File created: 8-Jul-92 17:20:45 **{DSK}<usr>local>lde>lispcore>sources>ATERM.;3**

changes to: (RECORDS LINEBUFFER)
(VARS ATERMCOMS)
(FNS \SETUP.DEFAULT.LINEBUF \LINEBUFFER.GETNEXTBUFFER)

previous date: 16-May-90 12:08:04 {DSK}<usr>local>lde>lispcore>sources>ATERM.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ ATERMCOMS
       (                                                          ; Line-buffering
        (FNS BKLINBUF CLEARBUF LINBUF PAGEFULLFN SETLINELENGTH SYSBUF TERMCHARWIDTH TERMINAL-INPUT
             TERMINAL-OUTPUT \CHDEL1 \CLOSELINE \DECPARENCOUNT \ECHOCHAR \FILLBUFFER \FILLBUFFER.WORDSEPRP
             \FILLBUFFER.BACKUP \GETCHAR \INCPARENCOUNT \RESETLINE \RESETTERMINAL \SAVELINEBUF \STOPSCROLL?)
        (COMS * BCPLDISPLAYCOMS)
        (COMS (FNS VIDEOCOLOR)
              (VARS (\VideoColor))
              (PROP ARGNAMES VIDEOCOLOR))
        [DECLARE%: DOCOPY DONTEVAL@LOAD (P (MOVD? 'NILL 'SETDISPLAYHEIGHT]
        (DECLARE%: DONTCOPY (MACROS \RAISECHAR \LINEBUFBOUT))
        (FNS \PEEKREFILL \READREFILL \RATOM/RSTRING-REFILL \READCREFILL)
        (FNS DRIBBLE DRIBBLEFILE)
        (FNS \SETUP.DEFAULT.LINEBUF \CREATELINEBUFFER \LINEBUF.READP \LINEBUF.EOFP \LINEBUF.PEEKBIN
             \LINEBUFFER.GETNEXTBUFFER \OPENLINEBUF)
        (COMS                                                     ; User entries to make up for fact that (EOFP T) = NIL now.
              (FNS LINEBUFFER-EOFP LINEBUFFER-SKIPSEPRS))
        (DECLARE%: DOCOPY DONTEVAL@LOAD (VARS (\#DISPLAYLINES 58)
                                              (\DISPLAYLINELENGTH 82)
                                              (\CURRENTDISPLAYLINE 0)
                                              (\STOPSCROLLMESSAGE "---MORE---"))
               (VARS (\SYSBUF NIL)
                     (\LINBUF NIL))
               (P (MOVD? '\OPENLINEBUF '\CREATE.TTYDISPLAYSTREAM))
               (VARS (\DEFAULTLINEBUF (\SETUP.DEFAULT.LINEBUF)))
               (P (\OPENLINEBUF)))
        (FNS \INTERMP \OUTTERMP)
        (EXPORT (DECLARE%: DONTCOPY (RECORDS LINEBUFFER)
                       (CONSTANTS * LINEBUFFERSTATES)
                       (MACROS \INTERMP \OUTTERMP)
                       (GLOBALVARS \DEFAULTLINEBUF)))
        (DECLARE%: DONTCOPY (CONSTANTS * FILLTYPES))
        (LOCALVARS . T)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA VIDEOCOLOR TERMINAL-OUTPUT
                                                                                    TERMINAL-INPUT)))
        (PROP FILETYPE ATERM)))
```

;; Line-buffering

```
(DEFINEQ

(BKLINBUF
  [LAMBDA (STR)                                                  (* bvm%: " 5-May-86 11:38")
    (COND
       ((STRINGP STR)
        (\RESETLINE)
        (for J C [SA _ (fetch READSA of (\DTEST *READTABLE* 'READTABLEP] from 1 while (SETQ C (NTHCHARCODE STR J))
           do (\OUTCHAR \LINEBUF.OFD C)
              (\INCPARENCOUNT (\SYNCODE SA C)))
        (\CLOSELINE)
        STR])

