https://invisible-island.net/xterm/

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 (2006-2018)
updated for XTerm Patch #336 (2018/09/16)

Definitions

c The literal character *c*.

C A single (required) character.

Ps A single (usually optional) numeric parameter, composed of one of more digits.

Pm A multiple numeric parameter composed of any number of single numeric parameters, separated by ; character(s). Individual values for the parameters are listed with *Ps* .

Pt 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), VT220.
     This affects next character only.

ESC O
     Single Shift Select of G3 Character Set (SS3 is 0x8f), VT220.
     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

In this document, "VT100" refers not only to VT100/VT102, but also to the succession of upward-compatible terminals produced by DEC (Digital Equipment Corporation) from the mid-1970s for about twenty years. For brevity, the document refers to the related models:
  "VT200" as VT220/VT240,

"VT300" as VT320/VT340,
"VT400" as VT420, and
"VT500" as VT510/VT520/VT525.

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 ECMA-48 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 ECMA-48, 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 Switch to *Standard Character Set* (Ctrl-O is Shift In or LS0). This invokes the G0 character set (the default) as GL. VT200 and up implement LS0.

SO Switch to *Alternate Character Set* (Ctrl-N is Shift Out or LS1). This invokes the G1 character set as GL. VT200 and up implement LS1.

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), VT220.

ESC SP G 8-bit controls (S8C1T), VT220.

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), VT100.

ESC # 4 DEC double-height line, bottom half (DECDHL), VT100.

ESC # 5 DEC single-width line (DECSWL), VT100.

ESC # 6 DEC double-width line (DECDWL), VT100.

ESC # 8 DEC Screen Alignment Test (DECALN), VT100.

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, VT100, ISO 2022.
Final character *C* for designating 94-character sets. In this list,

- 0 , A and B were introduced in the VT100,
- most were introduced in the VT200 series,
- a few were introduced in the VT300 series, and
- a few more were introduced in the VT500 series.

The VT220 character sets, together with a few others (such as Portuguese) are activated by the National Replacement Character Set (NRCS) controls. The term "replacement" says that the character set is formed by replacing some of the characters in a set (termed the *Multinational Character Set*) with more useful ones for a given language. The ASCII and DEC Supplemental character sets make up the two halves of the Multinational Character set, initially mapped to GL and GR.
The valid final characters *C* for this control are:

- *C* = A -> United Kingdom (UK), VT100.
- *C* = B -> United States (USASCII), VT100.
- *C* = 4 -> Dutch, VT200.
- *C* = C or 5 -> Finnish, VT200.
- *C* = R or f -> French, VT200.
- *C* = Q or 9 -> French Canadian, VT200.

C = K -> German, VT200.
C = " > -> Greek, VT500.
C = % = -> Hebrew, VT500.
C = Y -> Italian, VT200.
C = ` , E or 6 -> Norwegian/Danish, VT200.
C = % 6 -> Portuguese, VT300.
C = Z -> Spanish, VT200.
C = H or 7 -> Swedish, VT200.
C = = -> Swiss, VT200.
C = % 2 -> Turkish, VT500.

The final character A is a special case, since the same final character is used by the VT300-control for the 96-character British Latin-1.

There are a few other 94-character sets not documented as NRCS:

C = 0 -> DEC Special Character and Line Drawing Set, VT100.
C = < -> DEC Supplemental, VT200.
C = > -> DEC Technical, VT300.
C = % 5 -> DEC Supplemental Graphics, VT300.
C = " ? -> DEC Greek, VT500.
C = " 4 -> DEC Hebrew, VT500.
C = % 0 -> DEC Turkish, VT500.

The VT520 reference manual lists a few more, but no documentation has been found for the mappings:

C = & 4 -> DEC Cyrillic, VT500.
C = & 5 -> DEC Russian, VT500.
C = % 3 -> SCS NRCS, VT500.

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.
These controls apply only to 96-character sets. Unlike the 94-character sets, these can have different values than ASCII space and DEL for the mapping of 0x20 and 0x7f. The valid final characters *C* for this control are:

C = A -> ISO Latin-1 Supplemental (VT300).
C = F -> ISO Greek Supplemental (VT500).
C = H -> ISO Hebrew Supplemental (VT500).
C = L -> ISO Latin-Cyrillic (VT500).
C = M -> ISO Latin-5 Supplemental (VT500).

ESC . *C* Designate G2 Character Set, VT300.
The same character sets apply as for ESC - *C*.

ESC / *C* Designate G3 Character Set, VT300.
The same character sets apply as for ESC - *C*.

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

ESC 7 Save Cursor (DECSC), VT100.

ESC 8 Restore Cursor (DECRC), VT100.

ESC 9 Forward Index (DECFI), VT420 and up.

ESC = Application Keypad (DECKPAM).

ESC > Normal Keypad (DECKPNM), VT100.

ESC F Cursor to lower left corner of screen. This is enabled by the **hpLowerleftBugCompat** resource.

ESC c Full Reset (RIS), VT100.

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) as GL.

ESC o Invoke the G3 Character Set as GL (LS3) as GL.

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), VT100.

Application Program-Command functions

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

Device-Control functions

DCS *Ps* ; *Ps* | *Pt* ST
User-Defined Keys (DECUDK), VT220 and up.

The first parameter:
Ps = 0 -> Clear all UDK definitions before starting (default).
Ps = 1 -> Erase Below (default).

The second parameter:

Ps = 0 <- Lock the keys (default).
Ps = 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 *Pt* ST

Request Status String (DECRQSS), VT420 and up.
The string following the "q" is one of the following:

```
m       -> SGR
" p     -> DECSCL
SP q    -> DECSCUSR
" q     -> DECSCA
r       -> DECSTBM
s       -> DECSLRM
t       -> DECSLPP
$ |     -> DECSCPP
* |     -> DECSNLS
```

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

DCS *Ps* $ t *Pt* ST

Restore presentation status (DECRSPS), VT320 and up. The control can be converted from a response from DECCIR or DECTABSR by changing the first "u" to a "t"

Ps = 1 -> DECCIR
Ps = 2 -> DECTABSR

DCS + p *Pt* 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 *Pt* ST

Request Termcap/Terminfo String (*xterm*). 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.
A few 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).
- *RGB* for the ncurses direct-color extension.
  Only a terminfo name is provided, since termcap applications cannot use this information.

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

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

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

CSI *Ps* SP @
Shift left *Ps* columns(s) (default = 1) (SL), ECMA-48.

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

CSI *Ps* SP A
Shift right *Ps* columns(s) (default = 1) (SR), ECMA-48.

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

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

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

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

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

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

CSI *Ps* ; *Ps* H
Cursor Position [row;column] (default = [1,1]) (CUP).

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

CSI *Ps* J Erase in Display (ED), VT100.
Ps = 0 -> Erase Below (default).
Ps = 1 -> Erase Above.
Ps = 2 -> Erase All.
Ps = 3 -> Erase Saved Lines (*xterm*).

CSI ? *Ps* J
Erase in Display (DECSED), VT220.
Ps = 0 -> Selective Erase Below (default).
Ps = 1 -> Selective Erase Above.
Ps = 2 -> Selective Erase All.
Ps = 3 -> Selective Erase Saved Lines (*xterm*).

CSI *Ps* K Erase in Line (EL), VT100.

Ps = 0 -> Erase to Right (default).
Ps = 1 -> Erase to Left.
Ps = 2 -> Erase All.

CSI ? *Ps* K
Erase in Line (DECSEL), VT220.
Ps = 0 -> Selective Erase to Right (default).
Ps = 1 -> Selective Erase to Left.
Ps = 2 -> Selective Erase All.

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

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

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

CSI *Ps* S Scroll up *Ps* lines (default = 1) (SU), VT420, ECMA-48.

