XTerm Control Sequences

Edward Moy

University of California, Berkeley

Revised by

Stephen Gildea

X Consortium (1994)

Thomas Dickey

XFree86 Project (1996-2006)
invisible-island.net (2006-2017)
updated for XTerm Patch #329 (2017/06/11)

Definitions

`c`	The literal character *c*.
C	A single (required) character.
P_s	A single (usually optional) numeric parameter, composed of one of more digits.
P_m	A multiple numeric parameter composed of any number of single numeric parameters, separated by `;` character(s). Individual values for the parameters are listed with P_s .
P_t	A text parameter composed of printable characters.

Control Bytes, Characters, and Sequences

ECMA-48 (aka "ISO 6429") documents C1 (8-bit) and C0 (7-bit) codes. Those are respectively codes 128 to 159 and 0 to 31. ECMA-48 avoids referring to these codes as characters, because that term is associated with *graphic characters*. Instead, it uses “bytes” and “codes”, with occasional lapses to “characters” where the meaning cannot be mistaken.

Controls (including the escape code 27) are processed once:

- This means that a C1 control can be mistaken for badly-formed UTF-8 when the terminal runs in UTF-8 mode because C1 controls are valid *continuation bytes* of a UTF-8 encoded (multibyte) value.
- It is not possible to use a C1 control obtained from decoding the UTF-8 text, because that would require reprocessing the data. Consequently there is no ambiguity in the way this document uses the term “character” to refer to bytes in a control sequence.

The order of processing is a necessary consequence of the way ECMA-48 is designed:

- Each byte sent to the terminal can be unambiguously determined to fall into one of a few categories (C0, C1 and graphic characters).
- ECMA-48 is *modal*; once it starts processing a control sequence, the terminal continues until the sequence is complete, or some byte is found which is not allowed in the sequence.
- Intermediate, parameter and final bytes may use the same codes as graphic characters, but they are processed as part of a control sequence and are not actually graphic characters.
- Eight-bit controls can have intermediate, etc., bytes in the range 160 to 255. Those can be treated as their counterparts in the range 32 to 127.

- Single-byte controls can be handled separately from multi-byte control sequences because ECMA-48's rules are unambiguous.

  As a special case, ECMA-48 (section 9) mentions that the control functions shift-in and shift-out are allowed to occur within a 7-bit multibyte control sequence because those cannot alter the meaning of the control sequence.
- Some controls (such as OSC) introduce a string mode, which is ended on a ST (string terminator).

  Again, the terminal should accept single-byte controls within the string. However, *xterm* has a resource setting **brokenLinuxOSC** to allow recovery from applications which rely upon malformed palette sequences used by the Linux console.

C1 (8-Bit) Control Characters

The *xterm* program recognizes both 8-bit and 7-bit control characters. It generates 7-bit controls (by default) or 8-bit if S8C1T is enabled. The following pairs of 7-bit and 8-bit control characters are equivalent:

ESC D
: Index (IND is 0x84).

ESC E
: Next Line (NEL is 0x85).

ESC H
: Tab Set (HTS is 0x88).

ESC M
: Reverse Index (RI is 0x8d).

ESC N
: Single Shift Select of G2 Character Set (SS2 is 0x8e). This affects next character only.

ESC O
: Single Shift Select of G3 Character Set (SS3 is 0x8f). This affects next character only.

ESC P
: Device Control String (DCS is 0x90).

ESC V
: Start of Guarded Area (SPA is 0x96).

ESC W
: End of Guarded Area (EPA is 0x97).

ESC X
: Start of String (SOS is 0x98).

ESC Z
: Return Terminal ID (DECID is 0x9a). Obsolete form of CSI c (DA).

ESC [
: Control Sequence Introducer (CSI is 0x9b).

ESC \
: String Terminator (ST is 0x9c).

ESC]
: Operating System Command (OSC is 0x9d).

ESC ^
: Privacy Message (PM is 0x9e).

ESC _

Application Program Command (APC is 0x9f).

These control characters are used in the vtXXX emulation.

VT100 Mode

Most of these control sequences are standard VT102 control sequences, but there is support for later DEC VT terminals (i.e., VT220, VT320, VT420, VT510), as well as ISO 6429 and *aixterm* color controls. The only VT102 feature not supported is auto-repeat, since the only way X provides for this will affect all windows.

There are additional control sequences to provide *xterm*-dependent functions, such as the scrollbar or window size. Where the function is specified by DEC or ISO 6429, the code assigned to it is given in parentheses.

The escape codes to designate and invoke character sets are specified by ISO 2022 (see that document for a discussion of character sets).

Many of the features are optional; *xterm* can be configured and built without support for them.

Single-character functions

BEL	Bell (Ctrl-G).
BS	Backspace (Ctrl-H).
CR	Carriage Return (Ctrl-M).
ENQ	Return Terminal Status (Ctrl-E). Default response is an empty string, but may be overridden by a resource **answerbackString**.
FF	Form Feed or New Page (NP). (FF is Ctrl-L). FF is treated the same as LF.
LF	Line Feed or New Line (NL). (LF is Ctrl-J).
SI	Shift In (Ctrl-O) → Switch to Standard Character Set. This invokes the G0 character set (the default).
SO	Shift Out (Ctrl-N) → Switch to Alternate Character Set. This invokes the G1 character set.
SP	Space.
TAB	Horizontal Tab (HT) (Ctrl-I).
VT	Vertical Tab (Ctrl-K). This is treated the same as LF.

Controls beginning with ESC

This excludes controls where ESC is part of a 7-bit equivalent to 8-bit C1 controls, ordered by the final character(s).

ESC SP F	7-bit controls (S7C1T).
ESC SP G	8-bit controls (S8C1T).
ESC SP L	Set ANSI conformance level 1 (dpANS X3.134.1).
ESC SP M	Set ANSI conformance level 2 (dpANS X3.134.1).
ESC SP N	Set ANSI conformance level 3 (dpANS X3.134.1).
ESC # 3	DEC double-height line, top half (DECDHL).
ESC # 4	DEC double-height line, bottom half (DECDHL).
ESC # 5	DEC single-width line (DECSWL).
ESC # 6	DEC double-width line (DECDWL).

ESC # 8 DEC Screen Alignment Test (DECALN).

ESC % @ Select default character set. That is ISO 8859-1 (ISO 2022).

ESC % G Select UTF-8 character set (ISO 2022).

ESC (*C* Designate G0 Character Set (ISO 2022, VT100).

Final character *C* for designating 94-character sets. In this list, 0, A and B apply to VT100 and up, the remainder to VT220 and up. The VT220 character sets, together with the Portuguese character set are activated by the National Replacement Character controls. The A is a special case, since it is also activated by the VT300-control for British Latin-1 separately from the National Replacement Character controls.

C = 0 → DEC Special Character and Line Drawing Set.

C = < → DEC Supplementary (VT200).

C = % 5 → DEC Supplementary Graphics (VT300).

C = > → DEC Technical (VT300).

C = A → United Kingdom (UK).

C = B → United States (USASCII).

C = 4 → Dutch.

C = C or 5 → Finnish.

C = R or f → French.

C = Q or 9 → French Canadian (VT200, VT300).

C = K → German.

C = Y → Italian.

C = `, E or 6 → Norwegian/Danish.

C = % 6 → Portuguese (VT300).

C = Z → Spanish.

C = H or 7 → Swedish.

C = = → Swiss.

ESC) *C* Designate G1 Character Set (ISO 2022, VT100).

The same character sets apply as for ESC (*C*.

ESC * *C* Designate G2 Character Set (ISO 2022, VT220).

The same character sets apply as for ESC (*C*.

ESC + *C* Designate G3 Character Set (ISO 2022, VT220).

The same character sets apply as for ESC (*C*.

ESC - *C* Designate G1 Character Set (VT300).

The same character sets apply as for ESC (*C*.

ESC . *C* Designate G2 Character Set (VT300).

The same character sets apply as for ESC (*C*.

ESC / *C* Designate G3 Character Set (VT300).

These work for 96-character sets only.

C = A → ISO Latin-1 Supplemental.

ESC 6 Back Index (DECBI), VT420 and up.

ESC 7 Save Cursor (DECSC).
ESC 8 Restore Cursor (DECRC).
ESC 9 Forward Index (DECFI), VT420 and up.
ESC = Application Keypad (DECKPAM).
ESC > Normal Keypad (DECKPNM).
ESC F Cursor to lower left corner of screen. This is enabled by the **hpLowerleftBugCompat** resource.
ESC c Full Reset (RIS).
ESC l Memory Lock (per HP terminals). Locks memory above the cursor.
ESC m Memory Unlock (per HP terminals).
ESC n Invoke the G2 Character Set as GL (LS2).
ESC o Invoke the G3 Character Set as GL (LS3).
ESC | Invoke the G3 Character Set as GR (LS3R).
ESC } Invoke the G2 Character Set as GR (LS2R).
ESC ~ Invoke the G1 Character Set as GR (LS1R).

Application Program-Command functions

APC P_t ST None. *xterm* implements no APC functions; P_t is ignored. P_t need not be printable characters.

Device-Control functions

DCS P_s ; P_s | P_t ST

User-Defined Keys (DECUDK). The first parameter:

P_s = 0 → Clear all UDK definitions before starting (default).
P_s = 1 → Erase Below (default).

The second parameter:

P_s = 0 ← Lock the keys (default).
P_s = 1 ← Do not lock.

The third parameter is a ';'-separated list of strings denoting the key-code separated by a '/' from the hex-encoded key value. The key codes correspond to the DEC function-key codes (e.g., F6=17).

DCS $ q P_t ST

Request Status String (DECRQSS). The string following the "q" is one of the following:

" q → DECSCA
" p → DECSCL
r → DECSTBM
s → DECSLRM
m → SGR
SP q → DECSCUSR

xterm responds with DCS 1 $ r P_t ST for valid requests, replacing the P_t with the corresponding CSI string, or DCS 0 $ r P_t ST for invalid requests.

DCS + p P_t ST

Set Termcap/Terminfo Data (xterm, experimental). The string following the "p" is a name to use

for retrieving data from the terminal database. The data will be used for the "tcap" keyboard configuration's function- and special-keys, as well as by the Request Termcap/Terminfo String control.