(CLEARBUF
  [LAMBDA (FILE FLG)                                             ; Edited 17-Jan-87 16:08 by bvm:
    [PROG ([STRM (SELECTQ FILE
                      (T \LINEBUF.OFD)
                      (NIL *STANDARD-INPUT*)
                      (\GETSTREAM FILE 'INPUT]
           SYSBUF LINBUF)                                        ; Do the stream coercion in line so we don't needlessly create a
                                                                 ; tty window just to clear input
          (COND
             ((AND (EQ STRM \LINEBUF.OFD)
                   (NEQ STRM \DEFAULTLINEBUF))                   ; Don't do this if \LINEBUF.OFD is the default, since then there
                                                                 ; really isn't anything to save
              (COND
                 [FLG (SETQ LINBUF (\SAVELINEBUF))
```

```
                    (SETQ SYSBUF (\SAVESYSBUF))
                    (COND
                       ((OR LINBUF SYSBUF)                    ; note in manual: if both buffers are empty, don't change saved
                                                              ; ones.
                        (SETQ \LINBUF LINBUF)
                        (SETQ \SYSBUF SYSBUF]
                  (T (\CLEARSYSBUF)))                         ; check for mouse events enabled and coordinated with
                                                              ; keyboard
             (\RESETTERMINAL]
    NIL])
```

(LINBUF

```
  [LAMBDA (FLG)                                               (* rrb "21-JUL-83 15:33")
    (COND
       (FLG (AND \LINBUF (CONCAT \LINBUF)))
       (T (SETQ \LINBUF NIL])
```

(PAGEFULLFN

```
  [LAMBDA (STREAM)                                            (* lmm "10-Jan-86 01:19")
```

;; default function that is called by \STOPSCROLL? when more lines are printed in a row than will fit on the screen or window.

;; If no input is pending, it waits for a character to be typed.

```
    (LET ((KEYSTREAM (fetch (LINEBUFFER KEYBOARDSTREAM) of \LINEBUF.OFD)))
         (COND
            ((READP KEYSTREAM))
            ((DISPLAYSTREAMP STREAM)                          ; reverse only this window.
             (RESETLST
                 (RESETSAVE (SETDISPLAYHEIGHT T))
                 [COND
                    ((AND (NOT (TTY.PROCESSP))
                          (EQ (PROCESSPROP (THIS.PROCESS)
                                           'NAME)
                              'MOUSE))                        ; Running under mouse, so can't make this proc be the tty
                                                              ; process
                     (RESETSAVE (TTY.PROCESS (THIS.PROCESS]
                 (RESETSAVE (INVERTW STREAM)
                        (LIST (FUNCTION INVERTW)
                              STREAM))
                 (BIN KEYSTREAM)))
            (T (PRIN1 \STOPSCROLLMESSAGE STREAM)
               (BIN KEYSTREAM)                                ; Now erase the message
               (FRPTQ (NCHARS \STOPSCROLLMESSAGE)
                      (\OUTCHAR STREAM ERASECHARCODE))
               (BLOCK])
```

(SETLINELENGTH

```
  [LAMBDA (N)                                                 (* rrb "22-JUL-83 10:10")
    (LINELENGTH (OR N (fetch (STREAM LINELENGTH) of \TERM.OFD))
           T])
```

(SYSBUF

```
  [LAMBDA (FLG)                                               (* rrb "21-JUL-83 15:34")
    (COND
       (FLG (AND \SYSBUF (CONCAT \SYSBUF)))
       (T (SETQ \SYSBUF NIL])
```

(TERMCHARWIDTH

```
  [LAMBDA (CHARCODE STREAM TTBL)                              (* JonL " 8-NOV-83 03:28")
```

;; Returns the width that the printed representation of CHARCODE would occupy if printed on the terminal STREAM, allowing for the various
;; escape sequences. Used by \ECHOCHAR