CSI ? *Pi* ; *Pa* ; *Pv* S
If configured to support either **Sixel Graphics** or **ReGIS Graphics**, *xterm* accepts a three-parameter control sequence, where *Pi*, *Pa* and *Pv* are the *item*, *action* and *value*:

Pi = 1 -> item is number of color registers.
Pi = 2 -> item is Sixel graphics geometry (in pixels).
Pi = 3 -> item is ReGIS graphics geometry (in pixels).

Pa = 1 -> read
Pa = 2 -> reset to default
Pa = 3 -> set to value in *Pv*
Pa = 4 -> read the maximum allowed value

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

xterm replies with a control sequence of the same form:

CSI ? *Pi* ; *Ps* ; *Pv* S

where *Ps* is the status:
Ps = 0 -> success.
Ps = 1 -> error in *Pi*.
Ps = 2 -> error in *Pa*.
Ps = 3 -> failure.

On success, *Pv* represents the value read or set.

Notes:
o The current implementation allows reading the graphics

sizes, but disallows modifying those sizes because that is done once, using resource-values.

- **o** 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.
- **o** 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 *Ps* T Scroll down *Ps* lines (default = 1) (SD), VT420.

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

CSI > *Ps* ; *Ps* 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*.

Ps = 0 -> Do not set window/icon labels using hexadecimal.
Ps = 1 -> Do not query window/icon labels using hexadecimal.
Ps = 2 -> Do not set window/icon labels using UTF-8.
Ps = 3 -> Do not query window/icon labels using UTF-8.

(See discussion of **Title Modes**).

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

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

CSI *Ps* ^ Scroll down *Ps* lines (default = 1) (SD), ECMA-48.
This is probably an error in ECMA-48, because the standard writes codes as pairs of decimal values, and this is "14" versus the VT420 "4".

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

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

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

CSI *Ps* c Send Device Attributes (Primary DA).

Ps = 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 ; *Ps* c ("VT220")
-> CSI ? 6 3 ; *Ps* c ("VT320")
-> CSI ? 6 4 ; *Ps* 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:
Ps = 1 -> 132-columns.
Ps = 2 -> Printer.
Ps = 3 -> ReGIS graphics.
Ps = 4 -> Sixel graphics.
Ps = 6 -> Selective erase.
Ps = 8 -> User-defined keys.
Ps = 9 -> National Replacement Character sets.
Ps = 1 5 -> Technical characters.
Ps = 1 8 -> User windows.
Ps = 2 1 -> Horizontal scrolling.
Ps = 2 2 -> ANSI color, e.g., VT525.
Ps = 2 9 -> ANSI text locator (i.e., DEC Locator mode).

CSI = *Ps* c
Send Device Attributes (Tertiary DA).
Ps = 0 -> report Terminal Unit ID (default), VT400. XTerm uses zeros for the site code and serial number in its DECRPTUI response.

CSI > *Ps* c
Send Device Attributes (Secondary DA).
Ps = 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 > *Pp* ; *Pv* ; *Pc* c
where *Pp* denotes the terminal type
Pp = 0 -> "VT100".
Pp = 1 -> "VT220".
Pp = 2 -> "VT240".
Pp = 1 8 -> "VT330".
Pp = 1 9 -> "VT340".
Pp = 2 4 -> "VT320".
Pp = 4 1 -> "VT420".
Pp = 6 1 -> "VT510".
Pp = 6 4 -> "VT520".
Pp = 6 5 -> "VT525".

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

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

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

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

CSI *Ps* g Tab Clear (TBC).
Ps = 0 -> Clear Current Column (default).
Ps = 3 -> Clear All.

CSI *Pm* h Set Mode (SM).
Ps = 2 -> Keyboard Action Mode (AM).
Ps = 4 -> Insert Mode (IRM).
Ps = 1 2 -> Send/receive (SRM).
Ps = 2 0 -> Automatic Newline (LNM).

CSI ? *Pm* h
DEC Private Mode Set (DECSET).
Ps = 1 -> Application Cursor Keys (DECCKM), VT100.
Ps = 2 -> Designate USASCII for character sets G0-G3 (DECANM), VT100, and set VT100 mode.
Ps = 3 -> 132 Column Mode (DECCOLM), VT100.
Ps = 4 -> Smooth (Slow) Scroll (DECSCLM), VT100.
Ps = 5 -> Reverse Video (DECSCNM), VT100.
Ps = 6 -> Origin Mode (DECOM), VT100.
Ps = 7 -> Auto-wrap Mode (DECAWM), VT100.
Ps = 8 -> Auto-repeat Keys (DECARM), VT100.
Ps = 9 -> Send Mouse X & Y on button press. See the section **Mouse Tracking**. This is the X10 *xterm* mouse protocol.
Ps = 1 0 -> Show toolbar (rxvt).
Ps = 1 2 -> Start Blinking Cursor (AT&T 610).
Ps = 1 3 -> Start Blinking Cursor (set only via resource or menu).
Ps = 1 4 -> Enable XOR of Blinking Cursor control sequence and menu.
Ps = 1 8 -> Print form feed (DECPFF), VT220.
Ps = 1 9 -> Set print extent to full screen (DECPEX), VT220.
Ps = 2 5 -> Show Cursor (DECTCEM), VT220.
Ps = 3 0 -> Show scrollbar (rxvt).
Ps = 3 5 -> Enable font-shifting functions (rxvt).
Ps = 3 8 -> Enter Tektronix Mode (DECTEK), VT240, *xterm*.

Ps = 4 0 -> Allow 80 **->** 132 Mode, *xterm*.
Ps = 4 1 -> **more(1)** fix (see **curses(3)** resource).
Ps = 4 2 -> Enable National Replacement Character sets (DECNRCM), VT220.
Ps = 4 4 -> Turn On Margin Bell, *xterm*.
Ps = 4 5 -> Reverse-wraparound Mode, *xterm*.
Ps = 4 6 -> Start Logging, *xterm*. This is normally disabled by a compile-time option.
Ps = 4 7 -> Use *Alternate Screen Buffer*, *xterm*. This may be disabled by the **titeInhibit** resource.
Ps = 6 6 -> Application keypad (DECNKM), VT320.
Ps = 6 7 -> Backarrow key sends backspace (DECBKM), VT340, VT420.
Ps = 6 9 -> Enable left and right margin mode (DECLRMM), VT420 and up.
Ps = 9 5 -> Do not clear screen when DECCOLM is set/reset (DECNCSM), VT510 and up.
Ps = 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.
Ps = 1 0 0 1 -> Use Hilite Mouse Tracking, *xterm*.
Ps = 1 0 0 2 -> Use Cell Motion Mouse Tracking, *xterm*.
Ps = 1 0 0 3 -> Use All Motion Mouse Tracking, *xterm*.
Ps = 1 0 0 4 -> Send FocusIn/FocusOut events, *xterm*.
Ps = 1 0 0 5 -> Enable UTF-8 Mouse Mode, *xterm*.
Ps = 1 0 0 6 -> Enable SGR Mouse Mode, *xterm*.
Ps = 1 0 0 7 -> Enable *Alternate Scroll Mode*, *xterm*. This corresponds to the **alternateScroll** resource.
Ps = 1 0 1 0 -> Scroll to bottom on tty output (rxvt).
Ps = 1 0 1 1 -> Scroll to bottom on key press (rxvt).
Ps = 1 0 1 5 -> Enable urxvt Mouse Mode.
Ps = 1 0 3 4 -> Interpret "meta" key, *xterm*. This sets eighth bit of keyboard input (and enables the **eightBitInput** resource).
Ps = 1 0 3 5 -> Enable special modifiers for Alt and Num-Lock keys, *xterm*. This enables the **numLock** resource.
Ps = 1 0 3 6 -> Send ESC when Meta modifies a key, *xterm*. This enables the **metaSendsEscape** resource.
Ps = 1 0 3 7 -> Send DEL from the editing-keypad Delete key, *xterm*.
Ps = 1 0 3 9 -> Send ESC when Alt modifies a key, *xterm*. This enables the **altSendsEscape** resource, *xterm*.
Ps = 1 0 4 0 -> Keep selection even if not highlighted, *xterm*. This enables the **keepSelection** resource.
Ps = 1 0 4 1 -> Use the CLIPBOARD selection, *xterm*. This enables the **selectToClipboard** resource.
Ps = 1 0 4 2 -> Enable Urgency window manager hint when Control-G is received, *xterm*. This enables the **bellIsUrgent** resource.
Ps = 1 0 4 3 -> Enable raising of the window when Control-G