DCS + q P_t ST

Request Termcap/Terminfo String (xterm, experimental). The string following the "q" is a list of names encoded in hexadecimal (2 digits per character) separated by ; which correspond to termcap or terminfo key names.

Two special features are also recognized, which are not key names: *Co* for termcap colors (or *colors* for terminfo colors), and *TN* for termcap name (or *name* for terminfo name).

xterm responds with DCS 1 + r P_t ST for valid requests, adding to P_t an =, and the value of the corresponding string that *xterm* would send, or DCS 0 + r P_t ST for invalid requests. The strings are encoded in hexadecimal (2 digits per character).

Functions using CSI, ordered by the final character(s)

CSI P_s @ Insert P_s (Blank) Character(s) (default = 1) (ICH).

CSI P_s A Cursor Up P_s Times (default = 1) (CUU).

CSI P_s B Cursor Down P_s Times (default = 1) (CUD).

CSI P_s C Cursor Forward P_s Times (default = 1) (CUF).

CSI P_s D Cursor Backward P_s Times (default = 1) (CUB).

CSI P_s E Cursor Next Line P_s Times (default = 1) (CNL).

CSI P_s F Cursor Preceding Line P_s Times (default = 1) (CPL).

CSI P_s G Cursor Character Absolute [column] (default = [row,1]) (CHA).

CSI P_s ; P_s H

Cursor Position [row;column] (default = [1,1]) (CUP).

CSI P_s I Cursor Forward Tabulation P_s tab stops (default = 1) (CHT).

CSI P_s J Erase in Display (ED).

- P_s = 0 → Erase Below (default).
- P_s = 1 → Erase Above.
- P_s = 2 → Erase All.
- P_s = 3 → Erase Saved Lines (xterm).

CSI ? P_s J Erase in Display (DECSED).

- P_s = 0 → Selective Erase Below (default).
- P_s = 1 → Selective Erase Above.
- P_s = 2 → Selective Erase All.
- P_s = 3 → Selective Erase Saved Lines (xterm).

CSI P_s K Erase in Line (EL).

- P_s = 0 → Erase to Right (default).
- P_s = 1 → Erase to Left.
- P_s = 2 → Erase All.

CSI ? P_s K Erase in Line (DECSEL).

- P_s = 0 → Selective Erase to Right (default).

P_s = 1 → Selective Erase to Left.
P_s = 2 → Selective Erase All.

CSI P_s L Insert P_s Line(s) (default = 1) (IL).

CSI P_s M Delete P_s Line(s) (default = 1) (DL).

CSI P_s P Delete P_s Character(s) (default = 1) (DCH).

CSI P_s S Scroll up P_s lines (default = 1) (SU).

CSI ? P_i ; P_a ; P_v S

If configured to support either **Sixel Graphics** or **ReGIS Graphics**, *xterm* accepts a three-parameter control sequence, where P_i, P_a and P_v are the *item*, *action* and *value*:

P_i = 1 → item is number of color registers.
P_i = 2 → item is Sixel graphics geometry (in pixels).
P_i = 3 → item is ReGIS graphics geometry (in pixels).

P_a = 1 → read
P_a = 2 → reset to default
P_a = 3 → set to value in P_v
P_a = 4 → read the maximum allowed value

P_v can be omitted except when setting (P_a == 3).
P_v = *n* ← A single integer is used for color registers.
P_v = *width* ; *height* ← Two integers for graphics geometry.

xterm replies with a control sequence of the same form:

CSI ? P_i ; P_s ; P_v S

where P_s is the status:
P_s = 0 → success.
P_s = 1 → error in P_i.
P_s = 2 → error in P_a.
P_s = 3 → failure.

On success, P_v represents the value read or set.

Notes:

- The current implementation allows reading the graphics sizes, but disallows modifying those sizes because that is done once, using resource-values.
- Graphics geometry is not necessarily the same as "window size" (see the dtterm window manipulation extensions). For example, *xterm* limits the maximum graphics geometry at compile time (1000x1000 as of version 328) although the window size can be larger.

- While resizing a window will always change the current graphics geometry, the reverse is not true. Setting graphics geometry does not affect the window size.

CSI P_s T Scroll down P_s lines (default = 1) (SD).

CSI P_s ; P_s ; P_s ; P_s ; P_s T

Initiate highlight mouse tracking. Parameters are [func;startx;starty;firstrow;lastrow]. See the section **Mouse Tracking**.

CSI > P_s ; P_s T

Reset one or more features of the title modes to the default value. Normally, "reset" disables the feature. It is possible to disable the ability to reset features by compiling a different default for the title modes into *xterm*.

P_s = 0 → Do not set window/icon labels using hexadecimal.

P_s = 1 → Do not query window/icon labels using hexadecimal.

P_s = 2 → Do not set window/icon labels using UTF-8.

P_s = 3 → Do not query window/icon labels using UTF-8. (See discussion of "Title Modes").

CSI P_s X Erase P_s Character(s) (default = 1) (ECH).

CSI P_s Z Cursor Backward Tabulation P_s tab stops (default = 1) (CBT).

CSI P_m ` Character Position Absolute [column] (default = [row,1]) (HPA).

CSI P_m a Character Position Relative [columns] (default = [row,col+1]) (HPR).

CSI P_s b Repeat the preceding graphic character P_s times (REP).

CSI P_s c Send Device Attributes (Primary DA).

P_s = 0 or omitted → request attributes from terminal. The response depends on the **decTerminalID** resource setting.

→ CSI ? 1 ; 2 c (“VT100 with Advanced Video Option”)

→ CSI ? 1 ; 0 c (“VT101 with No Options”)

→ CSI ? 6 c (“VT102”)

→ CSI ? 6 2 ; P_s c (“VT220”)

→ CSI ? 6 3 ; P_s c (“VT320”)

→ CSI ? 6 4 ; P_s c (“VT420”)

The VT100-style response parameters do not mean anything by themselves. VT220 (and higher) parameters do, telling the host what features the terminal supports:

P_s = 1 → 132-columns.

P_s = 2 → Printer.

P_s = 3 → ReGIS graphics.

P_s = 4 → Sixel graphics.

P_s = 6 → Selective erase.

P_s = 8 → User-defined keys.

P_s = 9 → National Replacement Character sets.

P_s = 1 5 → Technical characters.

P_s = 1 8 → User windows.

P_s = 2 1 → Horizontal scrolling.

P_s = 2 2 → ANSI color, e.g., VT525.

P_s = 2 9 → ANSI text locator (i.e., DEC Locator mode).

CSI > P_s c Send Device Attributes (Secondary DA).

P_s = 0 or omitted → request the terminal's identification code. The response depends on the **decTerminalID** resource setting. It should apply only to VT220 and up, but *xterm* extends this to VT100.

→ CSI > P_p ; P_v ; P_c c

where P_p denotes the terminal type

P_p = 0 → "VT100".

P_p = 1 → "VT220".

P_p = 2 → "VT240".

P_p = 1 8 → "VT330".

P_p = 1 9 → "VT340".

P_p = 2 4 → "VT320".

P_p = 4 1 → "VT420".

P_p = 6 1 → "VT510".

P_p = 6 4 → "VT520".

P_p = 6 5 → "VT525".

and P_v is the firmware version (for *xterm*, this was originally the XFree86 patch number, starting with 95). In a DEC terminal, P_c indicates the ROM cartridge registration number and is always zero.

CSI P_m d Line Position Absolute [row] (default = [1,column]) (VPA).

CSI P_m e Line Position Relative [rows] (default = [row+1,column]) (VPR).

CSI P_s ; P_s f

Horizontal and Vertical Position [row;column] (default = [1,1]) (HVP).

CSI P_s g Tab Clear (TBC).

P_s = 0 → Clear Current Column (default).

P_s = 3 → Clear All.

CSI P_m h Set Mode (SM).

P_s = 2 → Keyboard Action Mode (AM).

P_s = 4 → Insert Mode (IRM).

P_s = 1 2 → Send/receive (SRM).

P_s = 2 0 → Automatic Newline (LNM).

CSI ? P_m h DEC Private Mode Set (DECSET).

P_s = 1 → Application Cursor Keys (DECCKM).

P_s = 2 → Designate USASCII for character sets G0-G3 (DECANM), and set VT100 mode.

P_s = 3 → 132 Column Mode (DECCOLM).

P_s = 4 → Smooth (Slow) Scroll (DECSCLM).

P_s = 5 → Reverse Video (DECSCNM).

P_s = 6 → Origin Mode (DECOM).

P_s = 7 → Wraparound Mode (DECAWM).

P_s = 8 → Auto-repeat Keys (DECARM).

P_s = 9 → Send Mouse X & Y on button press. See the section **Mouse Tracking**. This is the X10 xterm mouse protocol.

P_s = 1 0 → Show toolbar (rxvt).

P_s = 1 2 → Start Blinking Cursor (att610).

P_s = 1 8 → Print form feed (DECPFF).

P_s = 1 9 → Set print extent to full screen (DECPEX).

P_s = 2 5 → Show Cursor (DECTCEM).

P_s = 3 0 → Show scrollbar (rxvt).

P_s = 3 5 → Enable font-shifting functions (rxvt).

P_s = 3 8 → Enter Tektronix Mode (DECTEK).

P_s = 4 0 → Allow 80 ↔ 132 Mode.

P_s = 4 1 → *more*(1) fix (see **curses** resource).

P_s = 4 2 → Enable National Replacement Character sets (DECNRCM).

P_s = 4 4 → Turn On Margin Bell.

P_s = 4 5 → Reverse-wraparound Mode.

P_s = 4 6 → Start Logging. This is normally disabled by a compile-time option.

P_s = 4 7 → Use Alternate Screen Buffer. (This may be disabled by the **titeInhibit** resource).

P_s = 6 6 → Application keypad (DECNKM).

P_s = 6 7 → Backarrow key sends backspace (DECBKM).

P_s = 6 9 → Enable left and right margin mode (DECLRMM), VT420 and up.

P_s = 9 5 → Do not clear screen when DECCOLM is set/reset (DECNCSM), VT510 and up.

P_s = 1 0 0 0 → Send Mouse X & Y on button press and release. See the section **Mouse Tracking**. This is the X11 xterm mouse protocol.

