)

1	invalid	
2	5✻10 pixel cell	
3	6✻10 pixel cell	
4	7✻10 pixel cell	
5–10	specifies width in pixels	
	Values *2*, *3*, and *4* are VT200 compatible, automatically setting the height of the cell to 10 pixels.	
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.	
Pcmh	Character matrix height. This selects the maximum height of the cell. If this parameter is omitted or Ø, the default height is used. The default for the VT320 is 12 pixels high; the default if you're using Reflection for ReGIS Graphics for the VT330 and VT340 is 20 pixels high. Values from *1* to *16* (VT420), 1 to 12 (VT320), or 1 to 20 (VT340) select a character cell with that height. This para-meter is ignored if *Pcmw* is *2*, *3*, or *4*.	
Pcss	Character set size:	
Ø	94-character set *12*	36 rows, 132 columns
1	96-character set *21*	48 rows, 80 columns
2	24 rows, 132 columns *2248*	48 rows, 132 columns
11	36 rows, 80 columns	
	This also affects the *Pcn* starting position parameter. Not valid for the VT200 series.	
{	The final character of the string, marking the end of the parameter list and indicating that the string is a DECDLD function.	

Dscs   Soft character set name. This is name you use in the SCS escape
        sequence (see 72) to select the set. 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 can define a soft character set that is currently
        unused. The value of *Pcss* determines whether the set contains 94 or
        96 characters. The recommended default for a soft character set name
        is *<SP>*@.

UUUUUUUU/LLLLLLLL

        The ASCII characters that represent the sixel patterns defining each
        character. Sixel bit patterns are separated with a semicolon. Your
        character set can have 1 to 94 or 1 to 96 patterns, depending on the
        character set size (the *Pcss* parameter). *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. You can then map it into GL or GR of the in-use table.

Clearing a Soft Character Set (DECDLD)

^{DC}S1;1;2{<SP>@^ST

The device control string that clears a soft character set loaded into Reflection.

Any of the following actions also clears a soft character set:

• Quitting Reflection, then restarting.

• Resetting to initial state with ^{ES}Cc (see 106).

• Performing a soft reset with ^{CS}I!p (see 106).

User-Preferred Supplemental Character Set

You can assign the Digital Supplemental Graphic or ISO Latin-1 supplemental character set as the user-preferred supplemental set (UPSS). This is the supplemental set loaded by default when you start Reflection.

Assigning a User-Preferred Supplemental Set (DECAUPSS)

^{DC}S∅!u%5^{S}T Assign the Digital Supplemental Graphic set as the UPSS.

^{DC}S1!uA^{S}T   Assign the ISO Latin-1 supplemental set as the UPSS.

This control function is linked to the **Host character set** list on the Emulation tab of the Terminal Setup dialog box.

Requests and Reports

The host can request many types of information from Reflection, and depending on the response, can adapt the computing environment to match Reflection's capabilities. Reflection can provide the following kinds of information:

- Type of terminal (identification)

- Cursor state

- Operating status

- Operating level

In general, the host can request information about all the options you or a host can set in Reflection.

Device Attributes

When the host sends a device attributes (DA) request, Reflection provides conformance level (1, 2, or 3) and extensions, basic features, and identification code information. The host can use the information it receives to make the best use of Reflection's features. This information also allows the host to determine the cause of certain communication errors.

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

Primary DA Request (DA)

$^{C}S_{I}c$ or $^{C}S_{I}\text{Ø}c$

With either of these two control functions, the host can request Reflection's service code, conformance level, and basic attributes.[*]

Reflection replies according to the setting of the **Terminal ID** list on the Emulation tab of the Terminal Setup dialog box. The reply can be one of the following:

Primary DA Responses

Terminal ID	Reflection Reply
VT100	$^{E}S_{C}$[?1;2c
VT101	$^{E}S_{C}$[?1;0c
VT102	$^{E}S_{C}$[?6c
VT220	$^{E}S_{C}$[?62;1;2;6;7;8;9c
VT240	$^{E}S_{C}$[?62;1;2;3;4;6;7;8;9c
VT241	$^{E}S_{C}$[?62;1;2;3;4;6;7;8;9c
VT320	$^{E}S_{C}$[?63;1;2;6;7;8;9c
VT330	$^{E}S_{C}$[?63;1;2;3;4;6;7;8;9;13;15;16;18;19c
(Reflection for ReGIS Graphics, only)	
VT340	$^{E}S_{C}$[?63;1;2;3;4;6;7;8;9;13;15;16;18;19c
(Reflection for ReGIS Graphics, only)	
VT420	$^{E}S_{C}$[?64;1;2;6;7;8;9;15;18;19;21c
WRQ	$^{E}S_{C}$[?63;1;2;6;7;8;9;11;14;15;17c
(Reflection for UNIX and Digital and ReGIS Graphics, VT320)	
WRQ	$^{E}S_{C}$[?63;1;2;3;4;6;7;8;9;11;14;15;16;17c
(Reflection for ReGIS Graphics, VT340 emulation)	
WRQ	$^{E}S_{C}$[?64;1;2;6;7;8;9;15;18;19;21c
(Reflection for UNIX and Digital and ReGIS Graphics, VT420 emulation)	

---

[*] $^{E}S_{C}Z$ (DECID) is also a valid request, but its use is only recommended on a VT52 terminal.

The meaning of the numbers in the "Number" column indicate which features are supported:

Number	Meaning
62	VT200 series terminal
63	VT300 series terminal
64	VT400 series terminal
1	132 columns
2	Printer port
3	ReGIS display
4	Sixel graphics
6	Selective erase
7	Soft character set (DRCS)
8	User-defined keys (UDKs)
9	National replacement character sets
11	Status line
13	Local editing mode
14	8-bit interface
15	Digital Technical character set
16	Locator device port
17	Terminal state reports
18	Windowing capability
19	Dual sessions
21	Horizontal scrolling

With either of the following two control functions, the host can request a report of Reflection's parity and baud rate settings:

^{CS}I1x
^{CS}IØx

Reflection responds with the following for a ^{CS}IØx query:

^{CS}I2;<n1>;<n2>;<receive baud rate>;<transmit baud rate>;1;Øx

Reflection responds with the following for a ^{CS}I1x query:

^{CS}I3;<n1>;<n2>;<receive baud rate>;<transmit baud rate>;1;Øx

Note: Each of the response strings do the same thing: what differentiates them is the beginning number in the response (2 for^{CS}IØx, and 3 for^{CS}I1x).

The values for *<n1>* and *<n2>* in the responses are based on setting of parity:

Parity	<n1>	<n2>
8/None	1	1
8/Even	5	1
8/Odd	4	1
7/None	1	Ø
7/Even	5	Ø
7/Odd	4	Ø
7/1's (mark, or "ones" parity)	3	Ø
7/0's (space, or "zeros" parity)	2	Ø

The values for *<receive baud rate>* and *<transmit baud rate>* in the responses are based on the setting of the baud rate.

Baud	<Value>	Baud	<Value>
110	16	3600	96
150	32	7200	108
300	48	9600	112
600	56	19200	120
1200	64	38400	128
1800	72	57600	136
2400	88	115200	144
4800	104		

For example, when the host requests terminal parameters with ^{CS}I1x, Reflection might respond with:

   ^{CS}I4;1;1;144;144;1;0x

This means that parity is set to **8/None**. The receive and transmit baud rates are both **19200**.

Secondary DA Request (DA)

^{CS}I>c	With either of these two control functions, the host can request
^{CS}I>Øc	Reflection's identification code and hardware options (*the secondary device* attributes).
^{CS}I>41;11;Øc	Reflection response to the above request. The code *41* identifies Reflection as a VT420 terminal. The *11* indicates a firmware version, and the Ø indicates that there are no hardware options.

Locator Device Port

CSI?55n   Request from the host for the status of the locator device (the mouse). A response is sent only if the operating mode is VT200 or higher.

CSI?5Øn   Reflection response to CSI?55n, indicating that a locator device is present.

CSI?53n   Reflection response to CSI?55n, indicating that no locator device is present.

CSI?56n   Request from the host to identify the type of locator device. A response is sent only if the operating mode is VT200 or higher.

CSI?57;Øn   Reflection response to CSI?56n, indicating that no locator device is connected, or the device is not a mouse or tablet.

CSI?57;1n   Reflection response to CSI?56n, indicating that the locator device is a mouse.

Device Status Reports

The host can send a device status report request to Reflection for information on the following features:

- Operating status

- Cursor position

- Printer port

- User-defined keys

- Keyboard dialect

There are different DSR requests for different features. DSR requests and reports follow either ANSI standard or Digital private format.

Operating Status (DSR)

CSI5n   Host request for Reflection's operating status. Reflection always responds with CSIØn, indicating no malfunctions.

Cursor Position Report (CPR)

^CS_I6n   Host request for a cursor position report.

^CS_I<row>;<column>R

Reflection response to the above request. The <*row*> and <*column*> indicate the current row and column of the cursor.

Printer Status (DSR)

^CS_I?15n   Host request for the current printer status. Reflection responds with one of the following sequences.

^CS_I?13n   Reflection response to the above request, indicating that the host cannot print to a PC printer (there is an active printer).

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

UDK Status (DSR)

^CS_I?25n   Host request for the UDK status: locked or unlocked. This request is valid only for the VT200 or higher series terminals. Reflection responds with one of the following sequences:

^CS_I?2Øn   Reflection response to the above request, indicating that the UDKs are unlocked.

^CS_I?21n   Reflection response to the above request, indicating that the UDKs are locked.

Keyboard Dialect (DSR)

^CS_I?26n   Host request for keyboard dialect. Reflection responds with the following sequence:

^CS_I?27;<n>nReflection response to the above request. The possible values for <n> are:

<n>	Keyboard	<n>	Keyboard
1	North American	9	Italian
2	British	10	Swiss (French)
3	Flemish (French NRC)	11	Swiss (German)
4	Canadian (French NRC)	12	Swedish
5	Danish	13	Norwegian
6	Finnish	14	French/ Belgian
7	German	15	Spanish
8	Dutch	16	Portuguese

Terminal State Reports

With the request terminal state report sequence (DECRQTSR), the host can request information about Reflection's current operating state. The host can save the reported information; if an application makes temporary changes to the state, the original state can be restored. These functions are available only in VT200 mode and above.

Request Terminal State Report (DECRQTSR)

^CS_I1$u   Host request for a terminal state report.

^CS_I2$u   Host request for color table report (Reflection for ReGIS Graphics, VT340 only).If Reflection is configured as a VT340 terminal, it responds with the following DECCTR sequence. Otherwise, it responds with ^DC_SØ$s^S_T.

^CS_I2;<n>$u   This sequence also requests a color table report (Reflection for ReGIS Graphics, VT340 only). By including an additional parameter *<n>*, you can specify the color coordinate system to use in the report.

<n>	Color Coordinate System
0, 1, or none	HLS (default)
2	RGB

Terminal State Report (DECTSR)

^DC_S1$sD...D<checksum 1><checksum 2>^ST

The sequence sent in response to the DECRQTSR sequence ^CS_I1$u. DECTSR provides the host with a complete report on Reflection's current operating state, except UDKs and the soft character set.

D...D is the data string indicating the status of various features. *<checksum 1><checksum 2>* is a two-byte checksum of all the data in the report.

Color Table Report (DECCTR)

^DC_S2$sD...D^ST

Report the values of the current color table (Reflection for ReGIS Graphics, VT340 only). This is the response sent for the DECRQTSR sequences ^CS_I2$u and ^CS_I2;<n>$u.

If the color coordinate system is HLS, the format of the data string *D...D* is:

    <color 0>;1;H;L;S/<color 1>;1;H;L;S/.../<color 15>;1;H;L;S

If the color coordinate system is RGB, the format of the data string *D...D* is:

    <color 0>;2;R;G;B/<color 1>;2;R;G;B/.../<color 15>;2;R;G;B

Restore Terminal State (DECRSTS)

^DC_S1$pD...D^ST

Restore Reflection's operating state to the state stored with DECTSR. *D...D* indicates the same data string saved with DECTSR.

Restore Color Table (DECRSTS)

^DC_S2$pD...D^S_T

Restore the color table to the state stored with DECCTR (Reflection for ReGIS Graphics, VT340 only). For other terminal modes, applications cannot use DECRSTS to restore a color table; any attempt to restore a color table is ignored. The format of the data string D...D is identical to the format given above for the DECCTR function.

Note: The format of the data string used by the terminal state store and restore functions may change between software versions and is not intended to be parsed or used by host applications. The data received should be returned unmodified.

Presentation State Reports

With the DECRQPSR function, the host can request information about Reflection's presentation state, such as the current cursor position, character sets, and tab stops.

Depending on the type of request, Reflection responds with one of two presentation state reports:

- Cursor Information Report (DECCIR), see following

- Tab Stop Report (DECTABSR), see 99

Presentation State Report Request (DECRQPSR)

^CS_I<n>$w   Host request for a presentation state report.

<n>	Report requested
1	Cursor information report
2	Tab stop report

Cursor Information Report (DECCIR)

^DC_S1$uD...D^ST

Reflection response to ^CS_I1$w (the DECRQPSR sequence).

The cursor information report provides the status of the following:

- Cursor position
- Visual attributes
- Character protection attribute
- Origin mode (DECOM)
- Active character sets

D...D is the data string. Its format is:

    Pr;Pc;Pp;Srend;Satt;Sflag;Pgl;Pgr;Scss;Sdesig

Each element of the data string is:

Pr   The number of the line containing the cursor.

Pc   The number of the column containing the cursor.

Pp   The number of the current page; for the VT420 this is always 1.

Srend   One or more ASCII characters indicating the visual attributes, such as bold and underline, currently in use for writing (see 27).

To find out which attributes are set, you need to convert the character to an 8-bit binary number. After converting a character to its binary number, use the following table to determine the meaning of *Srend*. Bit 8 is the most significant bit and bit 1 is the least significant.

Bit	Attribute	Bit Value
8	—	Always 0 (off)
7	—	Always 1 (on)
6	Extension indicator	0 = No more attribute data
		1 = Another character of visual
		attribute data follows this one
5	—	Always 0 (off)

Bit	Attribute	Bit Value
4	Reverse video	0 = Off
		1 = On
3	Blinking	0 = Off
		1 = On
2	Underline	0 = Off
		1 = On
1	Bold	0 = Off
		1 = On

For example, suppose the *Srend* character reported is an ASCII uppercase L. The binary equivalent of L (decimal 76 in the ASCII character chart) is 01001100. The 7th bit is always on (1), and with the 4th and 3rd bits also on, this code means that the reverse video and blinking attributes are set.

Satt   One or more characters that indicate any selective erase attributes currently set for writing (see the information starting on 46). The same conversion method is used as explained above for the *Srend* parameter. The following table explains the meaning of the 8-bit binary number:

Bit	Attribute	Bit Value
8	—	Always 0 (off)
7	—	Always 1 (on)
6	Extension indicator	0 = No more protection data
		1 = Another character of selective
		erase data follows this one
5	—	0 = Reserved for future use
4	—	0 = Reserved for future use
3	—	0 = Reserved for future use
2	—	0 = Reserved for future use

Bit	Attribute	Bit Value
1	Selective erase	0 = Off
	(DECSCA)	1 = On

For example, if the *Satt* character reported is an ASCII uppercase A, selective erase is set for writing. The letter A has a decimal value of 65, giving a binary equivalent of 01000001. The 7th bit is always 1, and the 1st bit set to 1 means that selective erase is on.

Sflag   One or more characters that indicate several flags and modes Reflection must save. Again, to decode the character, you must convert it to an 8-bit binary number as explained above. The following table explains the meaning of the 8-bit binary number.

Bit	Attribute	Bit Value
8	—	Always 0 (off)
7	—	Always 1 (on)
6	Extension indicator	0 = No more flag data
		1 = Another character of flag data follows this one
5	—	0 = Reserved for future use
4	Autowrap	0 = Autowrap not pending
		1 = Autowrap pending
3	Single shift 3	0 = SS_3 is off
	(SS_3) setting	1 = G3 mapped into GL only for next typed character
2	Single shift 2	0 = SS_2 is off
	(SS_2) setting	1 = G2 mapped into GL only for next typed character
1	Origin mode	0 = Origin mode reset
		1 = Origin mode set

For example, if the *Sflag* character reported is an ASCII uppercase J, autowrap is pending, and a single shift 2 has been received. The letter J has a decimal value of 74 and a binary value of `01001010`. The 7th bit is always 1, and the 4th and 2nd bits set to one correspond to the autowrap pending and single shift 2 flags.

Pgl   The number of the logical character set mapped into GL:

Code	Meaning
0	G0 is in GL
1	G1 is in GL
2	G2 is in GL
3	G3 is in GL

Pgr   The number of the logical character set mapped into GR:

Code	Meaning
0	G0 is in GR
1	G1 is in GR
2	G2 is in GR
3	G3 is in GR

Scss   A character that indicates the size of the character sets in G0 through G3. Convert the character to an 8-bit binary number, and refer to the following table.

Bit	Attribute	Bit Value
8	—	Always 0 (off)
7	—	Always 1 (on)
6	Extension indicator	0 = No more size data
		1 = Another character of character
		size data follows this one
5	—	0 = Reserved for future use
4	G3 set size	0 = 94 characters
		1 = 96 characters
3	G2 set size	0 = 94 characters
		1 = 96 characters
2	G1 set size	0 = 94 characters
		1 = 96 characters
1	G0 set size	0 = 94 characters
		1 = 96 characters

For example, if ASCII is designated as G0, G1, and G3, and ISO Latin-1 supplemental graphic is designated as G2, then Scss is the ASCII uppercase D character. The letter D has a decimal value of 68, and a binary value of 01000100. The 7th bit is always on (1). The 4th, 2nd, and 1st bits are off (0), meaning the character sets in G3, G1, and G0 have 94 characters. The 3rd bit is on, meaning the character set in G2 (ISO Latin-1) has 96 characters.

Sdesig A string of intermediate and final characters that indicate the character sets designated as G0 through G3. The final characters are the same as those used in the SCS sequences (see 72).

For example, if ASCII is designated as G0 and G1, and Digital Supplemental Graphic is designated as G1 and G2, the *Sdesig* string would be reported as BB%5%5.

For example, a cursor information report might look like this:

$^{D}C_{S}1\$u1;1;1;@;@;@;0;2;@;BB%5\emptyset S_{T}$

1;1;1; Means that the cursor is at row 1, column 1, on the first page.

@;@;@; Means that no individual character attributes or selective erase attributes are set for writing, DECCOM is reset, and that there is no $^{SS}2$, $^{SS}3$, or autowrap pending.

0;2; Means that G0 is mapped into GL, and G2 is in GR.

@; Means all character sets have 94 characters.

BB%5Ø Means that ASCII is in G0 and G1, Digital Supplemental Graphic is in G2, and Digital Special Graphic is in G3.

Tab Stop Report (DECTABSR)

$^{D}C_{S}2\$uD...DS_{T}$

Reflection response to $^{C}S_{I}2\$w$ (the DECRQPSR sequence). This reports Reflection's current tab settings to the host. D...D is a data string indicating the column number location of each tab stop. Tab stops are separated by a forward slash (/).

For example, a tab stop report might look like this:

$^{D}C_{S}2\$u10/15/23/31S_{T}$

The column numbers for the tab stops are 10, 15, 23, and 31.

Restore Presentation State (DECRSPS)

DCS<n>$tD...DST

The host can send this sequence to restore Reflection to a previous state based on one of the two presentation state reports DECCIR and DECTABSR.

<n>	Presentation State Restored
0	An error occured
1	Restore the cursor state from DECCIR
2	Restore the tab state from DECTABSR

DECRSPS restores the information from only one report at a time, with the format of the data string *D...D* exactly the same as it was received.

Mode Settings

The host can request the current settings of any ANSI or Digital private modes using the DECRQM control string. In response, Reflection sends a DECRPM string, reporting which modes are currently set and reset.

The host can use the information in the report to save the current mode settings, and later restore Reflection's mode settings to their saved state (with an SM or RM sequence).

Mode Settings Request (DECRQM)

CSI<n>$p   Host request for ANSI mode settings, where *<n>* is an ANSI mode from the following table.

Mode	Mnemonic	<n>
Keyboard action	KAM	2
Control representation	CRM	3
Insert/replace	IRM	4
Send/receive	SRM	12
Linefeed/new line	LNM	20

CSI?<n>$p   Host request for Digital private mode setting, where <*n*> is a private
mode from the following table:

Mode	Mnemonic	<n>
Cursor keys	DECCKM	1
ANSI	DECANM	2
Column	DECCOLM	3
Scrolling	DECSCLM	4
Screen	DECSCNM	5
Origin	DECOM	6
Autowrap	DECAWM	7
Autorepeat	DECARM	8
Print form feed	DECPFF	18
Print extent	DECPEX	19
Text cursor enable	DECTCEM	25
NRC set	DECNRCM	42
Numeric keypad	DECNKM	66
Backarrow key	DECBKM	67
Keyboard usage	DECKBUM	68

Report Mode (DECRPM)

Reflection response to a request mode (DECRQM) function. This tells the host whether a certain mode is set or reset. As with the DECRQM function, there are two versions of DECRPM: one for reporting ANSI mode settings, and one for reporting Digital private mode settings.

^CS_I\<n1>;\<n2>$y   Reflection response to an ANSI mode request. *\<n1>* is the mode from the ANSI mode table on the previous page. *\<n2>* is one of the following mode states:

\<n2>	Mode State
0	Unknown
1	Set
2	Reset
3	Permanently set
4	Permanently reset

^CS_I?\<n1>;\<n2>$y   Reflection response to a Digital private mode request. *\<n1>* is the mode from the Digital private mode table on the previous page. *\<n2>* is the mode state; these are the same as for the ANSI mode states listed above.

Restoring Mode Settings

ANSI and Digital private modes are control functions that have only two settings: *set* and *reset*. Resetting Reflection affects many control functions, including some ANSI and Digital private modes. Separate set mode (SM) sequences set ANSI and Digital private modes, and separate reset mode (RM) sequences reset ANSI and Digital private modes. You cannot set and reset both ANSI and Digital private modes with the same SM or RM function.

Set Mode (SM)

^CS_I\<n>;...\<n>h   Set one or more ANSI modes, where *\<n>* is a mode from the ANSI table on 100.

^CS_I?\<n>;...\<n>h   Set one or more Digital private modes, where *\<n>* is a mode from Digital private table on 101.

Reset Mode (RM)

CSI<n>;...<n>l   Reset one or more ANSI modes, where *<n>* is a mode from the ANSI table on 100.

CSI?<n>;...<n>l   Reset one or more Digital private modes, where *<n>* is a mode from the Digital private table on 101.

Control Function Settings

With the DECRQSS function, the host can request from Reflection a variety of control function settings. The host can save this information, so if an application makes any temporary changes, the host can later restore Reflection to its previous state. Requests can be made for the following control function settings:

Control Function	Mnemonic	Intermediate and Final Character(s)
Select active display	DECSASD	$}
Select columns per page (VT340)	DECSCPP	$|
Set character attribute	DECSCA	"q
Set conformance level	DECSCL	"p
Set status line type	DECSSDT	$~
Set top and bottom margins	DECSTBM	r
Select graphic rendition	SGR	m

Control Function Settings Request (DECRQSS)

DCS$qD...DST   Host request for a selection or setting. *D...D* is the intermediate and final characters of the function. Information for only one function can be requested at once. For example, the following sequence asks about the DECSCL (select conformance level) function: DCS$q"pST.

Control Function Settings Response (DECRPSS)

DCS<n>$rD...DST   Reflection sends this sequence to the host in response to a DECRQSS sequence. If the value of ⟨n⟩ is Ø, then the host request is invalid. If ⟨n⟩ is *1*, the request is valid. The string *D...D* contains the current settings of valid control functions, and consists of all characters in the control function except CSI or ESC.

For example, if the host requests Reflection's conformance level (DECSCL; 23) as in the DECRQSS example above, the default settings for Reflection results in the following DECRPSS response:

DCS1$r63;2"pST

If the host requests a setting that Reflection does not recognize, Reflection responds with:

DCS0$rST

Saving and Restoring the Cursor State

With the DECSC function, a variety of Reflection's cursor state settings can be saved. The DECRC function restores the saved state. An application can use these functions to save the cursor state, make temporary changes, and later restore the state to its original settings. A separate cursor state is saved for the status line and the main display, whichever is active. Reflection, not the host, saves the cursor state information.

Save Cursor State (DECSC)

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 (DECOM)

- Selective erase attributes

- Any SS2 or SS3 functions sent

Restore Cursor State (DECRC)

E_SC8   Restore the cursor state saved with DECSC. If no previous DECSC was performed, DECRC does the following:

- Homes the cursor.

- Resets origin mode (DECOM).

- Turns off all character attributes.

- Maps the ASCII character set into GL, and the Digital Supplemental Graphic set into GR.

User-Preferred Supplemental Set Request

With the DECRQUPSS function, the host can determine which of the two user-preferred supplemental character sets is active: ISO Latin-1 or Digital Supplemental Graphic. Reflection responds with the DECAUPSS sequence (84).

Request User-Preferred Supplemental Set (DECRQUPSS)

C_SI&u   Request user-preferred supplemental character set. Reflection can respond with one of the following sequences:

D_CSØ!u%5_ST   Reflection response to the above request, indicating that the UPSS is Digital Supplemental Graphic.

D_CS1!uA_ST   Reflection response to the above request, indicating that the UPSS is ISO Latin-1 supplemental graphic.

Resetting and Testing

There are three control functions you can use to reset Reflection:

- Reset to initial state—hard terminal reset (RIS)

- Soft terminal reset (DECSTR)

- Tab clear (TBC)

Reset to Initial State (RIS)

^ES_{CC} Restore Reflection's terminal settings for the active connection only to their last saved values—this is also called a *hard reset*. Settings specific to Reflection, such as file transfer parameters, preferences, and printer options, are *not* reset. A hard reset is like reloading a Reflection settings file before any changes were saved.

In addition to recalling the last saved settings, a hard reset does the following:

- Clears the display and places the cursor in the upper-left corner.

- Sets the select graphic rendition (SGR) function to normal (27).

- Sets the selective erase attribute (DECSCA) to erasable (46).

- Clears the user-defined keys (see DECUDK on 52).

- Selects the default character sets: ASCII in GL and user-preferred supplemental set in GR (70).

- Cancels an executing macro or script file.

- Cancels a pending *Wait* method.

Soft Terminal Reset (DECSTR)

^CS_{I!p} Reset most of Reflection's default settings. This is available only with VT200 and higher emulation.

DECSTR is identical to clicking Reset on the cascading Connection menu. The following table shows the features reset by DECSTR and their mnemonic identifiers:

Feature	Mnemonic	Value After Reset
Autowrap	DECAWM	Cancel pending autowrap
Cursor keys	DECCKM	Normal

Feature	Mnemonic	Value After Reset
Insert/replace	IRM	Replace
Keyboard action	KAM	Unlocked
Numeric keypad	DECKPNM	Numeric characters
Origin	DECOM	Absolute
Text cursor enable	DECTCEM	Cursor enabled
Character sets	G0–G3, GL, GR	VT320 default settings
UPSS	DECAUPSS	Last saved value
Cursor state	DECSC	Home position
Active display	DECSASD	Main display
Select graphic rendition	SGR	Normal rendition
Selective erase	DECSCA	Normal attribute
Top/bottom margins	DECSTBM	Top/bottom of display
Use NRC characters	DECNRCM	No

Tab Clear (TBC)

^{CS}I3g   Clear all tab stops.

^{CS}IgorCsiØg   Clear only the tab stop at the current
                    position.

You can also set a tab stop at the current column with the escape sequence ^{ES}CH. You can also set tabs on the Tabs tab of the Terminal Setup dialog box. The TBC function only clears tab stops.

Testing Reflection

There are two control functions the host can send to test the terminal: one to invoke a confidence test and one to perform a screen alignment test.

Terminal Self-Test (DECTST)

^{CS}I4;<n>;...;<n>y

Invoke confidence test. The values of *<n>* determine which tests to perform. With Reflection, all values of *<n>* cause a communications line disconnect.

Screen Alignment Pattern (DECALN)

^{ES}C#8   Display a screen full of the letter E. On a terminal, this pattern can be
            used to align the screen display. With Reflection, it does not serve much
            purpose (unless you like the letter E).

VT52 Control Functions

The following functions control Reflection in VT52 mode. You can enter VT52 mode by setting this **Terminal type** option on the Terminal Type tab of the Terminal Setup dialog box to **VT52**, or with the DECANM control function below. In VT52 mode, ANSI-mode features and functions are ignored.

Enter VT52 Mode

$^{C}S_{I?2l}$ Set VT52 mode as the operating level; only VT52 control functions are recognized.

Exit VT52 Mode

$^{E}S_{C<}$ Exit VT52 mode and enter VT100 mode.

Cursor Positioning

$^{E}S_{CA}$ Move the cursor up.

$^{E}S_{CB}$ Move the cursor down.

$^{E}S_{CC}$ Move the cursor forward.

$^{E}S_{CD}$ Move the cursor backward.

$^{E}S_{CH}$ Move the cursor to the top of the display.

$^{E}S_{CY<r><c>}$ Move the cursor to row <r> and column <c>, where <r> and <c> are single ASCII characters whose decimal value is the desired row or column plus 31.