is received, *xterm*. This enables the **popOnBell** resource.
Ps = 1 0 4 4 -> Reuse the most recent data copied to CLIPBOARD, *xterm*. This enables the **keepClipboard** resource.
Ps = 1 0 4 6 -> Enable switching to/from *Alternate Screen Buffer*, *xterm*. This works for terminfo-based systems, updating the **titeInhibit** resource.
Ps = 1 0 4 7 -> Use *Alternate Screen Buffer*, *xterm*. This may be disabled by the **titeInhibit** resource.
Ps = 1 0 4 8 -> Save cursor as in DECSC, *xterm*. This may be disabled by the **titeInhibit** resource.
Ps = 1 0 4 9 -> Save cursor as in DECSC, *xterm*. After saving the cursor, switch to the *Alternate Screen Buffer*, clearing it first. This may be disabled by the **titeInhibit** resource. This control 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.
Ps = 1 0 5 0 -> Set terminfo/termcap function-key mode, *xterm*.
Ps = 1 0 5 1 -> Set Sun function-key mode, *xterm*.
Ps = 1 0 5 2 -> Set HP function-key mode, *xterm*.
Ps = 1 0 5 3 -> Set SCO function-key mode, *xterm*.
Ps = 1 0 6 0 -> Set legacy keyboard emulation (i.e, X11R6), *xterm*.
Ps = 1 0 6 1 -> Set VT220 keyboard emulation, *xterm*.
Ps = 2 0 0 4 -> Set bracketed paste mode, *xterm*.

CSI *Pm* i Media Copy (MC).
Ps = 0 -> Print screen (default).
Ps = 4 -> Turn off printer controller mode.
Ps = 5 -> Turn on printer controller mode.
Ps = 1 0 -> HTML screen dump, *xterm*.
Ps = 1 1 -> SVG screen dump, *xterm*.

CSI ? *Pm* i
Media Copy (MC), DEC-specific.
Ps = 1 -> Print line containing cursor.
Ps = 4 -> Turn off autoprint mode.
Ps = 5 -> Turn on autoprint mode.
Ps = 1 0 -> Print composed display, ignores DECPEX.
Ps = 1 1 -> Print all pages.

CSI *Pm* l Reset Mode (RM).
Ps = 2 -> Keyboard Action Mode (AM).
Ps = 4 -> Replace Mode (IRM).
Ps = 1 2 -> Send/receive (SRM).
Ps = 2 0 -> Normal Linefeed (LNM).

CSI ? *Pm* l
DEC Private Mode Reset (DECRST).
Ps = 1 -> Normal Cursor Keys (DECCKM), VT100.

Ps = 2 -> Designate VT52 mode (DECANM), VT100.
Ps = 3 -> 80 Column Mode (DECCOLM), VT100.
Ps = 4 -> Jump (Fast) Scroll (DECSCLM), VT100.
Ps = 5 -> Normal Video (DECSCNM), VT100.
Ps = 6 -> Normal Cursor Mode (DECOM), VT100.
Ps = 7 -> No Auto-wrap Mode (DECAWM), VT100.
Ps = 8 -> No Auto-repeat Keys (DECARM), VT100.
Ps = 9 -> Don't send Mouse X & Y on button press, *xterm*.
Ps = 1 0 -> Hide toolbar (rxvt).
Ps = 1 2 -> Stop Blinking Cursor (AT&T 610).
Ps = 1 3 -> Disable Blinking Cursor (reset only via resource or menu).
Ps = 1 4 -> Disable XOR of Blinking Cursor control sequence and menu.
Ps = 1 8 -> Don't print form feed (DECPFF).
Ps = 1 9 -> Limit print to scrolling region (DECPEX).
Ps = 2 5 -> Hide Cursor (DECTCEM), VT220.
Ps = 3 0 -> Don't show scrollbar (rxvt).
Ps = 3 5 -> Disable font-shifting functions (rxvt).
Ps = 4 0 -> Disallow 80 **->** 132 Mode, *xterm*.
Ps = 4 1 -> No **more(1)** fix (see **curses(3)** resource).
Ps = 4 2 -> Disable National Replacement Character sets (DECNRCM), VT220.
Ps = 4 4 -> Turn Off Margin Bell, *xterm*.
Ps = 4 5 -> No Reverse-wraparound Mode, *xterm*.
Ps = 4 6 -> Stop Logging, *xterm*. This is normally disabled by a compile-time option.
Ps = 4 7 -> Use Normal Screen Buffer, *xterm*.
Ps = 6 6 -> Numeric keypad (DECNKM), VT320.
Ps = 6 7 -> Backarrow key sends delete (DECBKM), VT340, VT420.
Ps = 6 9 -> Disable left and right margin mode (DECLRMM), VT420 and up.
Ps = 9 5 -> Clear screen when DECCOLM is set/reset (DEC-NCSM), VT510 and up.
Ps = 1 0 0 0 -> Don't send Mouse X & Y on button press and release. See the section **Mouse Tracking**.
Ps = 1 0 0 1 -> Don't use Hilite Mouse Tracking, *xterm*.
Ps = 1 0 0 2 -> Don't use Cell Motion Mouse Tracking, *xterm*.
Ps = 1 0 0 3 -> Don't use All Motion Mouse Tracking, *xterm*.
Ps = 1 0 0 4 -> Don't send FocusIn/FocusOut events, *xterm*.
Ps = 1 0 0 5 -> Disable UTF-8 Mouse Mode, *xterm*.
Ps = 1 0 0 6 -> Disable SGR Mouse Mode, *xterm*.
Ps = 1 0 0 7 -> Disable *Alternate Scroll Mode*, *xterm*. This corresponds to the **alternateScroll** resource.
Ps = 1 0 1 0 -> Don't scroll to bottom on tty output (rxvt).
Ps = 1 0 1 1 -> Don't scroll to bottom on key press (rxvt).
Ps = 1 0 1 5 -> Disable urxvt Mouse Mode.

Ps = 1 0 3 4 -> Don't interpret "meta" key, *xterm*. This disables the **eightBitInput** resource.
Ps = 1 0 3 5 -> Disable special modifiers for Alt and Num-Lock keys, *xterm*. This disables the **numLock** resource.
Ps = 1 0 3 6 -> Don't send ESC when Meta modifies a key, *xterm*. This disables the **metaSendsEscape** resource.
Ps = 1 0 3 7 -> Send VT220 Remove from the editing-keypad *Delete* key, *xterm*.
Ps = 1 0 3 9 -> Don't send ESC when Alt modifies a key, *xterm*. This disables the **altSendsEscape** resource.
Ps = 1 0 4 0 -> Do not keep selection when not highlighted, *xterm*. This disables the **keepSelection** resource.
Ps = 1 0 4 1 -> Use the PRIMARY selection, *xterm*. This disables the **selectToClipboard** resource.
Ps = 1 0 4 2 -> Disable Urgency window manager hint when Control-G is received, *xterm*. This disables the **bellIsUrgent** resource.
Ps = 1 0 4 3 -> Disable raising of the window when Control-G is received, *xterm*. This disables the **popOnBell** resource.
Ps = 1 0 4 6 -> Disable switching to/from *Alternate Screen Buffer*, *xterm*. This works for terminfo-based systems, updating the **titeInhibit** resource. If currently using the *Alternate Screen Buffer*, *xterm* switches to the Normal Screen Buffer.
Ps = 1 0 4 7 -> Use Normal Screen Buffer, *xterm*. Clear the screen first if in the *Alternate Screen Buffer*. This may be disabled by the **titeInhibit** resource.
Ps = 1 0 4 8 -> Restore cursor as in DECRC, *xterm*. This may be disabled by the **titeInhibit** resource.
Ps = 1 0 4 9 -> Use Normal Screen Buffer and restore cursor as in DECRC, *xterm*. 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.
Ps = 1 0 5 0 -> Reset terminfo/termcap function-key mode, *xterm*.
Ps = 1 0 5 1 -> Reset Sun function-key mode, *xterm*.
Ps = 1 0 5 2 -> Reset HP function-key mode, *xterm*.
Ps = 1 0 5 3 -> Reset SCO function-key mode, *xterm*.
Ps = 1 0 6 0 -> Reset legacy keyboard emulation (i.e, X11R6), *xterm*.
Ps = 1 0 6 1 -> Reset keyboard emulation to Sun/PC style, *xterm*.
Ps = 2 0 0 4 -> Reset bracketed paste mode, *xterm*.