P_s = 1 0 0 1 → Use Hilite Mouse Tracking.

P_s = 1 0 0 2 → Use Cell Motion Mouse Tracking.

P_s = 1 0 0 3 → Use All Motion Mouse Tracking.

P_s = 1 0 0 4 → Send FocusIn/FocusOut events.

P_s = 1 0 0 5 → Enable UTF-8 Mouse Mode.

P_s = 1 0 0 6 → Enable SGR Mouse Mode.

P_s = 1 0 0 7 → Enable Alternate Scroll Mode, i.e., the **alternateScroll** resource.

P_s = 1 0 1 0 → Scroll to bottom on tty output (rxvt).

P_s = 1 0 1 1 → Scroll to bottom on key press (rxvt).

P_s = 1 0 1 5 → Enable urxvt Mouse Mode.

P_s = 1 0 3 4 → Interpret "meta" key, sets eighth bit. (enables the **eightBitInput** resource).

P_s = 1 0 3 5 → Enable special modifiers for Alt and NumLock keys. (This enables the **numLock** resource).

P_s = 1 0 3 6 → Send ESC when Meta modifies a key. (This enables the **metaSendsEscape** resource).

P_s = 1 0 3 7 → Send DEL from the editing-keypad Delete key.

P_s = 1 0 3 9 → Send ESC when Alt modifies a key. (This enables the **altSendsEscape** resource).

P_s = 1 0 4 0 → Keep selection even if not highlighted. (This enables the **keepSelection** resource).

P_s = 1 0 4 1 → Use the CLIPBOARD selection. (This enables the **selectToClipboard** resource).

P_s = 1 0 4 2 → Enable Urgency window manager hint when Control-G is received. (This enables the **bellIsUrgent** resource).

P_s = 1 0 4 3 → Enable raising of the window when Control-G is received. (enables the **popOnBell** resource).

P_s = 1 0 4 4 → Reuse the most recent data copied to CLIPBOARD. (This enables the **keepClipboard** resource).

P_s = 1 0 4 7 → Use Alternate Screen Buffer. (This may be disabled by the **titeInhibit** resource).

P_s = 1 0 4 8 → Save cursor as in DECSC. (This may be disabled by the **titeInhibit** resource).

P_s = 1 0 4 9 → Save cursor as in DECSC and use Alternate Screen Buffer, clearing it first. (This may be disabled by the **titeInhibit** resource). This combines the effects of the 1 0 4 7 and 1 0 4 8 modes. Use this with terminfo-based applications rather than the 4 7 mode.

P_s = 1 0 5 0 → Set terminfo/termcap function-key mode.

P_s = 1 0 5 1 → Set Sun function-key mode.

P_s = 1 0 5 2 → Set HP function-key mode.

P_s = 1 0 5 3 → Set SCO function-key mode.

P_s = 1 0 6 0 → Set legacy keyboard emulation (X11R6).

P_s = 1 0 6 1 → Set VT220 keyboard emulation.

P_s = 2 0 0 4 → Set bracketed paste mode.

CSI P_m i Media Copy (MC).

P_s = 0 → Print screen (default).

P_s = 4 → Turn off printer controller mode.

P_s = 5 → Turn on printer controller mode.

P_s = 1 0 → HTML screen dump.

P_s = 1 1 → SVG screen dump.

CSI ? P_m i Media Copy (MC, DEC-specific).

P_s = 1 → Print line containing cursor.

P_s = 4 → Turn off autoprint mode.

P_s = 5 → Turn on autoprint mode.

P_s = 1 0 → Print composed display, ignores DECPEX.

P_s = 1 1 → Print all pages.

CSI P_m l Reset Mode (RM).

P_s = 2 → Keyboard Action Mode (AM).

P_s = 4 → Replace Mode (IRM).
P_s = 1 2 → Send/receive (SRM).
P_s = 2 0 → Normal Linefeed (LNM).

CSI ? P_m l DEC Private Mode Reset (DECRST).

P_s = 1 → Normal Cursor Keys (DECCKM).
P_s = 2 → Designate VT52 mode (DECANM).
P_s = 3 → 80 Column Mode (DECCOLM).
P_s = 4 → Jump (Fast) Scroll (DECSCLM).
P_s = 5 → Normal Video (DECSCNM).
P_s = 6 → Normal Cursor Mode (DECOM).
P_s = 7 → No Wraparound Mode (DECAWM).
P_s = 8 → No Auto-repeat Keys (DECARM).
P_s = 9 → Don't send Mouse X & Y on button press.
P_s = 1 0 → Hide toolbar (rxvt).
P_s = 1 2 → Stop Blinking Cursor (att610).
P_s = 1 8 → Don't print form feed (DECPFF).
P_s = 1 9 → Limit print to scrolling region (DECPEX).
P_s = 2 5 → Hide Cursor (DECTCEM).
P_s = 3 0 → Don't show scrollbar (rxvt).
P_s = 3 5 → Disable font-shifting functions (rxvt).
P_s = 4 0 → Disallow 80 ↔ 132 Mode.
P_s = 4 1 → No *more*(1) fix (see **curses** resource).
P_s = 4 2 → Disable National Replacement Character sets (DECNRCM).
P_s = 4 4 → Turn Off Margin Bell.
P_s = 4 5 → No Reverse-wraparound Mode.
P_s = 4 6 → Stop Logging. (This is normally disabled by a compile-time option).
P_s = 4 7 → Use Normal Screen Buffer.
P_s = 6 6 → Numeric keypad (DECNKM).
P_s = 6 7 → Backarrow key sends delete (DECBKM).
P_s = 6 9 → Disable left and right margin mode (DECLRMM), VT420 and up.
P_s = 9 5 → Clear screen when DECCOLM is set/reset (DECNCSM), VT510 and up.
P_s = 1 0 0 0 → Don't send Mouse X & Y on button press and release. See the section **Mouse Tracking**.
P_s = 1 0 0 1 → Don't use Hilite Mouse Tracking.
P_s = 1 0 0 2 → Don't use Cell Motion Mouse Tracking.
P_s = 1 0 0 3 → Don't use All Motion Mouse Tracking.
P_s = 1 0 0 4 → Don't send FocusIn/FocusOut events.
P_s = 1 0 0 5 → Disable UTF-8 Mouse Mode.
P_s = 1 0 0 6 → Disable SGR Mouse Mode.
P_s = 1 0 0 7 → Disable Alternate Scroll Mode, i.e., the **alternateScroll** resource.
P_s = 1 0 1 0 → Don't scroll to bottom on tty output (rxvt).

P_s = 1 0 1 1 → Don't scroll to bottom on key press (rxvt).
P_s = 1 0 1 5 → Disable urxvt Mouse Mode.
P_s = 1 0 3 4 → Don't interpret "meta" key. (This disables the **eightBitInput** resource).
P_s = 1 0 3 5 → Disable special modifiers for Alt and NumLock keys. (This disables the **numLock** resource).
P_s = 1 0 3 6 → Don't send ESC when Meta modifies a key. (This disables the **metaSendsEscape** resource).
P_s = 1 0 3 7 → Send VT220 Remove from the editing-keypad Delete key.
P_s = 1 0 3 9 → Don't send ESC when Alt modifies a key. (This disables the **altSendsEscape** resource).
P_s = 1 0 4 0 → Do not keep selection when not highlighted. (This disables the **keepSelection** resource).
P_s = 1 0 4 1 → Use the PRIMARY selection. (This disables the **selectToClipboard** resource).
P_s = 1 0 4 2 → Disable Urgency window manager hint when Control-G is received. (This disables the **bellIsUrgent** resource).
P_s = 1 0 4 3 → Disable raising of the window when Control-G is received. (This disables the **popOnBell** resource).
P_s = 1 0 4 7 → Use Normal Screen Buffer, clearing screen first if in the Alternate Screen. (This may be disabled by the **titeInhibit** resource).
P_s = 1 0 4 8 → Restore cursor as in DECRC. (This may be disabled by the **titeInhibit** resource).
P_s = 1 0 4 9 → Use Normal Screen Buffer and restore cursor as in DECRC. (This may be disabled by the **titeInhibit** resource). This combines the effects of the 1 0 4 7 and 1 0 4 8 modes. Use this with terminfo-based applications rather than the 4 7 mode.
P_s = 1 0 5 0 → Reset terminfo/termcap function-key mode.
P_s = 1 0 5 1 → Reset Sun function-key mode.
P_s = 1 0 5 2 → Reset HP function-key mode.
P_s = 1 0 5 3 → Reset SCO function-key mode.
P_s = 1 0 6 0 → Reset legacy keyboard emulation (X11R6).
P_s = 1 0 6 1 → Reset keyboard emulation to Sun/PC style.
P_s = 2 0 0 4 → Reset bracketed paste mode.

CSI P_m m Character Attributes (SGR).
P_s = 0 → Normal (default).
P_s = 1 → Bold.
P_s = 2 → Faint, decreased intensity (ISO 6429).
P_s = 3 → Italicized (ISO 6429).
P_s = 4 → Underlined.
P_s = 5 → Blink (appears as Bold).
P_s = 7 → Inverse.

P_s = 8 → Invisible, i.e., hidden (VT300).
P_s = 9 → Crossed-out characters (ISO 6429).
P_s = 2 1 → Doubly-underlined (ISO 6429).
P_s = 2 2 → Normal (neither bold nor faint).
P_s = 2 3 → Not italicized (ISO 6429).
P_s = 2 4 → Not underlined.
P_s = 2 5 → Steady (not blinking).
P_s = 2 7 → Positive (not inverse).
P_s = 2 8 → Visible, i.e., not hidden (VT300).
P_s = 2 9 → Not crossed-out (ISO 6429).
P_s = 3 0 → Set foreground color to Black.
P_s = 3 1 → Set foreground color to Red.
P_s = 3 2 → Set foreground color to Green.
P_s = 3 3 → Set foreground color to Yellow.
P_s = 3 4 → Set foreground color to Blue.
P_s = 3 5 → Set foreground color to Magenta.
P_s = 3 6 → Set foreground color to Cyan.
P_s = 3 7 → Set foreground color to White.
P_s = 3 9 → Set foreground color to default (original).
P_s = 4 0 → Set background color to Black.
P_s = 4 1 → Set background color to Red.
P_s = 4 2 → Set background color to Green.
P_s = 4 3 → Set background color to Yellow.
P_s = 4 4 → Set background color to Blue.
P_s = 4 5 → Set background color to Magenta.
P_s = 4 6 → Set background color to Cyan.
P_s = 4 7 → Set background color to White.
P_s = 4 9 → Set background color to default (original).