Erasing

$^{E}S_{CK}$ Erase from cursor to end of line.

$^{E}S_{CJ}$ Erase from cursor to end of screen.

Identification Request

$^{E}S_{CZ}$ Host issued identification request.

$^{E}S_{C/Z}$ Reflection's reply to host identification request, indicating the operating mode is VT52.

Keypad Mode

$^{E}S_{C=}$ Set keypad to application mode.

$^{E}S_{C>}$ Set keypad to normal mode (sends numeric sequences).

Printing Modes

$^{E}S_{C^}$ Turn on auto print mode.

$^{E}S_{C\_}$ Turn off auto print mode.

$^{E}S_{CW}$ Start controller mode.

$^{E}S_{CX}$ Stop controller mode.

$^{E}S_{CV}$ Print cursor line.

$^{E}S_{C]}$ Print screen.

Reverse Linefeed

$^{E}S_{CI}$ Reverse linefeed. The cursor moves up one row in the current column.

Reflection for ReGIS Graphics

ReGIS Graphics Support in Reflection

Note: This section only applies if you are using Reflection for ReGIS Graphics for Windows.

Reflection for ReGIS Graphics allows your PC to support Digital's Remote Graphics Instruction Set, or ReGIS, and the sixel features of Digital's VT340, VT330, VT241, and VT240 graphics terminals.

Reflection for ReGIS Graphics supports the following ReGIS features:

- Up to 16 colors

- Shading with selected patterns and polygon fill

- Rubberband cursors

- Rotated and italicized characters

- Mouse support

- A scaled image showing the complete ReGIS screen (800 ✥ 480 pixels) on the physical display

This section describes each ReGIS command in detail.

About ReGIS Graphics

For accurate ReGIS and sixel graphics emulation using Reflection for ReGIS Graphics for Windows, a 256-color display is recommended. In addition, the **Terminal ID** on the Emulation tab of the Terminal Setup dialog box should be set to one of the four graphics terminal types—host applications may need to determine the graphics capabilities of Reflection:

- The VT240 and VT330 are monochrome graphics terminals, providing up to four shades of gray at once.

- The VT241 and VT340 are color graphics terminals; the VT241 provides up to four different colors at once, while the VT340 provides up to 16 different colors.

The default **Terminal ID** for Reflection for ReGIS Graphics is **VT340**; make sure it is selected in the **Terminal ID** list (on the Emulation tab of the Terminal Setup dialog box). Further, you should also keep the **Terminal type** list on the Terminal Type tab set to its default value of **VT400-7**. Once the correct terminal type is set, simply run your host graphics applications as you normally would.

Sample Session

The following instructions show you how to use Reflection for ReGIS Graphics to display ReGIS graphics. You will create a short ReGIS program that places Reflection into ReGIS mode, then draws a box using ReGIS commands. The individual commands and their syntax are described in subsequent chapters in this chapter.

1. Start Reflection for ReGIS Graphics as you normally would.

2. On the Setup menu, click Terminal.

3. On the Emulation tab, clear the **Online** check box.

4. Click OK. Reflection is now in Local mode.

5. Press Alt+L to open the Reflection command line.

   In this sample session, you will use the Reflection command line to execute the ReGIS commands.

6. Type the following lines, pressing Enter after each one:

   ```
   Display "^[P3p"
   Display "P[100,100]"
   Display "V[+200] [,+100] [-200] [,-100]"
   Display "^[\\"
   ```

   The two characters ^[in the first line represent the ESc character that starts the sequence.

   The first command, EScP3p, puts Reflection into ReGIS mode at the start of a new command and displays the ReGIS command line. (See page 60 for a list of other sequences that enter ReGIS mode.) The final command instructs Reflection to exit ReGIS and return to text processing.

This is a very simple series of ReGIS commands that you could just as easily have entered directly at the ReGIS command line. You can write and debug more complicated ReGIS programs using the same method. Because the ReGIS command line does not allow you to edit your entries, though, it's easier to use the Reflection command line.

Mouse Support

The VT340 terminal supports locator devices with 2 to 4 buttons; a PC mouse can have 2 to 3 buttons. The following table shows what mouse clicks (and keystrokes used with mouse clicks) simulate the VT terminal mouse buttons.

Do this with your PC mouse...	To simulate this VT mouse button action
Click button 1 down	Click button 1 down
Click button 1 up	Click button 1 up
Click button 2 down	Click button 2 down
Click button 2 up	Click button 2 up
If you have a 3-button mouse...	
Click button 3 down	Click button 3 down
Click button 3 up	Click button 3 up
Otherwise...	
Press [Alt] and click button 1 down	Click button 3 down
Press [Alt] and click button 1 up	Click button 3 up
Press [Alt] and click button 2 down	Click button 4 down
Press [Alt] and click button 2 up	Click button 4 up

ReGIS Commands: Overview

Digital's Remote Graphics Instruction Set, or *ReGIS*, is a symbol system describing the parts of an image using standard geometric forms: lines, dots, arcs, circles, and curves. You can also create fill patterns and text. The graphics you create using ReGIS can be used for display or printing.

This chapter covers the conventions, syntax, and commands of the basic ReGIS commands. A brief explanation, syntax, and example are provided for each one.

Conventions

The following conventions apply to ReGIS commands as they are presented in this manual:

- When you enter commands, you can use upper or lowercase letters. Uppercase letters are used here for consistency.

- Angle brackets (< >) enclose a description of the type of information required. The brackets are *not* part of the syntax.

- The expression [X,Y] indicates that you can select an absolute or relative position on the screen. X (horizontal) and Y (vertical) are variables for a coordinate position; the brackets *are* part of the syntax.

- Spaces are included between some parts of ReGIS commands for readability only; they are never required.

- When values are given for character display and unit cell sizes, the values assume a full-size, unscaled ReGIS display similar to the VT340 terminal. Reflection automatically scales characters as needed to fit your actual display size.

Command Syntax

ReGIS command syntax is made up of relatively few elements. A complex image can be created by entering a straightforward, but rather long, series of commands. If you are creating a complex graphics image locally—that is, it is not being sent to you from the host—the most efficient way to do so is to create a Reflection macro (or Reflection Basic script) file with the ReGIS commands in it.

Command Key Letter

Each ReGIS command is identified by a single letter, its *command key letter*. The *vector* command, for example, is written simply as V. Selected options and arguments follow the command. For instance, the following sequence draws a line from the current position to a given pair of coordinates. The vector command is followed by its arguments:

```
V[300,400]
```

Until a new command key letter is given, all arguments apply to the current command. In the following example, the graphics cursor is positioned at [200,300], a vector is drawn from there to [400,100], and another vector is drawn from [400,100] to [200,50]. The last pair of coordinates uses the vector command, since it was the last command specified:

```
P[200,300] V[400,100] [200,50]
```

ReGIS Command Summary

The following is a brief summary of the command key letters. Each command is discussed in more detail beginning on page 123.

Command Key Letter	Command Name	Description
P	Position	Positions the graphics cursor without doing any drawing.
V	Vector	Draws straight lines.
C	Curve	Draws circles, arcs, and curves.
S	Screen	Controls elements, such as background intensity and erasing.
W	Write	Provides control over such elements as pen color and fill patterns.

Command Key Letter	Command Name	Description
F	Polygon Fill	Fills in a closed figure.
T	Text	Controls character display.
L	Load	Defines and loads alternate character sets displayed with the text command.
R	Report	Provides information about the current status of ReGIS operations and is used to enter graphics input mode.
@	Macrograph	Stores and recalls ReGIS command strings.

Parentheses

Parentheses enclose options and suboptions of ReGIS commands. The following shows the screen command with the erase option:

```
S(E)
```

In the next example the cursor is positioned, and a circle with a radius of 150 pixels is drawn:

```
P[200,200] C(W(P2)) [+150]
```

The option in this case is the write command, and its suboption is the selection of a pattern for the line—P2—that consists of dashes. The pattern applies only to circles drawn subsequently by the curve command. See page 146 for a listing of available patterns.

Brackets

Brackets are used to enclose the following types of numeric values:

- Coordinate positions, both absolute and relative

- Height and width values (for text commands only)

By default, the ReGIS screen has the following coordinate system:

[0,0]

[799,0]

[0,479]

[799,479]

When specifying a coordinate position, follow these rules:

- Specify the X (horizontal) coordinate first, then the Y (vertical) coordinate.

  For example, [100,200] indicates an X coordinate of 100, and a Y coordinate of 200.

- If the Y coordinate for a new position is unchanged from its current value, you can omit the Y coordinate and just specify the X coordinate.

  For example, [200] indicates a new X coordinate, and an unchanged Y coordinate.

- If the X coordinate for a new position is unchanged from its current value, you can just use a comma and then specify the new Y coordinate.

  For example, [,50] indicates an unchanged X coordinate, and a new Y coordinate.

- A coordinate value beginning with a + or − sign indicates a value relative to the previous one.

  For example, [+100] indicates a new X coordinate 100 units to the right of the previous X coordinate. The Y coordinate is unchanged.

The following example indicates a position that is at the intersection of 200 on the X axis and 400 on the Y axis:

    [200,400]

The next position is relative to the current one:

    [+50,-25]

In the following example, X is relative and Y is absolute:

    [+75,200]

Quotes

Quotes enclose the following ReGIS arguments:

- Text to display with text commands

- A character to use for polygon filling

- A character set name to select with the load command

- A character to use as an identifier for an argument to a load command

Either single quotes (') or double quotes (") can be used to define the beginning and end of the quoted argument; parentheses are not allowed.

Control Characters

ReGIS recognizes four control characters within a quoted string:

Character	Description	Keystroke
C_R	Carriage return	Ctrl + M
L_F	Linefeed	Ctrl + J
B_S	Backspace	Ctrl + H
H_T	Horizontal tab	Ctrl + I

If ReGIS encounters any control characters outside a quoted string, they are shown on the command line and ignored, with the exception of linefeed. When the linefeed control character is entered (the keystroke Ctrl + J), the ReGIS command line is erased and you are positioned at the beginning of the line.

Resynchronization

A semicolon (;) in a command string instructs ReGIS to abort the current command and begin a new command. There are three instances in which the semicolon is not recognized as a resynchronization command:

- Within a quoted text string

- When it is part of a macrograph definition

- When it terminates a load command

Pixel Vector Directions

Many ReGIS positioning and drawing commands can use the pixel vector (PV) system, for moving and drawing one pixel at a time. This system uses eight directions at 45-degree intervals. Each direction is assigned a number:
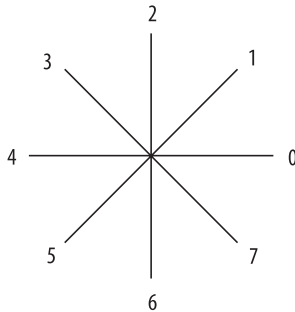
One PV number moves one pixel at a time in the direction specified. For example, the following command positions the cursor, then draws a line diagonally down to the right, then diagonally to the left. Each of the two lines is 10 pixels long:

```
P[100,100] V7777777775555555555
```

Moving and drawing one pixel at a time can be tedious. An option of the write command lets you give pixel vector points a multiplication factor, which specifies the number of times to repeat each point.

The following example sets a PV multiplier of 100 (the option `M100` to the write command), then draws a box by listing the four numbers that represent the PV directions for down, left, up, and right:

```
W(M100) V6420
```

Each side of the box is 100 pixels long.

ReGIS Command Descriptions

This chapter describes the following commands available when using ReGIS graphics: Position, Vector, Curve, Screen, Write, Polygon Fill, Text, Load, Report, and Macrograph.

Position Command

The position command moves the graphics cursor to an absolute or relative location on the screen.

Cursor Positioning with [X,Y]

Format: `P[X,Y]`

The cursor is positioned at the given X and Y coordinates. If the first value is omitted, its current value is assumed. In the following example, the second set of coordinates is assumed to be [100,300], even though the value 100 is not entered:

```
P[100,200] [,300]
```
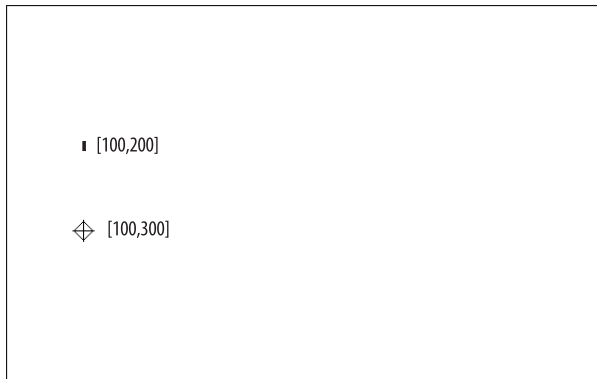
In the next example, the second pair of coordinates is assumed to be [400,300] even though only the X coordinate is given. The following two expressions are equivalent:

```
P[100,300] [400,]
P[100,300] [400]
```

Relative values are entered using a plus or minus sign. The following command positions the cursor at [100,100] and then draws a vector 200 pixels down:

```
P[100,100] V[,+200]
```

When specifying X and Y coordinates, the X coordinate must always be first, and the Y coordinate must always be preceded by a comma. The default position is [0,0].

Cursor Positioning with Pixel Vectors

Format: `P<pv>`

The cursor can be positioned with pixel vector values. The pixel vector directions are shown on page 122.

In the following example, the cursor is moved one pixel at a time, 10 pixels to the right and 10 down:

```
P00000000006666666666
```

Select Graphics Page

Format: `P(P<0 or 1>)`

the *P* option lets you move the graphics cursor from one page to another: *0* moves the cursor to the first page and *1* moves it to the second. Both the input cursor and the output cursor move to the corresponding position on the selected page.

Entering and exiting ReGIS does not change the currently active page. When you enter ReGIS, the currently displayed graphics page is the first graphics page. The active graphics page is the same page as the last time you exited ReGIS. If you did not use ReGIS since resetting Reflection, the active graphics page is the first graphics page.

PV Multiplication

Format: `P(W(M<n>)) <pv>`

Using a temporary write control option, you can specify a multiplier for moving the cursor using the pixel vector system. The default is a multiplication factor of 1.

The value of ⟨*n*⟩ is the PV multiplier. For example, the next command uses a multiplier of 50 (`M50`) to move the cursor 50 coordinates for each PV value:

```
P(W(M50))6420
```

Begin Bounded Position Stack

Format: `P(B) <options> (E)`

With the *B* option—begin bounded position stack—the current position of the graphics cursor is saved. When the *E* option—end of position stack—is encountered, the cursor is returned to the saved position. Position, vector, curve, and other commands can be specified between the options that begin and end the bounded position stack. For each position stack begun with the *B* option, there must be a corresponding *E* option to end that stack.

In the following example, the cursor begins at [300,100]. This becomes the bounded position when the *B* option is specified. The cursor is then moved to two other positions on the screen. When the *E* is encountered, the cursor returns to [300,100]:

```
P[300,100] (B) [400,200] [300,150] (E)
```

Bounded positions can be nested up to 16 levels. The following example shows the command nested one level:

```
P[300,100] V(B) [+100] (B) [+100,+50]
[-50,+100] (E) [+100,-100] [-50,+50] (E)
```

Start Unbounded Position Stack

Format: `P(S) <options> (E)`

The S option puts a dummy position on the stack. When the *E* option is encountered, the dummy stack position is discarded, and the current position remains unchanged. The purpose for this option becomes clear in the discussion of the curve command on page 129.

Null Position Argument

Format: [ ]

This argument is equivalent to moving the cursor to a relative position of [+0,+0]. Its use is explained more fully in the discussion of the curve command's open curve option on page 132.

Vector Command

The vector command draws a line between the cursor position and the absolute or relative position that you specify.

Draw Dot

Format: V[ ]

By specifying the vector command with a null position argument, you can draw a single dot (pixel) at the current cursor position.

The following commands position the cursor at [300,200] and draw a single dot; the cursor does not move.

    P[300,200] V[]

Draw Line

Format: V[X,Y]

Specifying X and Y coordinates with the vector command, you can draw a straight line from the current position to an absolute or relative position.

The following commands use absolute coordinates to draw a line from [200,100] to [250,300] and another one from [250,300] to [100,350]:

    P[200,100] V[250,300] [100,350]

In the second pair of coordinates after the vector command, the V command is assumed and does not need to be entered.

In the following example, a series of relative vector commands draws a pentagon after positioning the cursor at the absolute coordinates [200,100]:

    P[200,100] V[+50,+50] [,+50] [-100] [,-50] [+50,-50]

Draw Lines Using Pixel Vectors

Format: V<pv>

Using pixel vector numbers, you can draw a vector one pixel at a time from the current cursor position in any of the eight positions of the pixel vector system.

The following example draws a line to the right that is ten pixels long:

```
V0000000000
```

With the write option given as an option to the vector command, a multiplication factor for pixel vectors can be designated, just as in the position command.

The following example draws a square, each side of which is 200 pixels long:

```
P[300,150] V(W(M200))6420
```
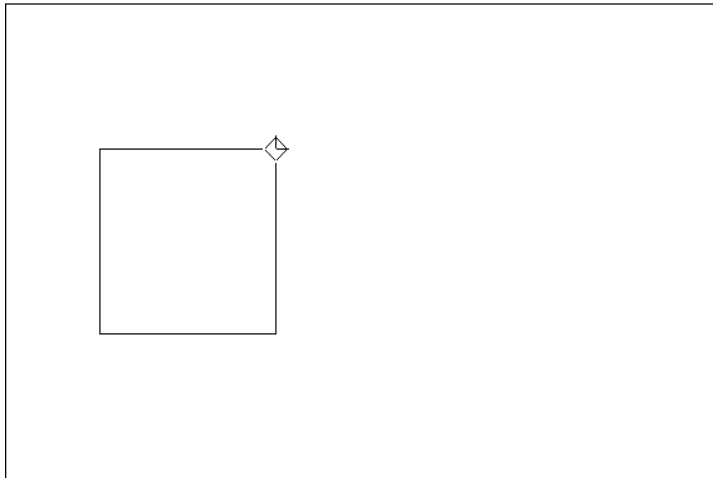
Begin Bounded Position Stack

Format: `V(B) <options> (E)`

When the *B* option (begin bounded position stack) is used, the current position of the graphics cursor is saved. When the *E* option (end stack) is entered, the cursor returns to the saved position. Other commands can be specified between the options that begin and end the bounded position stack.

The following draws a triangle:

```
P[100,100] V(B) [200,200] [50,250] (E)
```

The cursor begins at [100,100], and this position is made the start of a bounded position stack with the *B* option. A vector is then drawn to [200,200], then to [50,250], and the third side is drawn when *E* is entered.

Bounded position stacks can be nested up to 16 levels.

Start Unbounded Position Stack

Format: `V(S) <options> (E)`

The *S* option puts a dummy position on the stack. When the *E* option is encountered, the position stack is discarded, no line is drawn, and the current position of the cursor does not change.

The unbounded stack option serves little use here; it is primarily for consistency with the stack options of the curve command discussed below.

Temporary Write Control

Format: `V(W(<suboptions>))`

This option gives you temporary control over the write command values used by the vector command; it ends when a new command is given. See page 145 for details about the write command.

The following example temporarily defines P4 as the line pattern and then draws a line using that pattern to a point 100 pixels down. When the second *V* command is given, the temporary write control is canceled, and the new line is drawn with the default pattern:

```
V(W(P4)) [,+100] V[+50,+100]
```

Curve Command

Curve commands can draw circles, arcs, and other types of curves. Temporary write controls are also available, so that curves can be drawn in a given pattern.

Center of Circle at Current Position

Format: `C[X,Y]`

The coordinates [X,Y] define a point on the circumference of the circle. The current cursor position is the circle's center.

The following example positions the cursor at [400,250] and then draws a circle whose circumference passes through [270,250]. The cursor position is [400,250], the center of the circle:

```
P[400,250] C[270]
```
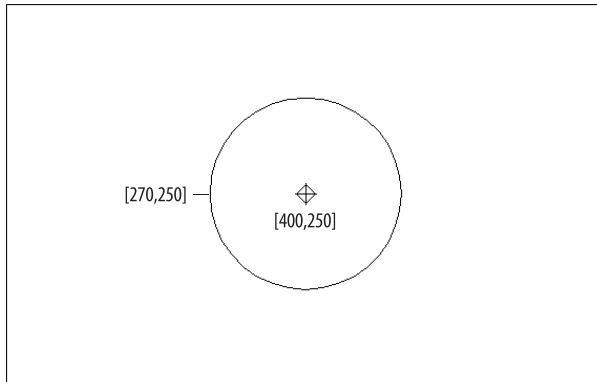

You can also use relative coordinates to specify the point to pass through. In effect, this specifies a radius for the circle. The following commands position the cursor at [200,300] and draw a circle that passes through a point 50 pixels to the right of the cursor position; that is, with a radius of 50 pixels:

```
P[200,300] C[+50]
```

Center of Circle at Specified Position

Format: `C(C) [X,Y]`

The coordinates [X,Y] define the center of the circle while the current position defines a point on the circumference. This option remains in effect until a new command is given.

The following commands position the cursor at [250,150] and draw a circle whose center is at [300,250] and whose circumference passes through [250,150]:

```
P[250,150] C(C)[300,250]
```
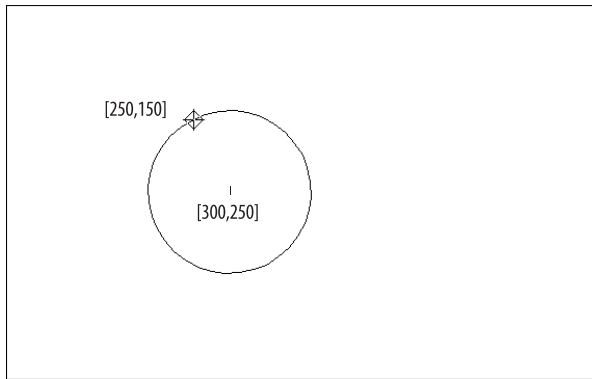

Arc with Center at Current Position

Format: `C(A<degrees>) [X,Y]`

Using a command similar to the circle drawing command, you can draw an arc—a section of circle—with its center at the current cursor position and with a point on the circumference specified.

The [X,Y] defines the starting point for the arc, in either absolute or relative coordinates. A is the arc option, and *<degrees>* determines the size and direction of the arc. If *<degrees>* is preceded by a minus sign, the direction of the arc is clockwise. If preceded by either a plus sign or no sign, the direction is counterclockwise. ReGIS draws arcs in 1 degree intervals, rounded to the closest integer degree.

The next example positions the cursor at [300,200] and draws a half circle counterclockwise starting 150 pixels to the right of the cursor position:

```
P[300,200] C(A180) [+150]
```

Arc with Center at Specified Position

Format: `C(A<degrees>C) [X,Y]`

This option lets you draw an arc that begins at the current position, and whose center is specified by [X,Y]. Using this option, you can link one arc to the next.

The following commands position the cursor at [400,200] and draw an arc counterclockwise 100 degrees. The center of the arc is 150 pixels to the left of the current cursor position:

```
P[400,200] C(A100C) [-150]
```
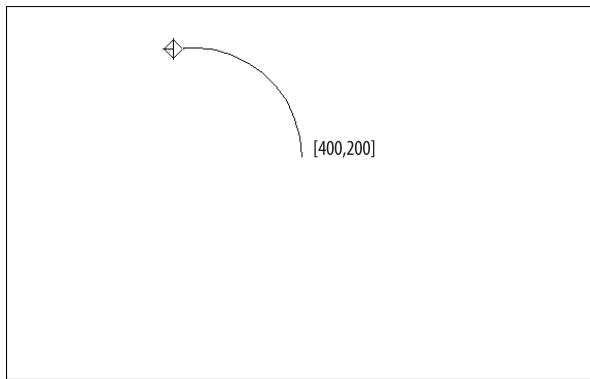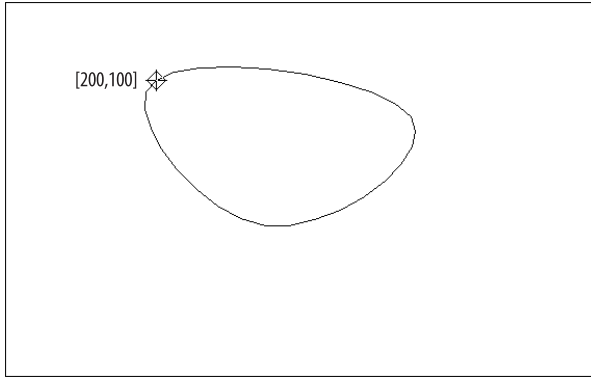

Closed Curve Sequence

Format: `C(B) <X,Y coordinates> (E)`

This command defines a closed curve based on a starting point, a minimum of two coordinates, and an endpoint; the curve is interpolated from the given coordinates. The B option indicates the start of the curve (similar to the option that starts a bounded position stack).

When the sequence is ended with *E*, the cursor moves to the beginning position, drawing a line. Unlike the bounded position stack *B* and *E* options, which allow nesting, closed curve sequences can have only one *B* and one *E*.

The X and Y coordinates can be absolute, relative, or a combination of the two. It may also include pixel vector values. Relative values are based on the last specified cursor position.

The following example draws an irregular, closed curve. Only two coordinates beyond the starting position are specified, and no lines are actually drawn until the *E* option ends the curve. This is because the curve drawing algorithm needs at least four positions to draw its curve. (The fourth point, or end point, is the same as the starting point.)

```
P[200,100] C(B) [350,+200] [+200,-150] (E)
```


## Open Curve Sequence

Format: `C(S) <X,Y coordinates> (E)`

This option lets you define an open curve based on a starting point, a minimum of three coordinates, and an endpoint. The *S* option indicates the start of the open curve (similar to the unbounded position stack option). When the sequence is ended with *E*, the cursor remains in the last position that you specified.

Unlike the unbounded position stack option, which allows nesting, an open curve sequence can have only one *S* and one *E*.

To draw the curve completely, you need to include a null position argument, [], at the beginning and end; otherwise, ReGIS only draws from the first position to the next-to-last one.

The following two examples illustrate this point: the first is drawn without null position arguments and the second is the same curve, further down on the screen, with null positions.

```
P[50,100]
C(S)
[125,175] [200,100] [275,175] [350,100]
[425,175] [500,100]
(E) P[50,300]
C(S) []
[125,375] [200,300] [275,375] [350,300]
[425,375] [500,300]
[] (E)
```
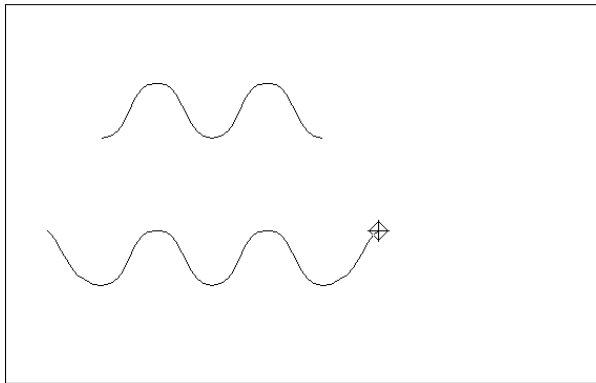

Temporary Write Control

Format: `C(W(<suboptions>))`

When you draw curves, any write control options currently active remain active. With temporary write control options, you can temporarily change the write controls for the curve you are drawing. The temporary controls remain in effect until you specify another temporary write control, or until a new command key letter is given. Any of the options discussed with the write command on page 145 can be used.

The following example draws an open curve similar to the one in the preceding figure, only now the P2 line pattern is used (for a list of patterns, see page 146):

```
P[50,300]
C(W(P2)) (S) []
[125,375] [200,300] [275,375]
[350,300] [425,375] [500,300] []
(E)
```

Screen Command

Using the screen command S, you can control attributes of the entire screen, including the background intensity, screen erasing, and the shape of the graphics cursor.

Display Addressing

Format: `S(A[x_1,y_1] [x_2,y_2])`

Normally, the screen has default coordinates of [0,0] for the upper-left corner and [799,479] for the lower-right corner. With the display address option, you can define the coordinates within which ReGIS images appear. If you create applications that are run on terminals other than the VT300 series, defining display coordinates can help make ReGIS images more portable. ReGIS automatically scales the images to fit correctly in the defined coordinates, so that squares appear square and angles appear at the proper angle.

The first set of coordinates is for the upper-left corner of the screen, and the second set is for the lower right corner.

Scrolling

Format: `S[X,Y]`   (using coordinates)

   `S<PV value>`   (using pixel vectors)

With a scroll argument to the screen command, you can offset data on the display, but leave the coordinate system unchanged; the image moves relative to the screen origin [0,0]. The scroll argument can use absolute or relative coordinate values, a combination of the two, or pixel vector (PV) values.

If you use PV values, the current PV multiplier is used. You can also specify a temporary PV multiplier, using a temporary write control as an option to the screen command:

Format: `S(W(M<multiplier>))<PV value>`

The temporary write control stays in effect until a new command key letter is used, or another temporary write control is given.

Any data that is scrolled outside of the display boundaries is lost; the data cannot be recovered by reversing the scrolling.

Print Screen

This command option lets you send a hard copy of your display to a PC printer or the host; the destination depends on the setting of the **Destination** list in the **Sixel** group box on the Graphics tab of the Terminal Setup dialog box. The destination can also be set using the MC function; see page 67.

