*File created:* 4-Jul-90 02:21:01 **{PELE:MV:ENVOS}<LISPCORE>LIBRARY>CHATTERMINAL.;3**

*changes to:* (VARS CHATTERMINALCOMS)

*previous date:* 14-Jun-90 15:57:05 {DSK}<users>sybalsky>bane>chatterminal.;1

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS

```
;;
;; Copyright (c) 1984, 1985, 1986, 1987, 1988, 1990 by VENUE & Xerox Corporation.  All rights reserved.

(RPAQQ CHATTERMINALCOMS
       ((FNS TERM.ADDCHAR TERM.ADDLINE TERM.CLEAR.TAB TERM.DELCHAR TERM.DELETELINE TERM.DOWN
             TERM.ERASE.IN.DISPLAY TERM.ERASE.IN.LINE TERM.ERASE.TO.EOL TERM.ERASEBITS TERM.GODOWN TERM.HOME
             TERM.IDENTIFY.SELF TERM.LEFT TERM.MODIFY.ATTRIBUTES TERM.MOVETO TERM.NEWLINE TERM.PRINTCHAR
             TERM.RESET.DISPLAY.PARMS TERM.RIGHT TERM.SCROLLDOWN TERM.SET.TAB TERM.SETMARGINS TERM.SMOOTHSCROLL
             TERM.TAB TERM.UP)
        (VARIABLES CHAT.UNDERLINE.DESCENT CHAT.UNDERLINE.METHOD)
        (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (SOURCE)
                                                CHATDECLS)
               (LOCALVARS . T))))

(DEFINEQ

(TERM.ADDCHAR
  [LAMBDA (CHAT.STATE)                                              (* ejs%: "12-May-85 14:43")
                                                                    ; Insert a space at cursor position, pushing rest of line to right
    (LET ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
          (Y (- (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE)))
          (FONTWIDTH (ffetch (CHAT.STATE FONTWIDTH) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE)))
         (BITBLT DSP XPOS Y DSP (+ XPOS FONTWIDTH)
                Y
                (+ (- (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE)
                      XPOS)
                   FONTWIDTH)
                FONTHEIGHT
                'INPUT
                'REPLACE)                                           ; Blt remainder of line to the right by FONTWIDTH, then erase
                                                                    ; the character under the cursor
         (TERM.ERASEBITS CHAT.STATE XPOS Y FONTWIDTH FONTHEIGHT])

(TERM.ADDLINE
  [LAMBDA (CHAT.STATE ATYPOS)                                       (* ejs%: "12-May-85 14:44")
    (LET [[DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
          (Y (- (OR ATYPOS (ffetch (CHAT.STATE YPOS) of CHAT.STATE))
                (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE]
        ;; To insert line at y, we blt everything below it down one, then clear the line at y
        (BITBLT DSP 0 FONTHEIGHT DSP 0 0 TTYWIDTH Y 'INPUT 'REPLACE)
        (TERM.ERASEBITS CHAT.STATE 0 Y TTYWIDTH FONTHEIGHT])

(TERM.CLEAR.TAB
  [LAMBDA (CHAT.STATE TERMINAL.X)                                   (* ejs%: "12-May-85 14:45")

;;; Clear a tab stop

    (freplace (CHAT.STATE TERM.TAB.STOPS) of (\DTEST CHAT.STATE 'CHAT.STATE) with (DREMOVE TERMINAL.X
                                                                                           (ffetch (CHAT.STATE
                                                                                                    TERM.TAB.STOPS
                                                                                                    )
                                                                                                   of])

(TERM.DELCHAR
  [LAMBDA (CHAT.STATE)                                              (* ejs%: "12-May-85 14:46")
    ;; Delete character under cursor, moving rest of line to left
    (LET [[DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
          (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
          (FONTWIDTH (ffetch (CHAT.STATE FONTWIDTH) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
          (Y (- (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE]
         (BITBLT DSP (+ XPOS FONTWIDTH)
```

```
               Y DSP XPOS Y (+ (- TTYWIDTH XPOS)
                               FONTWIDTH)
               FONTHEIGHT
               'INPUT
               'REPLACE)                                    ; Blt remainder of line to the left by FONTWIDTH, then erase the
                                                            ; rightmost character position
         (TERM.ERASEBITS CHAT.STATE (- TTYWIDTH FONTWIDTH)
               Y FONTWIDTH FONTHEIGHT])
```