```
    (\STREAMCHARWIDTH (LOGAND CHARCODE \CHARMASK)
           (\OUTSTREAMARG STREAM)
           (GETTERMTABLE TTBL])
```

(TERMINAL-INPUT

```
  [LAMBDA U                                                   ; Edited 17-Jan-87 16:08 by bvm:
```

;;; Return the current terminal output stream. If an argument is supplied, make it the new terminal output stream

```
    (PROG1 \LINEBUF.OFD
        [COND
           ((IGEQ U 1)
            (LET [(STREAM (GETSTREAM (ARG U 1)
                                     'INPUT]
                 (if (EQ *STANDARD-INPUT* \LINEBUF.OFD)
                     then (SETQ *STANDARD-INPUT* STREAM))
                 (SETQ \LINEBUF.OFD STREAM])])
```

(TERMINAL-OUTPUT

```
[LAMBDA U                                                    ; Edited 17-Jan-87 16:08 by bvm:
```

;;; Return the current terminal output stream. If an argument is supplied, make it the new terminal output stream

```
    (PROG1 \TERM.OFD
        [COND
           ((IGEQ U 1)
            (LET [(STREAM (GETSTREAM (ARG U 1)
                                     'OUTPUT]
                  (if (EQ *STANDARD-OUTPUT* \TERM.OFD)
                      then (SETQ *STANDARD-OUTPUT* STREAM))
                  (SETQ TtyDisplayStream (SETQ \TERM.OFD STREAM])])
```

(\CHDEL1

```
[LAMBDA NIL                                                  (* rmk%: "28-Mar-85 18:25")
   (COND
      ((\BACKNSCHAR \LINEBUF.OFD (UNFOLD \NORUNCODE 256))
       (PROG1 (\NSPEEK \LINEBUF.OFD (UNFOLD \NORUNCODE 256))
           (\SETEOFPTR \LINEBUF.OFD (GETFILEPTR \LINEBUF.OFD)))])
```

(\CLOSELINE

```
[LAMBDA NIL                                                  (* lmm "10-Jan-86 03:07")
   (SETQ \CURRENTDISPLAYLINE 0)
   (UNINTERRUPTABLY
       (\SETFILEPTR \LINEBUF.OFD 0)
       (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with READING.LBS))])
```

(\DECPARENCOUNT

```
[LAMBDA (RSNX)                                               (* bvm%: "14-Feb-85 00:29")
```

;; This updates parencounts as characters are removed from the buffer due to line-editting. RSNX is a readtable syntax code

```
   (COND
      [(EQ RSNX STRINGDELIM.RC)
       (replace (LINEBUFFER INSTRINGP) of \LINEBUF.OFD with (NOT (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD]
      ((NOT (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD))
       (SELECTC RSNX
            (LEFTPAREN.RC (COND
                              ((EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                                   0)
                               (add (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                                    -1))))
            (RIGHTPAREN.RC
                 (COND
                    ((EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                         0)
                     (add (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                          1))))
            (LEFTBRACKET.RC
                 (add (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                      -1))
            (RIGHTBRACKET.RC
                 (add (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                      1))
            NIL])
```

(\ECHOCHAR

```
[LAMBDA (C)                                                  ; Edited 15-Jun-87 16:58 by jds
```

;; Echo the character-code C appropriately. If it really got echoed, return T, otherwise NIL.