CSI *Pm* m Character Attributes (SGR).
Ps = 0 -> Normal (default), VT100.
Ps = 1 -> Bold, VT100.
Ps = 2 -> Faint, decreased intensity, ECMA-48 2nd.
Ps = 3 -> Italicized, ECMA-48 2nd.

Ps = 4 -> Underlined, VT100.
Ps = 5 -> Blink, VT100.
This appears as Bold in X11R6 xterm.
Ps = 7 -> Inverse, VT100.
Ps = 8 -> Invisible, i.e., hidden, ECMA-48 2nd, VT300.
Ps = 9 -> Crossed-out characters, ECMA-48 3rd.
Ps = 2 1 -> Doubly-underlined, ECMA-48 3rd.
Ps = 2 2 -> Normal (neither bold nor faint), ECMA-48 3rd.
Ps = 2 3 -> Not italicized, ECMA-48 3rd.
Ps = 2 4 -> Not underlined, ECMA-48 3rd.
Ps = 2 5 -> Steady (not blinking), ECMA-48 3rd.
Ps = 2 7 -> Positive (not inverse), ECMA-48 3rd.
Ps = 2 8 -> Visible, i.e., not hidden, ECMA-48 3rd, VT300.
Ps = 2 9 -> Not crossed-out, ECMA-48 3rd.
Ps = 3 0 -> Set foreground color to Black.
Ps = 3 1 -> Set foreground color to Red.
Ps = 3 2 -> Set foreground color to Green.
Ps = 3 3 -> Set foreground color to Yellow.
Ps = 3 4 -> Set foreground color to Blue.
Ps = 3 5 -> Set foreground color to Magenta.
Ps = 3 6 -> Set foreground color to Cyan.
Ps = 3 7 -> Set foreground color to White.
Ps = 3 9 -> Set foreground color to default, ECMA-48 3rd.
Ps = 4 0 -> Set background color to Black.
Ps = 4 1 -> Set background color to Red.
Ps = 4 2 -> Set background color to Green.
Ps = 4 3 -> Set background color to Yellow.
Ps = 4 4 -> Set background color to Blue.
Ps = 4 5 -> Set background color to Magenta.
Ps = 4 6 -> Set background color to Cyan.
Ps = 4 7 -> Set background color to White.
Ps = 4 9 -> Set background color to default, ECMA-48 3rd.

Some of the above note the edition of ECMA-48 which first describes a feature. In its successive editions from 1979 to 1991 (*2nd* 1979, *3rd* 1984, *4th* 1986, and *5th* 1991), ECMA-48 listed codes through 6 5 (skipping several toward the end of the range). Most of the ECMA-48 codes not implemented in *xterm* were never implemented in a hardware terminal. Several (such as 3 9 and 4 9) are either noted in ECMA-48 as implementation defined, or described in vague terms.

The successive editions of ECMA-48 give little attention to changes from one edition to the next, except to comment on features which have become obsolete. ECMA-48 1st (1976) is unavailable; there is no reliable source of information which states whether "ANSI" color was defined in that edition, or later (1979). The VT100 (1978) implemented the most commonly used non-color video attributes which are given in the 2nd edition.

While 8-color support is described in ECMA-48 2nd edition, the VT500 series (introduced in 1993) were the first DEC terminals implementing "ANSI" color. The DEC terminal's use of color is known to differ from *xterm*; useful documentation on this series became available too late to influence *xterm*.

If 16-color support is compiled, the following *aixterm* controls 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:

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

If *xterm* is compiled with the 16-color support disabled, it supports the following, from *rxvt*:
Ps = 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-8613-6 has been interpreted in more than one way; *xterm* allows the semicolons separating the subparameters in this control to be replaced by colons (but after the first colon, colons must be used).

These ISO-8613-6 controls (marked in ECMA-48 5th edition as "reserved for future standardization") are supported by *xterm*:
Pm = 3 8 ; 2 ; *Pi* ; *Pr* ; *Pg* ; *Pb* -> Set foreground color to the closest match in *xterm*'s palette for the given RGB *Pr/Pg/Pb*. The color space identifier *Pi* is ignored.
Pm = 3 8 ; 5 ; *Ps* -> Set foreground color to *Ps*.

Pm = 4 8 ; 2 ; *Pi* ; *Pr* ; *Pg* ; *Pb* -> Set background color to the closest match in *xterm*'s palette for the given RGB *Pr*/*Pg*/*Pb*. The color space identifier *Pi* is ignored.
Pm = 4 8 ; 5 ; *Ps* -> Set background color to *Ps*.

This variation on ISO-8613-6 is supported for compatibility with KDE konsole:
Pm = 3 8 ; 2 ; *Pr* ; *Pg* ; *Pb* -> Set foreground color to the closest match in *xterm*'s palette for the given RGB *Pr*/*Pg*/*Pb*.
Pm = 4 8 ; 2 ; *Pr* ; *Pg* ; *Pb* -> Set background color to the closest match in *xterm*'s palette for the given RGB *Pr*/*Pg*/*Pb*.

If *xterm* is compiled with direct-color support, and the resource **directColor** is true, then rather than choosing the closest match, *xterm* asks the X server to directly render a given color.

CSI > *Ps* ; *Ps* 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.

Ps = 0 -> modifyKeyboard.
Ps = 1 -> modifyCursorKeys.
Ps = 2 -> modifyFunctionKeys.
Ps = 4 -> modifyOtherKeys.

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

CSI *Ps* n Device Status Report (DSR).
Ps = 5 -> Status Report.
Result ("OK") is CSI 0 n
Ps = 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 > *Ps* n

Disable modifiers which may be enabled via the CSI > *Ps*; *Ps* 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:

Ps = 0 -> modifyKeyboard.
Ps = 1 -> modifyCursorKeys.
Ps = 2 -> modifyFunctionKeys.
Ps = 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 ? *Ps* n

Device Status Report (DSR, DEC-specific).
Ps = 6 -> Report Cursor Position (DECXCPR) [row;column] as CSI ? *r* ; *c* R (assumes the default page, i.e., "1").
Ps = 1 5 -> Report Printer status as CSI ? 1 0 n (ready). or CSI ? 1 1 n (not ready).
Ps = 2 5 -> Report UDK status as CSI ? 2 0 n (unlocked) or CSI ? 2 1 n (locked).
Ps = 2 6 -> Report Keyboard status as CSI ? 2 7 ; 1 ; 0 ; 0 n (North American).

The last two parameters apply to VT300 & up (keyboard ready) and VT400 & up (LK01) respectively.