## (TERM.DELETELINE

```
  [LAMBDA (CHAT.STATE ATYPOS)                               (* ejs%: "12-May-85 19:08")
    (LET ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
          (BOTTOMMARGIN (ffetch (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE)))
                                                            ; To delete line at ATYPOS, we blt everything below it up one,
                                                            ; then clear the bottom line
         (BITBLT DSP 0 BOTTOMMARGIN DSP 0 (+ FONTHEIGHT BOTTOMMARGIN)
               TTYWIDTH
               (- (OR ATYPOS (ffetch (CHAT.STATE YPOS) of CHAT.STATE))
                  (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE)
                  BOTTOMMARGIN)
               'INPUT
               'REPLACE)
         (TERM.ERASEBITS CHAT.STATE 0 BOTTOMMARGIN TTYWIDTH FONTHEIGHT])
```

## (TERM.DOWN

```
  [LAMBDA (CHAT.STATE NLINES)                               (* ejs%: "12-May-85 19:06")
```

;;; Move down NLINES (default = 1), pegging at bottom if NLINES not 1, else wrap or roll

```
    (LET [[DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
          (YPOS (ffetch (CHAT.STATE YPOS) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
          (BOTTOM (+ (ffetch (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE)
                     (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE]
         (COND
            ((> YPOS BOTTOM)
             (MOVETO XPOS [freplace (CHAT.STATE YPOS) of CHAT.STATE with (IMAX BOTTOM
                                                                                (- YPOS (ITIMES FONTHEIGHT
                                                                                               (OR NLINES 1]
                    DSP))
            ((NULL (ffetch (CHAT.STATE ROLLMODE) of CHAT.STATE))   ; Wraparound to top
             (MOVETO XPOS (freplace (CHAT.STATE YPOS) of CHAT.STATE with (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE))
                    DSP))
            (T                                              ; On bottom line in rollmode, scroll screen up one.  Ypos does
                                                            ; not change
               (TERM.DELETELINE CHAT.STATE (- (ffetch (CHAT.STATE TOPMARGIN) of CHAT.STATE)
                                              FONTHEIGHT])
```

## (TERM.ERASE.IN.DISPLAY

```
  [LAMBDA (CHAT.STATE PARAM)                                (* ejs%: "12-May-85 14:48")
                                                            ; Misc erasing functions

    (\DTEST CHAT.STATE 'CHAT.STATE)
    (LET ((TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
          (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
          (YPOS (ffetch (CHAT.STATE YPOS) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
          (FONTDESCENT (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE)))
         (SELECTQ PARAM
              (0                                            ; Erase to end of screen
                 (TERM.ERASE.TO.EOL CHAT.STATE)
                 (TERM.ERASEBITS CHAT.STATE 0 0 TTYWIDTH (- YPOS FONTDESCENT)))
              (1                                            ; Erase from HOME to current position
                 (TERM.ERASEBITS CHAT.STATE 0 (+ YPOS FONTHEIGHT)
                        TTYWIDTH
                        (- (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE)
                           (+ YPOS FONTHEIGHT)))
                 (TERM.ERASEBITS CHAT.STATE 0 (- YPOS FONTDESCENT)
                        XPOS FONTHEIGHT))
              (2                                            ; Erase screen
                 (CLEARW (ffetch (CHAT.STATE WINDOW) of CHAT.STATE))
                 (MOVETO XPOS YPOS (ffetch (CHAT.STATE DSP) of CHAT.STATE)))
              NIL])
```

## (TERM.ERASE.IN.LINE

```
  [LAMBDA (CHAT.STATE PARAM)                                (* ejs%: "12-May-85 14:48")
                                                            ; Do line-oriented erasing

    (CASE PARAM
        (0                                                  ; Erase to end-of-line
           (TERM.ERASE.TO.EOL CHAT.STATE))
        ((1 2)                                              ; Erase from beginning of line to current pos, or entire line
```

```
          (TERM.ERASEBITS CHAT.STATE 0 (- (ffetch (CHAT.STATE YPOS) of (\DTEST CHAT.STATE 'CHAT.STATE))
                                          (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE))
                 (if (EQ PARAM 1)
                     then (ffetch (CHAT.STATE XPOS) of CHAT.STATE)
                   else (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
                 (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))))])
```