An image sent to the host is sent as a series of sixels, which can be collected in a host file or sent to another graphics terminal.

There are three formats for the print command:

- Print the entire screen.

    Format: `S(H)`

- Print a rectangular portion of the screen, from the cursor position to a given set of coordinates.

    Format: `S(H[X,Y])`

- Print a rectangular portion of the screen, from one set of coordinates to another.

    Format: `S(H[x_1,y_1] [x_2,y_2])`

You can add a print offset to this command so the image is printed X number of pixels from the left printing margin and Y number of pixels from the top printing margin. The default offset is [50,0]. If you define a new offset, the new value remains in effect until redefined. Define an offset as follows:

Format: `S(H(P[X,Y]))`

You can also combine the print offset with the print command:

Format: S(H(P[X,Y]) [x$_1$,y$_1$] [x$_2$,y$_2$])

Output Mapping Controls

The VT240, VT241, and VT330 use a 2-bit code to represent each screen pixel, providing up to 4 different shades of gray (or colors in the case of the VT241). The VT340 uses a 4-bit code to represent each pixel, providing up to 16 different colors.

Each color is maintained in an *output map location*, a location in memory that stores a set of values that define the color. The VT240, VT241, and VT330, each with a 2-plane bitmap, have 4 output map locations, while the VT340, with a 4-plane bitmap, has 16 output map locations.

Using output mapping controls, you can change the colors stored in specific output map locations, letting you change the color of a ReGIS image without redrawing it (this is only possible when you run Windows in 256-color mode).

The output map locations for ReGIS graphics are linked to the **Text** and **Background** palettes on the Colors tab of the Display Setup dialog box; you can use this dialog box to manually change the colors in the output map. Color palette squares correspond to map locations 0 in the upper-left corner of the palette to 15 in the lower-right corner; locations are counted from left to right, and from top to bottom.

When an application changes ReGIS colors (using the mapping control options explained here), the new colors are shown on the Colors tab of the Display Setup dialog box. If the application does not restore the original colors after changing them, the color of text attributes that use those map locations will be changed as well.

Specifying Colors

Each color in the output map can be specified in one of two ways:

- Hue, lightness, and saturation (HLS) values.

  The hue can range from 0 to 360 degrees, lightness from 0 to 100 percent, and saturation from 0 to 100 percent.

- RGB color abbreviations.

  Eight single-letter abbreviations represent eight different colors. Some colors are secondary colors made by mixing the primary colors red, green, and blue.

Letter	Color	Letter	Color
D	Black (dark)	C	Cyan
R	Red	Y	Yellow
G	Green	M	Magenta
B	Blue	W	White

Monochrome Graphics

The L, or *lightness*, parameter of an HLS value defines a color's monochrome shade. For the VT240 and VT330 monochrome terminals, only the lightness parameter has meaning; saturation is always *0* for gray shades, and hue has no effect. If a full HLS specification is given (as explained in the next section), the hue and saturation are ignored. If RGB abbreviations are used with a monochrome terminal type, the lightness is set to *50* for all color abbreviations except D (which has a lightness of *0*) and W (which has a lightness of *99*).

When the terminal type is VT240 or VT330, the following command selects a monochrome shade for an output map location:

Format: `S(M<0 to 3>(L<value>))`

The appropriate color palette square on the Colors tab of the Display Setup dialog box is updated to show the new monochrome shade.

This command can also be used when the terminal type is VT241 or VT340; however, the color stored at the selected output map location is replaced by the specified monochrome shade (see below for more information).

The following table lists the monochrome output map locations for the VT240 and VT330 terminals and their default HLS and equivalent RGB values.

VT240 and VT330 Default Monochrome Mappings

Map Location	Shade	HLS Values			Equivalent RGB Values		
		H	L	S	R	G	B
0	Black	0	0	0	0	0	0
1	Dark gray	0	33	0	33	33	33
2	Light gray	0	66	0	66	66	66
3	White	0	100	0	100	100	100

Color Graphics

When Reflection is configured for VT241 graphics emulation, there are four output map locations. For VT340 graphics, there are 16 output map locations. Each location stores a monochrome shade and a color, and the two can be changed together or separately.

Changing Monochrome Shades

Format: `S(M<0 to 3>(L<value>))`   (VT241)

              `S(M<0 to 15>(L<value>))`   (VT340)

As mentioned in "Monochrome Graphics" on the previous page, specifying a lightness value alone changes the monochrome shade of an output map location.

To change the lightness of multiple output map locations at once, you can specify multiple output maps and lightness values in the same command. For example, the following changes the monochrome shades stored in output map locations 1, 2, and 4:

```
S (M1(L35)2(L45)4(L79))
```

Changing Color Values

Format: `S(M<0 to 3>(H<hue>L<lightness>S<saturation>))`   (VT241)

       `S(M<0 to 15>(H<hue>L<lightness>S<saturation>))`   (VT340)

Format: `S(M<0 to 3>(<RGB letter>))`   (VT241)

       `S(M<0 to 15>(<RGB letter>))`   (VT340)

You can change the color in an output map location by specifying a new color using either an HLS or RGB value. You can use either of the above two commands to select a new color value for an output map location. HLS values and RGB values should not be combined in one command. When you change colors, the appropriate color palette square on the Colors tab of the Display Setup dialog box is updated to show the change.

In the following example, the color in output map location 0 is changed to blue, using an RGB value:

```
S(M0(B))
```

To change the colors of multiple output map locations at once, you can specify multiple output map locations and color values in the same command. For example, the following command changes the colors in output maps 2 and 7, using HLS values:

```
S(M2(H300 L75 S60) 7(H180 L50 S100))
```

Remember that the VT241 and VT340 store both a monochrome and a color value for each output map location. This means that when you change a color using one of the commands above, the monochrome shade stored at that map location is changed to a shade of gray corresponding to the lightness of the color; if a monochrome value was already defined for that map location, it is replaced by the new monochrome shade.

For example, the following commands define a monochrome value for output map location 12, then define a color for the same location using HLS values:

```
S(M12(L33))
S(M12(H120 L46 S71))
```

The first command defines a shade of gray with a lightness value of 33. The second command, which defines red on the VT340, would specify a shade of gray with a lightness of 46 on a monochrome terminal; the value of 33 is overwritten.

You can change a color in the output map while retaining the monochrome value stored in the same location by using the following command:

Format: `S(M<0 to 3>(A <HLS or RGB values>))` (VT241)

`S(M<0 to 15>(A <HLS or RGB values>))` (VT340)

The A suboption specifies that only the *color* at the specified map location should be changed; the monochrome value should remain as it is.

Using the previous example, you could keep the monochrome shade at a lightness value of 33, and change only the color value to red with the A suboption. The two commands would then look like this:

```
S(M12(L33))
S(M12(AH120 L46 S71))
```

The A suboption can be useful when programming ReGIS graphics for different types of terminals, such as the VT330 and VT340. On the VT330, you would want the exact lightness value you specify, rather than a lightness value corresponding to the lightness of a color. On the VT340, however, you would want to show the actual color. Using the A suboption solves this problem.

The default color maps for the VT241 and VT340 terminals are shown in the tables below.

VT241 Default Color Map

Map Location	Shade	HLS Values			Equivalent RGB Values		
		H	L	S	R	G	B
0	Black	0	0	0	0	0	0
1	Blue	0	50	60	20	20	80
2	Red	120	46	71	78	14	14
3	Green	240	50	60	20	80	20

VT 340 Default Color Map

		HLS Values			Equivalent RGB Values		
Map Location	Shade	H	L	S	R	G	B
0	Black	0	0	0	0	0	0
1	Blue	0	50	60	20	20	80
2	Red	120	46	71	78	14	14
3	Green	240	50	60	20	80	20
4	Magenta	60	50	60	80	20	80
5	Cyan	300	50	60	20	80	80
6	Yellow	180	50	60	80	80	20
7	Gray 50%	0	46	0	46	46	46
8	Gray 25%	0	26	0	26	26	26
9	Blue	0	46	28	34	34	58
10	Red	120	43	37	58	28	28
11	Green	240	46	28	34	58	34
12	Magenta	60	46	28	58	34	58
13	Cyan	300	46	28	34	58	58
14	Yellow	180	46	28	58	58	34
15	Gray 75%	0	80	0	80	80	80

Background Intensity

Format: `S(I<n>)`
`S(I(<RGB>))`
`S(I(H<n>L<n>S<n>))`

To select a shade or color for the screen background, you can either give an output map location or provide an RGB or HLS value. The three formats are shown above.

With the second and third methods, the output map location that contains the color closest to the one you specify is selected. For example, if Reflection is configured as a VT241, which has four colors, entering `S(I(M))` gives you the color closest to magenta; on the VT241, this gives you red (unless the output map was changed from its defaults).

Selecting a background intensity does not change the color stored at the output map location.

Time Delay

Format: `S(T<0 to 32767>)`

You can delay the execution of a ReGIS command by specifying a time delay. The number you give is measured in ticks: 60 ticks equal 1 second. The maximum time you can specify as a delay is 32,767 ticks, which is equal to about 9.1 minutes.

Screen Erase

Format: `S(E)`

You can erase the entire screen to the background screen color.

This command does not change the color stored in any output map locations, and it does not move the cursor. You can combine the screen erase option with a command to change the background intensity. The comma in the command is required:

Format: `S(I<value>,E)`

Display Graphics Page

Format: `S(P<0 or 1>)`

the S option selects which of two position graphics pages Reflection displays: 0 displays the first page and 1 displays the second. Exiting ReGIS displays graphics page 0.

Graphics Cursor

ReGIS uses two types of graphics cursors: a graphics *output* cursor and a graphics *input* cursor.

- The output cursor appears when ReGIS is waiting for commands from the host (or from the ReGIS command line). You control whether the output cursor is displayed and what shape it should have with the ReGIS commands described below.

- The input cursor appears when ReGIS is waiting for graphics input; for example, when the host requests a cursor position report. You can move the input cursor with the mouse, and change its shape with a screen command option.

Output Cursor Control

Format: `S(C<0 or 1>)`

Use a screen command option to control whether or not the graphics output cursor is displayed.

A *0* turns the output cursor off; a *1* turns it on.

The style of the output cursor is determined by a suboption:

Format: `S(C(H<0 to 2>))`

If the number is omitted, or is 0 or 1, the cursor style is a diamond (the default). If the number is 2, the cursor appears as a crosshair.

Input Cursor Control

Format: `S(C(I<0 to 4>))`   (predefined styles)

`S(C(I[X,Y]"<characters>"))`   (user-defined input cursor)

The graphics input cursor is typically used to determine the current cursor position with the report command (see page 180 for details about the report command). The style of the input cursor can be selected from five predefined styles, or defined by the user.

The table below lists the predefined input cursor styles:

Input Cursor Styles

Number	Input Cursor Style
0	Crosshair (default)
1	Diamond
2	Crosshair
3	Rubberband line
4	Rubberband rectangle

A user-defined cursor is composed of two characters: one is displayed in the foreground shade or color, and the other is displayed in the background shade or color. Monochrome graphics emulation uses output map location 2 for the foreground and 1 for the background. The color graphics emulation uses location 14 for the foreground and 1 for the background.

The size of the input cursor is limited to 16 ✷ 24 pixels. The following example defines an input cursor that consists of a vertical bar inside of an uppercase O:

```
S(C(I[+4,+10]"O|"))
```

The parameter [+4,+10] selects the coordinate for the origin of the cursor.

The two characters you use to compose the cursor can come from either the built-in character sets or from characters you design and load. See the description of the load command on page 176 for more information. You can also use the text command (described on page 163) to change the size of the character.

Write Command

The write command gives you control over patterns, the foreground color, four types of writing, plane selection, and shading. When used alone, a write command affects all commands that follow it, until another write command is encountered. If used as a temporary write control, the W is enclosed in parentheses and it becomes a temporary option of the command key letter. For example, in the following sequence, the write command becomes an option of the vector command and has temporary effect:

```
V[+100](W(E))[+100]
```

The control remains in effect until another command is issued.

When you use the write command to control how vectors, curves, and shaded areas appear, you typically set three parameters first:

1. Select a pattern using the *P* option.

    The pattern can include a multiplication factor, specified with the *M* suboption.

2. Select a foreground color using the *I* option.

3. Select a write mode: overlay, replace, complement, or erase.

When an image is drawn, these and other parameters combine to control its appearance.

Select Standard Pattern

Format: `W(P<0 to 9>)`

The *P* option selects one of ten standard patterns for drawing; the patterns are shown in the figure below. The default pattern is *P1*.

```
W(P0)

W(P1)    ———————————————————————————

W(P2)    – – – – – – – – – – – – – – – –

W(P3)    – · – · – · – · – · – · – · – · – · – ·

W(P4)    ·······································

W(P5)    –··–·–··–·–··–·–··–·–··–·–··–·–··–

W(P6)    · · · · · · · · · · · · · · · · · · · · · · · · · · · · · ·

W(P7)    · · · · · ·· · · · · ·· · · · · ·· · · ·· ··

W(P8)    - - - - - · - - - - - · - - - - - · - - - - -

W(P9)    · ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ··
```

The following example selects line pattern 5, and draws a vector 100 pixels long:

```
W(P5)
V[+100]
```

Select Binary Pattern

Format: `W(P<binary pattern>)`

This option lets you define your own 8-bit writing pattern.

The binary pattern is made up of 2 to 8 bits that are either on or off. If you specify fewer than 8 bits, ReGIS repeats as much of the pattern as possible in what remains of the 8-bit segment. If more than 8 bits are specified, only the last 8 make up the defined pattern. For example, both of the following two binary pattern selections produce the same dotted pattern (identical to the standard pattern 4):

```
W(P01)
W(P01010101)
```

Both standard and binary patterns can be multiplied, letting you create writing patterns longer than 8 bits. Using a multiplication factor as a suboption to the P option, you specify the number of times each bit in the 8-bit pattern should repeat. The default multiplication factor is 2:

Format: `W(P<pattern>(M<1 to 16>))`

For example, when the standard pattern 4 is multiplied by a factor of 4, the bit pattern changes from `10101010` to `11110000111100001111000011110000`. The command that creates this pattern looks like this:

```
W(P4(M4))
```

In the following example, the binary pattern specified by `W(P01)` is multiplied by five, and a vector 100 pixels long is drawn:

```
W(P01(M5))
V[+100]
```

Pixel Vector Multiplication

Format: `W(M<multiplication factor>)`

The M option to the W command lets you assign a multiplication factor to the magnitude of the pixel vectors used for writing.

With the default multiplication factor of *1*, the following command draws four lines that are each two pixels long. The numbers given are the pixel vector directions (see page 122).

```
V00664422
```

The next example includes a multiplication factor of 100; this time the four lines are each 200 pixels long.

```
W(M100) V00664422
```

Foreground Intensity

This option determines the shade or color of the foreground image. (A similar option of the screen command determines the background color; see page 136.) On the VT240, VT241, and VT330, up to 4 shades (or colors for the VT241) are available. On the VT340, up to 16 shades or colors are available.

There are three ways to define the shade or color:

- An output map location can be given, meaning that whatever color is currently stored in that location is used. Output mapping is explained in the discussion of the screen command on page 136.

  Format: W(I<0 to 3>)   (VT240, VT241, or VT330)
            W(I<0 to 15>)   (VT340)

- An RGB abbreviation can be used to select an output map location. The output map location containing the color closest to the one you specify is selected.

  Format: W(I(<RGB>))

  The following RGB colors are available:

Letter	Color	Letter	Color
D	Black (dark)	C	Cyan
R	Red	Y	Yellow
G	Green	M	Magenta
B	Blue	W	White

- A hue, lightness, and saturation (HLS) value for the color can be given. The HLS value selects the output map location containing the color closest to the one you specify.

  Format: W(I(H<n>L<n>S<n>))

  Hue values can range from 0 to 360; blue has a hue value of 0, red a value of 120, and green a value of 240. Lightness and saturation values can range from 0 to 100.

The selected foreground intensity is used only for the arguments that follow the option. This way, different parts of the image can have different colors.

Writing Styles

The write command has four writing styles: overlay (the default), replace, complement, and erase. Each one affects the way ReGIS draws images into graphics memory.

Writing Style	Command	Effect
Overlay	W(V)	Foreground where pattern is on; no change where pattern is off.
Replace	W(R)	Background where pattern is off; foreground where pattern is on.
Complement	W(C)	Where pattern is on, the complementary pixel values are used.
Erase	W(E)	If negative pattern control is off, writing is done in the background color, regardless of the pattern. If negative pattern control is on, writing is done in the foreground color.

Only the bitmap planes that are enabled, as defined by the plane selection option described on page 158, are affected by each writing style. The following sections explain more about each style.

Overlay Writing

Format: W(V)

When overlay writing is selected, pixels are drawn with the foreground color where bits in the writing pattern are on (1). Bits in the pattern that are off (0) are ignored. Overlay writing is the default style; you do not need to specify an option to select it, unless you want to change from another style.

In the following example, the display is cleared with a screen command (screen controls are explained on page 134) and the cursor is positioned to [6,200]. A series of write commands select yellow for the foreground, overlay writing, and line pattern P2 (a series of dashes) with a multiplication factor of 1. The vector command is then used to draw a line 100 pixels long.

```
S(I0,E)
P[6,200]
W(I(Y)) (V) (P2(M1))
V[+100]
```


Replace Writing

Format: `W(R)`

When replace writing is selected, pixels are drawn with the foreground color where bits in the writing pattern are on (1). Pixels are drawn with the background color where bits in the pattern are off (0). (If the background color is not changed from its default using a screen command, the effect of replace writing appears the same as for overlay writing.)

In the following example, the display is first cleared and the cursor positioned as in the overlay writing example. Then, the background color is set to the color in output map location 3, the foreground color to output map location 6, and replace writing is selected. Line pattern 2 (dashes) is selected with a multiplication factor of 1, and finally a vector 100 pixels long is drawn.

```
S(I0,E)
P[6,200]
S(I3)
W(I6) (R)
W(P2(M1))
V[+100]
```

Complement Writing

Format: `W(C)`   (specifies complement writing alone)

          `W(F<plane code>,C)`   (specifies complement writing and a
                                    plane selection)

When complement writing is selected, the binary value of the pixel's color (that is, the value of its output map location) is complemented where bits in the writing pattern are on (1). Bits in the pattern that are off (0) are ignored.

For example, on the VT240 terminal (with a 2-bit code for each pixel), a pixel with a color value of 1 (binary 01) has a complemented value of 2 (binary 10), and complement writing is done using the color in output map location 2.

Likewise, on the VT340 terminal (with a 4-bit code for each pixel), a pixel with a color value of 7 (binary 0111) has a complemented value of 8 (binary 1000), and complement writing is done using the color in output map location 8.

The actual color that appears on the screen is not necessarily the true complementary color of the original color. The setting of the foreground intensity (the (I) option) has no effect on complement writing.

In this example, the first two lines clear the display and position the cursor. Then, the screen color is changed to the color in output map location 2, replace writing is selected, using the color in output map location 5 and line pattern 2 with a multiplication factor of 1, and a vector 100 pixels long is drawn. Finally, the cursor is repositioned, complement writing is selected with line pattern 1 (solid), and another vector 100 pixels long is drawn. Complement writing causes the new line to be drawn using the colors in output map locations 13 and 10.

```
S(I0,E)
P[6,100]
S(I2)
W(R,I5,P2(M1))
V[+100]
P[6,100]
W(C,P1)
V[+100]
```

Erase Writing

Format: `W(E)`

With erase writing, all pixels are drawn using the background color, regardless of whether the writing pattern bits are on or off. If negative pattern control is turned on (as described below), erase writing draws using the foreground color.

In the following example, the screen is cleared and the cursor positioned. Then the background color is set to the color in output map location 6, erase writing is set, line pattern P2 is selected with a multiplication factor of 1, and a vector 100 pixels long is drawn.

```
S(I0,E)
P[6,200]
S(I6)
W(E)
W(P2(M1))
V[+100]
```


Negative Pattern Control

Format: `W(N<0 or 1>)`

When negative pattern control is turned on with `W(N1)`, the current writing pattern is reversed. By default, this option is off: `W(N0)`.

Shading

Format: `W(S<0 or 1>)`

The shading option lets you shade the inside of an image in ReGIS as it is drawn. The current writing pattern determines the pattern for shaded areas, and the current writing style determines how 1 bits and 0 bits appear.

Shading occurs from the point being drawn by the vector or curve command to a *shading reference line*. The default shading reference line is a horizontal line defined by the Y coordinate of the current cursor position.

The option `(S1)` turns shading on. By default, shading is off.

Selecting a Horizontal Reference Line

Format: `W(S[,Y])`

Select a new horizontal reference line by specifying a Y coordinate as an argument to the shading option.

The following example draws a filled, bullet-shaped figure. The cursor is positioned, shading is turned on, a horizontal reference line 300 pixels down is defined, and a curve with a radius of 100 pixels is drawn at the top. After the commands are executed, the cursor remains at [125,125].

```
P[125,125]
W(S1[,+300])
C[+100]
```

Selecting a Vertical Reference Line

Format: `W(S(X) [X])`

Select a vertical reference line.

The next series of commands draws another filled, bullet-shaped figure, this time using a vertical reference line as the reference for shading.

```
P[600,375]
W(S1(X)[-340])
C[+50]
```

Selecting a Character for Shading

Format: `W(S'<character>')`

Select a given character to be used for shading.

The following example positions the cursor, selects a forward slash (/) as the character to be used for shading, and draws a filled circle. Selecting a character for shading also turns shading on.

```
P[400,200]
W(S'/')
C[+75]
```

To outline the shaded circle above, turn shading off and draw the circle again. You can do this by adding the following commands:

```
W(S0)
C(W(I2)) [+75]
```

The following figure shows all four of the shading command examples described.
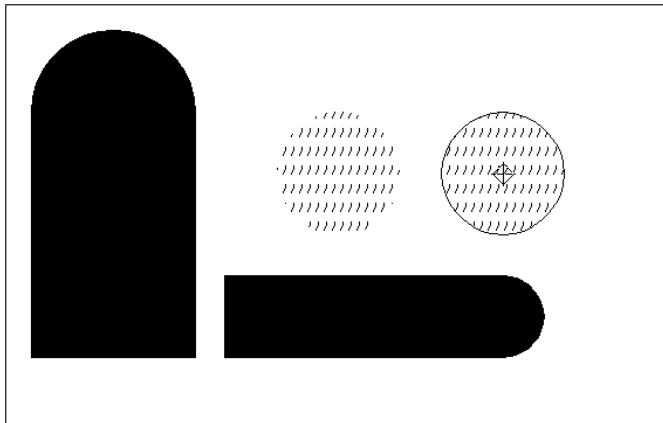
Examples of Writing Styles

The following are detailed examples of each style of writing. The polygon fill command F is used to draw each square; this command is explained further starting on page 160.

```
S(I15,E)
W(I0)
P[225,200]
F(V[+100] [,+100] [-100] [,-100])
W(I7,V)
P[250,225]
F(V[+100] [,+100] [-100] [,-100])
```
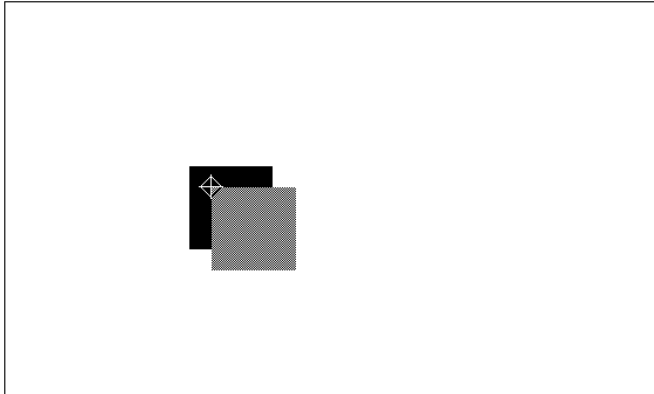
```
S(I15,E)
W(I0)
P[225,200]
F(V[+100] [,+100] [-100] [,-100])
S(I7)
W(I15,R)
P[250,230]
T 'Replace Writing'
```
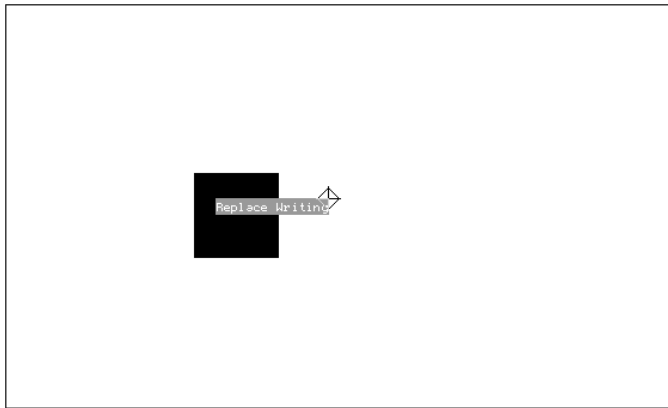
```
S(I15,E)
W(I0)
P[225,200]
W(S'#')
F(V[+100] [,+100] [-100] [,-100])
W(S1S0,I15)
P[268,240]
W(C)
F(V[+100] [,+100] [-100] [,-100])
```
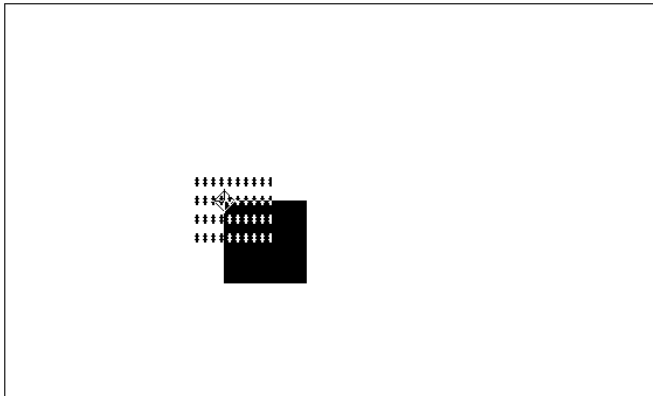
```
S(I15,E)
W(I0)
P[225,200]
F(V[+100] [,+100] [-100] [,-100])
W(E)
P[250,225]
F(V[+50] [,+50] [-50] [,-50])
```
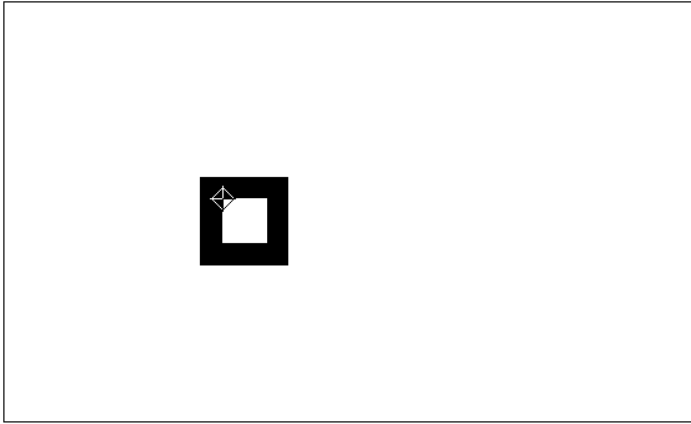

Plane Selection

Format: W(F<0 to 3>)   (VT240, VT241, or VT330)
        W(F<0 to 15>)   (VT340)

When the **Terminal ID** (on the Emulation tab of the Terminal Setup dialog box) is set to **VT240**, **VT241**, or **VT330**, ReGIS can write to 2 bitmap planes. When the **Terminal ID** is set to **VT340**, 4 bitmap planes are available. For the VT240, VT241, and VT330, each pixel has a 2-bit code—one bit for each plane—giving up to four different simultaneous shades of gray (or colors in the case of the VT241). For the VT340, each pixel has a 4-bit code, allowing for 16 simultaneous colors.

With the plane selection option to the write command, you can specify which bitmap planes can be written to. This lets you disable or enable each plane individually. This can affect how complement writing appears, how images are overlaid with one another, and so on.

The code numbers that select the bitmap planes (0 to 3 or 0 to 15) correspond to the binary representations of the planes you can write to. A 1 bit selects a plane for writing.

For example, with the VT330 terminal, the digit 2 in the command `W(F2)` corresponds to the binary code 10, allowing you to write to bitmap plane 1 (counting from 0, right to left). With the VT340 terminal, the digit 7 in the command `W(F7)` corresponds to the binary code 0111, letting you write to planes 0, 1, and 2; plane 3 is left disabled.

VT240, VT241, and VT330 Writing Planes

Command	Planes that can be written to
W(F0)	None
W(F1)	0
W(F2)	1
W(F3)	All

VT340 Writing Planes

Command	Planes that can be written to
W(F0)	None
W(F1)	0
W(F2)	1
W(F3)	0 and 1
W(F4)	2
W(F5)	0 and 2
W(F6)	1 and 2
W(F7)	0, 1, and 2
W(F8)	3
W(F9)	0 and 3
W(F10)	1 and 3
W(F11)	0, 1, and 3
W(F12)	2 and 3
W(F13)	0, 2, and 3
W(F14)	1, 2, and 3
W(F15)	All

After drawing with a restricted plane selection, you should restore writing to all planes with the command `W(F15)`.

Note: Plane selection requires that you run Windows in 256-color mode.