Ps = 5 3 -> Report Locator status as CSI ? 5 3 n Locator available, if compiled-in, or CSI ? 5 0 n No Locator, if not.
Ps = 5 5 -> Report Locator status as CSI ? 5 3 n Locator available, if compiled-in, or CSI ? 5 0 n No Locator, if not.
Ps = 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.
Ps = 6 2 -> Report macro space (DECMSR) as CSI *Pn* * { .
Ps = 6 3 -> Report memory checksum (DECCKSR) as DCS *Pt* ! x x x x ST .
Pt is the request id (from an optional parameter to the request).

The x's are hexadecimal digits 0-9 and A-F.
Ps = 7 5 -> Report data integrity as CSI ? 7 0 n (ready, no errors).
Ps = 8 5 -> Report multi-session configuration as CSI ? 8 3 n (not configured for multiple-session operation).

CSI > *Ps* 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:
Ps = 0 -> never hide the pointer.
Ps = 1 -> hide if the mouse tracking mode is not enabled.
Ps = 2 -> always hide the pointer, except when leaving the window.
Ps = 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), VT220 and up.

CSI *Ps* ; *Ps* " p
Set conformance level (DECSCL), VT220 and up.

Valid values for the first parameter:
Ps = 6 1 -> VT100.
Ps = 6 2 -> VT200.
Ps = 6 3 -> VT300.

Valid values for the second parameter:
Ps = 0 -> 8-bit controls.
Ps = 1 -> 7-bit controls (always set for VT100).
Ps = 2 -> 8-bit controls.

CSI *Ps* $ p
Request ANSI mode (DECRQM). For VT300 and up, reply DECRPM is
CSI *Ps*; *Pm*$ y
where *Ps* is the mode number as in SM/RM, and *Pm* is the mode value:
0 - not recognized
1 - set
2 - reset
3 - permanently set
4 - permanently reset

CSI ? *Ps* $ p
Request DEC private mode (DECRQM). For VT300 and up, reply DECRPM is
CSI ? *Ps*; *Pm*$ y

where *Ps* is the mode number as in DECSET/DECSET, *Pm* is the mode value as in the ANSI DECRQM.
Two private modes are read-only (i.e., 1 3 and 1 4), provided only for reporting their values using this control sequence. They correspond to the resources **cursorBlink** and **cursorBlinkXOR**.

CSI *Ps* q Load LEDs (DECLL), VT100.
Ps = 0 -> Clear all LEDS (default).
Ps = 1 -> Light Num Lock.
Ps = 2 -> Light Caps Lock.
Ps = 3 -> Light Scroll Lock.
Ps = 2 1 -> Extinguish Num Lock.
Ps = 2 2 -> Extinguish Caps Lock.
Ps = 2 3 -> Extinguish Scroll Lock.

CSI *Ps* SP q
Set cursor style (DECSCUSR), VT520.
Ps = 0 -> blinking block.
Ps = 1 -> blinking block (default).
Ps = 2 -> steady block.
Ps = 3 -> blinking underline.
Ps = 4 -> steady underline.
Ps = 5 -> blinking bar (*xterm*).
Ps = 6 -> steady bar (*xterm*).

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

CSI *Ps* ; *Ps* r
Set Scrolling Region [top;bottom] (default = full size of window) (DECSTBM), VT100.

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

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* ; *Ps* $ r
Change Attributes in Rectangular Area (DECCARA), VT400 and up.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.
Ps denotes the SGR attributes to change: 0, 1, 4, 5, 7.

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

CSI *Pl* ; *Pr* s

Set left and right margins (DECSLRM), VT420 and up. This is available only when DECLRMM is enabled.

CSI ? *Pm* s
Save DEC Private Mode Values. *Ps* values are the same as for DECSET.

CSI *Ps* ; *Ps* ; *Ps* t
Window manipulation (from *dtterm*, as well as extensions by *xterm*). These controls may be disabled using the *allowWindowOps* resource.

xterm uses *Extended Window Manager Hints* (EWMH) to maximize the window. Some window managers have incomplete support for EWMH. For instance, *fvwm*, *flwm* and *quartz-wm* advertise support for maximizing windows horizontally or vertically, but in fact equate those to the maximize operation.

Valid values for the first (and any additional parameters) are:
Ps = 1 -> De-iconify window.
Ps = 2 -> Iconify window.
Ps = 3 ; *x* ; *y* -> Move window to [x, y].
Ps = 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.
Ps = 5 -> Raise the *xterm* window to the front of the stacking order.
Ps = 6 -> Lower the *xterm* window to the bottom of the stacking order.
Ps = 7 -> Refresh the *xterm* window.
Ps = 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.
Ps = 9 ; 0 -> Restore maximized window.
Ps = 9 ; 1 -> Maximize window (i.e., resize to screen size).
Ps = 9 ; 2 -> Maximize window vertically.
Ps = 9 ; 3 -> Maximize window horizontally.
Ps = 1 0 ; 0 -> Undo full-screen mode.
Ps = 1 0 ; 1 -> Change to full-screen.
Ps = 1 0 ; 2 -> Toggle full-screen.
Ps = 1 1 -> Report *xterm* window state.
If the *xterm* window is non-iconified, it returns CSI 1 t .
If the *xterm* window is iconified, it returns CSI 2 t .
Ps = 1 3 -> Report *xterm* window position.
Result is CSI 3 ; *x* ; *y* t
Ps = 1 3 ; 2 -> Report *xterm* text-area position.

Result is CSI 3 ; *x* ; *y* t
Ps = 1 4 -> Report *xterm* text area size in pixels.
Result is CSI 4 ; *height* ; *width* t
Ps = 1 4 ; 2 -> Report *xterm* window size in pixels.
Normally *xterm*'s *window* is larger than its *text area*, since it includes the frame (or decoration) applied by the window manager, as well as the area used by a scroll-bar.
Result is CSI 4 ; *height* ; *width* t
Ps = 1 5 -> Report size of the screen in pixels.
Result is CSI 5 ; *height* ; *width* t
Ps = 1 6 -> Report *xterm* character size in pixels.
Result is CSI 6 ; *height* ; *width* t
Ps = 1 8 -> Report the size of the text area in characters.
Result is CSI 8 ; *height* ; *width* t
Ps = 1 9 -> Report the size of the screen in characters.
Result is CSI 9 ; *height* ; *width* t
Ps = 2 0 -> Report *xterm* window's icon label.
Result is OSC L *label* ST
Ps = 2 1 -> Report *xterm* window's title.
Result is OSC l *label* ST

Ps = 2 2 ; 0 -> Save *xterm* icon and window title on stack.
Ps = 2 2 ; 1 -> Save *xterm* icon title on stack.
Ps = 2 2 ; 2 -> Save *xterm* window title on stack.
Ps = 2 3 ; 0 -> Restore *xterm* icon and window title from stack.
Ps = 2 3 ; 1 -> Restore *xterm* icon title from stack.
Ps = 2 3 ; 2 -> Restore *xterm* window title from stack.

Ps >= 2 4 -> Resize to *Ps* lines (DECSLPP), VT340 and VT420. *xterm* adapts this by resizing its window.

CSI > *Ps* ; *Ps* t
This *xterm* control sets one or more features of the title modes. Each parameter enables a single feature.
Ps = 0 -> Set window/icon labels using hexadecimal.
Ps = 1 -> Query window/icon labels using hexadecimal.
Ps = 2 -> Set window/icon labels using UTF-8.
Ps = 3 -> Query window/icon labels using UTF-8. (See discussion of **Title Modes**)

CSI *Ps* SP t
Set warning-bell volume (DECSWBV), VT520.
Ps = 0 or 1 -> off.
Ps = 2 , 3 or 4 -> low.
Ps = 5 , 6 , 7 , or 8 -> high.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* ; *Ps* $ t
Reverse Attributes in Rectangular Area (DECRARA), VT400 and up.

Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.
Ps denotes the attributes to reverse, i.e., 1, 4, 5, 7.

CSI u Restore cursor (SCORC, also ANSI.SYS).