## (TERM.ERASE.TO.EOL

```
(TERM.ERASE.TO.EOL
  [LAMBDA (CHAT.STATE)                                          (* ejs%: "12-May-85 16:18")
    (LET [(XPOS (ffetch (CHAT.STATE XPOS) of (\DTEST CHAT.STATE 'CHAT.STATE]
         (TERM.ERASEBITS CHAT.STATE XPOS (- (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                                            (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE))
                (- (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE)
                   XPOS)
                (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE])
```

## (TERM.ERASEBITS

```
(TERM.ERASEBITS
  [LAMBDA (CHAT.STATE LEFT BOTTOM WIDTH HEIGHT)                 ; Edited  8-Dec-87 14:03 by jrb:
    (LET ((DSP (fetch (CHAT.STATE DSP) of CHAT.STATE)))
         (BITBLT NIL NIL NIL DSP LEFT BOTTOM WIDTH HEIGHT 'TEXTURE 'REPLACE (DSPTEXTURE NIL DSP])
```

## (TERM.GODOWN

```
(TERM.GODOWN
  [LAMBDA (CHAT.STATE NLINES)                                   (* ejs%: "12-May-85 14:49")
```

;;; Move down NLINES, pegging at the bottom of the window

```
    (LET [[YPOS (ffetch (CHAT.STATE YPOS) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (BOTTOM (+ (ffetch (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE)
                     (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE]
         (COND
            ((> YPOS BOTTOM)
             (MOVETO (ffetch (CHAT.STATE XPOS) of CHAT.STATE)
                    [freplace (CHAT.STATE YPOS) of CHAT.STATE with (IMAX BOTTOM (- YPOS (ITIMES (ffetch (CHAT.STATE
                                                                                                 FONTHEIGHT)
                                                                                            of CHAT.STATE)
                                                                                         (OR NLINES 1]
                    (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.HOME

```
(TERM.HOME
  [LAMBDA (CHAT.STATE)                                          (* ejs%: "12-May-85 16:59")
    (\DTEST CHAT.STATE 'CHAT.STATE)
    (MOVETO (freplace (CHAT.STATE XPOS) of CHAT.STATE with 0)
           (freplace (CHAT.STATE YPOS) of CHAT.STATE with (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE))
           (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.IDENTIFY.SELF

```
(TERM.IDENTIFY.SELF
  [LAMBDA (CHAT.STATE)                                          (* ejs%: "12-May-85 20:22")
```

;;; Identify self to the host operating system

```
    (with CHAT.STATE CHAT.STATE (PRIN3 TERM.IDENTITY.STRING OUTSTREAM)
          (FORCEOUTPUT OUTSTREAM])
```

## (TERM.LEFT

```
(TERM.LEFT
  [LAMBDA (CHAT.STATE NCHARS)                                   (* ejs%: "12-May-85 14:50")
```

;;; Move the cursor NCHARS (default = 1), pegging at the left margin

```
    (LET [(XPOS (ffetch (CHAT.STATE XPOS) of (\DTEST CHAT.STATE 'CHAT.STATE]
         (COND
            ((> XPOS 0)
             (MOVETO [freplace (CHAT.STATE XPOS) of CHAT.STATE with (IMAX 0 (- XPOS (ITIMES (ffetch (CHAT.STATE
                                                                                                  FONTWIDTH)
                                                                                             of CHAT.STATE)
                                                                                          (OR NCHARS 1]
                     (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                     (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.MODIFY.ATTRIBUTES

```
(TERM.MODIFY.ATTRIBUTES
  [LAMBDA (CHAT.STATE ATTRIBUTES INVERTFLG)                     (* ejs%: "18-Mar-86 16:40")
```

;;; Function to do character attribute setting.  Attributes is a list of attribute modifying commands