If 16-color support is compiled, the following apply. Assume that *xterm*'s resources are set so that the ISO color codes are the first 8 of a set of 16. Then the *aixterm* colors are the bright versions of the ISO colors:

P_s = 9 0 → Set foreground color to Black.
P_s = 9 1 → Set foreground color to Red.
P_s = 9 2 → Set foreground color to Green.
P_s = 9 3 → Set foreground color to Yellow.
P_s = 9 4 → Set foreground color to Blue.
P_s = 9 5 → Set foreground color to Magenta.
P_s = 9 6 → Set foreground color to Cyan.
P_s = 9 7 → Set foreground color to White.
P_s = 1 0 0 → Set background color to Black.

P_s = 1 0 1 → Set background color to Red.
P_s = 1 0 2 → Set background color to Green.
P_s = 1 0 3 → Set background color to Yellow.
P_s = 1 0 4 → Set background color to Blue.
P_s = 1 0 5 → Set background color to Magenta.
P_s = 1 0 6 → Set background color to Cyan.
P_s = 1 0 7 → Set background color to White.

If *xterm* is compiled with the 16-color support disabled, it supports the following, from *rxvt*:
P_s = 1 0 0 → Set foreground and background color to default.

Xterm maintains a color palette whose entries are identified by an index beginning with zero. If 88- or 256-color support is compiled, the following apply:

- All parameters are decimal integers.
- RGB values range from zero (0) to 255.
- ISO-8613-3 can be interpreted in more than one way; *xterm* allows the semicolons in this control to be replaced by colons (but after the first colon, colons must be used).

These ISO-8613-3 controls are supported:

P_m = 3 8 ; 2 ; P_r ; P_g ; P_b → Set foreground color to the closest match in *xterm*'s palette for the given RGB P_r/P_g/P_b.

P_m = 3 8 ; 5 ; P_s → Set foreground color to P_s.

P_m = 4 8 ; 2 ; P_r ; P_g ; P_b → Set background color to the closest match in *xterm*'s palette for the given RGB P_r/P_g/P_b.

P_m = 4 8 ; 5 ; P_s → Set background color to P_s.

CSI > P_s ; P_s m

Set or reset resource-values used by *xterm* to decide whether to construct escape sequences holding information about the modifiers pressed with a given key. The first parameter identifies the resource to set/reset. The second parameter is the value to assign to the resource. If the second parameter is omitted, the resource is reset to its initial value.

P_s = 0 → modifyKeyboard.
P_s = 1 → modifyCursorKeys.
P_s = 2 → modifyFunctionKeys.
P_s = 4 → modifyOtherKeys.

If no parameters are given, all resources are reset to their initial values.

CSI P_s n Device Status Report (DSR).

P_s = 5 → Status Report.

Result ("OK") is CSI 0 n

P_s = 6 → Report Cursor Position (CPR) [row;column].

Result is CSI *r* ; *c* R

Note: it is possible for this sequence to be sent by a function key. For example, with the default keyboard configuration the shifted F1 key may send (with shift-, control-, alt-modifiers)

CSI 1 ; 2 R, or

CSI 1 ; 5 R, or

CSI 1 ; 6 R, etc.

The second parameter encodes the modifiers; values range from 2 to 16. See the section **PC-Style Function Keys** for the codes. The **modifyFunctionKeys** and **modifyKeyboard** resources can change the form of the string sent from the modified F1 key.

CSI > P_s n Disable modifiers which may be enabled via the CSI > P_s ; P_s m sequence. This corresponds to a resource value of "-1", which cannot be set with the other sequence. The parameter identifies the resource to be disabled:

P_s = 0 → modifyKeyboard.

P_s = 1 → modifyCursorKeys.

P_s = 2 → modifyFunctionKeys.

P_s = 4 → modifyOtherKeys.

If the parameter is omitted, **modifyFunctionKeys** is disabled. When **modifyFunctionKeys** is disabled, *xterm* uses the modifier keys to make an extended sequence of functions rather than adding a parameter to each function key to denote the modifiers.

CSI ? P_s n Device Status Report (DSR, DEC-specific).

P_s = 6 → Report Cursor Position (DECXCPR) [row;column] as CSI ? r ; c R (assumes the default page, i.e., "1").

P_s = 1 5 → Report Printer status as CSI ? 1 0 n (ready). or CSI ? 1 1 n (not ready).

P_s = 2 5 → Report UDK status as CSI ? 2 0 n (unlocked) or CSI ? 2 1 n (locked).

P_s = 2 6 → Report Keyboard status as
CSI ? 2 7 ; 1 ; 0 ; 0 n (North American).

The last two parameters apply to VT400 & up, and denote keyboard ready and LK01 respectively.

P_s = 5 3 → Report Locator status as CSI ? 5 3 n Locator available, if compiled-in, or CSI ? 5 0 n No Locator, if not.

P_s = 5 5 → Report Locator status as CSI ? 5 3 n Locator available, if compiled-in, or CSI ? 5 0 n No Locator, if not.

P_s = 5 6 → Report Locator type as CSI ? 5 7 ; 1 n Mouse, if compiled-in, or CSI ? 5 7 ; 0 n Cannot identify, if not.

P_s = 6 2 → Report macro space (DECMSR) as CSI P_n * { .

P_s = 6 3 → Report memory checksum (DECCKSR) as DCS P_t ! x x x x ST .

P_t is the request id (from an optional parameter to the request).

The x's are hexadecimal digits 0-9 and A-F.

P_s = 7 5 → Report data integrity as CSI ? 7 0 n (ready, no errors).

P_s = 8 5 → Report multi-session configuration as CSI ? 8 3 n (not configured for

multiple-session operation).

CSI > P_s p Set resource value *pointerMode*. This is used by *xterm* to decide whether to hide the pointer cursor as the user types. Valid values for the parameter:

P_s = 0 → never hide the pointer.

P_s = 1 → hide if the mouse tracking mode is not enabled.

P_s = 2 → always hide the pointer, except when leaving the window.

P_s = 3 → always hide the pointer, even if leaving/entering the window. If no parameter is given, *xterm* uses the default, which is 1.

CSI ! p Soft terminal reset (DECSTR).

CSI P_s ; P_s " p

Set conformance level (DECSCL). Valid values for the first parameter:

P_s = 6 1 → VT100.

P_s = 6 2 → VT200.

P_s = 6 3 → VT300.

Valid values for the second parameter:

P_s = 0 → 8-bit controls.

P_s = 1 → 7-bit controls (always set for VT100).

P_s = 2 → 8-bit controls.

CSI P_s $ p

Request ANSI mode (DECRQM). For VT300 and up, reply is

CSI P_s ; P_m $ y

where P_s is the mode number as in RM, and P_m is the mode value:

0 - not recognized

1 - set

2 - reset

3 - permanently set

4 - permanently reset

CSI ? P_s $ p

Request DEC private mode (DECRQM). For VT300 and up, reply is

CSI ? P_s ; P_m $ y

where P_s is the mode number as in DECSET, P_m is the mode value as in the ANSI DECRQM.

CSI P_s q Load LEDs (DECLL).

P_s = 0 → Clear all LEDS (default).

P_s = 1 → Light Num Lock.

P_s = 2 → Light Caps Lock.

P_s = 3 → Light Scroll Lock.

P_s = 2 1 → Extinguish Num Lock.

P_s = 2 2 → Extinguish Caps Lock.

P_s = 2 3 → Extinguish Scroll Lock.

CSI P_s SP q Set cursor style (DECSCUSR, VT520).

P_s = 0 → blinking block.

P_s = 1 → blinking block (default).
P_s = 2 → steady block.
P_s = 3 → blinking underline.
P_s = 4 → steady underline.
P_s = 5 → blinking bar (xterm).
P_s = 6 → steady bar (xterm).

CSI P_s " q Select character protection attribute (DECSCA). Valid values for the parameter:
P_s = 0 → DECSED and DECSEL can erase (default).
P_s = 1 → DECSED and DECSEL cannot erase.
P_s = 2 → DECSED and DECSEL can erase.

CSI P_s ; P_s r
Set Scrolling Region [top;bottom] (default = full size of window) (DECSTBM).

CSI ? P_m r Restore DEC Private Mode Values. The value of P_s previously saved is restored. P_s values are the same as for DECSET.

CSI P_t ; P_l ; P_b ; P_r ; P_s $ r
Change Attributes in Rectangular Area (DECCARA), VT400 and up.
P_t ; P_l ; P_b ; P_r denotes the rectangle.
P_s denotes the SGR attributes to change: 0, 1, 4, 5, 7.

CSI s Save cursor (ANSI.SYS), available only when DECLRMM is disabled.

CSI P_l ; P_r s
Set left and right margins (DECSLRM), available only when DECLRMM is enabled (VT420 and up).

CSI ? P_m s Save DEC Private Mode Values. P_s values are the same as for DECSET.

CSI P_s ; P_s ; P_s t
Window manipulation (from *dtterm*, as well as extensions). These controls may be disabled using the *allowWindowOps* resource. Valid values for the first (and any additional parameters) are:
P_s = 1 → De-iconify window.
P_s = 2 → Iconify window.
P_s = 3 ; x ; y → Move window to [x, y].
P_s = 4 ; *height* ; *width* → Resize the *xterm* window to given height and width in pixels. Omitted parameters reuse the current height or width. Zero parameters use the display's height or width.
P_s = 5 → Raise the *xterm* window to the front of the stacking order.
P_s = 6 → Lower the *xterm* window to the bottom of the stacking order.
P_s = 7 → Refresh the *xterm* window.
P_s = 8 ; *height* ; *width* → Resize the text area to given height and width in characters. Omitted parameters reuse the current height or width. Zero parameters use the display's height or width.
P_s = 9 ; 0 → Restore maximized window.
P_s = 9 ; 1 → Maximize window (i.e., resize to screen size).
P_s = 9 ; 2 → Maximize window vertically.