CSI *Ps* SP u
Set margin-bell volume (DECSMBV), VT520.
Ps = 1 -> off.
Ps = 2 , 3 or 4 -> low.
Ps = 0 , 5 , 6 , 7 , or 8 -> high.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* ; *Pp* ; *Pt* ; *Pl* ; *Pp* $ v
Copy Rectangular Area (DECCRA), VT400 and up.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.
Pp denotes the source page.
Pt ; *Pl* denotes the target location.
Pp denotes the target page.

CSI *Ps* $ w
Request presentation state report (DECRQPSR), VT320 and up.
Ps = 0 -> error.
Ps = 1 -> cursor information report (DECCIR).
Response is
DCS 1 $ u *Pt* ST
Refer to the VT420 programming manual, which requires six pages to document the data string *Pt*,
Ps = 2 -> tab stop report (DECTABSR).
Response is
DCS 2 $ u *Pt* ST
The data string *Pt* is a list of the tab-stops, separated by "/" characters.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* ' 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 *Ps* x Request Terminal Parameters (DECREQTPARM).
if *Ps* 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:
Ps -> the given *Ps* incremented by 2.
Pn = 1 <- no parity.

Pn = 1 <- eight bits.
Pn = 1 <- 2 8 transmit 38.4k baud.
Pn = 1 <- 2 8 receive 38.4k baud.
Pn = 1 <- clock multiplier.
Pn = 0 <- STP flags.

CSI *Ps* * x
Select Attribute Change Extent (DECSACE), VT420 and up.
Ps = 0 -> from start to end position, wrapped.
Ps = 1 -> from start to end position, wrapped.
Ps = 2 -> rectangle (exact).

CSI *Pc* ; *Pt* ; *Pl* ; *Pb* ; *Pr* $ x
Fill Rectangular Area (DECFRA), VT420 and up.
Pc is the character to use.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.

CSI *Ps* # y
Select checksum extension (XTCHECKSUM), *xterm*. The bits of *Ps* modify the calculation of the checksum returned by DECRQCRA:
0 -> do not negate the result.
1 -> do not report the VT100 video attributes.
2 -> do not omit checksum for blanks.
3 -> omit checksum for cells not explicitly initialized.
4 -> do not mask cell value to 8 bits or ignore combining characters.
5 -> do not mask cell value to 7 bits.

CSI *Pi* ; *Pg* ; *Pt* ; *Pl* ; *Pb* ; *Pr* * y
Request Checksum of Rectangular Area (DECRQCRA), VT420 and up.
Response is
DCS *Pi* ! x x x x ST
Pi is the request id.
Pg is the page number.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.
The x's are hexadecimal digits 0-9 and A-F.

CSI *Ps* ; *Pu* ' z
Enable Locator Reporting (DECELR).
Valid values for the first parameter:
Ps = 0 -> Locator disabled (default).
Ps = 1 -> Locator enabled.
Ps = 2 -> Locator enabled for one report, then disabled.
The second parameter specifies the coordinate unit for locator reports.
Valid values for the second parameter:
Pu = 0 <- or omitted -> default to character cells.
Pu = 1 <- device physical pixels.
Pu = 2 <- character cells.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* $ z
Erase Rectangular Area (DECERA), VT400 and up.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.

CSI *Pm* ' {
Select Locator Events (DECSLE).
Valid values for the first (and any additional parameters) are:
Ps = 0 -> only respond to explicit host requests (DECRQLP).
This is default. It also cancels any filter rectangle.
Ps = 1 -> report button down transitions.
Ps = 2 -> do not report button down transitions.
Ps = 3 -> report button up transitions.
Ps = 4 -> do not report button up transitions.

CSI # {
CSI *Ps* ; *Ps* # {
Push video attributes onto stack (XTPUSHSGR), *xterm*. The optional parameters correspond to the SGR encoding for video attributes, except for colors (which do not have a unique SGR code):
Ps = 1 -> Bold.
Ps = 2 -> Faint.
Ps = 3 -> Italicized.
Ps = 4 -> Underlined.
Ps = 5 -> Blink.
Ps = 7 -> Inverse.
Ps = 8 -> Invisible.
Ps = 9 -> Crossed-out characters.
Ps = 1 0 -> Foreground color.
Ps = 1 1 -> Background color.
Ps = 2 1 -> Doubly-underlined.

If no parameters are given, all of the video attributes are saved. The stack is limited to 10 levels.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* $ {
Selective Erase Rectangular Area (DECSERA), VT400 and up.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.

CSI *Pt* ; *Pl* ; *Pb* ; *Pr* # |
Report selected graphic rendition (XTREPORTSGR), *xterm*. The response is an SGR sequence which contains the attributes which are common to all cells in a rectangle.
Pt ; *Pl* ; *Pb* ; *Pr* denotes the rectangle.

CSI *Ps* $ |
Select columns per page (DECSCPP), VT340.
Ps = 0 -> 80 columns, default if *Ps* omitted.
Ps = 8 0 -> 80 columns.

Ps = 1 3 2 -> 132 columns.

CSI *Ps* ' |
Request Locator Position (DECRQLP).
Valid values for the parameter are:
Ps = 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 *Pe* ; *Pb* ; *Pr* ; *Pc* ; *Pp* & w

Parameters are [*event*;*button*;*row*;*column*;*page*].
Valid values for the event:
Pe = 0 -> locator unavailable - no other parameters sent.
Pe = 1 -> request - *xterm* received a DECRQLP.
Pe = 2 -> left button down.
Pe = 3 -> left button up.
Pe = 4 -> middle button down.
Pe = 5 -> middle button up.
Pe = 6 -> right button down.
Pe = 7 -> right button up.
Pe = 8 -> M4 button down.
Pe = 9 -> M4 button up.
Pe = 1 0 -> locator outside filter rectangle.
The "*button*" parameter is a bitmask indicating which buttons are pressed:
Pb = 0 <- no buttons down.
Pb & 1 <- right button down.
Pb & 2 <- middle button down.
Pb & 4 <- left button down.
Pb & 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 *Ps* * |
Select number of lines per screen (DECSNLS), VT420 and up.

CSI # } Pop video attributes from stack (XTPOPSGR), *xterm*. Popping restores the video-attributes which were saved using XTPUSHSGR to their previous state.

CSI *Pm* ' }

Insert *Ps* Column(s) (default = 1) (DECIC), VT420 and up.

CSI *Pm* ' ~
Delete *Ps* Column(s) (default = 1) (DECDC), VT420 and up.

Operating System Commands

OSC *Ps* ; *Pt* BEL

OSC *Ps* ; *Pt* ST
Set Text Parameters. For colors and font, if *Pt* 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.
Ps = 0 -> Change Icon Name and Window Title to *Pt*.
Ps = 1 -> Change Icon Name to *Pt*.
Ps = 2 -> Change Window Title to *Pt*.
Ps = 3 -> Set X property on top-level window. *Pt* should be in the form "*prop=value*", or just "*prop*" to delete the property.
Ps = 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.

Ps = 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:

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

Ps = 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 correspond-

ing 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 *Pt*. Each successive parameter changes the next color in the list. The value of *Ps* 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.

Ps = 1 0 -> Change VT100 text foreground color to *Pt*.
Ps = 1 1 -> Change VT100 text background color to *Pt*.
Ps = 1 2 -> Change text cursor color to *Pt*.
Ps = 1 3 -> Change mouse foreground color to *Pt*.
Ps = 1 4 -> Change mouse background color to *Pt*.
Ps = 1 5 -> Change Tektronix foreground color to *Pt*.
Ps = 1 6 -> Change Tektronix background color to *Pt*.
Ps = 1 7 -> Change highlight background color to *Pt*.
Ps = 1 8 -> Change Tektronix cursor color to *Pt*.
Ps = 1 9 -> Change highlight foreground color to *Pt*.

Ps = 4 6 -> Change Log File to *Pt*. This is normally disabled by a compile-time option.