```
    (LET ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (PLAINFONT (ffetch (CHAT.STATE PLAINFONT) of CHAT.STATE)))
         (for A inside ATTRIBUTES
            do (PROG NIL
                  RETRY
                     (CASE A
                         (NORMAL                                ; What does inverting normal mean?
```

```
                        (DSPFONT PLAINFONT DSP)
                        (DSPSOURCETYPE 'INPUT DSP)
                        (freplace (CHAT.STATE UNDERLINEMODE) of CHAT.STATE with NIL))
                    (BRIGHT                                  ; Implement 'BRIGHT' by using a bold font
                        (DSPFONT [if INVERTFLG
                                     then PLAINFONT
                                 else (OR (ffetch (CHAT.STATE CHATBOLDFONT) of CHAT.STATE)
                                          (freplace (CHAT.STATE CHATBOLDFONT) of CHAT.STATE
                                             with (FONTCOPY PLAINFONT 'WEIGHT 'BOLD]
                                 DSP))
                    (ITALIC (DSPFONT [if INVERTFLG
                                         then PLAINFONT
                                     else (OR (ffetch (CHAT.STATE ITALICFONT) of CHAT.STATE)
                                              (freplace (CHAT.STATE ITALICFONT) of CHAT.STATE
                                                 with (FONTCOPY PLAINFONT 'SLOPE 'ITALIC]
                                     DSP))
                    ((BLINK UNDERLINE)                       ; Implement 'BLINK' with underline, for now.  Blinking characters
                                                             ; are probably a bit too expensive
                        (CASE CHAT.UNDERLINE.METHOD
                            ((BOLD BRIGHT)
                               (SETQ A 'BRIGHT)
                               (GO RETRY))
                            ((INVERSE INVERT)
                               (SETQ A 'INVERSE)
                               (GO RETRY))
                            (ITALIC
                               (SETQ A 'ITALIC)
                               (GO RETRY))
                            (T                               ; The default: underline
                               (freplace (CHAT.STATE UNDERLINEMODE) of CHAT.STATE with (NOT INVERTFLG)))))
                    (INVERSE                                 ; Inverse video we can do directly
                        (DSPSOURCETYPE (if INVERTFLG
                                           then              ; Inverse of inverse is normal
                                                'INPUT
                                       else 'INVERT)
                                DSP)))])
```

## (TERM.MOVETO

```
  [LAMBDA (CHAT.STATE CX CY)                                 (* ejs%: "12-May-85 14:53")
```

;;; Set our cursor position

```
    (LET [(FONTWIDTH (ffetch (CHAT.STATE FONTWIDTH) of (\DTEST CHAT.STATE 'CHAT.STATE]
         (MOVETO (freplace (CHAT.STATE XPOS) of CHAT.STATE with (IMIN (ITIMES CX FONTWIDTH)
                                                                      (- (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE)
                                                                         FONTWIDTH)))
                 [freplace (CHAT.STATE YPOS) of CHAT.STATE with (IMAX (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE)
                                                                      (- (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE)
                                                                         (ITIMES CY (ffetch (CHAT.STATE FONTHEIGHT)
                                                                                       of CHAT.STATE]
                 (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.NEWLINE

```
  [LAMBDA (CHAT.STATE)                                       (* ejs%: "12-May-85 14:54")
                                                             ; Do a CRLF.
    (TERM.DOWN (\DTEST CHAT.STATE 'CHAT.STATE))
    (MOVETO (freplace (CHAT.STATE XPOS) of CHAT.STATE with 0)
           (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
           (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.PRINTCHAR

```
  [LAMBDA (CHAT.STATE CHAR WRAPFN)                           ; Edited  2-Sep-88 10:35 by jds
```

;;; Print a character.  If this char fills the last position on the line, then the next action is determined by WRAPFN: if it is given, we call it with
;;; CHAT.STATE as arg.  Otherwise, if WRAPMODE is on in the state, we perform an explict newline, else we peg at the right margin.