P_s = 9 ; 3 → Maximize window horizontally.
P_s = 1 0 ; 0 → Undo full-screen mode.
P_s = 1 0 ; 1 → Change to full-screen.
P_s = 1 0 ; 2 → Toggle full-screen.
P_s = 1 1 → Report *xterm* window state. If the *xterm* window is open (non-iconified), it returns CSI 1 t. If the *xterm* window is iconified, it returns CSI 2 t.
P_s = 1 3 → Report *xterm* window position.
Result is CSI 3 ; *x* ; *y* t
P_s = 1 4 → Report *xterm* window in pixels.
Result is CSI 4 ; *height* ; *width* t
P_s = 1 8 → Report the size of the text area in characters.
Result is CSI 8 ; *height* ; *width* t
P_s = 1 9 → Report the size of the screen in characters.
Result is CSI 9 ; *height* ; *width* t
P_s = 2 0 → Report *xterm* window's icon label.
Result is OSC L *label* ST
P_s = 2 1 → Report *xterm* window's title.
Result is OSC l *label* ST
P_s = 2 2 ; 0 → Save *xterm* icon and window title on stack.
P_s = 2 2 ; 1 → Save *xterm* icon title on stack.
P_s = 2 2 ; 2 → Save *xterm* window title on stack.
P_s = 2 3 ; 0 → Restore *xterm* icon and window title from stack.
P_s = 2 3 ; 1 → Restore *xterm* icon title from stack.
P_s = 2 3 ; 2 → Restore *xterm* window title from stack.
P_s >= 2 4 → Resize to P_s lines (DECSLPP).

CSI > P_s ; P_s t

Set one or more features of the title modes. Each parameter enables a single feature.

P_s = 0 → Set window/icon labels using hexadecimal.
P_s = 1 → Query window/icon labels using hexadecimal.
P_s = 2 → Set window/icon labels using UTF-8.
P_s = 3 → Query window/icon labels using UTF-8. (See discussion of "Title Modes")

CSI P_s SP t Set warning-bell volume (DECSWBV, VT520).

P_s = 0 or 1 → off.
P_s = 2, 3 or 4 → low.
P_s = 5, 6, 7, or 8 → high.

CSI P_t ; P_l ; P_b ; P_r ; P_s $ t

Reverse Attributes in Rectangular Area (DECRARA), VT400 and up.

P_t ; P_l ; P_b ; P_r denotes the rectangle.
P_s denotes the attributes to reverse, i.e., 1, 4, 5, 7.

CSI u Restore cursor (ANSI.SYS).

CSI P_s SP u Set margin-bell volume (DECSMBV, VT520).

P_s = 1 → off.

P_s = 2, 3 or 4 → low.

P_s = 0, 5, 6, 7, or 8 → high.

CSI P_t ; P_l ; P_b ; P_r ; P_p ; P_t ; P_l ; P_p $ v

Copy Rectangular Area (DECCRA, VT400 and up).

P_t ; P_l ; P_b ; P_r denotes the rectangle.

P_p denotes the source page.

P_t ; P_l denotes the target location.

P_p denotes the target page.

CSI P_t ; P_l ; P_b ; P_r ' w

Enable Filter Rectangle (DECEFR), VT420 and up.

Parameters are [top;left;bottom;right].

Defines the coordinates of a filter rectangle and activates it. Anytime the locator is detected outside of the filter rectangle, an outside rectangle event is generated and the rectangle is disabled. Filter rectangles are always treated as "one-shot" events. Any parameters that are omitted default to the current locator position. If all parameters are omitted, any locator motion will be reported. DECELR always cancels any prevous rectangle definition.

CSI P_s x Request Terminal Parameters (DECREQTPARM).

if P_s is a "0" (default) or "1", and *xterm* is emulating VT100, the control sequence elicits a response of the same form whose parameters describe the terminal:

P_s → the given P_s incremented by 2.

P_n = 1 ← no parity.

P_n = 1 ← eight bits.

P_n = 1 ← 2 8 transmit 38.4k baud.

P_n = 1 ← 2 8 receive 38.4k baud.

P_n = 1 ← clock multiplier.

P_n = 0 ← STP flags.

CSI P_s * x Select Attribute Change Extent (DECSACE).

P_s = 0 → from start to end position, wrapped.

P_s = 1 → from start to end position, wrapped.

P_s = 2 → rectangle (exact).

CSI P_c ; P_t ; P_l ; P_b ; P_r $ x

Fill Rectangular Area (DECFRA), VT420 and up.

P_c is the character to use.

P_t ; P_l ; P_b ; P_r denotes the rectangle.

CSI P_i ; P_g ; P_t ; P_l ; P_b ; P_r * y

Request Checksum of Rectangular Area (DECRQCRA), VT420 and up. Response is

DCS P_i ! x x x x ST

P_i is the request id.

P_g is the page number.

P_t ; P_l ; P_b ; P_r denotes the rectangle.

The x's are hexadecimal digits 0-9 and A-F.

CSI P_s ; P_u ' z

Enable Locator Reporting (DECELR).

Valid values for the first parameter:

P_s = 0 → Locator disabled (default).

P_s = 1 → Locator enabled.

P_s = 2 → Locator enabled for one report, then disabled.

The second parameter specifies the coordinate unit for locator reports.

Valid values for the second parameter:

P_u = 0 ← or omitted → default to character cells.

P_u = 1 ← device physical pixels.

P_u = 2 ← character cells.

CSI P_t ; P_l ; P_b ; P_r $ z

Erase Rectangular Area (DECERA), VT400 and up.

P_t ; P_l ; P_b ; P_r denotes the rectangle.

CSI P_m ' { Select Locator Events (DECSLE).

Valid values for the first (and any additional parameters) are:

P_s = 0 → only respond to explicit host requests (DECRQLP).
(This is default). It also cancels any filter
rectangle.

P_s = 1 → report button down transitions.

P_s = 2 → do not report button down transitions.

P_s = 3 → report button up transitions.

P_s = 4 → do not report button up transitions.

CSI P_t ; P_l ; P_b ; P_r $ {

Selective Erase Rectangular Area (DECSERA), VT400 and up.

P_t ; P_l ; P_b ; P_r denotes the rectangle.

CSI P_s ' | Request Locator Position (DECRQLP).

Valid values for the parameter are:

P_s = 0, 1 or omitted → transmit a single DECLRP locator report.

If Locator Reporting has been enabled by a DECELR, *xterm* will respond with a DECLRP Locator Report. This report is also generated on button up and down events if they have been enabled with a DECSLE, or when the locator is detected outside of a filter rectangle, if filter rectangles have been enabled with a DECEFR.

→ CSI P_e ; P_b ; P_r ; P_c ; P_p & w

Parameters are [*event*;*button*;*row*;*column*;*page*].

Valid values for the event:

P_e = 0 → locator unavailable - no other parameters sent.

P_e = 1 → request - *xterm* received a DECRQLP.
P_e = 2 → left button down.
P_e = 3 → left button up.
P_e = 4 → middle button down.
P_e = 5 → middle button up.
P_e = 6 → right button down.
P_e = 7 → right button up.
P_e = 8 → M4 button down.
P_e = 9 → M4 button up.
P_e = 1 0 → locator outside filter rectangle.

The "*button*" parameter is a bitmask indicating which buttons are pressed:

P_b = 0 ← no buttons down.
P_b & 1 ← right button down.
P_b & 2 ← middle button down.
P_b & 4 ← left button down.
P_b & 8 ← M4 button down.

The "*row*" and "*column*" parameters are the coordinates of the locator position in the *xterm* window, encoded as ASCII decimal.

The "*page*" parameter is not used by xterm.

CSI P_m ' } Insert P_s Column(s) (default = 1) (DECIC), VT420 and up.
CSI P_m ' ~ Delete P_s Column(s) (default = 1) (DECDC), VT420 and up.

Operating System Commands

OSC P_s ; P_t BEL
OSC P_s ; P_t ST

Set Text Parameters. For colors and font, if P_t is a "?", the control sequence elicits a response which consists of the control sequence which would set the corresponding value. The *dtterm* control sequences allow you to determine the icon name and window title.

P_s = 0 → Change Icon Name and Window Title to P_t.
P_s = 1 → Change Icon Name to P_t.
P_s = 2 → Change Window Title to P_t.
P_s = 3 → Set X property on top-level window. P_t should be in the form "*prop=value*", or just "*prop*" to delete the property
P_s = 4 ; *c* ; *spec* → Change Color Number *c* to the color specified by *spec*. This can be a name or RGB specification as per *XParseColor*. Any number of *c*/*spec* pairs may be given. The color numbers correspond to the ANSI colors 0-7, their bright versions 8-15, and if supported, the remainder of the 88-color or 256-color table.

If a "?" is given rather than a name or RGB specification, *xterm* replies with a control sequence of the same form which can be used to set the corresponding color. Because more than one pair of color number and specification can be given in one control sequence, *xterm* can make more than

one reply.

P_s = 5 ; *c* ; *spec* → Change Special Color Number *c* to the color specified by *spec*. This can be a name or RGB specification as per *XParseColor*. Any number of *c*/*spec* pairs may be given. The special colors can also be set by adding the maximum number of colors to these codes in an OSC 4 control:

P_c = 0 ← resource **colorBD** (BOLD).
P_c = 1 ← resource **colorUL** (UNDERLINE).
P_c = 2 ← resource **colorBL** (BLINK).
P_c = 3 ← resource **colorRV** (REVERSE).
P_c = 4 ← resource **colorIT** (ITALIC).

P_s = 6 ; *c* ; *f* → Enable/disable Special Color Number *c*. OSC 6 is the same as OSC 1 0 6.

The 10 colors (below) which may be set or queried using 1 0 through 1 9 are denoted *dynamic colors*, since the corresponding control sequences were the first means for setting *xterm*'s colors dynamically, i.e., after it was started. They are not the same as the ANSI colors. These controls may be disabled using the *allowColorOps* resource. At least one parameter is expected for P_t. Each successive parameter changes the next color in the list. The value of P_s tells the starting point in the list. The colors are specified by name or RGB specification as per *XParseColor*.