Ps = 5 0 -> Set Font to *Pt*. These controls may be disabled using the *allowFontOps* resource. If *Pt* 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 *Pt* 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.

Ps = 5 1 -> reserved for Emacs shell.

Ps = 5 2 -> Manipulate Selection Data. These controls may

be disabled using the *allowWindowOps* resource. The parameter *Pt* is parsed as

Pc ; *Pd*

The first, *Pc*, may contain zero or more characters from the set c , p , s , 0 , 1 , 2 , 3 , 4 , 5 , 6 , and 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, *Pd*, 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.

Ps = 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.

Ps = 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).

Ps = 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.

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

The *dynamic colors* can also be reset to their default (resource) values:

Ps = 1 1 0 -> Reset VT100 text foreground color.

Ps = 1 1 1 -> Reset VT100 text background color.
Ps = 1 1 2 -> Reset text cursor color.
Ps = 1 1 3 -> Reset mouse foreground color.
Ps = 1 1 4 -> Reset mouse background color.
Ps = 1 1 5 -> Reset Tektronix foreground color.
Ps = 1 1 6 -> Reset Tektronix background color.
Ps = 1 1 7 -> Reset highlight color.
Ps = 1 1 8 -> Reset Tektronix cursor color.
Ps = 1 1 9 -> Reset highlight foreground color.

Ps = I ; *c* -> Set icon to file. Sun shelltool, CDE dtterm. The file is expected to be XPM format, and uses the same search logic as the **iconHint** resource.

Ps = l ; *c* -> Set window title. Sun shelltool, CDE dtterm.

Ps = L ; *c* -> Set icon label. Sun shelltool, CDE dtterm.

Privacy Message

PM *Pt* ST *xterm* implements no PM functions; *Pt* is ignored. *Pt* 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:

o prefix a key with the ESC character.

o 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
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
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.

The codes used for the *PC-style function keys* were inspired by a feature of the VT510, referred to in its reference manual as DECFNK. In the DECFNK scheme, codes 2-8 identify modifiers for function-keys and cursor-, editing-keypad keys. Unlike *xterm*, the VT510 limits the modifiers which can be used with cursor- and editing-keypad keys. Although the name "DECFNK" implies that it is a mode, the VT510 manual mentions it only as a feature, which (like *xterm*) interacts with the DECUDK feature. Unlike *xterm*, VT510/VT520 provide an extension to DECUDK (DECPFK and DECPAK) which apparently was the reason for the feature in those terminals, i.e., for identifying a programmable key rather than making it simple for applications to obtain modifier information. It is not described in the related VT520 manual. Neither manual was readily available at the time the feature was added to *xterm*.

On the other hand, the VT510 and VT520 reference manuals do document a related feature. That is its emulation of the SCO console, which is similar to the "xterm-sco" terminal description. The SCO console function-keys are less useful to applications developers than the approach used by *xterm* because

- the relationship between modifiers and the characters sent by function-keys is not readily apparent, and

- the scheme is not extensible, i.e., it is an *ad hoc* asssignment limited to two modifiers (*shift* and *control*).

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
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 *Ps* ; *Ps* ' 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 ? *Pm* h " or "DECRST CSI ? *Pm* 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 compatibility 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 *CbCxCy* (6 characters).

- *Cb* is button-1.

- *Cx* and *Cy* 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 *CbCxCy*.

- The low two bits of *Cb* 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(1)**).

- *Cx* and *Cy* 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 dis-

playing the *Alternate Screen Buffer*. 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 *Ps* ; *Ps* ; *Ps* ; *Ps* ; *Ps* 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 *CxCy*.

- otherwise:
  CSI T *CxCyCxCyCxCy*.
  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, *Cb*).

- 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 @ *CxCy*. (@ = 32 + 0 (button 1) + 32 (motion indicator)). Similarly, motion with button 3 down is reported as CSI M B *CxCy*. (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 *Cx* and *Cy* 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 *Cx* and *Cy* 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, *Cb* 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 *Cx* and *Cy* 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 *Cx* and *Cy* 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 *Ps* c *xterm* responds to Send Device Attributes (Primary DA) with these additional codes:
Ps = 4 -> Sixel graphics.

CSI ? *Pm* h
xterm has these additional private Set Mode values:
Ps = 8 0 -> Sixel scrolling.
Ps = 1 0 7 0 -> use private color registers for each graphic.
Ps = 8 4 5 2 -> Sixel scrolling leaves cursor to right of graphic.

DCS *Pa* ; *Pb* ; *Ph* q *Ps*..*Ps 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.
Pa -> pixel aspect ratio
Pb -> background color option
Ph -> horizontal grid size (ignored).
Ps -> 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 *Ps* c *xterm* responds to Send Device Attributes (Primary DA) with these additional codes:
Ps = 3 -> ReGIS graphics.

CSI ? *Pm* h
xterm has these additional private Set Mode values:
Ps = 1 0 7 0 -> use private color registers for each graphic.

DCS *Pm* p *Pr*..*Pr ST*
See:

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

The ReGIS data device control string has one positional parameter with four possible values:
Pm = 0 -> resume command, use fullscreen mode.
Pm = 1 -> start new command, use fullscreen mode.
Pm = 2 -> resume command, use command display mode.
Pm = 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.

Some of the sequences are specific to *xterm*. The Tektronix emulation was added in X10R4 (1986). The VT240, introduced two years earlier, also supported Tektronix 4010/4014. Unlike *xterm*, the VT240 documentation implies (there is an obvious error in section 6.9 "Entering and

Exiting 4010/4014 Mode") that exiting back to ANSI mode is done by resetting private mode 3 8 (DECTEK) rather than ESC ETX . A real Tektronix 4014 would not respond to either.

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 *Ps* ; *Pt* BEL
Set Text Parameters of VT window.
Ps = 0 -> Change Icon Name and Window Title to *Pt*.
Ps = 1 -> Change Icon Name to *Pt*.
Ps = 2 -> Change Window Title to *Pt*.

Ps = 4 6 -> Change Log File to *Pt*. 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 *Ps Ps*
  Move the cursor to given row and column.

ESC Z Identify.
  -> ESC / Z ("I am a VT52.").

Further reading
Technical manuals
Manuals for *hardware* terminals are more readily available than similarly-detailed documentation for terminal *emulators* such as *aixterm*, *shelltool*, *dtterm*.

However long, the technical manuals have problems:

- DEC's manuals did not provide a comprehensive comparison of the features in different model.

  Peter Sichel's *Host Interface Functions Checklist* spreadsheet is useful for noting which model introduced a given feature (although there are a few apparent errors such as the **DECRQSS** feature cited for VT320 whereas the technical manual omits it).

- Sometimes the manuals disagree. For example, DEC's standard document (DEC STD 070) for terminals says that **DECSCL** performs a *soft* reset (DECSTR), while the VT420 manual says it does a *hard* reset (RIS).

- Sometimes the manuals are simply incorrect. For example, testing a

DEC VT420 in 1996 showed that the documented code for a valid or invalid response to **DECRQSS** was reversed.

The VT420 test results were incorporated into *vttest* program. At the time, DEC STD 070 was not available, but it also agrees with *vttest*. Later, documentation for the DEC VT525 was shown to have the same flaw.

- Not all details are clear even in DEC STD 070 (which is more than twice the length of the VT520 programmer's reference manual, and almost three times longer than the VT420 reference manual). However, as an internal standards document, DEC STD 070 is more likely to describe the actual behavior of DEC's terminals than the more polished user's guides.

That said, here are technical manuals which have been used in developing *xterm*. Not all were available initially. In August 1996 for instance, the technical references were limited to EK-VT220-HR-002 and EK-VT420-UG.002. Shortly after, Richard Shuford sent a copy of EK-VT3XX-TP-001. Still later (beginning in 2003), Paul Williams' vt100.net site provided EK-VT102-UG-003, EK-VT220-RM-002, EK-VT420-RM-002, EK-VT520-RM A01, EK-VT100-TM-003, and EK-VT102-UG-003. The remaining documents were found on the bitsavers site.