```
    (LET* ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
           (DISPLAYDATA (fetch (STREAM IMAGEDATA) of DSP))
           (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
           IMAGEWIDTH CHARWIDTH)
          (if (NEQ (ffetch (\DISPLAYDATA DDCHARSET) of DISPLAYDATA)
                   (\CHARSET CHAR))
              then ;; The display stream's caches are invalid.  Fix them up for the new character set (this also cleans up after font changes, etc,
                   ;; and at initial window opening)
                   (\CHANGECHARSET.DISPLAY DISPLAYDATA (\CHARSET CHAR)))
          ;; These two SETQs can't be in the LET* because the charset change may need to happen first:
          (SETQ IMAGEWIDTH (\DSPGETCHARIMAGEWIDTH CHAR DISPLAYDATA))
          (SETQ CHARWIDTH (\DSPGETCHARWIDTH CHAR DISPLAYDATA))
          (if (NEQ IMAGEWIDTH CHARWIDTH)
              then ;; Take care of the case where the character's image isn't the same as its escapement, by filling in the background properly for
                   ;; the intervening space.  We wouldn't have to worry about this nonsense if ns fonts did their character bitmaps properly.
```

```
                (\BLTSHADE.DISPLAY (CASE (fetch DDSOURCETYPE of DISPLAYDATA)
                                         (INVERT BLACKSHADE)
                                         (T WHITESHADE))
                       DSP
                       (+ XPOS IMAGEWIDTH)
                       (- (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                          (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE))
                       (- CHARWIDTH IMAGEWIDTH)
                       (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE)))
        (\BLTCHAR CHAR DSP DISPLAYDATA)
        (if (ffetch (CHAT.STATE UNDERLINEMODE) of CHAT.STATE)
            then                                                    ; Underline what we just drew
                 (\BLTSHADE.DISPLAY BLACKSHADE DSP XPOS (- (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                                                           CHAT.UNDERLINE.DESCENT)
                       CHARWIDTH 1 'INVERT))
        (if (>= (freplace (CHAT.STATE XPOS) of CHAT.STATE with (+ XPOS (ffetch (CHAT.STATE FONTWIDTH) of CHAT.STATE)
                                                                  ))
                (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
            then                                                    ; Have reached right margin, so wrap around
                 (if WRAPFN
                     then                                           ; Terminal-specific wrap handler
                          (CL:FUNCALL WRAPFN CHAT.STATE)
                   elseif (ffetch (CHAT.STATE WRAPMODE) of CHAT.STATE)
                     then (TERM.NEWLINE CHAT.STATE)
                     else                                           ; No, don't wrap--stay on the last character
                          (MOVETO (freplace (CHAT.STATE XPOS) of CHAT.STATE with XPOS)
                                  (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                                  DSP])
```

## (TERM.RESET.DISPLAY.PARMS

```
  [LAMBDA (CHAT.STATE)                                              ; Edited 21-May-90 00:00 by jrb:
```

;; Reset state, assuming window coords are as if CLEARW was just called.

```
    (LET* ([WINDOW (ffetch (CHAT.STATE WINDOW) of (\DTEST CHAT.STATE 'CHAT.STATE]
           (DSP (ffetch (CHAT.STATE DSP) of CHAT.STATE))
           (FONT (PROGN (DSPFONT (OR CHAT.FONT (DEFAULTFONT 'DISPLAY))
                                 DSP)                               ; Reset default font, and read it back after display has coerced it
                                                                    ; as necessary
                        (DSPFONT NIL DSP)))
           (FONTDESCENT (FONTPROP FONT 'DESCENT))
           (FONTWIDTH (CHARWIDTH (CHARCODE A)
                                 FONT))
           (FONTHEIGHT (FONTPROP FONT 'HEIGHT))
           (CLEARMODEFN (ffetch (CHAT.STATE CLEARMODEFN) of CHAT.STATE))
           TERM.STATE)
          (freplace (CHAT.STATE PLAINFONT) of CHAT.STATE with (freplace (CHAT.STATE FONT) of CHAT.STATE with FONT))
          (freplace (CHAT.STATE CHATBOLDFONT) of CHAT.STATE with (freplace (CHAT.STATE ITALICFONT) of CHAT.STATE
                                                                           with NIL))
          (freplace (CHAT.STATE FONTHEIGHT) of CHAT.STATE with FONTHEIGHT)
          (freplace (CHAT.STATE FONTWIDTH) of CHAT.STATE with FONTWIDTH)
          (freplace (CHAT.STATE FONTDESCENT) of CHAT.STATE with FONTDESCENT)
```

;; We use just the part of window that is even multiple of the font width and height