If a "?" is given rather than a name or RGB specification, *xterm* replies with a control sequence of the same form which can be used to set the corresponding dynamic color. Because more than one pair of color number and specification can be given in one control sequence, *xterm* can make more than one reply.

P_s = 1 0 → Change VT100 text foreground color to P_t.
P_s = 1 1 → Change VT100 text background color to P_t.
P_s = 1 2 → Change text cursor color to P_t.
P_s = 1 3 → Change mouse foreground color to P_t.
P_s = 1 4 → Change mouse background color to P_t.
P_s = 1 5 → Change Tektronix foreground color to P_t.
P_s = 1 6 → Change Tektronix background color to P_t.
P_s = 1 7 → Change highlight background color to P_t.
P_s = 1 8 → Change Tektronix cursor color to P_t.
P_s = 1 9 → Change highlight foreground color to P_t.

P_s = 4 6 → Change Log File to P_t. (This is normally disabled by a compile-time option).

P_s = 5 0 → Set Font to P_t. These controls may be disabled using the *allowFontOps* resource. If P_t begins with a "#", index in the font menu, relative (if the next character is a plus or minus sign) or absolute. A number is expected but not required after the sign (the default is the current entry for relative, zero for absolute indexing).

The same rule (plus or minus sign, optional number) is used when querying the font. The remainder of P_t is ignored.
A font can be specified after a "#" index expression, by adding a space and then the font specifier.
If the "TrueType Fonts" menu entry is set (the **renderFont** resource), then this control sets/queries the **faceName** resource.

P_s = 5 1 → reserved for Emacs shell.

P_s = 5 2 → Manipulate Selection Data. These controls may be disabled using the *allowWindowOps* resource. The parameter P_t is parsed as

P_c ; P_d

The first, P_c, may contain zero or more characters from the set c p s 0 1 2 3 4 5 6 7. It is used to construct a list of selection parameters for clipboard, primary, select, or cut buffers 0 through 7 respectively, in the order given. If the parameter is empty, *xterm* uses s 0, to specify the configurable primary/clipboard selection and cut buffer 0.
The second parameter, P_d, gives the selection data. Normally this is a string encoded in base64. The data becomes the new selection, which is then available for pasting by other applications.
If the second parameter is a ?, *xterm* replies to the host with the selection data encoded using the same protocol.
If the second parameter is neither a base64 string nor ?, then the selection is cleared.

P_s = 1 0 4 ; c → Reset Color Number c. It is reset to the color specified by the corresponding X resource. Any number of c parameters may be given. These parameters correspond to the ANSI colors 0-7, their bright versions 8-15, and if supported, the remainder of the 88-color or 256-color table. If no parameters are given, the entire table will be reset.

P_s = 1 0 5 ; c → Reset Special Color Number c. It is reset to the color specified by the corresponding X resource. Any number of c parameters may be given. These parameters correspond to the special colors which can be set using an OSC 5 control (or by adding the maximum number of colors using an OSC 4 control).

P_s = 1 0 6 ; c ; f → Enable/disable Special Color Number c. The second parameter tells *xterm* to enable the corresponding color mode if nonzero, disable it if zero.

P_c = 0 ← resource **colorBDMode** (BOLD).
P_c = 1 ← resource **colorULMode** (UNDERLINE).
P_c = 2 ← resource **colorBLMode** (BLINK).
P_c = 3 ← resource **colorRVMode** (REVERSE).
P_c = 4 ← resource **colorITMode** (ITALIC).
P_c = 5 ← resource **colorAttrMode** (Override ANSI).

The *dynamic colors* can also be reset to their default (resource) values:
P_s = 1 1 0 → Reset VT100 text foreground color.
P_s = 1 1 1 → Reset VT100 text background color.

P_s = 1 1 2 → Reset text cursor color.
P_s = 1 1 3 → Reset mouse foreground color.
P_s = 1 1 4 → Reset mouse background color.
P_s = 1 1 5 → Reset Tektronix foreground color.
P_s = 1 1 6 → Reset Tektronix background color.
P_s = 1 1 7 → Reset highlight color.
P_s = 1 1 8 → Reset Tektronix cursor color.
P_s = 1 1 9 → Reset highlight foreground color.

Privacy Message

PM P_t ST *xterm* implements no PM functions; P_t is ignored. P_t need not be printable characters.

Alt and Meta Keys

Many keyboards have keys labeled "Alt". Few have keys labeled "Meta". However, *xterm*'s default translations use the *Meta* modifier. Common keyboard configurations assign the *Meta* modifier to an "Alt" key. By using *xmodmap* one may have the modifier assigned to a different key, and have "real" alt and meta keys. Here is an example:

```
! put meta on mod3 to distinguish it from alt
keycode 64 = Alt_L
clear mod1
add mod1 = Alt_L
keycode 115 = Meta_L
clear mod3
add mod3 = Meta_L
```

The **metaSendsEscape** resource (and **altSendsEscape** if **altIsNotMeta** is set) can be used to control the way the *Meta* modifier applies to ordinary keys unless the **modifyOtherKeys** resource is set:

- prefix a key with the ESC character.

- shift the key from codes 0-127 to 128-255 by adding 128.

The table shows the result for a given character "x" with modifiers according to the default translations with the resources set on or off. This assumes **altIsNotMeta** is set:

key	altSendsEscape	metaSendsEscape	result
x	off	off	x
Meta-x	off	off	shift
Alt-x	off	off	shift
Alt+Meta-x	off	off	shift
x	ON	off	x
Meta-x	ON	off	shift
Alt-x	ON	off	ESC x
Alt+Meta-x	ON	off	ESC shift
x	off	ON	x
Meta-x	off	ON	ESC x
Alt-x	off	ON	shift
Alt+Meta-x	off	ON	ESC shift
x	ON	ON	x
Meta-x	ON	ON	ESC x
Alt-x	ON	ON	ESC x
Alt+Meta-x	ON	ON	ESC x

PC-Style Function Keys

If *xterm* does minimal translation of the function keys, it usually does this with a PC-style keyboard, so PC-style function keys result. Sun keyboards are similar to PC keyboards. Both have cursor and scrolling operations printed on the keypad, which duplicate the smaller cursor and scrolling keypads.

X does not predefine NumLock (used for VT220 keyboards) or Alt (used as an extension for the Sun/PC keyboards) as modifiers. These keys are recognized as modifiers when enabled by the **numLock** resource, or by the "DECSET 1 0 3 5 " control sequence.

The cursor keys transmit the following escape sequences depending on the mode specified via the **DECCKM** escape sequence.

Key	Normal	Application
Cursor Up	CSI A	SS3 A
Cursor Down	CSI B	SS3 B
Cursor Right	CSI C	SS3 C
Cursor Left	CSI D	SS3 D

The home- and end-keys (unlike PageUp and other keys also on the 6-key editing keypad) are considered "cursor keys" by *xterm*. Their mode is also controlled by the **DECCKM** escape sequence:

Key	Normal	Application
Home	CSI H	SS3 H
End	CSI F	SS3 F

The application keypad transmits the following escape sequences depending on the mode specified via the **DECKPNM** and **DECKPAM** escape sequences. Use the NumLock key to override the application mode.

Not all keys are present on the Sun/PC keypad (e.g., PF1, Tab), but are supported by the program.

Key	Numeric	Application	Terminfo	Termcap
Space	SP	SS3 SP	-	-
Tab	TAB	SS3 I	-	-
Enter	CR	SS3 M	kent	@8
PF1	SS3 P	SS3 P	kf1	k1
PF2	SS3 Q	SS3 Q	kf2	k2
PF3	SS3 R	SS3 R	kf3	k3

Key	Numeric	Application	Terminfo	Termcap
PF4	SS3 S	SS3 S	kf4	k4
* (multiply)	*	SS3 j	-	-
+ (add)	+	SS3 k	-	-
, (comma)	,	SS3 l	-	-
- (minus)	-	SS3 m	-	-
. (Delete)	.	CSI 3 ~	-	-
/ (divide)	/	SS3 o	-	-
0 (Insert)	0	CSI 2 ~	-	-
1 (End)	1	SS3 F	kc1	K4
2 (DownArrow)	2	CSI B	-	-
3 (PageDown)	3	CSI 6 ~	kc3	K5
4 (LeftArrow)	4	CSI D	-	-
5 (Begin)	5	CSI E	kb2	K2
6 (RightArrow)	6	CSI C	-	-
7 (Home)	7	SS3 H	ka1	K1
8 (UpArrow)	8	CSI A	-	-
9 (PageUp)	9	CSI 5 ~	ka3	K3
= (equal)	=	SS3 X	-	-

They also provide 12 function keys, as well as a few other special-purpose keys:

Key	Escape Sequence
F1	SS3 P
F2	SS3 Q
F3	SS3 R
F4	SS3 S
F5	CSI 1 5 ~
F6	CSI 1 7 ~
F7	CSI 1 8 ~
F8	CSI 1 9 ~
F9	CSI 2 0 ~
F10	CSI 2 1 ~
F11	CSI 2 3 ~
F12	CSI 2 4 ~

Note that F1 through F4 are prefixed with SS3, while the other keys are prefixed with CSI. Older versions of *xterm* implement different escape sequences for F1 through F4, with a CSI prefix. These can be activated by setting the **oldXtermFKeys** resource. However, since they do not correspond to any hardware terminal, they have been deprecated. (The DEC VT220 reserves F1 through F5 for local functions such as **Setup**).

Key	Escape Sequence
F1	CSI 1 1 ~
F2	CSI 1 2 ~
F3	CSI 1 3 ~
F4	CSI 1 4 ~

In normal mode, i.e., a Sun/PC keyboard when the **sunKeyboard** resource is false (and none of the other keyboard resources such as **oldXtermFKeys** resource is set), *xterm* encodes function key modifiers as parameters appended before the *final* character of the control sequence. As a special case, the SS3 sent before F1 through F4 is altered to CSI when sending a function key modifier as a parameter.