Polygon Fill Command

The polygon fill command *F* is used to draw and fill closed figures such as squares, triangles, and circles. There are four options to the polygon fill command, letting you position the cursor, draw vectors, draw curves, and specify temporary write controls. In general, the fill command accepts any ReGIS commands as options (enclosed in parentheses). The current foreground color specified by a screen or write command is used as the color for polygon filling.

Vector

Format: `F(V<coordinates>)`

All of the options and arguments for the vector command (page 126) can be used as suboptions to the fill command.

The following example creates an irregular, filled, 5-sided image:

```
P[50,250]
F(V[200,250] [,+50] [-50,+50] [-80,-40] [-20,-60])
```

Because there is no way for ReGIS to know when the polygon is complete, it is not drawn and filled until the closing parenthesis is entered.
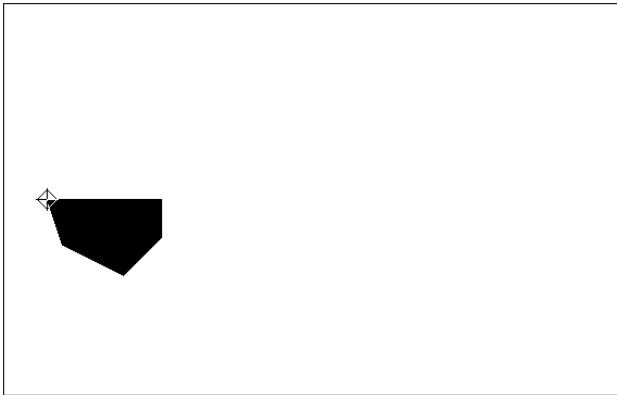
Curve

Format: `F(C<coordinates>)`

The options and arguments for the curve command (page 129) can all be used as suboptions to the fill command.

The following creates a filled ellipse. The *B* and *E* options are used to define the beginning and end of the curve:

```
P[150,200]
F(C(B) [+300] [,+200] [-300] (E))
```

To draw a filled circle with its center at [400,200] and a radius of 100 pixels, you would use the following commands:

```
P[400,200]
F(C[+100])
```

Position with Curve

Format: `F(C(A+<degrees>) <position 1> P<position2>)`

This option is used to connect curves, arcs, and vectors.

The C identifies the curve option, and the A+*<degrees>* parameter indicates the arc suboption. Both are explained on page 130.

The next example produces a slanted, rectangular shape with rounded ends. An initial position is specified at [200,200]. Then the fill command is given, first defining a 90 degree counterclockwise arc that starts 50 pixels right and 50 pixels up from the current position. The current position is then moved down and right 100 pixels, and a second 90 degree arc is defined from a point 50 pixels left and 50 pixels down from the new position. When the closing parenthesis is encountered, the shape defined by the coordinates is filled.

```
P[200,200]
F(C(A+90) [+50,-50]
P[+100,+100]
C(A+90)
[-50,+50])
```
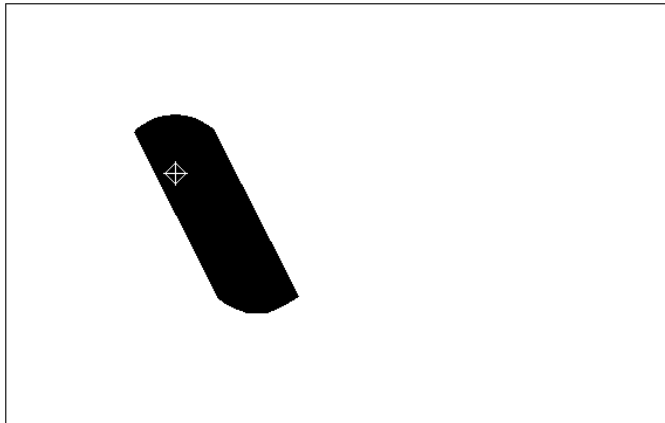
Temporary Write Control

Temporary write control can be used either as an option of the fill command or as a subsection of the curve and vector options. As an option of the F command, the syntax is as follows:

Format: `F(W(<suboptions>) <options>)`

Any of the options for the write command can be used in the parameter W(*suboptions*). In the format above, the temporary write control applies only to the *<options>* of the polygon fill command.

The syntax for using the temporary write control as a suboption of the C or V options is as follows:

Format: `F(C(W(<sub-suboptions>) <suboptions>) <options>)`

In this format, the W(*sub-suboptions*) parameter has an effect only on the curve command and its suboptions; it does not necessarily affect the fill command.

Text Command

The text command *T* determines the size and orientation of graphics text characters. Options of the text command let you select a character set, the spacing between characters, the size and height of characters, and so on. Until you define new text characteristics, the ones you select remain in effect, unless they have been defined with temporary write controls.

In discussing characters cells and sizes, this manual uses values corresponding to a full-sized, unscaled ReGIS display of 800 ⚥ 480 pixels, similar to the VT340 terminal. Reflection automatically scales characters to fit the actual size of your display.

About Character Formats

When you use the text command, ReGIS selects each character from a stored character set and draws it to the screen, scaling and orienting the character according to the values defined. This lets you align subsequent characters using PV values and position commands.

Each character is drawn with the upper-left pixel of the character cell at the cursor position. The following figure shows a few sample characters from the ASCII character set.
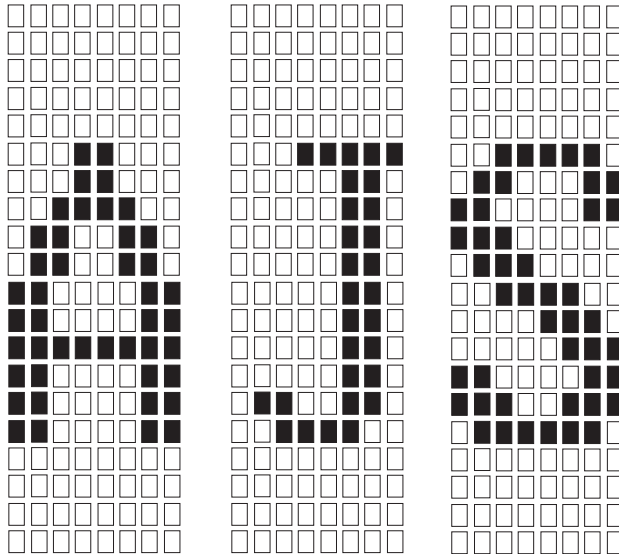
When characters are drawn normally—without any tilt—each character appears below and to the right of the cursor. If a 180 degree tilt is applied, characters appear upside-down, with each one starting above and to the left of the cursor. The upper-left pixel of the character cell acts as a "pivot" around which the character cell is rotated.

Knowing information about the size and orientation of character cells can be useful when working with the size and tilt options of the text command, explained on page 169 and 171.

About Text Formats

The following conventions are used to specify text strings in ReGIS:

• Single or double quotes can be used to mark the beginning and end of the string. The same type of marks must delimit the string.

For example, `"this is text"` and `'this is text'` are equivalent.

• If one type of quote is used to delimit the string, the other type can be used within the string.

For example:

`"it's a Weimaraner"`	appears as	`it's a Weimaraner`
`"Wow,"she exclaimed'`	appears as	`"Wow," she exclaimed`

• Two quote marks in a row, with no space separating them, are used to include both types of quotes within a string.

For example:

`"Quote mark: "" "`	appears as	`Quote mark: "`
`'it's a boy'`	appears as	`it's a boy`

• A comma placed between two strings that have the same type of quote marks combines the strings.

For example:

`"Press ","enter"`	appears as	`Press enter`
`"Press ""enter"`	appears as	`Press "enter`

ReGIS also recognizes four control characters within a text string:

Character	Keystroke	Description
C_R	Ctrl + M	Carriage return—horizontally returns the cursor to the position where the string started
L_F	Ctrl + J	Linefeed—moves the cursor down one line
B_S	Ctrl + H	Backspace—moves the cursor back one position; this lets you overstrike a character
H_T	Ctrl + I	Horizontal tab—moves the cursor forward one space, using the current value for text spacing

Select Character Set

The character set option lets you select which character set to use for ReGIS text strings. You can use the built-in character sets shown starting on page 250, or you can load your own (see page 176 for more on the load command). The character sets selected for ReGIS text do not change when you exit or enter ReGIS. The default character sets loaded for ReGIS text are the ASCII character set and the ISO Latin-1 supplemental graphic set.

Character sets are accessed the same way in ReGIS mode as they are in text mode (see page 68 for information about text mode character sets). However, in ReGIS mode, the "in-use" table is distinct and can contain different character sets from the in-use table for text mode. See page 70 for more information about how the in-use tables are defined.

The following format selects a character set for ReGIS text:

Format: `T(A<0 to 3>)`

A0 selects a set from the built-in character sets, while *A1*, *A2*, or *A3* selects a set that can be loaded. The built-in set occupies an 8-bit character code table, with the left half (GL) used for 7-bit characters and the right half (GR) for 8-bit characters. By default, the ASCII character set is automatically mapped into GL when you enter ReGIS, and the ISO Latin-1 supplemental graphic set is mapped into GR. The set you can load with A1, A2, or A3 can include only 7-bit characters, and cannot have more than 96 characters.

When selecting the built-in character sets with *A0*, you can use the following commands to select a character set to map into GL, GR, or both halves of the in-use table. Only 7-bit character sets can be mapped into GL; either 7-bit or 8-bit character sets can be mapped into GR.

Format: `T(A0(L"<character set designator>"))`
`T(A0(R"<character set designator>"))`
`T(A0(L"<designator>",R"<designator>"))`

The following two tables list the designators for the built-in 7-bit and 8-bit character sets.

7-Bit Character Set Designators

Character Set	Text Command
ASCII (default)	(B
DEC Special Graphic	(0
DEC Technical	(>

National Replacement Character Sets	
British	(A
Dutch	(4
Finnish	(5
French	(R
Canadian	(9
German	(K
Italian	(Y
Norwegian/Danish	(`
Spanish	(Z
Swedish	(7
Swiss	(=
Portuguese	(%6

8-bit Character Set Designators	
DEC supplemental graphic	)%5
ISO Latin-1 supplemental (default)	-A
User-preferred supplemental (94-character set)	)<

In the following example, the DEC Technical character set is selected for GL:

```
T(A0(L"(>"))
```

In this example, the ISO Latin-1 character set is selected for GR:

```
T(A0(R"-A"))
```

In this example, the British character set is selected for GL, and the DEC Technical set is selected for GR:

```
T(A0(L"(A",R"(>"))
```

Character Spacing

You can space between ReGIS text characters in three ways:

- Select a standard character cell size (explained in the next section). The spacing value for that size is used.

- Select a character orientation.

- Specify a relative X and Y value for spacing, using the format below.

To specify a relative X and Y value for spacing between ReGIS characters, use the following command:

Format: `T<[X,Y] position>`

The X and Y positions are relative values, even if no sign is given. After each character is drawn, the X and Y values tell ReGIS how far to move the cursor before drawing the next character. A negative X spacing value writes text from right to left; a Y spacing value writes text vertically. Changing the character spacing does not change the baseline orientation of characters.

In the following example, the cursor is positioned, a spacing value of 25 pixels is specified, and four letters are written.

```
P[100,100]
T[25]"ABCD"
```

Character Size

There are three ways to select a character size:

- Select a standard character size.

- Define a display cell size.

- Define a unit cell size.

The *display cell size* is the area that a given character takes up on the screen. This area includes spacing between characters and lines. The *unit cell size* is the size of the character within the display cell.

If you think of the display cell as a box, the unit cell is its contents. Increasing only the display cell size of a character does not increase the size of the character itself; it increases only the area around the character. To increase the size of the characters, you must increase the unit cell size.

The following command selects a standard character cell size. When you select a standard cell size, the display cell size, unit cell size, and character spacing are already defined.

Format: `T(S<0 to 16>)`

The number 0 to 16 selects a standard size from the table below. Character spacing is expressed as in the previous section.

T Command: Standard Character Cell Sizes

Standard Size	Display Cell Size	Unit Cell Size	Character Spacing
S0	[9,10]	[8,10]	[9,]
S1 (default)	[9,20]	[8,20]	[9,]
S2	[18,30]	[16,30]	[18,]
S3	[27,45]	[24,45]	[27,]
S4	[36,60]	[32,60]	[36,]
S5	[45,75]	[40,75]	[45,]
S6	[54,90]	[48,90]	[54,]
S7	[63,105]	[56,105]	[63,]

Standard Size	Display Cell Size	Unit Cell Size	Character Spacing
S8	[72,120]	[64,120]	[72,]
S9	[81,135]	[72,135]	[81,]
S10	[90,150]	[80,150]	[90,]
S11	[99,165]	[88,165]	[99,]
S12	[108,180]	[96,180]	[108,]
S13	[117,195]	[104,195]	[117,]
S14	[126,210]	[112,210]	[126,]
S15	[135,225]	[120,225]	[135,]
S16	[144,240]	[128,240]	[144,]

You can also define your own display and unit cell size. The following is the command for defining a display cell size:

Format: `T(S[<width,height>])`

The following is the command for defining a unit cell size:

Format: `T(U[<width,height>])`

The unit cell size, which defines the width and height of characters, should be close to the display cell size. Otherwise, the following can occur:

- If the display cell size is larger than the unit cell size, the excess area around each character is treated as pattern bits. If replace writing or erase writing mode is in effect, the excess area is drawn in the background color; otherwise, the excess area remains unchanged.

- If the display cell size is smaller than the unit cell size, only the part of the character that can fit in the display cell is displayed.

Height Multiplier

Format: `T(H<1 to 256>)`

By specifying a height multiplier with the text command, you can change the height of the display and unit cells to the same value without changing their width.

The height for the new character is the value you specify (from 1 to 256 pixels) multiplied by 10. The horizontal character width and the spacing between characters remain unchanged.

For example, the following commands position the cursor, set the character size to the standard size S2 (16 pixel wide by 30 pixel high characters), and write a string of text. Then, the default character height is multiplied by 5, and another string is written. The character width and spacing are not changed.

```
P[50,50]
T(S2)'30 pixels high'
T(H5)'50 pixels high'
```

Size Multiplier

Format: `T(M[width<1 to 16>,height>])`

By specifying a size multiplier, you can multiply both the standard character height and width by the same or different factors.

This sets the unit cell size to the width you specify multiplied by 8, and the height you specify multiplied by 10.

String Tilt

Format: `T(D<text string angle>,S<0 to 16>)`

This option determines a tilt angle for text strings. Each character lies along the same tilted baseline, so that the entire string appears pivoted around the starting point of the string.

The tilt angle must be in 45 degree increments. The S*<0 to 16>* option selects a character cell size from one of the 17 standard cell sizes (see the table on page 169); the cell size is used to determine the spacing between characters in the tilted string. (By itself, the *D* option indicates only character rotation.)

Because screen pixels are further apart diagonally than they are horizontally or verti-cally, char-acters will appear distorted when writing strings at angles other than 0 or 180 degrees. There is only slight distortion at 90 and 270 degrees, while the distortion is more apparent at 45, 135, 225, and 315 degrees.

You can partially correct for character distortion when writing tilted strings by following this general guideline:

• Select a character size for the tilted text that has a width about two-thirds the width of the untilted text.

For example, if you write a string of untilted text using the standard character size S15, writing a tilted string using standard size S10 will produce characters that look approximately the same size. Size 10 characters are 80 pixels wide, which is two-thirds the width of the size 15, 120-pixel wide characters.

Likewise, if you write a string of untilted characters using size 5 characters, which are 40 pixels wide, you would write a tilted string using size 3 characters, which are 24 pixels wide. The size 3 characters are approximately two-thirds as big as size 5 characters, and look about the same size as the untilted, size 5 characters.

However, even the resized characters may appear distorted, and you may need to make a char-acter height correction. Do this by specifying a height multiplier in addition to a character size. The syntax of the text command becomes the following:

Format: `T(D<text string angle>,S<0 to 16>,H<1 to 256>)`

As explained for the height multiplier option on page 171, the value for H is multiplied by 10, the height of the default character cell.

For example, the following command writes a string of text at 45 degrees, using character size 3, and applying a height multiplier of 5. This results in characters that are 24 pixels wide and 50 pixels high.

```
T(D45,S3,H5)
```

String and Character Tilt

Format: `T(D<text string angle>,S<0 to 16>,D<character angle>)`

A variation on the string tilt option lets you specify both the tilt of the text string and the tilt of each character within the string. By using different string and character tilts, the entire string can be pivoted around a point at the start of the string, with each character pivoted around a point at the start of the character.

Like the string tilt option, character tilts must be at 45 degree increments. If no character tilt is specified, characters are tilted at the same angle as the text string. To offset distortion that may appear, you can apply a height multiplication factor; for example, `T(D315,S3,H5,D180)`'This is the text'.

Italics

Format: `T(I<angle>)`

With the italics option *I*, you can indicate the degree and direction of character slant: a positive number makes characters appear slanted up and to the left, while a negative number makes them appear slanted up and to the right. The angle of the slant can be plus or minus 0 degrees, 22 degrees, or 45 degrees.

The following example shows a line of text written using size S2 at 22 degrees, 45 degrees, minus 22 degrees, and minus 45 degrees. Italics slant is *not* the same as character or string tilt.

```
P[100,100]
T(S2)
T(I22) "Message for Bryan (I22)"
P[100,125]
T(I45) "Message for Bryan (I45)"
P[100,150]
T(I-22) "Message for Bryan (I-22)"
P[100,175]
T(I-45) "Message for Bryan (I-45)"
```

Message for Bryan (I22)
Message for Bryan (I45)
Message for Bryan (I-22)
Message for Bryan (I-45)
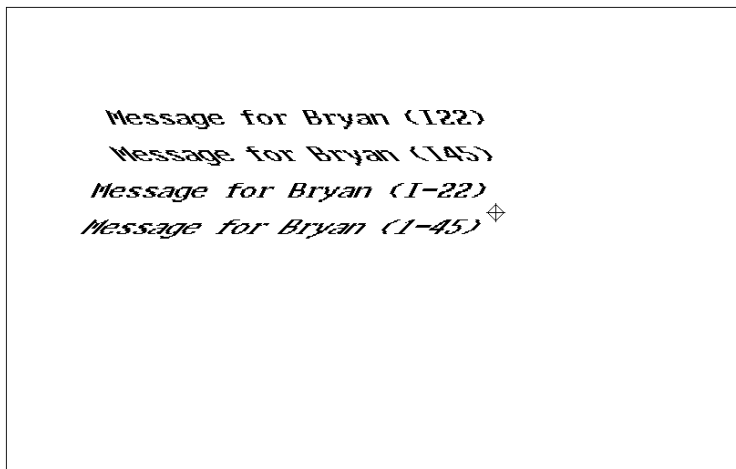
Temporary Text Control

Format: `T(B) <options> (E)`

Temporary text control options let you control the characteristics of a limited string of text; when the string ends, text controls are returned to their previous values.

The following commands write the phrase "Reflection 4," begin a temporary text control to write the phrase upside down, and then end the temporary control. After the temporary control is ended, the text has the same attributes it did before the temporary control was begun.

```
P[100,100]
T(D0,S3,I0) "Reflection 4"
P[430,250]
T(B) (D180,S3) "Reflection 4" (E)
P[110,330]
T "Reflection 4"
```

```
Reflection 4

⟘ uoᴉʇɔǝꞁℲǝꓤ  ㄣ

Reflection 4
```
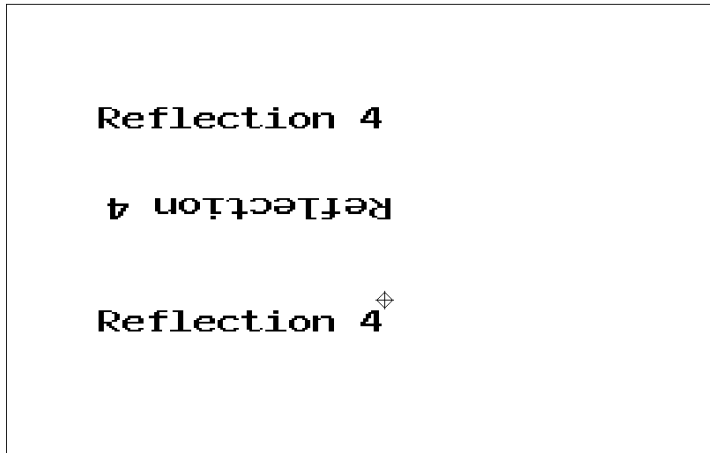
Temporary Write Control

Format: `T(W(<suboptions>)<text controls>)`

Temporary write control options provide temporary control over text characteristics such as intensity and the kind of writing: overlay, erase, replace, or complement. See page 145 for details about write command options.

The temporary write controls remain in effect until a new command key letter is used, another temporary write control command is encountered, or the ReGIS command line is resynchronized (;). The temporary write controls apply to the specified text controls only.

Positioning Text with PV Values

Format: `T<PV value>`

You can use the pixel vector system to move text relative to the current baseline. Superscripting, subscripting, and overstriking are all possible. Each PV value moves the character one half of the current display cell size, along the PV direction specified. PV multiplication factors have no effect.

The following table lists the effect of each of the eight PV values.

PV Value	Character Movement
0	Forward one half character cell
1	Superscript: Up from baseline and to right of the previous character
2	Superscript: Straight up from baseline
3	Up from baseline and toward the previous character, partially overwriting the previous character
4	Overstrike: Backward one half character cell; use 44 to overstrike
5	Down from baseline and toward the previous character, partially overwriting the previous character
6	Subscript: Straight down from baseline
7	Subscript: Down from baseline and to right of the previous character

The following commands write the mnemonic for form feed (FF) and then return to the baseline:

```
P[300,200]
T(S2)"F"6"F"1
```

Load Command

The text command selects any one of the built-in character sets; the load command, on the other hand, lets you design your own character sets. There are two steps required to load your own character set:

1. Select a number from 1 to 3 for the character set you want to load.

2. Load the character cells (up to 96 characters per set are possible).

Once the character set is created, the select option of the load command is used to load or reload a given set. Although you can select the built-in character set 0, you cannot load it with the select option.

Select Character Set

Format: `L(A<0 to 3>)`

When you select a character set, all ReGIS load commands apply to this set until you select another set. Other types of ReGIS commands can be used without affecting the selected character set. For example, you can change the appearance of text with the text command, without affecting the selected character set.

Specify Name

Format: `L(A"<name>")`

You can specify a name up to ten characters long for the character set you are loading. By giving the character set a name, you can receive a meaningful response when requesting a report about the loaded character set.

You can also give a name to the default character set, which is selected by the text command `T(A0)`. By default, the A0 set has no name. When you use the report command to get the name of a character set, an unnamed set is reported as `(A"")`.

Load Character Cell

Format: `L"<ASCII character>" <hex pairs...>`

The above command lets you create each character in your selected set. The defined character occupies an 8 pixel wide by 10 pixel high unit cell.

The <*ASCII character*> is a single letter or number used to identify the loaded character; it does not represent the shape of the loaded character. For example, if you want to create the symbol for copyright, you might call the character "c".

The <*hex pairs*> define which pixels are on and off within the 8 ⚡ 10 character cell. The pairs can be separated by commas for better readability. Each hex code in a hex pair represents four pixels of a cell row: the first hex code of the first hex pair defines the pattern of the left four pixels in the top row of the cell; the second hex code of the first hex pair defines the right four pixels in the first row. Moving from top to bottom through the unit cell, the second hex pair defines the second row of the cell, the third pair defines the third row of the cell, and so on.

The hex codes that create the 16 possible four-pixel patterns are shown in the following table. An example follows the table.

L Command: Hex Codes
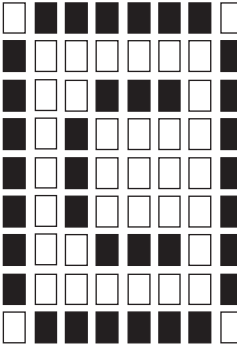
Hex Code	Binary Code	4-Pixel Bit Pattern
0	0000	
1	0001	■
2	0010	■
3	0011	■ ■
4	0100	■
5	0101	■ ■
6	0110	■ ■
7	0111	■ ■ ■
8	1000	■
9	1001	■ ■
A	1010	■ ■
B	1011	■ ■ ■
C	1100	■ ■
D	1101	■ ■ ■
E	1110	■ ■ ■
F	1111	■ ■ ■ ■

Blank rows that are above rows with pixels on must be defined; undefined rows at the bottom of the character cell, however, are assumed to be blank.

The following example selects the character set 1, assigns it the name symbols, and loads a character with the copyright symbol. Its call letter is c.

```
L(A1"symbols") "c" 7E,81,9D,A1,A1,A1,9D,81,7E
```

The commas between the hex pairs are not required: they are included for readability only. The next figure shows an enlarged view of the character cell with its corresponding hex pairs.

Character Cell   Hex Code

⬜⬛⬛⬛⬛⬛⬜	7E
⬛⬜⬜⬜⬜⬜⬛	81
⬛⬜⬛⬛⬛⬜⬛	9D
⬛⬜⬛⬜⬜⬜⬛	A1
⬛⬜⬛⬜⬜⬜⬛	A1
⬛⬜⬛⬜⬜⬜⬛	A1
⬛⬜⬜⬛⬛⬛⬜⬛	9D
⬛⬜⬜⬜⬜⬜⬛	81
⬜⬛⬛⬛⬛⬛⬜	7E

The following figure shows what the character looks like when it is displayed on the screen using a standard cell size of S12. The command to select character set 1 and display the character is `T(A1,S12)"c"`.
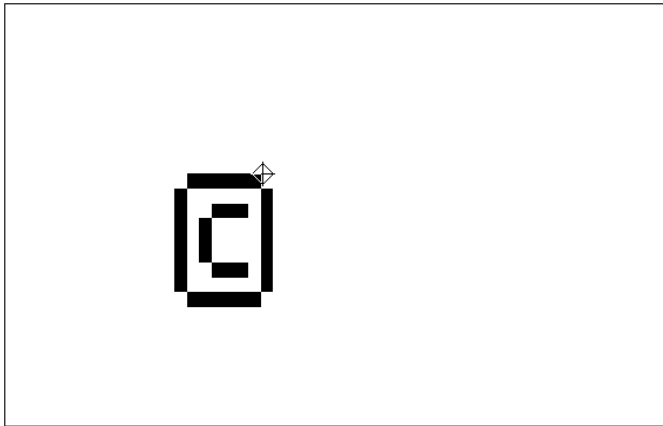
Report Command

The report command can perform two different functions:

- Request information on the current status of ReGIS operations.

- Enter graphics input mode.

Cursor Position

Format: `R(P)`

This option requests the current cursor position; ReGIS reports the absolute screen coordinates.

Macrograph Contents and Storage

Format: `R(M(<call letter>))`

ReGIS can report on the contents of a specific macrograph with the above command. The <*call letter*> is the letter that identifies the macrograph. The report that is sent to the host begins with `@=<call letter>` and ends with `@;`C_R. This option is affected by the setting of the **Macrograph reports** check box in the **ReGIS** group box on the Graphics tab of the Terminal Setup dialog box. By default, reports are enabled. You may want to disable macrograph reporting for security reasons.

The next command tells ReGIS to report the total amount of space allotted to macrograph storage, and the amount of free space.

Format: `R(M(=))`

The report sent to the host is in the form `aaaa,tttt`, where *aaaa* is the amount of storage still available, and *tttt* is the total amount of storage that was allocated.

Character Set

Format: R(L)

ReGIS can report the name of the character set currently selected for load command operations.

The name of the set is reported in the following format:

```
(A"<character set name>")
```

Error

Format: R(E)

The following option requests a report on the last error detected by the ReGIS command parser; it must be given before the ReGIS command line is resynchronized. The resynchronization character (;) clears all errors.

The latest error is reported in the form "<error code>,<ASCII code>". The <*error code*> is a decimal value from the table below. The <*ASCII code*> is the decimal ASCII code of the character that caused the error. The following table lists the possible error codes and their meanings.

R Command: Error Conditions

Error condition	Report	Meaning
No error	"0,0"	No error since last synchronization.
Ignore character	"1,<n>"	A character with an ASCII decimal value of <*n*> was ignored.
Extra coordinate	"3,48"	[X,Y] contains more than two coordinates.
Alphabet out of range	"4,48"	L(A<0 to 3>) contains a number outside of the <0 to 3> range.
Begin/start overflow	"7,66" or "7,83"	The limit of 16 (B) and (S) options for position and vector commands has been exceeded.
Begin/start underflow	"8,69"	A position or vector command contains an (E) option without a matching (B) option.

Error condition	Report	Meaning
Text standard size error	`"9,48"`	A text command has selected a standard character size outside of the <0 to 16> range.

Graphics Input Modes

Graphics input modes let you position the cursor and send a position report. The report is sent in response to the interactive position report command described in the next section.

Input mode can be either *one-shot* (the default) or *multiple*. In one-shot mode, the application running on the host is suspended until ReGIS sends a position report. In multiple input mode, a number of cursor position reports can be sent without leaving graphics input mode.

The following command indicates that the interactive position report should use one-shot mode:

Format: `R(I0)`

The input cursor appears on the screen, and you use the mouse or arrow keys to position it. You remain in one-shot mode until a cursor position request is received; a position report is sent when you press the mouse button or an active non-arrow key. Once the position report is sent, you exit one-shot mode.