```
          (LET ((TTYHEIGHT (+ (ITIMES (IQUOTIENT (WINDOWPROP WINDOW 'HEIGHT)
                                                 FONTHEIGHT)
                                      FONTHEIGHT)
                              FONTDESCENT)))
               (freplace (CHAT.STATE TTYHEIGHT) of CHAT.STATE with (freplace (CHAT.STATE TOPMARGIN) of CHAT.STATE
                                                                             with TTYHEIGHT))
```

;; JRB Just guessing that nobody sets BOTTOMMARGIN, or that somebody is clobbering it...

```
               (freplace (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE with 0)
               (freplace (CHAT.STATE HOMEPOS) of CHAT.STATE with (- TTYHEIGHT FONTHEIGHT))
               (freplace (CHAT.STATE TTYWIDTH) of CHAT.STATE with (ITIMES (IQUOTIENT (WINDOWPROP WINDOW
                                                                                                 'WIDTH)
                                                                                     FONTWIDTH)
                                                                         FONTWIDTH)))
          (if (AND CLEARMODEFN (SETQ TERM.STATE (ffetch (CHAT.STATE TERM.STATE) of CHAT.STATE)))
              then ;; Clear any funny mode the terminal might have gotten into.  Test for TERM.STATE is because when we are called at startup,
                   ;; TERM.STATE might not be filled in yet (what a crock).
                   (CL:FUNCALL CLEARMODEFN CHAT.STATE TERM.STATE])
```

## (TERM.RIGHT

```
  [LAMBDA (CHAT.STATE NCHARS)                                       (* ejs%: "12-May-85 15:33")
```

;;; Move the cursor NCHARS to the right, pegging at the right margin

```
    (LET ([TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (XPOS (ffetch (CHAT.STATE XPOS) of CHAT.STATE))
          (FONTWIDTH (ffetch (CHAT.STATE FONTWIDTH) of CHAT.STATE)))
         (COND
            ((< (+ XPOS FONTWIDTH)
                TTYWIDTH)
              (MOVETO [freplace (CHAT.STATE XPOS) of CHAT.STATE with (IMIN TTYWIDTH (+ XPOS (ITIMES FONTWIDTH
```

```
                                                                                    (OR NCHARS 1]
                    (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                    (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.SCROLLDOWN

```
[LAMBDA (CHAT.STATE TOP)                                    (* ejs%: "12-May-85 14:56")
```

;;; Scroll down a line, from the line at TOP

```
    (LET* ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
           (TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
           (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE))
           (BOTTOMMARGIN (ffetch (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE))
           (NEARBOTTOM (+ BOTTOMMARGIN FONTHEIGHT)))
          ;; Move most of window down one line, then erase the top line
          (BITBLT DSP 0 NEARBOTTOM DSP 0 BOTTOMMARGIN TTYWIDTH (- TOP NEARBOTTOM)
                 'INPUT
                 'REPLACE)
          (TERM.ERASEBITS CHAT.STATE 0 (- TOP FONTHEIGHT (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE))
                 TTYWIDTH FONTHEIGHT])
```

## (TERM.SET.TAB

```
[LAMBDA (CHAT.STATE TERMINAL.X)                             (* ejs%: "26-Aug-85 12:28")
```

;;; Set a new tab stop for the terminal

```
    (LET [(TERM.TAB.STOPS (ffetch (CHAT.STATE TERM.TAB.STOPS) of (\DTEST CHAT.STATE 'CHAT.STATE]
         (freplace (CHAT.STATE TERM.TAB.STOPS) of CHAT.STATE with (COND
                                                                    ((NULL TERM.TAB.STOPS)
                                                                     (LIST TERMINAL.X))
                                                                    (T (SORT (CONS TERMINAL.X TERM.TAB.STOPS])
```