Code	Modifiers
2	Shift
3	Alt

Code	Modifiers
4	Shift + Alt
5	Control
6	Shift + Control
7	Alt + Control
8	Shift + Alt + Control
9	Meta
10	Meta + Shift
11	Meta + Alt
12	Meta + Alt + Shift
13	Meta + Ctrl
14	Meta + Ctrl + Shift
15	Meta + Ctrl + Alt
16	Meta + Ctrl + Alt + Shift

For example, shift-F5 would be sent as CSI 1 5 ; 2 ~

If the **alwaysUseMods** resource is set, the Meta modifier also is recognized, making parameters 9 through 16.

VT220-Style Function Keys

However, *xterm* is most useful as a DEC VT102 or VT220 emulator. Set the **sunKeyboard** resource to true to force a Sun/PC keyboard to act like a VT220 keyboard.

The VT102/VT220 application keypad transmits unique escape sequences in application mode, which are distinct from the cursor and scrolling keypad:

Key	Numeric	Application
Space	SP	SS3 SP
Tab	TAB	SS3 I
Enter	CR	SS3 M
PF1	SS3 P	SS3 P
PF2	SS3 Q	SS3 Q
PF3	SS3 R	SS3 R
PF4	SS3 S	SS3 S
* (multiply)	*	SS3 j
+ (add)	+	SS3 k
, (comma)	,	SS3 l
- (minus)	-	SS3 m
. (period)	.	SS3 n
/ (divide)	/	SS3 o
0	0	SS3 p
1	1	SS3 q
2	2	SS3 r
3	3	SS3 s
4	4	SS3 t
5	5	SS3 u
6	6	SS3 v
7	7	SS3 w
8	8	SS3 x
9	9	SS3 y
= (equal)	=	SS3 X

The VT220 provides a 6-key editing keypad, which is analogous to that on the PC keyboard. It is not affected by **DECCKM** or **DECKPNM**/**DECKPAM**:

Key	Normal	Application
Insert	CSI 2 ~	CSI 2 ~
Delete	CSI 3 ~	CSI 3 ~
Home	CSI 1 ~	CSI 1 ~
End	CSI 4 ~	CSI 4 ~
PageUp	CSI 5 ~	CSI 5 ~
PageDown	CSI 6 ~	CSI 6 ~

The VT220 provides 8 additional function keys. With a Sun/PC keyboard, access these keys by Control/F1 for F13, etc.

Key	Escape Sequence
F13	CSI 2 5 ~
F14	CSI 2 6 ~
F15	CSI 2 8 ~
F16	CSI 2 9 ~
F17	CSI 3 1 ~
F18	CSI 3 2 ~
F19	CSI 3 3 ~
F20	CSI 3 4 ~

VT52-Style Function Keys

A VT52 does not have function keys, but it does have a numeric keypad and cursor keys. They differ from the other emulations by the prefix. Also, the cursor keys do not change:

Key	Normal/Application
Cursor Up	ESC A
Cursor Down	ESC B
Cursor Right	ESC C
Cursor Left	ESC D

The keypad is similar:

Key	Numeric	Application
Space	SP	ESC ? SP
Tab	TAB	ESC ? I
Enter	CR	ESC ? M
PF1	ESC P	ESC P
PF2	ESC Q	ESC Q
PF3	ESC R	ESC R
PF4	ESC S	ESC S
* (multiply)	*	ESC ? j
+ (add)	+	ESC ? k
, (comma)	,	ESC ? l
- (minus)	-	ESC ? m
. (period)	.	ESC ? n
/ (divide)	/	ESC ? o
0	0	ESC ? p
1	1	ESC ? q
2	2	ESC ? r
3	3	ESC ? s
4	4	ESC ? t
5	5	ESC ? u
6	6	ESC ? v

Key	Numeric	Application
7	7	ESC ? w
8	8	ESC ? x
9	9	ESC ? y
= (equal)	=	ESC ? X

Sun-Style Function Keys

The *xterm* program provides support for Sun keyboards more directly, by a menu toggle that causes it to send Sun-style function key codes rather than VT220. Note, however, that the *sun* and *VT100* emulations are not really compatible. For example, their wrap-margin behavior differs.

Only function keys are altered; keypad and cursor keys are the same. The emulation responds identically. See the xterm-sun terminfo entry for details.

HP-Style Function Keys

Similarly, *xterm* can be compiled to support HP keyboards. See the xterm-hp terminfo entry for details.

The Alternate Screen Buffer

Xterm maintains two screen buffers. The normal screen buffer allows you to scroll back to view saved lines of output up to the maximum set by the **saveLines** resource. The alternate screen buffer is exactly as large as the display, contains no additional saved lines. When the alternate screen buffer is active, you cannot scroll back to view saved lines. *Xterm* provides control sequences and menu entries for switching between the two.

Most full-screen applications use terminfo or termcap to obtain strings used to start/stop full-screen mode, i.e., *smcup* and *rmcup* for terminfo, or the corresponding *ti* and *te* for termcap. The **titeInhibit** resource removes the *ti* and *te* strings from the TERMCAP string which is set in the environment for some platforms. That is not done when *xterm* is built with terminfo libraries because terminfo does not provide the whole text of the termcap data in one piece. It would not work for terminfo anyway, since terminfo data is not passed in environment variables; setting an environment variable in this manner would have no effect on the application's ability to switch between normal and alternate screen buffers. Instead, the newer private mode controls (such as 1 0 4 9) for switching between normal and alternate screen buffers simply disable the switching. They add other features such as clearing the display for the same reason: to make the details of switching independent of the application that requests the switch.

Bracketed Paste Mode

When bracketed paste mode is set, pasted text is bracketed with control sequences so that the program can differentiate pasted text from typed-in text. When bracketed paste mode is set, the program will receive:
ESC [2 0 0 ~,
followed by the pasted text, followed by
ESC [2 0 1 ~.

Title Modes

The window- and icon-labels can be set or queried using control sequences. As a VT220-emulator, *xterm* "should" limit the character encoding for the corresponding strings to ISO-8859-1. Indeed, it used to be the case (and was documented) that window titles had to be ISO-8859-1. This is no longer the case. However, there are many applications which still assume that titles are set using ISO-8859-1. So that is the default behavior.

If *xterm* is running with UTF-8 encoding, it is possible to use window- and icon-labels encoded using UTF-8. That is because the underlying X libraries (and many, but not all) window managers support this feature.

The **utf8Title** X resource setting tells *xterm* to disable a reconversion of the title string back to ISO-8859-1, allowing the title strings to be interpreted as UTF-8. The same feature can be enabled using the title mode control sequence described in this summary.

Separate from the ability to set the titles, *xterm* provides the ability to query the titles, returning them either in ISO-8859-1 or UTF-8. This choice is available only while *xterm* is using UTF-8 encoding.

Finally, the characters sent to, or returned by a title control are less constrained than the rest of the control sequences. To make them more manageable (and constrained), for use in shell scripts, *xterm* has an optional feature which decodes the string from hexadecimal (for setting titles) or for encoding the title into hexadecimal when querying the value.

Mouse Tracking

The VT widget can be set to send the mouse position and other information on button presses. These modes are typically used by editors and other full-screen applications that want to make use of the mouse.

There are two sets of mutually exclusive modes:

- mouse protocol
- protocol encoding

The mouse protocols include DEC Locator mode, enabled by the DECELR CSI P_s ; P_s ' z control sequence, and is not described here (control sequences are summarized above). The remaining five modes of the mouse protocols are each enabled (or disabled) by a different parameter in the "DECSET CSI ? P_m h" or "DECRST CSI ? P_m l" control sequence.

Manifest constants for the parameter values are defined in **xcharmouse.h** as follows:

```
#define SET_X10_MOUSE               9
#define SET_VT200_MOUSE             1000
#define SET_VT200_HIGHLIGHT_MOUSE   1001
#define SET_BTN_EVENT_MOUSE         1002
#define SET_ANY_EVENT_MOUSE         1003

#define SET_FOCUS_EVENT_MOUSE       1004

#define SET_EXT_MODE_MOUSE          1005
#define SET_SGR_EXT_MODE_MOUSE      1006
#define SET_URXVT_EXT_MODE_MOUSE    1015

#define SET_ALTERNATE_SCROLL        1007
```

The motion reporting modes are strictly *xterm* extensions, and are not part of any standard, though they are analogous to the DEC VT200 DECELR locator reports.

Normally, parameters (such as pointer position and button number) for all mouse tracking escape sequences generated by *xterm* encode numeric parameters in a single character as *value*+32. For example, ! specifies the value 1. The upper left character position on the terminal is denoted as 1,1. This scheme dates back to X10, though the normal mouse-tracking (from X11) is more elaborate.

X10 compatbility mode

X10 compatibility mode sends an escape sequence only on button press, encoding the location and the mouse button pressed. It is enabled by specifying parameter 9 to DECSET. On button press, *xterm* sends CSI M $C_b C_x C_y$ (6 characters).

- C_b is button−1.
- C_x and C_y are the x and y coordinates of the mouse when the button was pressed.

Normal tracking mode

Normal tracking mode sends an escape sequence on both button press and release. Modifier key (shift, ctrl, meta) information is also sent. It is enabled by specifying parameter 1000 to DECSET. On button press or release, *xterm* sends CSI M $C_b C_x C_y$.

- The low two bits of C_b encode button information: 0=MB1 pressed, 1=MB2 pressed, 2=MB3 pressed, 3=release.

- The next three bits encode the modifiers which were down when the button was pressed and are added together: 4=Shift, 8=Meta, 16=Control. Note however that the shift and control bits are normally unavailable because *xterm* uses the control modifier with mouse for popup menus, and the shift modifier is used in the default translations for button events. The *Meta* modifier recognized by *xterm* is the *mod1* mask, and is not necessarily the "Meta" key (see *xmodmap*).
- C_x and C_y are the x and y coordinates of the mouse event, encoded as in X10 mode.

Wheel mice

Wheel mice may return buttons 4 and 5. Those buttons are represented by the same event codes as buttons 1 and 2 respectively, except that 64 is added to the event code. Release events for the wheel buttons are not reported. By default, the wheel mouse events are translated to *scroll-back* and *scroll-forw* actions. Those actions normally scroll the whole window, as if the scrollbar was used. However if Alternate Scroll mode is set, then cursor up/down controls are sent when the terminal is displaying the alternate screen. The initial state of Alternate Scroll mode is set using the **alternateScroll** resource.