- *DECscope User's Manual*.
  Digital Equipment Corporation (EK-VT5X-OP-001 1975).

- *VT100 Series Video Terminal Technical Manual*.
  Digital Equipment Corporation (EK-VT100-TM-003, July 1982).

- *VT100 User Guide*.
  Digital Equipment Corporation (EK-VT100-UG-003, June 1981).

- *VT102 User Guide*.
  Digital Equipment Corporation (EK-VT102-UG-003, February 1982).

- *VT220 Programmer Pocket Guide*.
  Digital Equipment Corporation (EK-VT220-HR-002, July 1984).

- *VT220 Programmer Reference Manual*.
  Digital Equipment Corporation (EK-VT220-RM-002, August 1984).

- *VT240 Programmer Reference Manual*.
  Digital Equipment Corporation (EK-VT240-RM-002, October 1984).

- *VT330/VT340 Programmer Reference Manual*
  Volume 1: Text Programming.
  Digital Equipment Corporation (EK-VT3XX-TP-001, March 1987).

- *VT330/VT340 Programmer Reference Manual*

Volume 2: Graphics Programming.
Digital Equipment Corporation (EK-VT3XX-GP-001, March 1987).

- *Installing and Using*
  The VT420 Video Terminal
  (North American Model).
  Digital Equipment Corporation (EK-VT420-UG.002, February 1990).

- *VT420 Programmer Reference Manual*.
  Digital Equipment Corporation (EK-VT420-RM-002, February 1992).

- *VT510 Video Terminal*
  Programmer Information.
  Digital Equipment Corporation (EK-VT510-RM B01, November 1993).

- *VT520/VT525 Video Terminal*
  Programmer Information.
  Digital Equipment Corporation (EK-VT520-RM A01, July 1994).

- *Digital ANSI-Compliant Printing Protocol*
  Level 2 Programming Reference Manual
  Digital Equipment Corporation (EK-PPLV2-PM. B01, August 1994).

- *4014 and 4014-1 Computer Display Terminal*
  User's Manual.
  Tektronix, Inc. (070-1647-00, November 1979).

Standards

The DEC terminal family (VT100 through VT525) is upward-compatible, using standards plus *extensions*, e.g., "private modes". Not all commonly-used features are standard. For example, scrolling regions are not found in ECMA-48.

- *ECMA-35: Character Code Structure and Extension Techniques*
  (6th Edition, December 1994).

- *ECMA-48: Control Functions for Coded Character Sets*
  (5th Edition, June 1991).

- *DEC STD 070 Video Systems Reference Manual*.
  Digital Equipment Corporation (A-MN-ELSM070-00-0000 Rev H, December 3, 1991).

Miscellaneous

A few hardware terminals survived into the 1990s only as terminal emulators. Documentation for these and other terminal emulators which have influenced *xterm* are generally available only in less-accessible and less-detailed manual pages.

- *XTerm* supports control sequences for manipulating its *window* which

were implemented by Sun's *shelltool* program. This was part of SunView (SunOS 3.0, 1986). The change-notes for *xterm*'s *resize* program in X10.4 (1986) mention its use of these "Sun tty emulation escape sequences" for resizing the window. The X10.4 *xterm* program recognized these sequences for resizing the terminal, except for the iconfig/deiconfy pair. SunView also introduced the **SIGWINCH** signal, used by the X10.4 *xterm* and mentioned in its *CHANGES* file:

> The window size is passed to the operating system via TIOCSWINSZ (4.3) or TIOCSSIZE (sun). A SIGWINCH signal is sent if the vtXXX window is resized.

While support for the Sun control-sequences remained in *resize*, the next release of *xterm* (X11R1 in 1987) omitted the code for interpreting them.

Later, the SunView program was adapted for the *OPEN LOOK* environment introduced 1988-1990.

Still later, in 1995, *OPEN LOOK* was abandoned in favor of *CDE*. The *CDE* terminal emulator *dtterm* implemented those controls, with a couple of additions.

Starting in July 1996, *xterm* re-implemented those control sequences (based on the *dtterm* manual pages) and further extended the group of window controls.

There were two sets of controls (CSI *Ps*[; *Pm* ; *Pm*]t , and OSC *Pstext*ST) implemented by *shelltool*, documented in appendix E of both *PHIGS Programming Manual* (1992), and the unpublished *X Window System User's Guide (OPEN LOOK Edition)* (1995). The *CDE* program kept those, and added a few new ones.

Code	**Sun**	**CDE**	*xterm*	*Description*
CSI 1 t	yes	yes	yes	de-iconify
CSI 2 t	yes	yes	yes	iconify
CSI 3 t	yes	yes	yes	move window to pixel-position
CSI 4 t	yes	yes	yes	resize window in pixels
CSI 5 t	yes	yes	yes	raise window to front of stack
CSI 6 t	yes	yes	yes	raise window to back of stack
CSI 7 t	yes	yes	yes	refresh window
CSI 8 t	yes	yes	yes	resize window in chars
CSI 9 t	-	-	yes	maximize/unmaximize window
CSI 1 0 t	-	-	yes	to/from full-screen
CSI 1 1 t	yes	yes	yes	report if window is iconified
CSI 1 2 t	-	-	-	-
CSI 1 3 t	yes	yes	yes	report window position
CSI 1 4 t	yes	yes	yes	report window size in pixels
CSI 1 5 t	-	-	yes	report screen size in pixels

CSI 1 6 t	-	-	yes	report character cell in pixels
CSI 1 7 t	-	-	-	-
CSI 1 8 t	yes	yes	yes	report window size in chars
CSI 1 9 t	-	-	yes	report screen size in chars
CSI 2 0 t	-	yes	yes	report icon label
CSI 2 1 t	-	yes	yes	report window title
CSI 2 2 t	-	-	yes	save window/icon title
CSI 2 3 t	-	-	yes	restore window/icon title
CSI 2 4 t	-	-	yes	resize window (DECSLPP)
OSC 0 ST	-	yes	yes	set window and icon title
OSC 1 ST	-	yes	yes	set icon label
OSC 2 ST	-	yes	yes	set window title
OSC 3 ST	-	n/a	yes	set X server property
OSC I ST	yes	yes	yes	set icon to file
OSC l ST	yes	yes	yes	set window title
OSC L ST	yes	yes	yes	set icon label

Besides the Sun-derived OSC controls for setting window title and icon label, *dtterm* also supported the *xterm* controls for the same feature.

The *CDE* source was unavailable for inspection until 2012, so that clarification of the details of the window operations relied upon *vttest*.

o The control sequences for saving/restoring the cursor and for saving/restoring "DEC Private Mode Values" may appear to be related (since the "save" controls both end with s), but that is coincidental. The latter was introduced in X10.4:

> Most Dec Private mode settings can be save away internally using \E[?ns, where n is the same number to set or reset the Dec Private mode. The mode can be restored using \E[?nr. This can be used in termcap for vi, for example, to turn off saving of lines, but restore whatever the original state was on exit.

while the SCOSC/SCORC pair was added in 1995 by XFree86 (and documented long afterwards).

o The *aixterm* manual page gives the format of the control sequence for foreground and background colors 8-15, but does not specify what those colors are. That is implied by the description's mention of *HFT*:

> The aixterm command provides a standard terminal type for programs that do not interact directly with Enhanced X-Windows. This command provides an emulation for a VT102 terminal or a high function terminal (HFT). The VT102 mode is activated by the -v flag.

Unlike *xterm*, there are no resource names for the 16 colors, leaving the reader to assume that the mapping is hard-coded. The control sequences for colors 8-15 are not specified by ECMA-48, but rather (as done in other instances by *xterm*) chosen to not conflict with current or future standards.