The following command indicates that the interactive position report should use multiple input mode:

Format: `R(I1)`

The input cursor appears, and you use the mouse to position it. Multiple mode allows more than one cursor position report to be sent without exiting input mode. Cursor position reports are sent whenever you press the mouse button, or when an interactive position report command is received. To exit multiple input mode, use the command that indicates one-shot mode: `R(I0)`.

Interactive Position of Input Cursor

Format: `R(P(I))`

With this option, an application can request an input cursor position in either one-shot or multiple graphics input mode.

In one-shot mode, ReGIS sends the position report when you press an active non-arrow key or the mouse button. In one-shot mode, you can use the mouse or arrow keys to position the input cursor. In multiple mode, ReGIS sends a position report immediately when you press the mouse button. You cannot use the arrow keys in multiple input mode to position the cursor.

Cursor Movement

If you use a locator device, such as a mouse or graphics tablet, the input cursor moves when you move your device. When the graphics input cursor appears on the screen, it can be moved with the locator device or with the arrow keys on the numeric keypad.

Keystroke	Effect
↑ ↓ → ←	Moves cursor 1 pixel at a time
Shift+↑ ↓ → ←	Moves cursor 10 pixels at a time

Position Report Format

In one-shot mode, a position report is sent only after the interactive cursor position command is received by ReGIS. The report contains the code for the key pressed (or a mouse button code from the table below), followed by the cursor's absolute position and a carriage return.

For example, a position report of `a[225,134]` means that when the interactive cursor position command was sent, the user placed the input cursor at coordinates [225,134] and pressed the A key. A position report of CSI241˜[350,75] means the user placed the input cursor at [350,75] and pressed the mouse button.

In multiple input mode, a cursor position is sent whenever one is requested by the host application or a mouse button is pressed. If an interactive position was not requested, the format of the position report is the same as above for one-shot input mode. If an interactive position was requested, the format of the position report is ^{CS}I240˜[X,Y]. The ^{CS}I240˜ is a "null button" sequence; it indicates that a mouse button was pressed in response to an interactive position request.

Button	Code (when pressed)
1 (left)	^CS_I241~
2 (middle)	^CS_I243~
3 (right)	^CS_I245~
4	^CS_I247~

The left, middle, and right designations in this table refer to the buttons on Digital's three-button mouse. The response that the mouse button sends when the button is pressed and released can be defined using the DECLBD function; see page 61.

Macrograph Command

A macrograph is a macro that stores ReGIS graphics commands. Instead of redrawing the same image each time you need it, you can store the ReGIS commands for the image under a macrograph call letter, then replay the macrograph each time you want the image. You can store up to 26 macrographs. You can also nest macrographs within other macrographs, up to 16 levels deep.

Define Macrograph

Format: @:<call letter> <definition>@;

A macrograph is identified by a call letter that is not case sensitive. The commands in the <definition> are not drawn as the macrograph is defined. You must invoke the macrograph to draw the image.

The following example defines a macrograph, identified as S, to draw a square relative to the cursor position. Each side of the square is 100 pixels long.

```
@:S V[+100] [,+100] [-100] [,-100]@;
```

Invoke Macrograph

Format: `@<call letter>`

This command invokes a macrograph. The following example invokes the macrograph defined above:

```
@S
```

Clear Macrographs

Format: `@:<call letter>@;`

This command deletes a single macrograph.

Format: `@.`

This command clears all defined macrographs (the period is required).

Tektronix Graphics

Tektronix Graphics Support in Reflection

Note: This section only applies if you are using Reflection for ReGIS Graphics.

Reflection for ReGIS Graphics supports all the Tektronix 4010/4014 features available on VT terminals. It also includes the Tektronix 4105 commands for line style and color.

Reflection emulates the Tektronix terminals in essentially all aspects, so you interact with graphics programs exactly as they are documented for the terminal. Reflection even rescales or repaints the graphics when you change the size of Reflection's terminal window.

This section first describes how to configure Reflection for Tektronix emulation and explains the three main display modes.

About Tektronix Graphics

Reflection for ReGIS Graphics supports all the Tektronix 4010/4014 features available on VT terminals. The Tektronix screen uses vectors (lines) to display both alphanumeric characters and graphics. In most cases the application software you are running automatically switches between the VT and Tektronix emulations. However, you can switch modes manually as explained below. The items in Reflection's **Tektronix** group box on the Graphics tab in the Terminal Setup dialog box allow you to configure the behavior of Reflection for Tektronix emulation.

A Tektronix terminal with Enhanced Graphics Module enters Special Point Plot mode upon receiving the escape sequence ᴱˢᶜꜰˢ from the host. Reflection, unlike the VT340, provides partial support for Special Point Plot mode.

Configuring for Tektronix Graphics Emulation

Normally the host switches Reflection into Tektronix mode. To manually begin Tektronix emulation:

1. Start Reflection for ReGIS Graphics as you normally do.

2. On the Setup menu, click Terminal.

3. On the Terminal Type tab in the Terminal Setup dialog box, select **401X** as the **Terminal type** group box option.

4. Click OK.

If there was any text on the terminal window, it is cleared and the window appears blank with the cursor in the upper-left corner and the mouse pointer changing to a crosshair pointer.

Sample Tektronix Graphics Session

The graphics features of Reflection's Tektronix emulation are primarily for using graphics programs running on a host computer. However, you can perform some graphics operations independent of a host application.

After manually configuring Reflection for Tektronix emulation (see above), this example shows you how to display a Tektronix image on the screen:

1. During Tektronix emulation, alphanumeric characters are displayed as vector graphics. If you have an active host session, type some characters to see how they are displayed on the screen.

2. On the Edit menu, click Clear Display.

3. Then, you need to go "off line:"

    a. On the Setup menu, click Terminal.

    b. On the Emulation tab of the Terminal Setup dialog box, clear the Online check box.

    c. Click OK.

4. Press ⟨Alt⟩+⟨L⟩ to open the Reflection command line.

5. Type the following command, and press ⟨Enter ↵⟩:

    ```
    DisplayFile "C:\Program Files\Reflection\VT\Samples\Demo.pic"
    ```

(The Demo.pic file is only available if you performed a **Custom installation** when you ran the Setup program, and then selected the **Utility and Support Files** check box.)

The Tektronix graphics display is turned on and a chart is displayed in the terminal window.

Zooming and Scrolling a Tektronix Image

Go through the following steps to learn how to control the Tektronix display and move the cursor:

1. First, place a Tektronix image on the display (explained earlier).

2. Press ⟨Alt⟩+⟨Z⟩ to zoom the image. Because the larger Tektronix screen may not fit on the PC's screen, only a portion of the image will be shown.

    Each time you press ⟨Alt⟩+⟨Z⟩, the image alternates from scaled to unscaled.[*] This is equivalent to selecting and clearing the **Display unscaled** check box in the **Graphics** group box on the Screen tab of the Display Setup dialog box.

3. Move the cursor around on the unscaled image using either the scroll bars or by holding down the ⟨Ctrl⟩ key while pressing the arrow keys located on your PC's cursor pad (*not* the numeric keypad arrow keys).

    Use ⟨Ctrl⟩-(Cp)⟨PgUp⟩ and ⟨Ctrl⟩-(Cp)⟨PgDn⟩ to move the image up or down one screen at a time.

---

*   This keystroke is remappable; "TekZoom" is the Macro Function name.

4. Press `Esc`, and then press `Ctrl`+`Z` to turn on the crosshair cursor. You see a fine crosshair on the screen as you move the mouse in the terminal window.

5. The mouse and the keypad arrow keys move the crosshair cursor when it is showing. Practice moving the crosshair cursor around the screen using either of these methods.

6. To clear the image, press (Cp)`PgDn`. This keystroke is equivalent to the Tektronix terminal PAGE CLEAR function.

7. To go back online with the host:

   a. On the Setup menu, click Terminal and then click the Emulation tab.

   b. Select the **Online** check box.

   c. Click OK.

Tektronix Display Modes

During Tektronix emulation using Reflection for ReGIS Graphics, there are three main modes:

- Alphanumeric (or text) mode
- Graph mode, which is broken down into the following modes:
    - Graphics (for drawing vectors)
    - Point Plot
    - Special Point Plot
    - Incremental Plot
- Graphics INput cursor (GIN) mode

This chapter describes the Tektronix modes in detail.

Alphanumeric Mode

In Alphanumeric mode, also known simply as Alpha mode, characters are displayed as they are received from the host; this is the initial state of 4014 Tektronix emulation. Alpha mode displays 35 rows with 74 characters per row; the position of the alpha cursor shows the next character position. Alpha mode is entered upon receipt of one of the following characters: CR, US, or ESCUS.

There are several character sizes that can be displayed in Alpha mode. Because of limited display resolutions, the smaller character sizes may be difficult to read, however you can always zoom the display (press Alt+Z) to read even the smallest characters.

Alpha Mode Margins

There are two specific margins in Alpha mode.

Margin 0 is flush with the left margin of the screen. When the right margin is reached, a carriage return and linefeed are automatically generated and Margin 0 becomes the active margin. It can also be made active by:

- Pressing `Ctrl`+`F2`; the Reset terminal keystroke.

- Pressing (Cp)`PgDn`; the Tektronix PAGE CLEAR function.

- Receiving the sequence ^{E}S$_C$F$_F$.

When the 35th line of input is reached, the cursor is repositioned at the top row halfway across the screen. This is the *Margin 1* position, which is halfway in the middle of the screen.

Having both Margin 0 and Margin 1, known as a *dual-margin* feature, you can view two columns of text at one time.

Note that when entering or editing lines in Margin 0, it is possible to overwrite characters in Margin 1. Linefeeds past the last line while Margin 0 is the current margin will make Margin 1 the current margin. Similarly, linefeeds past the last line while Margin 1 is the current margin will make Margin 0 the current margin. This allows twice as many lines of information to be displayed before the screen is filled.

Graphics Mode

Graphics mode is used to draw vectors by giving the absolute coordinates of their endpoints. Graphics mode is entered from Alpha mode upon receipt of ^{G}S or ^{E}S$_C$G$_S$ from the host.

Determining Graphics Coordinates

In Graphics mode, characters received from the host are interpreted by Reflection as vector endpoints; after receipt of the first endpoint, which is considered a dark vector point (defined as an unwritten vector that always occurs after execution of the first vector to be received after a ^{G}S command),[*] lines are drawn between endpoints.

---

* If you want the first vector to appear, send a ^{B}E$_L$ character before any of the coordinates.

Each vector is defined by sending its endpoints as a sequence of four ASCII characters. This four-character sequence consists of the high and low-order portions of the Y coordinate, and the high and low-order portions of the X coordinate, in that order. Reflection supports the 4014 vector styles.

In Graphics mode there are 1024 by 1024 addressable points. Each point on the plane is called a *tekpoint*. The addressable range of the displayed plane is 0–4095x and 0–4095y. In unscaled mode, only 780 of these tekpoints can be seen on the vertical axis. The tekpoint in the lower-left hand corner has the address 0,0. A Tektronix terminal's screen only displays 3120 tekpoints vertically, therefore Reflection creates images on a 4096 by 3120 plane.

Several of the Graph modes must manipulate coordinates describing the position of endpoints. Each coordinate consists of a 12-bit x value and a 12-bit y value used to represent a number in the range of 0–4095. Each character (byte) contains 2 tag bits plus 5 data bits for either the high-order or low-order bits of the number. Each xy coordinate pair is encoded as five ASCII characters. These five characters, in the order they are transmitted, are called:

- High Y
- Extra
- Low Y
- High X
- Low X

If the coordinates in binary are:

```
xxxxx xxxxx xx
```

and:

```
yyyyy yyyyy yy
```

where x and y represent single bits of the 12 bit addresses, then this describes the five characters:

High Y   Contains the five most significant bits of y coordinate. The high order two bits are set to 01. The complete byte has the format:

```
01yyyyy
```

Extra   Contains the two low order bits of the x and the y coordinates. The high order three bits are set to 110; the next two bits contain the low order two bits of the y coordinate; the final two bits contain the low order two bits of the x coordinate. The complete byte has the format:

```
110yyxx
```

Low Y   Contains the five intermediate bits of the y coordinate. The high order two bits of Low Y are set to 11. The complete byte has the format:

```
11yyyyy
```

On some systems, the ^{D}EL character (7F hex) is used for padding. Since the Low Y byte may take on the 7F value as part of a coordinate specification, a conflict can arise; ^{D}EL characters intended as padding may be interpreted as coordinates. To ignore the ^{D}EL characters and treat them as pads, select **Ignored** from the **Del processing** list in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box.

High X   Contains the five most significant bits of the x coordinate. The high order two bits of the High X byte are set to 01. The complete byte has the format:

```
01xxxxx
```

Low X   Contains the five intermediate bits of the x coordinate. The high order two bits of Low X are set to 10. The complete byte has the format:

```
10xxxxx
```

Tutorial: Drawing a Vector

The graphic features of Reflection's Tektronix emulation are primarily for using graphics programs running on a host computer. The following section simply shows you how to enter Graphics mode and draw a vector on the screen, so you can better understand how Tektronix applications work.

To Enter Graphics Mode…

To manually begin Tektronix emulation and go "off line" to simulate the commands normally sent by the host:

1. On the Setup menu, click Terminal.

2. On the Terminal Type tab, select **401X** as the **Terminal type**.

3. On the Emulation tab, clear the **Online** check box.

4. Click OK.

    The mouse pointer changes to a crosshair when emulating a Tektronix terminal.

5. To enter Graphics mode, press Ctrl+] ; notice that the alpha cursor disappears. This transmits the Gs control character to Reflection, as shown in the table on page 244.

Now that Reflection is in graphics mode, it is ready to receive vector coordinates.

Establishing a Starting Point…

As described on page 195, an address is made up of four characters that direct the display writing beam. To start drawing from approximately the middle of your screen:

• While still in Graphics mode from the steps above, type the following characters (without spaces between them) in the order shown—note that the first character is a comma and the third character is the number zero:

    , f 0 @

    This sets the address of 390Y, 512X; how these values were calculated is explained on page 199.

After executing this command, you will not see any visible movement of the cursor. What you have actually just done is to execute an unwritten (dark) vector.

Writing a Point Vector...

To write a point (zero-length) vector:

• Type the @ symbol (that is, Shift + 2).

Again, you will not see any visible movement of the cursor on your display.

Drawing a Vector...

To draw a line (that is, a vector) on the screen:

1. Press Enter ↵ to see the alpha cursor. Notice how it is blinking halfway down your screen (this is where you placed it, two steps earlier).

2. Press Ctrl +] to send the group separator character to Reflection; the alpha cursor disappears.

3. Type the following characters in the order shown:

    f Spacebar @ f 0 @

    A vector is drawn to the center of the screen.

4. Any address different from the preceding one causes beam movement when an execution character is included. The beam will be turned on during movement to cause vector drawing, unless the command is immediately preceded by Gs.

    To draw another vector, type the following characters in the order shown—note that the second character is an accent (the unshifted state of the tilde ˜ key):

    Spacebar ` Spacebar   @

    A vector is drawn to the lower-left corner (address 0,0).

Ending Graphics Mode...

To exit Graphics mode and return to Alpha mode, press the following key on the keyboard's cursor pad (*not* the numeric keypad):

- Press (Cp)PgDn.

To Exit Tektronix Emulation...

To return to VT emulation and go back "on line" with the host:

1. On the Setup menu, click Terminal.

2. On the Terminal Type tab, select **VT400-7** as the **Terminal type**.

3. On the Emulation tab, select the **Online** check box.

4. Click OK.

The crosshair pointer changes back to a mouse pointer.

Calculating An Address

In the above tutorial, you typed the following characters to send the address 390Y, 512X:

```
, f 0 @
```

Following are the calculations used to derive this character sequence for the 390Y, 512X address.

To send the endpoint (X,Y) where X = 512 and Y = 390, use:

```
512 = 200 (Hex), 390 = 186h (Hex)
```

Reflection receives the endpoint as a sequence of 4 bytes in the order High Y, Low Y, High X, and Low X.

The Y coordinate is as follows:

186h	= 0110000110
High Y byte	= tag bits (01), 5 high-order bits of Y value
	= 01 01100
	= 2Ch
	= the , character is sent to the terminal
Low Y byte	= tag bits (11), 5 low-order bits of Y value
	= 11 00110
	= 66h
	= the f character is sent to the terminal

The X coordinate is as follows:

200h	= 1000000000
High X byte	= tag bits (01), 5 high-order bits of X value
	= 01 10000
	= 30h
	= the 0 character is sent to the terminal
Low X byte	= tag bits (10), 5 low-order bits of X value
	= 10 00000
	= 40h
	= the @ character is sent to the terminal

Subsequent Graphics Transmissions

Reflection remembers the last coordinate sent. Subsequent transmissions will not send bytes whose values have not changed with the following exceptions:

- The Low X byte will always be sent.

- If either the High X byte or the Extra byte is sent, then the Low Y byte must also be sent.

Point Plot Mode

Point Plot mode is identical to Graphics mode, except that lines are not drawn. Instead, only the endpoints are displayed. Dark vector is set upon entry to Point Plot mode.

Point Plot mode is entered when the host sends Reflection the FS control code.

Incremental Plot Mode

Incremental Plot mode emulates a plotter by responding to pen up, pen down, and direction motion commands. Each motion command moves one tekpoint in the given direction. The sequences for moving tekpoints are listed on page 208.

Incremental Plot mode is entered when the host sends Reflection either RS or ESCRS.

Special Point Plot Mode

A Tektronix terminal with Enhanced Graphics Module enters Special Point Plot mode upon receiving the escape sequence ESCFS from the host. Reflection, unlike the VT340, supports Special Point Plot mode commands. However, Reflection ignores the grey scaling intensity characters that precede each point address in this mode. Therefore, although Special Point Plot mode is supported, the results are identical to Point Plot mode.

GIN Mode

GIN (Graphics INput) mode allows the host to request the user to specify a screen location; you do this by moving the crosshair cursor to the desired screen position and clicking a mouse button or typing a key from the keyboard. This sends the crosshair cursor position report to the host.

GIN mode is entered when the host sends Reflection ESCSUB; upon receipt of this sequence, Reflection displays a crosshair cursor. Any data from the host is buffered until you return to Alpha mode.

To exit GIN mode:

- Click either mouse button or type any key on the keyboard that transmits a character.

  This sends the optional character, an optional terminator, the information listed above, and enters Alpha mode.

- The host can also send the sequence ᴱsᴄᴱɴǫ to exit GIN mode.

A cursor position report is four bytes long and does not include the extra byte which is usually part of an address. The position report consists of a High X byte, a Low X byte, a High Y byte, and a Low Y byte (described on page 195). Each byte contains five bits. The position report is contained to the range 0–1023.

The **GIN terminator** list in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box determines what characters are sent to terminate the position of the report. When set to **CR**, only a ᶜʀ is sent. When set to **None**, no terminating characters are sent. Choose the **CR-EOT** value to send a ᶜʀ and a Ctrl-D character (EOT).

Tektronix Escape Sequences

Reflection for ReGIS Graphics supports the full array of VT terminal control functions when emulating a VT terminal; the section starting on page 9 describes these functions in detail.

When emulating a Tektronix terminal, there are a unique subset of escape sequences supported, in addition to the Reflection-specific control functions listed on page 19. This chapter first describes the supported escape sequences, then the control characters used by Tektronix emulation.

Tektronix Escape Sequences

The following escape sequences are supported when running Reflection for ReGIS Graphics in Tektronix 401X mode:

ESC BEL	In Alpha mode, rings the bell and clears the bypass condition. In Graphics mode, if the dark vector condition is in effect, this condition is cleared—otherwise, this simply rings the bell.
ESC BS	In Alpha mode, moves the cursor left one position. If this action causes the cursor to move past the left margin, the cursor is then moved to the right margin on the same line.
ESC HT	In Alpha mode, moves the cursor to the right one character.
ESC LF	Ignore receipt of LF characters until a BEL character is received.
ESC VT	In Alpha mode, moves the cursor up one line. The cursor is not moved if it is already at the top margin.
ESC FF	Enters Alpha mode, clears the bypass condition, and erases the screen; positioning the cursor in the upper-left hand corner.
ESC CR	Ignores receipt of CR characters until a BEL character is received.
ESC ETB	Performs a graphics print and clears the bypass condition.
ESC CAN	Sets the bypass condition.
ESC SUB	Enters GIN mode and sets the bypass condition.

ESC FS	Enters Special Point Plot mode.
ESC GS	Enters Graphics mode, and begins the first dark vector point.
ESC RS	Enters Incremental Plot mode.
ESC US	Enters Alpha mode, clears the bypass condition, and positions the cursor at the last beam drawing position.
ESC /0d	Raster writing feature;* sets the dots to on, or normal (VT340 Tektronix 4105 extensions).
ESC /1d	Raster writing feature;* sets the dots to off, or erase (VT340 Tektronix 4105 extensions). **Note:** This sequence can be used in alphanumeric or Graphics mode to simulate the unsupported Tektronix write-through feature.
ESC /2d	Raster writing feature;[a] complement dots (VT340 Tektronix 4105 extensions).
ESC ?	This sequence is interpreted as if a DEL was received; this is useful when the **Del processing** list in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box) is set to **Ignored**.
CSI ?38h	Enter Tektronix mode (VT340).
CSI ?38l	Exit Tektronix mode (VT340).

a. The 4010 and 4014 Tektronix terminals have a "write-through" mode. A VT340 terminal and Reflection do not support this mode. Instead, VT340 and Reflection have raster writing modes.

Line Style Escape Sequences

Reflection maps the line styles listed below for 4014 Tektronix emulation to the closest available Windows line style, as noted. (Line styles supported by the 4105 terminal are listed on page 210.)

ESC `	Solid line style.
ESC a	Dotted line style.
ESC b	Dot-dash line style.
ESC c	Short dash line style.
ESC d	Long dash line style.
ESC e	Dot-dot-dash line style (the VT340 terminal draws a solid line).
ESC f	Long/short dash line style (the VT340 terminal draws a solid line).
ESC g	Medium line style (the VT340 terminal draws a solid line).
ESC h	Bold Solid line style.
ESC i	Bold Dotted line style.
ESC j	Bold Dot-dash line style.
ESC k	Bold Short line style.
ESC l	Bold Long line style.
ESC m	Bold Dot-dot-dash line style.
ESC n	Bold Long/short dash line style.
ESC o	Bold Medium dash line style.

Vector Character Size Escape Sequences

ESC 8	Select character size: 56 (x) by 88 (y) tekpoints.
ESC 9	Select character size: 51 (x) by 82 (y) tekpoints.
ESC :	Select character size: 34 (x) by 53 (y) tekpoints.
ESC ;	Select character size: 31 (x) by 48 (y) tekpoints.

Tektronix ASCII Control Codes

The following control codes are used by Reflection when running in Tektronix 401X mode:

B_{EL} In Alpha mode, rings the bell and clears the bypass condition (defined on page 209). In Graphics mode, if the dark vector condition is in effect, this condition is cleared—otherwise, this simply rings the bell.

B_S In Alpha mode, moves the cursor left one position. If this action causes the cursor to move past the left margin, the cursor is then moved to the right margin on the same line.

C_R In Alpha mode, moves the cursor to the current margin; either the left edge of the display if the current margin is Margin 0, or the center of the display for Margin 1 (see page 194 for more information). This also cancels the crosshair cursor and clears the bypass condition. If the **CR processing** list in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box is set to **CR-LF**, then a linefeed is also performed.

In Graphics and Incremental Plot mode, C_R enters Alpha mode. The lower-left edge of the cursor is positioned at the y beam coordinate and current margin.

D_{EL} In Graphics mode, D_{EL} is interpreted as Low Y (explained on page 195). To ignore the D_{EL} characters and treat them as pads, choose **Ignored** from the **Del processing** list in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box.

F_S Enters Point Plot mode.

G_S Enters Graphics mode, and begins the first dark vector point. (If you already were in Graphics mode, then this just sets the dark vector condition.)

H_T In Alpha mode, moves the cursor to the right one character; this can also be performed using Tab from the keyboard. If this action causes the cursor to move past the right margin, the cursor is then moved to the left margin on the next line ($C_R L_F$).

L_F   In Alpha mode, moves the cursor down one line and clears the bypass
             condition. If the **LF processing** list in the **Tektronix** group box on the
             Graphics tab in the Terminal Setup dialog box is set to **LF-CR**, then a
             carriage return is also performed. When L_F moves the cursor past the
             bottom margin, the current margin is adjusted and the cursor is moved
             to the top of the new current margin. The cursor maintains the same
             horizontal position with respect to the new margin as it had with respect to
             the old.

             In Graphics mode, L_F causes the beam drawing position to move down by
             one character space. The distance depends on the character's size. The
             position will wrap to the top edge of the screen if L_F would move it off the
             bottom edge.

R_S   Enters Incremental Plot mode.

S_I   Clears the bypass condition.

S_P   In Alpha mode, moves the cursor one space to the right. If the effect of S_P
             moves the cursor past the right margin, a carriage return and linefeed
             are executed.

             In Incremental Plot mode, turns the beam off. The current position can
             be moved but the motion is not displayed on the screen until a beam on is
             received.

R_S   Enters Alpha mode, clears the bypass condition, and positions the cursor
             at the last beam drawing position.

V_T   In Alpha mode, moves the cursor up one line. The cursor is not moved if
             it is already at the top margin.

! through ~ In Alpha mode, displays the printing character, then moves one space
             to the right. If this would move the cursor past the right margin, a
             carriage return and linefeed are executed.

` to ~   In Graphics mode, all codes in this range correspond to a Low Y or Extra
             byte address (explained on 195).

S_P to !   In Graphics mode, all codes in this range correspond to a High X or a
             High Y address.

The following sequences only apply when Reflection is in Increment Plot mode (explained on page 201):

A   Move one tekpoint right.

B   Move one tekpoint left.

D   Move one tekpoint up.

E   Move one tekpoint up and right.

F   Move one tekpoint up and left.

H   Move one tekpoint down.

I   Move one tekpoint down and right.

J   Move one tekpoint down and left.

P   Turn on the beam. Once the beam is on, subsequent moves causes visible vectors.

SP   Turn off the beam.

Obtaining Status Information

When the host computer sends an ^{ESC}^{ENQ} sequence, Reflection responds in the following manner:

Alpha Mode Sets the bypass condition, and returns one byte of terminal status information followed by the four-byte address of the lower-left corner of the cursor followed by the GIN terminator (this option is set to **None** in the **Tektronix** group box on the Graphics tab of the Terminal Setup dialog box by default).

Graphics Mode Sets the bypass condition, and returns one byte of terminal status information followed by the four-byte address of the Graphics mode beam position followed by the GIN terminator.

GIN Mode Sets the bypass condition, and returns four-byte address of the crosshair cursor position followed by the GIN terminator.

The first byte transmitted to the host in response to an $^{E}S_{C}$$^{E}N_{Q}$ terminal status request is a status byte with the following format:

Bit 7   Parity bit
Bit 6   Always 0
Bit 5   Always 1
Bit 4   1 indicates a configured printer
Bit 3   0 in alpha or GIN mode, 1 in Graphics mode
Bit 2   0 in Graphics mode, 1 in alpha or GIN mode
Bit 1   1 if margin 1 is set (that is, alpha cursor is in the right half of the screen)
Bit 0   Always 1

Bypass Condition

Any one of the above modes (Alpha, Graphics, and GIN) indicates that Reflection is in bypass condition. Bypass prevents Reflection from processing or displaying any GIN mode data sent to or echoed from the host. If the GIN terminator contains a ^{C}R character, bypass is cleared when ^{C}R is echoed from the host.

If the GIN terminator does not contain a ^{C}R, then bypass must be cleared by the host sending (or echoing) any of the following ASCII characters:

BEL	$^{B}E_{L}$	Ctrl + G
CR	^{C}R	Enter ↵
ESC ETB	$^{E}S_{C}$$^{E}T_{B}$	Ctrl + W
ESC FF	$^{E}S_{C}$$^{F}F$	Ctrl + L
LF	^{L}F	Ctrl + J
US	^{R}S	Ctrl + Shift + -
SI	^{S}I	Ctrl + 0

4105 Tektronix Escape Sequences

The following escape sequences are provided for support with the 4105 Tektronix terminal.

4105 Color Escape Sequences

$^{E}S_{C}ML$<n>   4105 color index escape sequences, where <n> is a value from the table below.

$^{E}S_{C}MT$<n>   4105 color index escape sequences, where <n> is a value from the table below.

ASCII <n> Value	Keystroke	Definition	Color
0	Ctrl+@	Null	Black
1	Ctrl+A	Start of header	Bold white
2	Ctrl+B	Start of text	Bold red
3	Ctrl+C	End of text	Bold green
4	Ctrl+D	End of transmission	Bold blue
5*	Ctrl+E	^{E}Q	Bold cyan
6	Ctrl+F	^{A}K	Bold magenta
7*	Ctrl+G	^{B}L	Bold yellow
8*	Ctrl+H	^{B}S	Yellow
9*	Ctrl+I	^{H}T	Green
10*	Ctrl+J	^{L}F	Cyan
11*	Ctrl+K	^{V}T	Blue
12*	Ctrl+L	^{F}F	Magenta
13*	Ctrl+M	^{C}R	Red
14*	Ctrl+N	^{S}O	Gray
15*	Ctrl+O	^{S}I	White

4105 Line Style Escape Sequences

^{E}SCMV0   Solid line style.