Highlight tracking

Mouse highlight tracking notifies a program of a button press, receives a range of lines from the program, highlights the region covered by the mouse within that range until button release, and then sends the program the release coordinates. It is enabled by specifying parameter 1001 to DECSET. Highlighting is performed only for button 1, though other button events can be received.

Warning: use of this mode requires a cooperating program or it will hang *xterm.*

On button press, the same information as for normal tracking is generated; *xterm* then waits for the program to send mouse tracking information. *All X events are ignored until the proper escape sequence is received from the pty:* CSI P_s ; P_s ; P_s ; P_s ; P_s T. The parameters are *func, startx, starty, firstrow,* and *lastrow. func* is non-zero to initiate highlight tracking and zero to abort. *startx* and *starty* give the starting x and y location for the highlighted region. The ending location tracks the mouse, but will never be above row *firstrow* and will always be above row *lastrow.* (The top of the screen is row 1.) When the button is released, *xterm* reports the ending position one of two ways:

- if the start and end coordinates are the same locations:
  CSI t $C_x C_y$.
- otherwise:
  CSI T $C_x C_y C_x C_y C_x C_y$.
  The parameters are *startx, starty, endx, endy, mousex,* and *mousey.*
  - *startx, starty, endx,* and *endy* give the starting and ending character positions of the region.
  - *mousex* and *mousey* give the location of the mouse at button up, which may not be over a character.

Button-event tracking

Button-event tracking is essentially the same as normal tracking, but *xterm* also reports button-motion events. Motion events are reported only if the mouse pointer has moved to a different character cell. It is enabled by specifying parameter 1002 to DECSET. On button press or release, *xterm* sends the same codes used by normal tracking mode.

- On button-motion events, *xterm* adds 32 to the event code (the third character, C_b).
- The other bits of the event code specify button and modifier keys as in normal mode. For example, motion into cell x,y with button 1 down is reported as CSI M @ $C_x C_y$. (@ = 32 + 0 (button 1) + 32 (motion indicator)). Similarly, motion with button 3 down is reported as CSI M B $C_x C_y$. (B = 32 + 2 (button 3) + 32 (motion indicator)).

Any-event tracking

Any-event mode is the same as button-event mode, except that all motion events are reported, even if no mouse button is down. It is enabled by specifying 1003 to DECSET.

FocusIn/FocusOut

FocusIn/FocusOut can be combined with any of the mouse events since it uses a different protocol. When set, it causes *xterm* to send CSI I when the terminal gains focus, and CSI O when it loses focus.

Extended coordinates

The original X10 mouse protocol limits the C_x and C_y ordinates to 223 (=255 - 32). *Xterm* supports more than one scheme for extending this range, by changing the protocol encoding:

UTF-8 (1005) This enables UTF-8 encoding for C_x and C_y under all tracking modes, expanding the maximum encodable position from 223 to 2015. For positions less than 95, the resulting output is identical under both modes. Under extended mouse mode, positions greater than 95 generate "extra" bytes which will confuse applications which do not treat their input as a UTF-8 stream. Likewise, C_b will be UTF-8 encoded, to reduce confusion with wheel mouse events.

Under normal mouse mode, positions outside (160,94) result in byte pairs which can be interpreted as a single UTF-8 character; applications which do treat their input as UTF-8 will almost certainly be confused unless extended mouse mode is active.

This scheme has the drawback that the encoded coordinates will not pass through *luit* unchanged, e.g., for locales using non-UTF-8 encoding.

SGR (1006) The normal mouse response is altered to use CSI < followed by semicolon-separated encoded button value, the C_x and C_y ordinates and a final character which is M for button press and m for button release.

- The encoded button value in this case does not add 32 since that was useful only in the X10 scheme for ensuring that the byte containing the button value is a printable code.
- The modifiers are encoded in the same way.
- A different final character is used for button release to resolve the X10 ambiguity regarding which button was released.

The highlight tracking responses are also modified to an SGR-like format, using the same SGR-style scheme and button-encodings.

URXVT (1015) The normal mouse response is altered to use CSI followed by semicolon-separated encoded button value, the C_x and C_y ordinates and final character M.

This uses the same button encoding as X10, but printing it as a decimal integer rather than as a single byte.

However, CSI M can be mistaken for DL (delete lines), while the highlight tracking CSI T can be mistaken for SD (scroll down), and the Window manipulation controls. For these reasons, the 1015 control is not recommended; it is not an improvement over 1005.

Sixel Graphics

If *xterm* is configured as VT240, VT241, VT330, VT340 or VT382 using the **decTerminalID** resource, it supports Sixel Graphics controls, a palleted bitmap graphics system using sets of six vertical pixels as the basic element.

CSI P_s c *xterm* responds to Send Device Attributes (Primary DA) with these additional codes:

P_s = 4 → Sixel graphics.

CSI ? P_m h *xterm* has these additional private Set Mode values:

P_s = 8 0 → Sixel scrolling.

P_s = 1 0 7 0 → use private color registers for each graphic.

P_s = 8 4 5 2 → Sixel scrolling leaves cursor to right of graphic.

DCS P_a ; P_b ; P_h q P_s..P_s ST

See:

http://vt100.net/docs/vt3xx-gp/chapter14.html

The sixel data device control string has three positional parameters, following the q with sixel data.

$P_a \rightarrow$ pixel aspect ratio

$P_b \rightarrow$ background color option

$P_h \rightarrow$ horizontal grid size (ignored).

$P_s \rightarrow$ sixel data

ReGIS Graphics

If *xterm* is configured as VT125, VT240, VT241, VT330 or VT340 using the **decTerminalID** resource, it supports Remote Graphic Instruction Set, a graphics description language.

CSI P_s c *xterm* responds to Send Device Attributes (Primary DA) with these additional codes:

P_s = 3 → ReGIS graphics.

CSI ? P_m h *xterm* has these additional private Set Mode values:

P_s = 1 0 7 0 → use private color registers for each graphic.

DCS P_m p P_r..P_r ST

See:

http://vt100.net/docs/vt3xx-gp/chapter1.html

The ReGIS data device control string has one positional parameter with four possible values:

P_m = 0 → resume command, use fullscreen mode.

P_m = 1 → start new command, use fullscreen mode.

P_m = 2 → resume command, use command display mode.

P_m = 3 → start new command, use command display mode.

Tektronix 4014 Mode

Most of these sequences are standard Tektronix 4014 control sequences. Graph mode supports the 12-bit addressing of the Tektronix 4014. The major features missing are the write-through and defocused modes. This document does not describe the commands used in the various Tektronix plotting modes but does describe the commands to switch modes.

BEL Bell (Ctrl-G).

BS Backspace (Ctrl-H).

TAB Horizontal Tab (Ctrl-I).

LF Line Feed or New Line (Ctrl-J).

VT Cursor up (Ctrl-K).

FF Form Feed or New Page (Ctrl-L).

CR Carriage Return (Ctrl-M).

ESC ETX Switch to VT100 Mode (ESC Ctrl-C).

ESC ENQ Return Terminal Status (ESC Ctrl-E).

ESC FF PAGE (Clear Screen) (ESC Ctrl-L).

ESC SO — Begin 4015 APL mode (ESC Ctrl-N). (This is ignored by *xterm*).

ESC SI — End 4015 APL mode (ESC Ctrl-O). (This is ignored by *xterm*).

ESC ETB — COPY (Save Tektronix Codes to file COPY*yyyy-mm-dd.hh:mm:ss*).

ETB (end transmission block) is the same as Ctrl-W.

ESC CAN — Bypass Condition (ESC Ctrl-X).

ESC SUB — GIN mode (ESC Ctrl-Z).

ESC FS — Special Point Plot Mode (ESC Ctrl-\).

ESC 8 — Select Large Character Set.

ESC 9 — Select #2 Character Set.

ESC : — Select #3 Character Set.

ESC ; — Select Small Character Set.

OSC P_s ; P_t BEL

Set Text Parameters of VT window.

P_s = 0 → Change Icon Name and Window Title to P_t.

P_s = 1 → Change Icon Name to P_t.

P_s = 2 → Change Window Title to P_t.

P_s = 4 6 → Change Log File to P_t. (This is normally disabled by a compile-time option).

ESC ` — Normal Z Axis and Normal (solid) Vectors.

ESC a — Normal Z Axis and Dotted Line Vectors.

ESC b — Normal Z Axis and Dot-Dashed Vectors.

ESC c — Normal Z Axis and Short-Dashed Vectors.

ESC d — Normal Z Axis and Long-Dashed Vectors.

ESC h — Defocused Z Axis and Normal (solid) Vectors.

ESC i — Defocused Z Axis and Dotted Line Vectors.

ESC j — Defocused Z Axis and Dot-Dashed Vectors.

ESC k — Defocused Z Axis and Short-Dashed Vectors.

ESC l — Defocused Z Axis and Long-Dashed Vectors.

ESC p — Write-Thru Mode and Normal (solid) Vectors.

ESC q — Write-Thru Mode and Dotted Line Vectors.

ESC r — Write-Thru Mode and Dot-Dashed Vectors.

ESC s — Write-Thru Mode and Short-Dashed Vectors.

ESC t — Write-Thru Mode and Long-Dashed Vectors.

FS — Point Plot Mode (Ctrl-\).

GS — Graph Mode (Ctrl-]).

RS — Incremental Plot Mode (Ctrl-^).

US — Alpha Mode (Ctrl-_).

VT52 Mode

Parameters for cursor movement are at the end of the ESC Y escape sequence. Each ordinate is encoded in a single character as *value*+32. For example, ! is 1. The screen coordinate system is 0-based.

ESC <	Exit VT52 mode (Enter VT100 mode).
ESC =	Enter alternate keypad mode.
ESC >	Exit alternate keypad mode.
ESC A	Cursor up.
ESC B	Cursor down.
ESC C	Cursor right.
ESC D	Cursor left.
ESC F	Enter graphics mode.
ESC G	Exit graphics mode.
ESC H	Move the cursor to the home position.
ESC I	Reverse line feed.
ESC J	Erase from the cursor to the end of the screen.
ESC K	Erase from the cursor to the end of the line.
ESC Y P_s P_s	Move the cursor to given row and column.
ESC Z	Identify.
	→ ESC / Z (“I am a VT52.”).