## (TERM.SETMARGINS

```
[LAMBDA (CHAT.STATE TOP BOTTOM)                             (* ejs%: "12-May-85 14:58")
```

;;; Function to set top and bottom margins, for terminals that implement a programmable scrolling region

```
    (LET ([FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (HOMEPOS (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE)))
         (freplace (CHAT.STATE TOPMARGIN) of CHAT.STATE with (- HOMEPOS (ITIMES (- TOP 2)
                                                                               FONTHEIGHT)))
         (freplace (CHAT.STATE BOTTOMMARGIN) of CHAT.STATE with (- HOMEPOS (ITIMES (SUB1 BOTTOM)
                                                                                   FONTHEIGHT)
                                                                    (ffetch (CHAT.STATE FONTDESCENT) of CHAT.STATE])
```

## (TERM.SMOOTHSCROLL

```
[LAMBDA (CHAT.STATE)                                        (* ejs%: "12-May-85 14:58")
```

;;; For those of you who can stand smooth scrolling, this will scroll in the normal direction, one (pixel) line at a time

```
    (LET ([DSP (ffetch (CHAT.STATE DSP) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (TTYWIDTH (ffetch (CHAT.STATE TTYWIDTH) of CHAT.STATE))
          (TTYHEIGHT (ffetch (CHAT.STATE TTYHEIGHT) of CHAT.STATE))
          (FONTHEIGHT (ffetch (CHAT.STATE FONTHEIGHT) of CHAT.STATE)))
         (for I from 1 to FONTHEIGHT do (BITBLT DSP 0 0 DSP 0 1 TTYWIDTH TTYHEIGHT 'INPUT 'REPLACE))
         (TERM.ERASEBITS CHAT.STATE 0 0 TTYWIDTH FONTHEIGHT])
```

## (TERM.TAB

```
[LAMBDA (CHAT.STATE)                                        (* edited%: "26-Mar-86 11:06")
```

;; Advance x to next tab stop on the line, if there is one.

```
(LET* ([XPOS (ffetch (CHAT.STATE XPOS) of (\DTEST CHAT.STATE 'CHAT.STATE]
       (FONTWIDTH (ffetch (CHAT.STATE FONTWIDTH) of CHAT.STATE))
       (CURSORX (ADD1 (IQUOTIENT XPOS FONTWIDTH)))
       NEXT.STOP)
      (COND
         ((SETQ NEXT.STOP (for CX in (ffetch (CHAT.STATE TERM.TAB.STOPS) of CHAT.STATE)
                             thereis (IGEQ CX CURSORX)))
          (MOVETO (freplace (CHAT.STATE XPOS) of CHAT.STATE with (ITIMES NEXT.STOP FONTWIDTH))
                  (ffetch (CHAT.STATE YPOS) of CHAT.STATE)
                  (ffetch (CHAT.STATE DSP) of CHAT.STATE])
```

## (TERM.UP

```
[LAMBDA (CHAT.STATE NLINES)                                 (* ejs%: "12-May-85 14:59")
```

;;; Go up NLINES (default = 1), pegging at top

```
    (LET ([YPOS (ffetch (CHAT.STATE YPOS) of (\DTEST CHAT.STATE 'CHAT.STATE]
          (HOMEPOS (ffetch (CHAT.STATE HOMEPOS) of CHAT.STATE)))
         (COND
```

```
            ((< YPOS HOMEPOS)
             (MOVETO (ffetch (CHAT.STATE XPOS) of CHAT.STATE)
                    [freplace (CHAT.STATE YPOS) of CHAT.STATE with (IMIN HOMEPOS (+ YPOS (ITIMES
                                                                                          (ffetch (CHAT.STATE
                                                                                                   FONTHEIGHT)
                                                                                             of CHAT.STATE)
                                                                                          (OR NLINES 1]
                     (ffetch (CHAT.STATE DSP) of CHAT.STATE])

)


(DEFGLOBALVAR CHAT.UNDERLINE.DESCENT 3
   "Number of pixels below baseline to draw character underline")


(DEFGLOBALVAR CHAT.UNDERLINE.METHOD 'UNDERLINE "How to handle terminal 'underline' or 'blink' attributes:
                                               one of UNDERLINE, INVERT, BOLD")

(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (SOURCE)
       CHATDECLS)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)

(PUTPROPS CHATTERMINAL COPYRIGHT ("VENUE & Xerox Corporation" 1984 1985 1986 1987 1988 1990))
```

## FUNCTION INDEX



## VARIABLE INDEX