^{E}SCMV1   Dotted line style.

^{E}SCMV2   Dot-dash line style.

^{E}SCMV3   Short dash line style; this is mapped in Windows to dash.

^{E}SCMV4   Long dash line style; this is mapped in Windows to dash.

^{E}SCMV5   Dot-dash-dot line style.

^{E}SCMV6   Long/short line style; this is mapped in Windows to dash-dot.

^{E}SCMV7   Medium dash line style.

SECTION **5**

DG Control Sequences

DG Control Sequences

This chapter details how to use Reflection for UNIX and Digital and Reflection for ReGIS Graphics to emulate the Data General 215 terminal.

Configuring Reflection for a DG Terminal

To configure Reflection for DG emulation:

1. From the Setup menu, click Terminal, then click the Terminal Type tab.

2. Select the **DG 215** terminal type in the **Terminal type** group box.

3. Click OK.

DG Terminal Escape Sequences

Control Sequences	Description
ctrl A	print form, from start of cursor line to end of page
ctrl B	reverse attribute off
ctrl C	enable blink attribute
ctrl D	disable blink attribute
ctrl E	read window address
ctrl G	sound bell
ctrl H	window home
ctrl J	new line
ctrl K	erase eol
ctrl L	erase window
ctrl M	carriage return
ctrl N	blink attribute on

Control Sequences	Description
`ctrl O`	blink attribute off
`ctrl P <col><row>`	write cursor address
`ctrl Q`	print window
`ctrl R`	roll enable
`ctrl S`	roll disable
`ctrl T`	underscore attribute on
`ctrl U`	underscore attribute off
`ctrl V`	reverse attribute on
`ctrl W`	cursor up
`ctrl X`	cursor right
`ctrl Y`	cursor left
`ctrl Z`	cursor down
`ctrl \`	dim attribute on
`ctrl ]`	dim attribute off

Escape Sequences	Description
`ctrl ^ C`	read model id
`ctrl ^ E`	reverse video off
`ctrl ^ D`	reverse video on
`ctrl ^ O`	shift in, selects primary character set
`ctrl ^ N`	shift out, selects secondary character set
`ctrl ^ F@`	select ANSI mode
`ctrl ^ FU0`	select 7 bit transmission
`ctrl ^ FU1`	select 8 bit transmission

WYSE Escape Sequences

WYSE Escape Sequences

This chapter details how to use Reflection for UNIX and Digital and Reflection for ReGIS Graphics to emulate the WYSE 50 or 60 terminals.

Configuring Reflection for the WYSE Terminal

To configure Reflection for WYSE emulation:

1. From the Setup menu, click Terminal, then click the Terminal Type tab.

2. Select a terminal type in the **Terminal type** group box:

   - **WYSE 50**+: Emulate the WYSE 50 terminal.

   - **WYSE 60**: Emulate the WYSE 60 terminal.

3. Click OK.

WYSE Terminal Escape Sequences

Escape Sequence	Description	Wyse Emulation Type	
ctrl Q	Enable Transmission		60
ctrl S	Stop Transmission		60
ctrl E	Send Ack (Answerback)	50	60
esc e 6	Ack Mode Off		60
esc e 7	Ack Mode On		60
esc C esc D F	Full-duplex Mode On	50	60
esc C esc D H	Half-duplex Mode On	50	60
esc B	Block Mode On	50	60
esc C	Block Mode Off	50	60
esc e 8	Data/Printer = Modem/Aux		60

Escape Sequence	Description	Wyse Emulation Type	
esc e 9	Data/Printer = Aux/Modem		60
esc c 0 <bd><st><pr><wd>	Set Modem Port Parameters		60
esc c 1 <bd><st><pr><wd>	Set Aux Port Parameters		60
esc c 2 <handshake>	Set Modem Port Rcv Handshake		60
esc c 3 <handshake>	Set Aux Port Rcv Handshake		60
esc c 4 <handshake>	Set Modem Port Xmt Handshake		60
esc c 5 <handshake>	Set Aux Port Xmt Handshake		60
esc c 6 <max>	Set Maximum Transmission		60
esc SPACE	Send Terminal ID	50	60
esc c ; <answer> ^Y	Program Answerback Message		60
esc c <	Send Answerback Message		60
esc c =	Conceal Answerback Message		60
esc e SPACE	Answerback Mode Off		60
esc e !	Answerback Mode On		60
esc c 8 <hh><mm>	Load Time of Day		60
esc e R	Ignore Received Nulls		60
esc e S	Accept Received Nulls		60
esc U	Monitor Mode On	50	60
esc u	Monitor Mode Off	50	60
esc X	Monitor Mode Off	50	60
esc k	Local Edit Mode On	50	60
esc l	Local Edit Mode Off	50	60
ctrl G	Sound Bell	50	60
ctrl N	Unlock Keyboard	50	60

Escape Sequence	Description	Wyse Emulation Type	
esc "	Unlock Keyboard	50	60
ctrl O	Lock Keyboard	50	60
esc #	Lock Keyboard	50	60
esc e $	Keyclick Off		60
esc e %	Keyclick On		60
esc e &	Caps Lock On		60
esc e '	Caps Lock Off		60
esc e L	Margin Bell Off		60
esc e M	Margin Bell On		60
esc ` J	Set Margin Bell at Cursor		60
esc e ,	Key Repeat Off		60
esc e -	Key Repeat On		60
esc e T	Caps Lock = Caps		60
esc e U	Caps Lock = Reverse		60
esc c T	Default all Modes		60
esc c V	Save Modes to NVR		60
esc c W	Save Modes and Tabs to NVR		60
esc c X	Power-on Reset		60
esc ~ /	Wyseword Mode On		60
esc ~ .	Wyseword Mode Off		60
esc ~ 2	Application Key Mode On		60
esc ~ 3	Application Key Mode Off		60
esc z <fkey><sequence> DEL	Program Function Key	50	60
esc z <fkey> DEL	Clear Function Key	50	60

Escape Sequence	Description	Wyse Emulation Type	
esc Z <dir><key><seq> DEL	Program Key Direction and Def		60
esc Z <dir><key> DEL	Clear Key Definition		60
esc Z ~ <key>	Read Key Direction and Def		60
esc c U	Clear all Redefinable Keys		60
esc c 7 <max>	Set Maximum Fkey Xmt Speed		60
esc ` 8	Screen Off		60
esc ` 9	Screen On		60
esc e P	Screen Saver Off		60
esc e Q	Screen Saver On		60
esc ^ 1	Reverse Screen		60
esc ^ 0	Normal Screen		60
esc ` <scroll>	Set Scrolling Speed/Type		60
esc ` <cursor>	Set Cursor Display Features		60
esc ` 0	Cursor Display Off		60
esc ` 1	Cursor Display On		60
esc ` a	Editing Status Line On		60
esc ` b	Standard Status Line On		60
esc ` c	Status Line Off		60
esc F <message> CR	Program/Display Msg on Status	50	60
esc F CR	Clear Message on Status Line	50	60
esc z (<text> CR	Program Unshifted Label Line	50	60
esc z) <text> CR	Program Shifted Label Line	50	60
esc z (CR	Erase Unshifted Label Line	50	60
esc z) CR	Erase Shifted Label Line	50	60

Escape Sequence	Description	Wyse Emulation Type	
esc z <field><label> CR	Program Function Key Label	50	60
esc z <field> CR	Erase Function Key Label	50	60
esc z DEL	Shifted Label Line Off	50	60
esc f	Message to User Line		60
esc ` :	80 Column Display		60
esc ` ;	132 Column Display		60
esc e F	Economy 80 Col Mode Off		60
esc e G	Economy 80 Col Mode On		60
esc e .	Width Change Clear Off		60
esc e /	Width Change Clear On		60
esc e (	24 Data Lines		60
esc e)	25 Data Lines		60
esc e *	42 Data Lines		60
esc e +	43 Data Lines		60
esc w <length>	Divide Memory into Pages		60
esc w B	Display Previous Page		60
esc J	Display Previous Page	50	60
esc w C	Display Next Page		60
esc K	Display Next Page	50	60
esc w <page>	Display Specified Page		60
esc x A <line>	Split Screen Horz. (Simple)		60
esc x 1 <line>	Split Screen Horz. Clr(Simpl)		60
esc x C <line>	Split Screen Horz. (Adjust)		60
esc x 3 <line>	Split Screen Horz. Clr(Adj.)		60

Escape Sequence	Description	Wyse Emulation Type	
esc]	Activate Upper Window		60
esc }	Activate Lower Window		60
esc J	Activate Other Window		60
esc K	Activate Other Window		60
esc x P	Lower Horizontal Split		60
esc x R	Raise Horizontal Split		60
esc w E	Roll Window Up in Page		60
esc w F	Roll Window Down in Page		60
esc x @	Redefine Screen as 1 Window		60
esc x 0	Redefine Screen 1 Window Clr		60
esc d (	Autodrag Cursor Off		60
esc d)	Autodrag Cursor On		60
esc w D	Reposition Workspace		60
esc A <mf><attr>	Set Attribute to Message	50	60
esc G <attr>	Set Character Display attribute	50	60
esc e 0	Character Attribute Mode Off		60
esc e 1	Character Attribute Mode On		60
esc e 2	Page Attribute Mode On		60
esc e 3	Line Attribute Mode On		60
esc ` <wpca>	Assign Write Protect Attrib		60
esc ! <attr>	Clear Unprotected Page to attribute	50	
esc G <lattr>	Assign Line Attribute		60
esc (	Write-protect Mode Off	50	60
esc)	Write-protect Mode On	50	60

Escape Sequence	Description	Wyse Emulation Type	
esc V	Clr Cursor Col to write protect space	50	60
esc '	Protect Mode Off	50	60
esc &	Protect Mode On	50	60
esc H <ldraw>	Display Graphic Character	50	60
esc H ctrl B	Line Draw Graphics Mode On	50	60
esc H ctrl C	Line Draw Graphics Mode Off	50	60
ctrl H	Cursor Left (Backspace)	50	60
ctrl L	Cursor Right	50	60
ctrl K	Cursor Up; No Scroll	50	60
esc j	Cursor Up; Scroll (Revers Lf)		60
ctrl V	Cursor Down; No Scroll	50	60
ctrl J	Cursor Down; Scroll	50	60
ctrl M	Cursor to Start of Line	50	60
ctrl _	Cursor to Start of Next Line	50	60
esc {	Home Cursor		60
ctrl ^	Home Cursor		60
esc _ <col>	Cursor to Specific Column		60
esc [<line>	Cursor to Specific Line		60
esc d .	End-of-Line Wrap Off		60
esc d /	End-of-Line Wrap On		60
esc e 4	Received CR = CR		60
esc e 5	Received CR = CRLF		60
esc d *	Autopage Mode Off		60
esc d +	Autopage Mode On		60

Escape Sequence	Description	Wyse Emulation Type	
`esc O`	Autoscroll Mode On	50	60
`esc N`	Autoscroll Mode Off	50	60
`esc ` H`	Line Lock Mode On		60
`esc ` I`	Line Lock Mode Off		60
`esc = <line><col>`	Cursor Address 80 Col Page	50	60
`esc w @<pg><ln><col>`	Cursor Add Specific 80 column page		60
`esc - <wnd/pg><ln><col>`	Address Specific 80 Column window/page	50	60
`esc a <lll>R<ccc>C`	Address 80/120 Col Page	50	60
`esc ?`	Read Cursor Address 80 Col		60
`esc w ``	Read 80 Col Pg and Address		60
`esc /`	Read 80 Col Window/Page & address	60	
`esc b`	Read Address 80/132 Column page	50	60
`esc 0`	Clear all Tab Stops	50	60
`esc 1`	Set Tab Stop at Cursor	50	60
`esc 2`	Clear Tab Stop at Cursor	50	60
`esc i`	Tab Cursor	50	60
`ctrl I`	Tab Cursor	50	60
`esc I`	Backtab	50	60
`esc q`	Insert Mode On, Replace Off	50	60
`esc r`	Replace Mode On, Insert Off	50	60
`esc e "`	Page Edit Mode Off		60
`esc e #`	Page Edit Mode On		60
`esc Q`	Insert Space Character	50	60

Escape Sequence	Description	Wyse Emulation Type	
esc E	Insert Line of Spaces	50	60
esc c M	Insert Column of Nulls		60
esc W	Delete Cursor Character	50	60
esc R	Delete Cursor Line	50	60
esc c J	Delete Cursor Column		60
esc *	Clear Page to Nulls	50	60
esc +	Clear Page to Spaces	50	60
esc ,	Clear Page to Write-prot space	50	60
esc ;	Clr Unprot Page to Spaces		60
ctrl Z	Clr Unprot Page to Spaces	50	60
esc :	Clr Unprot Page to Nulls	50	60
esc . <char>	Clr Unprot Page to Character	50	60
esc Y	Clr Unprot Pg to Sp from cursor	50	60
esc y	Clr Unprot Pg to Nul from cursor	50	60
esc c P	Clr Unprot Pg Foreground to space		60
esc c Q	Clr Unprot Pg Foreground to Nul		60
esc T	Clr Unprot Ln to Sp from Cur	50	60
esc t	Clr Unprot Ln to Nul from cursor	50	60
esc c O	Clr Unprot for Curs to EOL sp		60
esc c L	Clr Unprot for Curs to EOL Nul		60
esc c R	Clr Unprot Ln Fore to Sp Cursor		60
esc c S	Clr Unprot Ln Fore to Nul cursor		60
esc c I <char>	Clr Unprot Col to Character		60
esc c K	Clr Unprot Col to Nulls		60

Escape Sequence	Description	Wyse Emulation Type	
`esc c N <width><height>`	Draw Box Right of Cursor		60
`esc c G <line><col>`	Draw Box 80 Col Page		60
`esc c G <line>~<col>`	Draw Box 132 Col Page		60
`esc c F <line><col><char>`	Fill Unprot with Char 80 Col		60
`esc c F <line>~<col><char>`	Fill Unprot with Char 132Col		60
`esc c H <line><col><char>`	Fill Entire with Char 80 Col		60
`esc c H <line>~<col><char>`	Fill Entire with Char 132Col		60
`esc d '`	Begin Send/Print at Top of page		60
`esc d &`	Begin Send/Print at Top of screen		60
`esc M`	Send Cursor Character	50	60
`esc 6`	Send Cursor Line	50	60
`esc 4`	Send Unprotected Cursor Line	50	60
`esc 7`	Send Page	50	60
`esc 5`	Send Unprotected Page	50	60
`esc 8`	Mark Block Beginning	50	60
`esc 9`	Mark Block Ending	50	60
`esc s`	Send Entire Block	50	60
`esc S`	Send Unprotected Block	50	60
`esc P`	Print Formatted Page to Cursr	50	60
`esc @`	Print Format Unprot Page to cursor50	60	
`esc p`	Print Unformatted Page to cursor	50	60
`esc L`	Print Unformatted Page to	50	60
`ctrl T`	Aux/Transparent Print Off	50	60

Escape Sequence	Description	Wyse Emulation Type	
ctrl R	Aux Print Mode On	50	60
ctrl X	Transparent Print Mode On	50	60
esc d #	Transparent Print Mode On		60
esc d SPACE	Aux Receive Mode Off		60
esc d !	Aux Receive Mode On		60
esc d $	Bidirectional Mode Off		60
esc d %	Bidirectional Mode On		60
esc c D	Select Primary Character Set		60
esc c E	Select Secondary Character Set		60
esc c B <bank>	Define Primary Character Set		60
esc c C <bank>	Define Secondary Character Set		60
esc c @ <bank><Set>	Load Font Bank with Character Set		60
esc c ? <bank>	Clear Font Bank		60
esc c A >bank><pp><b..b>^Y	Define and Load Soft Font		60
esc e N	Auto Font Load Off		60
esc e O	Auto Font Load On		60

SECTION **7**

Unisys A Series T27 Escape Sequences

Unisys T27 Escape Sequences

This chapter details how to use Reflection for UNIX and Digital and Reflection for ReGIS Graphics to emulate the Unisys A Series T27 terminal.

Configuring Reflection for the T27 Terminal

To configure Reflection for T27 emulation:

1. From the Setup menu, click Terminal, then click the Terminal Type tab.

2. Select the **UNISYS T27** terminal type in the **Terminal type** group box.

3. Click OK.

Unisys T27 Terminal Escape Sequences

Escape Sequences	Description
`<ESC>W`	Set forms mode
`<ESC>X`	Reset forms mode
`<DC2>`	Toggle forms mode
`<SOH>`	Exit forms mode
`<ESC>E`	Set search mode
`<ESC>-<x>`	Set search character (x < 0x80)
`<ESC>-` `<ESC><SO><z><ESC><SI>`	Set search character (z >= 0x80)
`<ESC>F`	Reset search mode
`<LF>`	Move pointer down
`<DC3>`	Move pointer up
`<ESC>C`	Move pointer right
`<DC2>`	Move pointer right
`<BS>`	Move pointer left (backspace)
`<ESC>&`	Align kbc to dcp
`<HT>`	Tab right
`<ESC>#`	Clear variable tabs
`<VT>`	Vertical tab down
`<DC4>`	Home
`<ESC>$<n>`	Jump to page n
`<ESC>^<hh><kk>`	Position pointer
`<ESC>"<c><r>`	Position pointer
`<CR>`	Carriage return
`<LF>`	Carriage return

Escape Sequences	Description
<ESC>6	Prevent align in cursor page
<ESC>!	Insert character by line
<ESC>@	Insert character by page
<ESC>%	Delete character by line
<ESC>M	Line delete
<ESC>P	Delete character by page
<ESC>L	Line insert
<ESC>>	Move line up
<ESC><	Move line down
<ESC>K	Clear to end of line
<DC1>	Clear to end of line
<ESC>J	Clear to end of page
<SOH>	Clear page
<ESC>S	Roll page up
<ESC>T	Roll page down
<ESC>]	Print all
<ESC>;	Print all with form feed
<ESC>:	Print unprotected data
<ESC>)1	Query printer status
<FF>	Clear page
<ESC><FF>	Clear page
<ETB>	End highlight
<CAN>	Start video blink
<EM>	Start secure video

Escape Sequences	Description
`<SUB>`	Start bright video
`<ESC>[<n>a`	Printer space compression
`<ESC>3`	Underline video highlight
`<ESC>4`	Reverse video highlight
`<ESC>N`	Set reverse video
`<ESC>O`	Set normal video
`<ESC>Raaaaaccdd..dd`	Store ASCII codes as data
`<ESC>RBmmmmnnnnppp`	Configure data comm, screen and kpt buffers
`<ESC>RC`	Reconfigure terminal
`<ESC>RDhhpp..pp00iiqq..qq00nnzz..zz0000`	Selective key programming
`<ESC>Rhaaaacchhhh..hh`	Store data in hex codes
`<ESC>RKppphhkk..kkApiiqq..qqA9nnzz..zzA9A9`	Program function keys f1..f10
`<ESC>RL`	Transmit error log
`<ESC>RP`	Copy temporary storage into nv ram
`<ESC>RSccdd..dd`	Display message in environmental user status line
`<ESC>Rtaaaacc`	Transmit memory contents to host
`<ESC>`	Escape
`<ESC><SO>`	Set shift out
`<SO>`	Set shift out
`<ESC><SI>`	Set shift in
`<SI>`	Set shift in
`<ESC>'x`	Character translation
`<ESC>=`	Reset keystroke lockout

Escape Sequences	Description
`<ESC>.`	Toggle variable tabs
`<VT>`	Toggle variable tabs
`<ESC>D`	Set mobile home
`<ESC>Y`	Lower case disable
`<ESC>Z`	Lower case enable
`<ESC><space>E`	Continuous confidence test
`<ESC><space>F`	Printer interface test
`<ESC><space>V`	Display firmware version
`<ESC><space>C`	Display character set
`<ESC><space>D`	Load contents of permanent storage and reconfigure
`<ESC>_<x>`	Fill with x
`<ESC>(`	Transmit terminal screen to host
`<ESC>?`	Toggle audible alarm
`<BEL>`	Toggle audible alarm
`<ETX>`	End of text processing
`<EOT>`	End of transmission

Characters and Character Sets

Introduction to Characters and Character Sets

Reflection supports the following VT character sets:

- ASCII; see page 250

- DEC Supplemental Graphic; see page 252

- ISO Latin-1 supplemental graphic; see page 253

- National replacement sets (12); see 267

- DEC Special Graphic; see page 254

- DEC Technical; see page 255

- DRCS; see 78

This section describes how to enter characters in both 7- and 8-bit environments, then shows a figure of each character set.

Supported Character Sets

Reflection supports the character sets listed above. All characters in each set are represented by 8-bit binary codes. They include displayable characters, such as letters and numbers, and control characters, such as carriage return (C_R) and linefeed (L_F).

The number of characters available at any one time depends on the type of computing environment:

- In a 7-bit environment, only the last 7 bits of the character code define the character (the 8th bit is typically a parity bit used for error checking). A total of 128 characters is available.

- In an 8-bit environment, all 8 bits define the character. A total of 256 characters is available.

When characters are displayed on screen, they are retrieved from an *in-use* table. The in-use table stores two character sets at a time, defining the entire set of characters available for immediate use.

The in-use table consists of a graphic left (GL) table, where characters are retrieved by 7-bit codes, and a graphic right (GR) table, where characters are retrieved by 8-bit codes. In a 7-bit environment, only the GL table (and therefore only 128 characters) is accessible. In an 8-bit environment, both the GL and GR tables are accessible.

Together, the ASCII character set and the DEC Supplemental Graphic set comprise the *DEC Multinational* character set. The ASCII set and the ISO Latin-1 supplemental character set comprise the *ISO Latin Alphabet Number 1* character set. The two sets have only a few differences, as shown in the character charts in this section.

These charts show each character's position by decimal and hexadecimal value. Values are given based on the character set's most common placement in the in-use table. For example, the DEC Special Graphic character set would typically be placed into GR, so the values given for this set are those of the GR table. If DEC Special Graphic were placed in GL, each character would have a decimal value 128 less than that shown (or less 80 in hexadecimal).

For more information about character sets, and how to map different sets into the in-use table, see page 68.

The Host Character Set vs. Windows Characters

Because Reflection is designed to be used as a DEC terminal, it uses the ASCII character set for the most common characters (such as letters and numbers) and, by default, the DEC Supplemental Graphic character set for national and special characters. Microsoft Windows, on the other hand, uses 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.[*]

---

[*] When you select ISO Latin-1 as your UPS Set, there is no difference at all between the upper range of character values; they are identical to the characters used by the ANSI character set.

Even when identical characters exist in both character sets, they are often repre-
sented by different numbers. For example, in the DEC Supplemental Graphic
character set (page 252), the y with a diaeresis (ÿ) is represented by decimal 253,
while in ANSI it is represented by decimal 255. When you use the [Alt] key method
(explained on page 257) 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 space.

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

• Entering special characters in a host application, explained on page 257.

• Translating characters when writing and reading ASCII disk files, explained on
page 265.

• Entering foreign characters in a file name when transferring files to a UNIX
host.

• Copying textual information from Reflection's terminal or command 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 _.)

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 **Display** is selected in the
Control characters group box on the Screen tab of the Display Setup dialog box,
control characters appear as a two-letter mnemonic.

Reflection recognizes the following codes denoted by an asterisk (*):

ASCII Decimal	Keystroke	Display	Definition
0	Ctrl+@	N_U	Null
1	Ctrl+A	S_H	Start of header
2	Ctrl+B	S_X	Start of text
3	Ctrl+C	E_X	End of text
4	Ctrl+D	E_T	End of transmission
5*	Ctrl+E	E_Q	Enquiry
6	Ctrl+F	A_K	Acknowledge
7*	Ctrl+G	B_L	Bell
8*	Ctrl+H	B_S	Backspace
9*	Ctrl+I	H_T	Horizontal tab
10*	Ctrl+J	L_F	Linefeed
11*	Ctrl+K	V_T	Vertical tab
12*	Ctrl+L	F_F	Form feed
13*	Ctrl+M	C_R	Carriage return
14*	Ctrl+N	S_O	Shift out
15*	Ctrl+O	S_I	Shift in
16	Ctrl+P	D_L	Data link escape
17*	Ctrl+Q	D_1	Device control 1 (XON)
18	Ctrl+R	D_2	Device control 2
19*	Ctrl+S	D_3	Device control 3 (XOFF)
20	Ctrl+T	D_4	Device control 4
21	Ctrl+U	N_K	Negative acknowledge

ASCII Decimal	Keystroke	Display	Definition
22	Ctrl+V	S_Y	Synchronous idle
23	Ctrl+W	E_B	End of transmission block
24*	Ctrl+X	C_N	Cancel
25	Ctrl+Y	E_M	End of medium
26	Ctrl+Z	S_B	Substitute
27*	Ctrl+[	E_C	Escape
28	Ctrl+\	F_S	Field separator
29	Ctrl+]	G_S	Group separator
30	Ctrl+Shift+6	R_S	Record separator
31	Ctrl+Shift+-	U_S	Unit separator
32	Ctrl+Shift+2		Space (blank)

ASCII Decimal	Display	7-bit Equiv.	Definition
128	8_0		Ignored
129	8_1		Ignored
130	8_2		Ignored
131	8_3		Ignored
132*	I_N	E_{SC} D	Index
133*	N_{EL}	E_{SC} E	Next line
134	S_S		Start selected area
135	E_S		End selected area
136*	H_S	E_{SC} H	Horizontal tab set
137	H_J		Horizontal tab with justification

ASCII Decimal	Display	7-bit Equiv.	Definition
138	V$_S$		Vertical tab set
139	P$_D$		Partial line down
140	P$_U$		Partial line up
141*	R$_I$	^{E}S$_C$ M	Reverse index
142*	S$_2$	^{E}S$_C$ N	Single shift 2
143*	S$_3$	^{E}S$_C$ O	Single shift 3
144*	D$_C$	^{E}S$_C$ P	Device control string
145	P$_1$		Private use 1
146	P$_2$		Private use 2
147	S$_E$		Ignored
148	C$_C$		Cancel character
149	M$_W$		Message waiting
150	S$_P$		Start protected area
151	E$_P$		End protected area
152	9$_8$		Ignored
153	9$_9$		Ignored
154	9$_A$		Ignored
155*	C$_S$	^{E}S$_C$ [	Control sequence introducer
156*	S$_T$	^{E}S$_C$ \	String terminator
157	O$_S$		Operating system command
158	P$_M$		Privacy message
159	A$_P$		Application program command

Some functions still work when you display the control codes:

- ^{L}F, ^{F}F, and ^{V}T cause a carriage return and linefeed that move the cursor to a new line.

  Reflection displays the character before performing the new line function.

- ^{D}C1 (XON) and ^{D}C3 (XOFF) maintain flow control for serial connections if the **Xon/Xoff** option is selected in the **Pacing** group box on the More Settings-Connection Setup dialog box.

Display Control Character Set

The display control character set is a special character set used if **Display** is selected in the **Control characters** group box on the Screen tab of the Display Setup dialog box. It is loaded temporarily into C0, GL, C1, and GR to show how control characters appear on screen.

The ISO Latin-1 supplemental character set is used in GR to present displayable graphic characters (numbers, letters, and so on) with decimal values 160–255.

Decimal → Hex →							
00 / 00 N$_U$	16 / 10 D$_L$	32 / 20 (space)	48 / 30 0	64 / 40 @	80 / 50 P	96 / 60 `	112 / 70 p
01 / 01 S$_H$	17 / 11 D$_1$	33 / 21 !	49 / 31 1	65 / 41 A	81 / 51 Q	97 / 61 a	113 / 71 q
02 / 02 S$_X$	18 / 12 D$_2$	34 / 22 "	50 / 32 2	66 / 42 B	82 / 52 R	98 / 62 b	114 / 72 r
03 / 03 E$_X$	19 / 13 D$_3$	35 / 23 #	51 / 33 3	67 / 43 C	83 / 53 S	99 / 63 c	115 / 73 s
04 / 04 E$_T$	20 / 14 D$_4$	36 / 24 $	52 / 34 4	68 / 44 D	84 / 54 T	100 / 64 d	116 / 74 t
05 / 05 E$_Q$	21 / 15 N$_K$	37 / 25 %	53 / 35 5	69 / 45 E	85 / 55 U	101 / 65 e	117 / 75 u
06 / 06 A$_K$	22 / 16 S$_Y$	38 / 26 &	54 / 36 6	70 / 46 F	86 / 56 V	102 / 66 f	118 / 76 v
07 / 07 B$_L$	23 / 17 E$_B$	39 / 27 '	55 / 37 7	71 / 47 G	87 / 57 W	103 / 67 g	119 / 77 w
08 / 08 B$_S$	24 / 18 C$_N$	40 / 28 (	56 / 38 8	72 / 48 H	88 / 58 X	104 / 68 h	120 / 78 x
09 / 09 H$_T$	25 / 19 E$_M$	41 / 29)	57 / 39 9	73 / 49 I	89 / 59 Y	105 / 69 i	121 / 79 y
10 / 0A L$_F$	26 / 1A ₵	42 / 2A *	58 / 3A :	74 / 4A J	90 / 5A Z	106 / 6A j	122 / 7A z
11 / 0B V$_T$	27 / 1B E$_C$	43 / 2B +	59 / 3B ;	75 / 4B K	91 / 5B [	107 / 6B k	123 / 7B {
12 / 0C F$_F$	28 / 1C F$_S$	44 / 2C ,	60 / 3C <	76 / 4C L	92 / 5C \	108 / 6C l	124 / 7C	
13 / 0D C$_R$	29 / 1D G$_S$	45 / 2D -	61 / 3D =	77 / 4D M	93 / 5D]	109 / 6D m	125 / 7D }
14 / 0E S$_O$	30 / 1E R$_S$	46 / 2E .	62 / 3E >	78 / 4E N	94 / 5E ^	110 / 6E n	126 / 7E ~
15 / 0F S$_I$	31 / 1F U$_S$	47 / 2F /	63 / 3F ?	79 / 4F O	95 / 5F _	111 / 6F o	127 / 7F D$_T$

CO CODES | GL CODES ASCII GRAPHIC

Display Controls (Left Half) with C0 Codes

Decimal →
Hex →

Dec/Hex		Dec/Hex		Dec/Hex		Dec/Hex		Dec/Hex		Dec/Hex		Dec/Hex		Dec/Hex	
128 80	8_0	144 90	D_C	160 A0	A_0	176 B0	°	192 C0	À	208 D0	Ð	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 ————

Display Controls (Right Half) with C1 Codes

ASCII Character Set

The ASCII character set is the default character set loaded into GL when you start Reflection. This set is identical to the Windows character set (ANSI) codes 32–126.

Decimal →					
Hex → 20	(space)	48 30 **0**	64 40 **@**	80 50 **P**	96 60 **`**
33 21 **!**		49 31 **1**	65 41 **A**	81 51 **Q**	97 61 **a**
34 22 **"**		50 32 **2**	66 42 **B**	82 52 **R**	98 62 **b**
35 23 **#**		51 33 **3**	67 43 **C**	83 53 **S**	99 63 **c**
36 24 **$**		52 34 **4**	68 44 **D**	84 54 **T**	100 64 **d**
37 25 **%**		53 35 **5**	69 45 **E**	85 55 **U**	101 65 **e**
38 26 **&**		54 36 **6**	70 46 **F**	86 56 **V**	102 66 **f**
39 27 **'**		55 37 **7**	71 47 **G**	87 57 **W**	103 67 **g**
40 28 **(**		56 38 **8**	72 48 **H**	88 58 **X**	104 68 **h**
41 29 **)**		57 39 **9**	73 49 **I**	89 59 **Y**	105 69 **i**
42 2A *****		58 3A **:**	74 4A **J**	90 5A **Z**	106 6A **j**
43 2B **+**		59 3B **;**	75 4B **K**	91 5B **[**	107 6B **k**
44 2C **,**		60 3C **<**	76 4C **L**	92 5C ****	108 6C **l**

ASCII Character Set

ANSI Character Set

Reflection can send and receive ASCII, DEC Supplemental Graphic, ISO Latin-1, DEC Special Graphic, or DEC Technical characters. Windows, however, uses the ANSI character set. This does not present a problem for the characters in the range 32–126 (because 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 on page 257.

Decimal → 128 Hex → 80 ■	144 90 ■	160 A0	176 B0 °	192 C0 À	208 D0 Ð	224 E0 à	240 F0 ð
129 81 ■	145 ʻ 91	161 A1 ¡	177 B1 ±	193 C1 Á	209 D1 Ñ	225 E1 á	241 F1 ñ
130 82 ■	146 ʼ 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.

ANSI Character Set (128–255)

DEC Supplemental Graphic Character Set

The DEC Supplemental Graphic character set contains 94 graphic characters, including accented characters for many national languages. This set is the default character set loaded into GR when you start Reflection.

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 Supplemental Graphic Character Set

ISO Latin-1 Supplemental Graphic Character Set

The ISO Latin-1 supplemental graphic character set contains 96 graphic characters, including many of the same characters as the DEC Supplemental Graphic set.[*] You select **ISO Latin-1** in the **Host character set** list in the **Emulation options** group box on the Emulation tab of the Terminal Setup dialog box.

Decimal → Hex →												
160 A0	(space)	176 B0	°	192 C0	À	208 D0	Ð	224 E0	à	240 F0	ð
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	×	231 E7	ç	247 F7	÷
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	Ý	237 ED	í	253 FD	ý
174 AE	®	190 BE	3/4	206 CE	Î	222 DE	Þ	238 EE	î	254 FE	þ
175 AF	—	191 BF	¿	207 CF	Ï	223 DF	ß	239 EF	ï	255 FF	ÿ

☐ Unique to the ISO Latin-1 character set; the Host Character Set on the Emulation panel in the Terminal Setup dialog box must be set to ISO Latin-1.

ISO Latin-1 Supplemental Graphic Character Set

[*] The ISO Latin-1 character set is identical to the Windows character set (ANSI) codes 160–255.

DEC Special Graphic Character Set

The DEC Special Graphic character set contains 94 graphic characters, including letters, numbers, special symbols, and line-drawing characters. When mapped to GL, this set includes the C0 codes shown in the table on page 248.

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 H_T	242 F2 — SCAN 7
163 A3 #	179 B3 3	195 C3 C	211 D3 S	227 E3 F_F	243 F3 — SCAN 9
164 A4 $	180 B4 4	196 C4 D	212 D4 T	228 E4 C_R	244 F4 ├
165 A5 %	181 B5 5	197 C5 E	213 D5 U	229 E5 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 N_L	248 F8 │
169 A9)	185 B9 9	201 C9 I	217 D9 Y	233 E9 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.

DEC Special Graphic Character Set

DEC Technical Character Set

The DEC Technical character set contains 94 graphic characters, including symbols and characters used in technical applications. When mapped to GL, it includes the C0 codes shown in the table on page 248.

Decimal → Hex →											
160 A0		176 B0	⌡	192 C0	∴	208 D0	Π	224 E0	¬	240 F0	π
161 A1	√	177 B1	⟍	193 C1	∝	209 D1	Ψ	225 E1	α	241 F1	ψ
162 A2	Γ	178 B2	∠	194 C2	∞	210 D2	⌠	226 E2	β	242 F2	ρ
163 A3	—	179 B3	\	195 C3	÷	211 D3	Σ	227 E3	χ	243 F3	σ
164 A4	⌠	180 B4	/	196 C4	Δ	212 D4	⌡	228 E4	δ	244 F4	τ
165 A5	⌡	181 B5	⌐	197 C5	∇	213 D5	⌡	229 E5	ε	245 F5	⌠
166 A6	|	182 B6	⌐	198 C6	Φ	214 D6	√	230 E6	φ	246 F6	ƒ
167 A7	⌈	183 B7	〉	199 C7	Γ	215 D7	Ω	231 E7	Υ	247 F7	ω
168 A8	L	184 B8	⌠	200 C8	¨	216 D8	Ξ	232 E8	η	248 F8	ξ
169 A9	⌐	185 B9	⌠	201 C9	≃	217 D9	γ	233 E9	ι	249 F9	υ
170 AA	⌋	186 BA	⌠	202 CA	Θ	218 DA	⊂	234 EA	θ	250 FA	ζ
171 AB	(	187 BB	⌠	203 CB	×	219 DB	⊃	235 EB	κ	251 FB	←
172 AC	\	188 BC	≤	204 CC	Λ	220 DC	∩	236 EC	λ	252 FC	↑
173 AD	\	189 BD	≠	205 CD	⇔	221 DD	∪	237 ED	⌠	253 FD	→
174 AE	)	190 BE	≥	206 CE	⇒	222 DE	∧	238 EE	ν	254 FE	↓

DEC Technical Character Set

IBM PC Extended Character Set

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

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.

IBM PC Extended Character Set (Code Page 437)

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

Using the Alt key, you provide a decimal value from the ANSI character set to enter a character in the terminal window. To do this:

1. Log in to the host as you would normally.

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 table on page 251.

   Because the ANSI character set is identical to the host's ASCII and ISO Latin-1 character sets, you can also refer to the tables on 250 and 253 for decimal values.

4. If NumLock is on, press Shift+NumLock to turn it 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, a ¢ appears when you release the Alt key.

When the **Host character set** list is set to **DEC Supplemental** (the default), you can display only those characters that also appear in the ISO Latin-1 character set. If you want to enter one of the 18 characters unique to ISO Latin-1 (see the figure on page 253), you must first select this set from the **Host character set** list.

The decimal values of the three characters unique to the DEC Supplemental graphic character set are 215, 221, and 247; the only way to create these characters is to use the Reflection Display method, explained next.

Using the Reflection Display Method

You can also enter characters in the terminal window using the Reflection Display method. With this command, 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.

For example:

1.   Press [Alt]+[L] to open the Reflection command line.

2.   Find the decimal value for the character from the ASCII character set (page 250) or the current **Host character set**; by default, **DEC Supplemental** (page 252) is selected. ISO Latin-1 is shown on page 253.

3.   Execute the Display method syntax as follows, and press [Enter ↵]:

```
Display "^(decimal value)"
```

For example, with the default DEC Supplemental graphic character set loaded, the following displays the £ symbol in Reflection's terminal window:

```
Display "^(163)"
```

And to display œ (an o e ligature character unique to DEC Supplemental), use the following command:

```
Display "^(247)"
```

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 the terminal window from the IBM extended character set:

1. Login as you would normally do.

2. Make sure the **National Replacement Set** list in the Advanced Options dialog box 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 table on page 256.

4. If NumLock is on, press Shift+NumLock to turn it off.

5. Hold down Alt and type the 3-digit decimal code on your numeric keypad (do not type a 0 before the code, as you would to enter a national character using the method on page 257).

6. 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.

Entering Characters Using Compose Sequences

A VT terminal has a Compose key used for creating special characters. In Reflection, you can start a compose sequence in one of two ways:

- Click the Compose indicator in Reflection's status bar.

- Press Alt+F8.

After entering this mode, type the two characters that make up the character. Once you enter the second character, the Compose indicator is turned off. For example, to create the â symbol, click the Compose indicator, then type a followed by ^ (that is, press A+Shift+6).

For most sequences, the order of the two characters does not matter. However, the ligatures, micron, one-quarter, and one-half fraction must be entered in the order indicated. In addition, the Ý, Œ , and œ sequences are only valid when the **Host character set** is set to **DEC Supplemental** (the default). Click Advanced on the Emulation tab of the Terminal Setup dialog box to see this. If an invalid compose character is entered, the compose sequence is ignored and a bell sounds.

The following table shows what characters to type after starting a compose sequence to form multinational characters.

Description	Character	Sequence
A acute	Á	A'
a acute	á	a'
A circumflex	Â	A^
a circumflex	â	a^
A grave	À	A`
a grave	à	a`
A ring	Å	A* or Aº
a ring	å	a* or aº
A tilde	Ã	A~
a tilde	ã	a~
A diaeresis	Ä	A"
a diaeresis	ä	a"
A E ligature	Æ	AE
a e ligature	æ	ae
C cedilla	Ç	C,
c cedilla	ç	c,
E acute	È	E'"
e acute	è	e'"
E circumflex	Ê	E^

Description	Character	Sequence
e circumflex	ê	e^
E grave	É	E`
e grave	é	e`
E diaeresis	Ë	E"
e diaeresis	ë	e"
I acute	Ì	I'"
i acute	ì	i'"
I circumflex	Î	I^
i circumflex	î	i^
I grave	Í	I`
i grave	í	i`
I diaeresis	Ï	I"
i diaeresis	î	i"
N tilde	Ñ	N~
n tilde	ñ	n~
O acute	Ò	O'
o acute	ò	o'
O circumflex	Ô	O^
o circumflex	ô	o^
O grave	Ó	O`
o grave	ó	o`
O slash	Ø	O/
o slash	ø	o/
O tilde	Õ	O~
o tilde	õ	o~

Description	Character	Sequence
O diaeresis	Ö	O"
o diaeresis	ö	o"
O E ligature	Œ	OE
o e ligature	œ	oe
German sharp s	ß	ss
U acute	Ù	U'"
u acute	ù	u'"
U circumflex	Û	U^
u circumflex	û	u^
U grave	Ú	U`
u grave	ú	u`
U diaeresis	Ü	U"
u diaeresis	ü	u"
Y diaeresis	Ÿ	Y"
y diaeresis	ÿ	y"
quotation mark	"	"(space)
number sign	#	++
apostrophe	‘	(space)"
commercial at	@	a a or A A
opening bracket	[	((
closing bracket	]	))
backslash	\	// or /<
single quote	‘	`(space)
opening brace	{	(-
closing brace	}	)-

Description	Character	Sequence
vertical line	|	/^
tilde	~	˜(space)
inverted !	¡	!!
cent sign	¢	c/ or C/ or c✠ or C✠
pound sign	£	l- or L- or l= or L=
yen sign	¥	y- or Y- or y= or Y=
section sign	§	so or SO or S! or s! or sØ or SØ
currency sign	€	xo or XO or xØ or XØ
copyright sign	©	co or CO or cØ or CØ
feminine ordinal indicator	ª	a_ or A_
angle quotation mark left	«	<<
angle quotation mark right	»	>>
degree sign	º	Ø^
plus minus sign	ffl	+-
superscript 1	¹	1^
superscript 2	²	2^
superscript 3	³	3^
micron	✠	/u or /U
paragraph sign	¶	p! or P!
middle dot	•	.^
masculine ordinal indicator	º	o_ or O_
one-quarter fraction	✠	1 4
one-half fraction	✠	1 2

Description	Character	Sequence
inverted ?	¿	??

The following table of sequences are available when **ISO Latin-1** is selected in the **Host character set** list on the Emulation tab of the Terminal Setup dialog box.

Description	Character	Sequence
No break space		(space)(space)
Broken vertical bar	¤	¤ ¤ or !^
Diaeresis (umlaut)*	"	" " or "(space)
Logical not*	⊕	-,
Multiplication symbol	¤	xx
Division symbol	÷	-:
Soft (syllable) hyphen	-	--
Registered trademark	®	RO
Macron	-	-^ or _^
Acute accent	´	”
Three quarters*	¤	34
Uppercase Icelandic Eth	Ð	D-
Lowercase Icelandic Eth*	ð	d-
Uppercase Icelandic thorn*	Þ	TH
Lowercase Icelandic thorn	þ	th
Uppercase Y acute	Ý	Y’”
Lowercase y acute	ý	y’”
Cedilla	،	”

During a compose sequence, keys ⌨1 – ⌨6 on the main keyboard (not the keypad) act as accent characters to accommodate national keyboards and drivers that do not provide all of the diacritical marks. These marks are:

⌨1   `

⌨2   ´

⌨3   ^

⌨4   ~

⌨5   ¨

⌨6   º

Translation of Characters

Reflection translates between 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 Reflection property from the command line:

```
TranslateCharacters=False
```

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	**VAX 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 also 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).

Transmitting National Characters

Together, the ASCII character set and the DEC Supplemental Graphic set comprise the DEC Multinational character set. You can transmit national characters using either the Multinational character set (MCS) or a national replacement character (NRC) set—Reflection's **Parity** setting in the More Settings-Connection Setup dialog box determines which type of characters are transmitted.

MCS	Set **Parity** to 8 data bits (**8/None**, **8/Even**, or **8/Odd**). Because multinational characters all use the 8th bit, they cannot be directly sent or received in 7-bit modes.
NRC	Set **Parity** to 7 data bits (**7/0's**, **7/1's**, **7/Odd**, **7/Even**, or **7/None**).
	The NRC sets are 7-bit character sets for national languages. Setting **Parity** to 7 data bits puts Reflection in 7-bit mode—that is, seven data bits and one (or no) parity bit. The 8th bit of each byte is used for parity and is not available for use as data.

When national characters are written to or read from disk files, they are converted to either the multinational or ANSI character set (depending on the character translation setting; see page 265).

National Replacement Characters

Some host systems use national replacement characters (NRC) to encode characters that are not available in the ASCII character set. In 7-bit operations when you select a **National Replacement Set** option other than **None**, and select the **Use NRC (7-bit) Set** check box in the Advanced Options dialog box (click Advanced on the Emulation tab of the Terminal Setup dialog box to see this), certain characters in the ASCII character set are replaced by the characters shown in the following table:

NRC Set	Characters											
ASCII Decimal	35	64	91	92	93	94	95	96	123	124	125	126
U.S. ASCII	#	@	[	\	]	^	_	`	{	|	}	~
Dutch	£	3/4	ÿ *	1/2	|	^	_	`	¨	f	1/4	´
Finnish	#	@	Ä	Ö	Å	Ü	_	é	ä	ö	å	ü
French	£	à	°	ç	§	^	_	`	é	ù	è	¨
Canadian	#	à	â	ç	ê	î	_	ô	é	ù	è	û
German	#	§	Ä	Ö	Ü	^	_	`	ä	ö	ü	ß
Italian	£	§	°	ç	é	^	_	ù	à	ò	è	ì
Norwegian/Danish	#	@	Æ	Ø	Å	^	_	`	æ	ø	å	~
Portuguese	#	@	Ã	Ç	Õ	^	_	`	ã	ç	õ	~
Spanish	£	§	¡	Ñ	¿	^	_	`	°	ñ	ç	~
Swedish	#	É	Ä	Ö	Å	Ü	_	é	ä	ö	å	ü
Swiss	ù	à	é	ç	ê	î	è	ô	ä	ö	ü	û
British	£	@	[	\	]	^	_	`	{	|	}	~

* The ÿ represents ij

Control Function Index

Control Functions: Sorted by Sequence

Control Function	Mnemonic	Description	Location
C_{SI}!p	DECSTR	Reset terminal (soft reset)	106
C_{SI}&u	DECRQUPSS	Request UPSS	105
C_{SI}>Øc	DA	Request secondary device attributes	88
C_{SI}>24;11;Øc	DA	Response to secondary device attributes	88
C_{SI}>c	DA	Request secondary device attributes	88
C_{SI}?<n>$p	DECRQM	Request DEC private mode settings	100
C_{SI}?<n>;…<n>l	RM	Reset DEC private mode	103
C_{SI}?<n>;…<n>h	SM	Set DEC private mode	102
C_{SI}?<n1>;<n2>$y	DECRPM	Report DEC private mode setting	102
C_{SI}?Øi	MC	Send graphics to printer	67
C_{SI}?ØJ	DECSED	Erase unprotected characters from cursor to end of screen	46
C_{SI}?ØK	DECSEL	Erase unprotected characters from cursor to end of line	46
C_{SI}?1Øn	DSR	Report printer ready	90
C_{SI}?13n	DSR	Report no printer	90
C_{SI}?15n	DSR	Request printer status	90
C_{SI}?18h	DECPFF	Send form feed after printing	57
C_{SI}?18l	DECPFF	No form feed after printing	57
C_{SI}?19h	DECPEX	Print full screen	56
C_{SI}?19l	DECPEX	Print scrolling region	56

Control Function	Mnemonic	Description	Location
CSI?1h	DECCKM	Cursor keys application	48
CSI?1i	MC	Print line	58
CSI?1J	DECSED	Erase unprotected characters from top of screen to cursor	46
CSI?27;\<n\>n	DSR	Report keyboard dialect (\<n\> = 1 for North American keyboards)	91
CSI?2J	DECSED	Erase unprotected characters from screen	46
CSI?2K	DECSEL	Erase unprotected characters from line	46
CSI?2l	DECANM	VT52 emulation	24, 109
CSI?3h	DECCOLM	Columns 132	42
CSI?3l	DECCOLM	Columns 80	42
CSI?42h	DECNRCM	Use national replacement set: 7-bit	78
CSI?42l	DECNRCM	Use national replacement set: 7-bit and 8-bit	78
CSI?43h	DECGEPM	Expanded print mode	66
CSI?43l	DECGEPM	Compressed print mode	66
CSI?44h	DECGPCM	Print color mode	66
CSI?44l	DECGPCM	Print monochrome mode	66
CSI?45h	DECGPCS	Print color syntax set to RGB	66
CSI?45l	DECGPCS	Print color syntax set to HLS	66
CSI?46h	DECGPBM	Print background	67
CSI?46l	DECGPBM	Do not print background	67
CSI?47h	DECGRPM	Print rotated mode	66
CSI?47l	DECGRPM	Print compressed mode	66
CSI?4h	DECSCLM	Smooth scroll	40

Control Function	Mnemonic	Description	Location
$^C_{SI}$?4l	DECSCLM	Jump scroll	40
$^C_{SI}$?4i	MC	Auto print off	58
$^C_{SI}$?5Øn	DSR	Report a locator device detected	89
$^C_{SI}$?53n	DSR	Report no locator device detected	89
$^C_{SI}$?55n	DSR	Request status of locator device	89
$^C_{SI}$?56n	DSR	Request type of locator device	89
$^C_{SI}$?57;Øn	DSR	Report no locator device connected	89
$^C_{SI}$?57;1n	DSR	Report locator device is a mouse	89
$^C_{SI}$?1K	DECSEL	Erase unprotected characters from beginning of line to cursor	46
$^C_{SI}$?1l	DECCKM	Cursor keys normal	48
$^C_{SI}$?2Øn	DSR	Report UDKs unlocked	90
$^C_{SI}$?21n	DSR	Report UDKs locked	90
$^C_{SI}$?2i	MC	Send graphics to host	67
$^C_{SI}$?25h	DECTCEM	Cursor visible	34
$^C_{SI}$?25l	DECTCEM	Cursor invisible	34
$^C_{SI}$?25n	DSR	Request UDK status (VT200)	90
$^C_{SI}$?26n	DSR	Request keyboard dialect	91
$^C_{SI}$?5h	DECSCNM	Inverse video	40
$^C_{SI}$?5l	DECSCNM	Normal video	40
$^C_{SI}$?5i	MC	Auto print on	58
$^C_{SI}$?60h	DECHCCM	Horizontal cursor: couple	38
$^C_{SI}$?60l	DECHCCM	Horizontal cursor: decouple	38
$^C_{SI}$?61h	DECVCCM	Vertical cursor: couple	39
$^C_{SI}$?61l	DECVCCM	Vertical cursor: decouple	39

Control Function	Mnemonic	Description	Location
CSI?62n	DECMSR	Request macro report	60
CSI?63;<n>n	DECCKSR	Request memory checksum of macros	34
CSI?64h	DECPCCM	Page cursor: couple	39
CSI?64l	DECPCCM	Page cursor: decouple	39
CSI?66h	DECNKM	Numeric keypad mode application	48
CSI?66l	DECNKM	Numeric keypad mode numeric	48
CSI?67h	DECBKM	Backarrow key set to BS	49
CSI?67l	DECBKM	Backarrow key set to DT	49
CSI?6h	DECOM	Origin mode set	43
CSI?6l	DECOM	Origin mode reset	43
CSI?7h	DECAWM	Autowrap on	48
CSI?7l	DECAWM	Autowrap off	48
CSI?8Øh	DECSDM	Disable sixel scrolling	65
CSI?8Øl	DECSDM	Enable sixel scrolling	65
CSI?8h	DECARM	Keyboard auto repeat on	47
CSI?8l	DECARM	Keyboard auto repeat off	47
CSI?i	MC	Send graphics to printer	67
CSI<n>*x	DECSACE	Select attribute change extent	32
CSI<n>*z	DECIMAC	Invoke macro	60
CSI<n>*}	none	Macro space report	60
CSI<n>$p	DECRQM	Request ANSI mode settings	100
CSI<n1>;<n2>$y	DECRPM	Report ANSI mode settings	102
CSI<n>@	ICH	Insert <n> characters	44
CSI<n>A	CUU	Cursor up	35

Control Function	Mnemonic	Description	Location
$^{C}S_{I}$<n>B	CUD	Cursor down	36
$^{C}S_{I}$<n>C	CUF	Cursor forward	35
$^{C}S_{I}$<n>D	CUB	Cursor backward	35
$^{C}S_{I}$<n>S	SU	Pan down	40
$^{C}S_{I}$<n>T	SD	Pan up	41
$^{C}S_{I}$<n>U	NP	Next page	37
$^{C}S_{I}$<n>V	PP	Previous page	37
$^{C}S_{I}$<n>Z	CBT	Cursor backtab	36
$^{C}S_{I}$?<n1>;<n2>;<n3>;<n4>;<n5>...<nn>$t			
	DECRARA	Reverse attributes in rectangular area	33
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>${			
	DECSERA	Selective erase of rectangular area	31
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>;<n5>;...<nn>$r			
	DECCARA	Change attributes of rectangular area	32
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>;<n5>;<n6>*y			
	DECRQCRA	Request memory checksum	33
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>;<n5>;<n6>;<n7>;<n8>$v			
	DECCRA	Copy rectangular area	30
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>$z			30
	DECERA	Erase rectangular area	page 30
$^{C}S_{I}$<n1>;<n2>;<n3>;<n4>$x			
	DECFRA	Fill rectangular area	31
$^{C}S_{I}$<r>;<c>f	HVP	Move cursor to horizontal and vertical position	35
$^{C}S_{I}$<n>;…<n>h	SM	Set ANSI mode	102

Control Function	Mnemonic	Description	Location
C_{SI}<r>;<c>H	CUP	Cursor position	35
C_{SI}<n>;…<n>l	RM	Reset ANSI mode	103
C_{SI}<n>L	IL	Insert <n> lines	44
C_{SI}<n>;...<n>m	SGR	Select graphic rendition	27
C_{SI}<n>M	DL	Delete <n> lines from cursor	44
C_{SI}<n>P	DCH	Delete <n> characters from cursor	44
C_{SI}<n><SP>P	PPA	Page position absolute	37
C_{SI}<n><SP>Q	PPR	Page position relative	37
C_{SI}<n><SP>R	PPB	Page position backward	38
C_{SI}<t>;<b>r	DECSTBM	Set top and bottom scrolling region margins	42
C_{SI}<row>;<column>R	CPR	Report cursor position	89
C_{SI}<n>X	ECH	Erase <n> characters on line	45
C_{SI}Ø"q	DECSCA	Select erasable character	46
C_{SI}Ø$}	DECSASD	Top 24 lines = active display	42
C_{SI}Ø$~	DECSSDT	Do not display status line	41
C_{SI}Øc	DA	Request primary device attributes	85
C_{SI}Øg	TBC	Tab clear	108
C_{SI}Øi	MC	Print screen	58
C_{SI}ØJ	ED	Erase from cursor to end of screen	45
C_{SI}ØK	EL	Erase from cursor to end of line	45
C_{SI}Øn	DSR	Report no device malfunction	89
C_{SI}Øx	DA	Request terminal parameters	87
C_{SI}Øz	DECVERP	Print 6 lines per inch, 63 lines per page	57
C_{SI}1z	DECVERP	Print 6 lines per inch, 63 lines per page	57

Control Function	Mnemonic	Description	Location
$C_{S_I}2z$	DECVERP	Print 8 lines per inch, 84 lines per page	57
$C_{S_I}3z$	DECVERP	Print 12 lines per inch, 125 lines per page	57
$C_{S_I}4z$	DECVERP	Print 2 lines per inch, 21 lines per page	57
$C_{S_I}5z$	DECVERP	Print 3 lines per inch, 32 lines per page	57
$C_{S_I}6z$	DECVERP	Print 4 lines per inch, 42 lines per page	57
$C_{S_I}1"q$	DECSCA	Select protected character	46
$C_{S_I}1\$\}$	DECSASD	Status line set to active display	42
$C_{S_I}1\$\sim$	DECSSDT	Status line set to indicator	41
$C_{S_I}1\$u$	DECRQTSR	Request terminal state report	91
$C_{S_I}1\$w$	DECRQPSR	Request cursor information report	93
$C_{S_I}12h$	SRM	Local echo off	41
$C_{S_I}12l$	SRM	Local echo on	41
$C_{S_I}1J$	ED	Erase from top of screen to cursor	45
$C_{S_I}1K$	EL	Erase from beginning of line to cursor	46
$C_{S_I}1x$	DA	Request terminal parameters	87
$C_{S_I}0w$	DECSHORP	Print 10 characters per inch, 80 characters per row	57
$C_{S_I}1w$	DECSHORP	Print 10 characters per inch, 80 characters per row	57
$C_{S_I}2w$	DECSHORP	Print 12 characters per inch, 96 characters per row	57
$C_{S_I}4w$	DECSHORP	Print 16.5 characters per inch, 132 characters per row	57
$C_{S_I}2"q$	DECSCA	Select erasable character	46
$C_{S_I}2\$\sim$	DECSSDT	Status line: host writable	41

Control Function	Mnemonic	Description	Location
$^{CS}_{I}$2$u	DECRQTSR	Request color table report (VT340)	91
$^{CS}_{I}$2$w	DECRQPSR	Request tab stop report	93
$^{CS}_{I}$2;<n1>;<n2>;<receive baud>;<transmit baud>;1;Øx			
	DA	Response to $^{CS}_{I}$Øx	87
$^{CS}_{I}$2Øh	LNM	Auto linefeed on	47
$^{CS}_{I}$2Øl	LNM	Auto linefeed off	47
$^{CS}_{I}$2h	KAM	Keyboard lock	47
$^{CS}_{I}$2J	ED	Erase entire screen	45
$^{CS}_{I}$2K	EL	Erase entire line	45
$^{CS}_{I}$2l	KAM	Keyboard unlock	47
$^{CS}_{I}$3;<n1>;<n2>;<receive baud>;<transmit baud>;1;Øx			
	DA	Response to $^{CS}_{I}$1x	87
$^{CS}_{I}$3g	TBC	Clear all tabs	108
$^{CS}_{I}$3h	—	Display controls on	43
$^{CS}_{I}$3l	—	Display controls off	43
$^{CS}_{I}$4h	IRM	Insert mode	44
$^{CS}_{I}$4i	MC	Printer controller mode off	58
$^{CS}_{I}$5i	MC	Printer controller mode on	58
$^{CS}_{I}$4l	IRM	Replace mode	44
$^{CS}_{I}$5n	DSR	Report device malfunction	89
$^{CS}_{I}$5n	DSR	Request operating status	89
$^{CS}_{I}$24t	DECSLPP	Select lines per page: 24 lines set to 6 pages	29
$^{CS}_{I}$25t	DECSLPP	Select lines per page: 25 lines set to 5 pages	29

Control Function	Mnemonic	Description	Location
^CS_I36t	DECSLPP	Select lines per page: 36 lines set to 4 pages	29
^CS_I48t	DECSLPP	Select lines per page: 48 lines set to 3 pages	29
^CS_I72t	DECSLPP	Select lines per page: 72 lines set to 2 pages	29
^CS_I144t	DECSLPP	Select lines per page: 144 lines set to 1 page	29
^CS_I61"p	DECSCL	VT102 emulation	23
^CS_I62"p	DECSCL	VT200 emulation: 8-bit controls	23
^CS_I62;∅"p	DECSCL	VT200 emulation: 8-bit controls	23
^CS_I62;1"p	DECSCL	VT200 emulation: 7-bit controls	23
^CS_I62;2"p	DECSCL	VT200 emulation: 8-bit controls	23
^CS_I63"p	DECSCL	VT300 emulation: 8-bit controls	23
^CS_I63;∅"p	DECSCL	VT300 emulation: 8-bit controls	23
^CS_I63;1"p	DECSCL	VT300 emulation: 7-bit controls	23
^CS_I63;2"p	DECSCL	VT300 emulation: 8-bit controls	23
^CS_I64;1"p	DECSCL	VT400 emulation: 7-bit controls	23
^CS_I64"p	DECSCL	VT400 emulation: 8-bit controls	23
^CS_I64;0"p	DECSCL	VT400 emulation: 8-bit controls	23
^CS_I64;2"p	DECSCL	VT400 emulation: 8-bit controls	23
^CS_I6n	CPR	Request cursor position	90
^CS_Ic	DA	Request primary device attributes	85
^CS_Ig	TBC	Tab clear	108
^CS_Ii	MC	Print screen	58
^CS_I4;<n>;...;<n>y	DECTST	Terminal test (disconnect)	108

Control Function	Mnemonic	Description	Location
DCS$q...ST	DECRQSS	Request control function setting	103
DCS<n>;<n>;P3;qs...sST	—	Sixel data format	63
DCS<n1>;<n2>;<n3>!zD...DSt			
	DECDMAC	Define macro (VT420)	59
DCS<C>$w<def1>;<def2>;...;<defn>ST			
	DECLBD	Define locator device buttons (mouse)	61
DCS<C>;<L>|<def1>;<def2>;...;<defn>ST			
	DECUDK	Load user-defined keys	52
DCS<n>$rD...DST	DECRPSS	Report control function setting	104
DCSØ!u%5ST	DECAUPSS	Assign UPSS as DEC Supplemental Graphic	105
DCSØp	—	Enter ReGIS at point command exited	60
DCS1!uAST	DECAUPSS	Assign UPSS as ISO Latin-1	105
DCS1$p....ST	DECRSTS	Restore terminal state from DECTSR	92
DCS1$sD....DST	DECTSR	Report terminal state	91
DCS1$tD....DST	DECRSPS	Restore cursor from DECCIR	100
DCS1$uD....DST	DECCIR	Report cursor information	93
DCS1234;Ps{<cmd>ST	WRQCMD	Invoke Reflection command	19
DCSØp	—	Enter ReGIS at point command exited; ReGIS commands are not displayed	60
DCS1p	—	Enter ReGIS and begin new command; ReGIS commands are not displayed	60
DCS2p	—	Enter ReGIS at point command exited; ReGIS commands are not displayed	60
DCS3p	—	Enter ReGIS and begin new command; ReGIS commands are not displayed	60

Control Function	Mnemonic	Description	Location
DCSp	—	Enter ReGIS at point command exited; ReGIS commands are not displayed	60
DCS2$p\<data>ST	DECRSTS	Restore color table (VT340)	91
DCS2$s\<data>ST	DECCTR	Report color table (VT340)	91
DCS2$tD....DST	DECRSPS	Restore tab from DECTABSR	100
DCS2$uD....DST	DECTABSR	Report tab stop	99
DCSPfn;Pcn;Pe;PcmW;Pw;Pt;Pcmh;Pcss{ Dscs UUUUUUUU/LLLLLLLL;...ST			
	DECDLD	Downline loadable character set	81
ESC(\<chr>	SCS	Selects GØ	72
ESC)\<chr>	SCS	Selects G1, 94-character set	72
ESC-\<chr>	SCS	Selects G1, 96-character set	72
ESC*\<chr>	SCS	Selects G2 (VT200), 94-character set	72
ESC+\<chr>	SCS	Selects G3 (VT200), 94-character set	72
ESC.\<chr>	SCS	Selects G2 (VT200), 96-character set	72
ESC/\<chr>	SCS	Selects G3 (VT200), 96-character set	72
ESC<	—	ANSI mode—exit VT52 emulation mode	24, 109
ESC=	DECKPAM	Keypad mode application	48
ESC=	DECKPAM	Keypad mode application (VT52)	110
ESC>	DECKPNM	Keypad mode normal	48
ESC>	DECKPNM	Keypad mode normal (VT52)	110
ESC\	—	Exit ReGIS mode	60
ESC]	MC	Print screen (VT52)	110
ESC^	MC	Auto print mode on (VT52)	110
ESC_	MC	Auto print mode off (VT52)	110
ESC|	LS3R	Map G3 into GR	74

Control Function	Mnemonic	Description	Location
^{E}s$_C$}	LS2R	Map G2 into GR	74
^{E}s$_C$~	LS1R	Map G1 into GR	74
^{E}s$_C$#3	DECDHL	Double-width and height line (top half)	28
^{E}s$_C$#4	DECDHL	Double-width and height line (bottom half)	28
^{E}s$_C$#5	DECSWL	Single-width and height line	28
^{E}s$_C$#6	DECDWL	Double-width and single height line	28
^{E}s$_C$#8	DECALN	Test pattern	108
^{E}s$_C$<S$_P$>F	S7C1T	VT200 emulation: 7-bit controls	24
^{E}s$_C$<S$_P$>G	S8C1T	VT200 emulation: 8-bit controls	24
^{E}s$_C$7	DECSC	Save cursor state	104
^{E}s$_C$8	DECRC	Restore cursor state	105
^{E}s$_C$A	—	Cursor up (VT52)	109
^{E}s$_C$B	—	Cursor down (VT52)	109
^{E}s$_C$C	—	Cursor forward (VT52)	109
^{E}s$_C$c	RIS	Reset to initial state (hard reset)	106
^{E}s$_C$D	—	Cursor backward (VT52)	109
^{E}s$_C$H	—	Home cursor (VT52)	109
^{E}s$_C$H	—	Set tab	108
^{E}s$_C$I	—	Reverse linefeed (VT52)	110
^{E}s$_C$J	—	Erase from cursor to end of screen (VT52)	109
^{E}s$_C$K	—	Erase to end of line (VT52)	109
^{E}s$_C$n	LS2	Map G2 into GL	74
^{E}s$_C$N	SS2	Map G2 into GL for next character	74

Control Function	Mnemonic	Description	Location
$^{E}S_{CO}$	LS3	Map G3 into GL	74
$^{E}S_{CO}$	SS3	Map G3 into GL for next character	74
$^{E}S_{CV}$	—	Print cursor line (VT52)	110
$^{E}S_{CW}$	—	Printer controller mode on (VT52)	110
$^{E}S_{CX}$	—	Printer controller mode off (VT52)	110
$^{E}S_{CY}$<r><c>	—	Cursor to <r> row <c> column (VT52)	109
$^{E}S_{CZ}$	—	Request primary device attributes (VT52); Reflection replies $^{E}S_{C}$/Z	110
$^{I}N_{D}$	IND	Index	36
$^{N}E_{L}$	NEL	Next line	36
^{R}I	RI	Reverse linefeed	36
SI	LS0	Map GØ into GL	74
SO	LS1	Map G1 into GL	74

Control Functions: Sorted by Mnemonic

Mnemonic	Function	Description	Location
CBT	^CS_I<n>Z	Cursor backtab	36
CPR	^CS_I6n	Request cursor position	90
CPR	^CS_I<row>;<column>R	Report cursor position	89
CUD	^CS_I<n>B	Cursor down	36
CUF	^CS_I<n>C	Cursor forward	35
CUB	^CS_I<n>D	Cursor backward	35
CUP	^CS_I<r>;<c>H	Cursor position	35
CUU	^CS_I<n>A	Cursor up	35
DA	^CS_Ic	Request primary device attributes	85
DA	^CS_IØc	Request primary device attributes	85
DA	^CS_IØx	Request terminal parameters	87
DA	^CS_I1x	Request terminal parameters	87
DA	^CS_I2;<n1>;<n2>;<rbaud>;<tbaud>;1;Øx	Response to ^CS_IØx	87
DA	^CS_I3;<n1>;<n2>;<rbaud>;<tbaud>;1;Øx	Response to ^CS_I1x	87
DA	^CS_I>c	Request secondary device attributes	88
DA	^CS_I>Øc	Request secondary device attributes	88
DA	^CS_I>24;11;Øc	Response to secondary device attributes	88

Mnemonic	Function	Description	Location
DCH	^{C}S$_I$<n>P	Delete <n> characters from cursor	44
DECALN	^{E}S$_C$#8	Test pattern	108
DECANM	^{C}S$_I$?2l	VT52 emulation	24, 109
DECARM	^{C}S$_I$?8h	Keyboard auto repeat on	47
DECARM	^{C}S$_I$?8l	Keyboard auto repeat off	47
DECAUPSS	^{D}C$_S$Ø!u%5S$_T$	Report UPSS is DEC Supplemental Graphic	105
DECAUPSS	^{D}C$_S$1!uAS$_T$	Report UPSS is ISO Latin-1 supplemental	105
DECAWM	^{C}S$_I$?7h	Autowrap on	48
DECAWM	^{C}S$_I$?7l	Autowrap off	48
DECBKM	^{C}S$_I$?67h	Backarrow key set to BS	49
DECBKM	^{C}S$_I$?67l	Backarrow key set to US	49
DECCARA ^{C}S$_I$<n1>;<n2>;<n3>;<n4>;<n5>;...<nn>$r		Change rectangular area attributes	
DECCIR	^{D}C$_S$1$uD....DS$_T$	Report cursor information	93
DECCKM	^{C}S$_I$?1h	Cursor keys application	48
DECCKM	^{C}S$_I$?1l	Cursor keys normal	48
DECCKSR	^{C}S$_I$?63;<n>n	Request memory checksum of macros	34
DECCOLM	^{C}S$_I$?3h	Columns 132	42
DECCOLM	^{C}S$_I$?3l	Columns 80	42
DECCRA ^{C}S$_I$<n1>;<n2>;<n3>;<n4>;<n5>;<n6>;<n7>;<n8>$v		Copy rectangular area	
DECCTR	^{D}C$_S$2$s<data>S$_T$	Report color table	91

Mnemonic	Function	Description	Location
DECDHL	ᴱS𝒸#3	Double-width and height line (top half)	28
DECDHL	ᴱS𝒸#4	Double-width and height line (bottom half)	28
DECDLD	ᴰ𝒸S1;1;2 {<SP>@Sᴛ	Clear downline loadable character set	83
DECDLD set		Downline loadable character	
	ᴰ𝒸SPfn;Pcn;Pe;PcmW;Pw;Pt{ Dscs UUUUUUUU/LLLLLLLL;...Sᴛ		
DECDMAC	ᴰ𝒸S<n1>;<n2>;<n3>!zD...DSt	Define macro (VT420)	59
DECDWL	ᴱS𝒸#6	Double-width and single height line	28
DECERA	ᶜSɪ<n1>;<n2>;<n3>;<n4>$z	Erase rectangular area	30
DECFRA	ᶜSɪ<n1>;<n2>;<n3>;<n4>$x	Fill rectangular area	31
DECGEPM	ᶜSɪ?43h	Expanded print mode	66
DECGEPM	ᶜSɪ?43l	Compressed print mode	66
DECGPBM	ᶜSɪ?46h	Print background	67
DECGPBM	ᶜSɪ?46l	Do not print background	67
DECGPCM	ᶜSɪ?44h	Print color mode	66
DECGPCM	ᶜSɪ?44l	Print monochrome mode	66
DECGPCS	ᶜSɪ?45h	Print color syntax set to RGB	66
DECGPCS	ᶜSɪ?45l	Print color syntax set to HLS	66
DECGRPM	ᶜSɪ?47h	Print rotated mode	66
DECGRPM	ᶜSɪ?47l	Print compressed mode	66
DECHCCM	ᶜSɪ?60h	Horizontal cursor coupling	38
DECHCCM	ᶜSɪ?60l	Horizontal cursor uncouple	38
DECIMAC	ᶜSɪ<n>*z	Invoke macro	60

Mnemonic	Function	Description	Location
DECKPAM	`ESC=`	Keypad mode application	48
DECKPNM	`ESC>`	Keypad mode normal	48
DECMSR	`CSI?62n`	Request macro report	60
DECNKM	`CSI?66h`	Numeric keypad mode application	48
DECNKM	`CSI?66l`	Numeric keypad mode numeric	48
DECNRCM	`CSI?42h`	Use national replacement set: 7-bit	78
DECNRCM	`CSI?42l`	Use national replacement set: 7-bit and 8-bit	78
DECOM	`CSI?6h`	Origin mode set	43
DECOM	`CSI?6l`	Origin mode reset	43
DECPCCM	`CSI?64h`	Page cursor coupling	39
DECPCCM	`CSI?64l`	Page cursor uncouple	39
DECPEX	`CSI?19h`	Print full screen	56
DECPEX	`CSI?19l`	Print scrolling region	56
DECPFF	`CSI?18h`	Send form feed after printing	57
DECPFF	`CSI?18l`	No form feed after printing	57
DECRARA		Reverse attributes in rectangular area	
`CSI?<n1>;<n2>;<n3>;<n4>;<n5>...<nn>$t`			
DECRC	`ESC8`	Restore cursor state	105
DECRPM	`CSI<n1>;<n2>$y`	Report ANSI private mode settings	102
DECRPM	`CSI?<n1>;<n2>$y`	Report DEC private mode settings	102
DECRPSS	`DCS<n>$rD...DST`	Report control function setting	104

Mnemonic	Function	Description	Location
DECRQCRA CSI<n1>;<n2>;<n3>;<n4>;<n5>;<n6>*y		Request memory checksum	
DECRQM	CSI?<n>$p	Request DEC private mode settings	100
DECRQM	CSI<n>$p	Request ANSI private mode settings	100
DECRQPSR	CSI1$w	Request cursor information report	93
DECRQPSR	CSI2$w	Request tab stop report	93
DECRQSS	DCS$q...ST	Request control function setting	103
DECRQTSR	CSI1$u	Request terminal state report	91
DECRQTSR	CSI2$u	Request color table report (VT340)	91
DECRQUPSS	CSI&u	Request UPSS	105
DECRSPS	DCS1$t....ST	Restore cursor from DECCIR	100
DECRSPS	DCS2$t....ST	Restore tab from DECTABSR	100
DECRSTS	DCS1$p....ST	Restore terminal state from DECTSR	92
DECRSTS	DCS2$p<data>ST	Restore color table	91
DECSACE	CSI<n>*x	Select attribute change extent	32
DECSASD	CSIØ$}	Top 24 lines = active display	42
DECSASD	CSI1$}	Status line = active display	42
DECSC	ESC7	Save cursor state	104
DECSCA	CSIØ"q	Select erasable character	46
DECSCA	CSI1"q	Select protected character	46

Mnemonic	Function	Description	Location
DECSCA	$^{C}S_{I}2"q$	Select erasable character	46
DECSCL	$^{C}S_{I}61"p$	VT102 emulation	23
DECSCL	$^{C}S_{I}62"p$	VT200 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}62;\varnothing"p$	VT200 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}62;1"p$	VT200 emulation: 7-bit controls	23
DECSCL	$^{C}S_{I}62;2"p$	VT200 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}63"p$	VT300 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}63;\varnothing"p$	VT300 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}63;1"p$	VT300 emulation: 7-bit controls	23
DECSCL	$^{C}S_{I}63;2"p$	VT300 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}64;1"p$	VT400 emulation: 7-bit controls	23
DECSCL	$^{C}S_{I}64"p$	VT400 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}64;0"p$	VT400 emulation: 8-bit controls	23
DECSCL	$^{C}S_{I}64;2"p$	VT400 emulation: 8-bit controls	23
DECSCL	$^{E}S_{C}<$	ANSI mode—exit VT52 mode	24, 109
DECSCLM	$^{C}S_{I}?4h$	Smooth scroll	40
DECSCLM	$^{C}S_{I}?4l$	Jump scroll	40

Mnemonic	Function	Description	Location
DECSCNM	$^{C}_{SI}$?5h	Inverse video	40
DECSCNM	$^{C}_{SI}$?5l	Normal video	40
DECSDM	$^{C}_{SI}$?80h	Disable sixel scrolling	65
DECSDM	$^{C}_{SI}$?80l	Enable sixel scrolling	65
DECSED	$^{C}_{SI}$?0J	Erase unprotected characters from cursor to end of screen	46
DECSED	$^{C}_{SI}$?1J	Erase unprotected characters from top of screen to cursor	46
DECSED	$^{C}_{SI}$?2J	Erase unprotected characters from screen	46
DECSEL	$^{C}_{SI}$?0K	Erase unprotected characters from cursor to end of line	46
DECSEL	$^{C}_{SI}$?1K	Erase unprotected characters from beginning of line to cursor	46
DECSEL	$^{C}_{SI}$?2K	Erase unprotected characters from line	46
DECSERA	$^{C}_{SI}$<n1>;<n2>;<n3>;<n4>${	Selective erase of rectangular area	31
DECSHORP	$^{C}_{SI}$0w	Print 10 characters per inch, 80 characters per row	57
DECSHORP	$^{C}_{SI}$1w	Print 10 characters per inch, 80 characters per row	57
DECSHORP	$^{C}_{SI}$2w	Print 12 characters per inch, 96 characters per row	57
DECSHORP	$^{C}_{SI}$4w	Print 16.5 characters per inch, 132 characters per row	57
DECSLPP	$^{C}_{SI}$24t	Select lines per page: 24 lines set to 6 pages	29

Mnemonic	Function	Description	Location
DECSLPP	CSI25t	Select lines per page: 25 lines set to 5 pages	29
DECSLPP	CSI36t	Select lines per page: 36 lines set to 4 pages	29
DECSLPP	CSI48t	Select lines per page: 48 lines set to 3 pages	29
DECSLPP	CSI72t	Select lines per page: 72 lines set to 2 pages	29
DECSLPP	CSI144t	Select lines per page: 144 lines set to 1 page	29
DECSSDT	CSIØ$~	Do not display status line	41
DECSSDT	CSI1$~	Status line: indicator	41
DECSSDT	CSI2$~	Status line: host writable	41
DECSTBM	CSI<t>;<b>r	Set top and bottom scrolling region margins	42
DECSTR	CSI!p	Reset to defaults (VT300 soft reset)	106
DECSWL	ESC#5	Single-width and height line	28
DECTABSR	DCS2$u....ST	Report tab stop	99
DECTCEM	CSI?25h	Cursor visible	34
DECTCEM	CSI?25l	Cursor invisible	34
DECTSR	DCS1$s....ST	Report terminal state	91
DECTST	CSIy	Terminal test (disconnect)	108
DECUDK	DCS<C>;<L>|<defn>ST	Load user-defined keys	52
DECVERP	CSI0z	Print 6 lines per inch, 63 lines per page	57
DECVERP	CSI1z	Print 6 lines per inch, 63 lines per page	57

Mnemonic	Function	Description	Location
DECVERP	^{C}S$_I$2z	Print 8 lines per inch, 84 lines per page	57
DECVERP	^{C}S$_I$3z	Print 12 lines per inch, 125 lines per page	57
DECVERP	^{C}S$_I$4z	Print 2 lines per inch, 21 lines per page	57
DECVERP	^{C}S$_I$5z	Print 3 lines per inch, 32 lines per page	57
DECVERP	^{C}S$_I$6z	Print 4 lines per inch, 42 lines per page	57
DECVCCM	^{C}S$_I$?61h	Vertical cursor coupling	39
DECVCCM	^{C}S$_I$?61l	Vertical cursor uncouple	39
DL	^{C}S$_I$<n>M	Delete <n> lines from cursor	44
DSR	^{C}S$_I$?25n	Request UDK status (VT200)	90
DSR	^{C}S$_I$5n	Report device malfunction	89
DSR	^{C}S$_I$?26n	Request keyboard dialect	91
DSR	^{C}S$_I$?27;<n>n	Report keyboard dialect	91
DSR	^{C}S$_I$?20n	Report UDKs locked	90
DSR	^{C}S$_I$?21n	Report UDKs unlocked	90
DSR	^{C}S$_I$?10n	Report printer ready	90
DSR	^{C}S$_I$?13n	Report no printer	90
DSR	^{C}S$_I$?15n	Request printer status	90
DSR	^{C}S$_I$5n	Request operating status	89
DSR	^{C}S$_I$0n	Report no device malfunction	89
DSR	^{C}S$_I$?50n	Report a locator device detected	89
DSR	^{C}S$_I$?53n	Report no locator device detected	89

Mnemonic	Function	Description	Location
DSR	$^C\text{S}_\text{I}$?55n	Request status of locator device	89
DSR	$^C\text{S}_\text{I}$?56n	Request type of locator device	89
DSR	$^C\text{S}_\text{I}$?57;Øn	Report no locator device connected	89
DSR	$^C\text{S}_\text{I}$?57;1n	Report locator device is a mouse	89
ECH	$^C\text{S}_\text{I}$<n>X	Erase <n> characters on line	45
ED	$^C\text{S}_\text{I}$1J	Erase from top of screen to cursor	45
ED	$^C\text{S}_\text{I}$ØJ	Erase from cursor to end of screen	45
ED	$^C\text{S}_\text{I}$2J	Erase entire screen	45
EL	$^C\text{S}_\text{I}$ØK	Erase from cursor to end of line	45
EL	$^C\text{S}_\text{I}$1K	Erase from beginning of line to cursor	46
EL	$^C\text{S}_\text{I}$2K	Erase entire line	45
HVP	$^C\text{S}_\text{I}$<r>;<c>f	Move cursor to horizontal and vertical position	35
ICH	$^C\text{S}_\text{I}$<n>@	Insert <n> characters	44
IL	$^C\text{S}_\text{I}$<n>L	Insert <n> lines	44
IND	$^I\text{N}_\text{D}$	Index	36
IRM	$^C\text{S}_\text{I}$4h	Insert mode	44
IRM	$^C\text{S}_\text{I}$4l	Replace mode	44
KAM	$^C\text{S}_\text{I}$2h	Keyboard lock	47
KAM	$^C\text{S}_\text{I}$2l	Keyboard unlock	47
LNM	$^C\text{S}_\text{I}$2Øh	Auto linefeed on	47
LNM	$^C\text{S}_\text{I}$2Øl	Auto linefeed off	47

Mnemonic	Function	Description	Location
LS0	^{S}I	Map GØ into GL	74
LS1	^{S}O	Map G1 into GL	74
LS1R	$^{E}S_{C}\sim$	Map G1 into GR	74
LS2	$^{E}S_{C}n$	Map G2 into GL	74
LS2R	$^{E}S_{C}\}$	Map G2 into GR	74
LS3	$^{E}S_{C}o$	Map G3 into GL	74
LS3R	$^{E}S_{C}|$	Map G3 into GR	74
MC	$^{C}S_{I}?5i$	Auto print on	58
MC	$^{C}S_{I}?4i$	Auto print off	58
MC	$^{C}S_{I}5i$	Printer controller mode on	58
MC	$^{C}S_{I}4i$	Printer controller mode off	58
MC	$^{C}S_{I}i$	Print screen	58
MC	$^{C}S_{I}Øi$	Print screen	58
MC	$^{C}S_{I}?Øi$	Send graphics to printer	67
MC	$^{C}S_{I}?1i$	Print line	58
MC	$^{C}S_{I}?2i$	Send graphics to host	67
MC	$^{C}S_{I}?i$	Send graphics to printer	67
MC	$^{C}S_{I}i$	Print screen	58
NEL	$^{E}S_{C}E$	Next line	36
NP	$^{C}S_{I}<n>U$	Next page	37
PP	$^{C}S_{I}<n>V$	Previous page	37
PPA	$^{C}S_{I}<n><SP>P$	Page position absolute	37
PPB	$^{C}S_{I}<n><SP>R$	Page position backward	38
PPR	$^{C}S_{I}<n><SP>Q$	Page position relative	37
RI	^{R}I	Reverse linefeed	36

Mnemonic	Function	Description	Location
RIS	^{E}S$_C$C	Reset to initial state (hard reset)	106
RM	^{C}S$_I$<n>;…<n>l	Reset ANSI private mode	103
RM	^{C}S$_I$?<n>;…<n>l	Reset DEC private mode	103
S7C1T	^{E}S$_C$<S$_P$>F	VT200 emulation: 7-bit controls	24
S8C1T	^{E}S$_C$<S$_P$>G	VT200 emulation: 8-bit controls	24
SCS	^{E}S$_C$(<chr>	Selects GØ	72
SCS	^{E}S$_C$)<chr>	Selects G1, 94-character set	72
SCS	^{E}S$_C$-<chr>	Selects G1, 96-character set	72
SCS	^{E}S$_C$*<chr>	Selects G2 (VT200), 94-character set	72
SCS	^{E}S$_C$+<chr>	Selects G3 (VT200), 94-character set	72
SCS	^{E}S$_C$.<chr>	Selects G2 (VT200), 96-character set	72
SCS	^{E}S$_C$/<chr>	Selects G3 (VT200), 96-character set	72
SD	^{C}S$_I$<n>T	Pan up	41
SGR	^{C}S$_I$<n>;...<n>m	Select graphic rendition (display enhancements)	27
SM	^{C}S$_I$?<n>;…<n>h	Set DEC private mode	102
SM	^{C}S$_I$<n>;…<n>h	Set ANSI private mode	102
SRM	^{C}S$_I$12h	Local echo off	41
SRM	^{C}S$_I$12l	Local echo on	41
SS2	^{E}S$_C$N	Map G2 into GL for next character	74

Mnemonic	Function	Description	Location
SS3	ESCO	Map G3 into GL for next character	74
SU	CSI\<n>S	Pan down	40
TBC	CSIg	Tab clear	108
TBC	CSI3g	Clear all tabs	108
WRQCMD	DCS1234;Ps{\<cmd>ST	Invoke Reflection command	19
WRQRQSN	CSIØ;1234c	Serial number request	19
—	CSI3h	Display controls on	43
—	CSI3l	Display controls off	43
—	DCSØp	Enter ReGIS at point command exited	60
—	DCSp	Enter ReGIS at point command exited	60
—	DCS1p	Enter ReGIS and begin new command	60
—	DCS2p	Enter ReGIS at point command exited	60
—	DCS3p	Enter ReGIS and begin new command	60
—	DCS\<Ø-9>;\<Ø-2>;P3;qs...sST	Sixel data format	63
—	ESC=	Keypad mode application (VT52)	110
—	ESC>	Keypad mode normal (VT52)	110
—	ESC]	Print screen (VT52)	110
—	ESC^	Auto print mode on (VT52)	110
—	ESC_	Auto print mode off (VT52)	110
—	ESCA	Cursor up (VT52)	109
—	ESCB	Cursor down (VT52)	109

Mnemonic	Function	Description	Location
—	ᴱₛCC	Cursor forward (VT52)	109
—	ᴱₛCD	Cursor backward (VT52)	109
—	ᴱₛC\	Exit ReGIS mode	60
—	ᴱₛCH	Set tab	108
—	ᴱₛCH	Home cursor (VT52)	109
—	ᴱₛCI	Reverse linefeed (VT52)	110
—	ᴱₛCJ	Erase from cursor to end of screen (VT52)	109
—	ᴱₛCK	Erase to end of line (VT52)	109
—	ᴱₛCV	Print cursor line (VT52)	110
—	ᴱₛCW	Printer controller mode on (VT52)	110
—	ᴱₛCX	Printer controller mode off (VT52)	110
—	ᴱₛCY<r><c>	Cursor to <r> row <c> column	
—	ᴱₛCZ	Request primary device attributes	

Control Functions: Sorted by Function

Character and Line Attributes

Character Sets

Cursor

Editing and Erasing

Graphics Functions

Keyboard

Operating Level

Printing

Reflection Control Functions

Requests and Reports

Resets and Tests

Screen Display

Selective Erase (VT200 and Above)

Tab Stops

Tektronix

VT52 Escape Sequences