```
   (COND
      ((fetch ECHOFLG of \PRIMTERMTABLE)
       [COND
          ((AND (EQ (fetch RAISEFLG of \PRIMTERMTABLE)
                    0)
                (IGEQ C (CHARCODE a))
                (ILEQ C (CHARCODE z)))                       ; This is doing a raise if flag is set
           (SETQ C (IDIFFERENCE C 32]
       (\OUTCHAR \TERM.OFD C)
       T])
```

(\FILLBUFFER

```
[LAMBDA (FILLTYPE)                                           ; Edited 20-Aug-87 17:52 by jds
```

;; While filling the line, the current file pointer is the end of the line. When the line is closed, this is made the eof. *READTABLE* is used for
;; syntactic delimiters and paren counting on READ and RATOM calls but isn't referenced (or bound) for READC

```
   (DECLARE (USEDFREE *READTABLE* *READ-NEWLINE-SUPPRESS*))
   (\RESETLINE)
   (PROG ((ILB (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD))
          (ISP (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD))
          (ILP (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD))
          (KEYSTREAM (fetch (LINEBUFFER KEYBOARDSTREAM) of \LINEBUF.OFD))
          (RTBLSA (fetch READSA of *READTABLE*))
```

```
        (CONTROLTON (fetch CONTROLFLG of \PRIMTERMTABLE))
        RSNX TCLASS CHAR RAISEDCHAR PEEKEDECHOED)
```

;; AR 8999/9000 the RTBLSA init code used to set it to nil if FILLTYPE were READC.FT; alas, RTBLSA is used even when that's true. --JDS
;; 8/20/87

```
    (DECLARE (SPECVARS RTBLSA))                          ; TCLASS is terminal syntax class, RSNX is read-table code
    [COND
       ((SETQ CHAR (fetch (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD))
                                                         ; Account for peeked character, adn remember if for further
                                                         ; down.
        (SETQ CHAR (IABS CHAR))                          ; The peeked char may be negative because it was BIN'ed
                                                         ; earlier.  Make sure it is positive.
        (COND
           ((NOT (fetch (LINEBUFFER PEEKEDECHOFLG) of \LINEBUF.OFD))
                                                         ; It wasn't echoed when first read, so echo it now if desired
                                                         ; Incompatible with I-10 to do it this way
            (\ECHOCHAR CHAR)))
        (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with NIL)
        (replace (LINEBUFFER PEEKEDECHOFLG) of \LINEBUF.OFD with NIL)
        (SETQ PEEKEDECHOED T)
        (SETQ RAISEDCHAR (\RAISECHAR CHAR]
    (COND
       ((AND CONTROLTON (EQ FILLTYPE READC.FT))
        (\LINEBUFBOUT \LINEBUF.OFD (OR CHAR (\GETCHAR)))
        (GO EXIT)))
    (COND
       (CHAR (GO NEXTTCLASS)))
NEXT
    (SETQ CHAR (BIN KEYSTREAM))
NEXTTCLASS
    [SETQ TCLASS (fetch TERMCLASS of (\SYNCODE \PRIMTERMSA (SETQ RAISEDCHAR (\RAISECHAR CHAR]
REDO
    (SELECTC TCLASS
        (RETYPE.TC (\OUTCHAR \TERM.OFD (CHARCODE EOL))
                   (\SETEOFPTR \LINEBUF.OFD (\GETFILEPTR \LINEBUF.OFD))
                   ;; Make the EOF be accurate during retyping, in case an interrupt happens and the buffer gets saved via
                   ;; \SAVELINEBUF.
                   (UNINTERRUPTABLY
                       (\SETFILEPTR \LINEBUF.OFD 0)
                       (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with RETYPING.LBS))
                   [until (\PAGEDEOFP \LINEBUF.OFD) do (\OUTCHAR \TERM.OFD (\NSIN \LINEBUF.OFD
                                                                              (UNFOLD \NORUNCODE 256]
                   (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with FILLING.LBS)
                   (GO NEXT))
        (CHARDELETE.TC
            (COND
               ((SETQ CHAR (\CHDEL1))
                (\FILLBUFFER.BACKUP CHAR)))
            (GO RECOMPUTE))
        (LINEDELETE.TC
            (while (SETQ CHAR (\CHDEL1)) do (\FILLBUFFER.BACKUP CHAR))
            (GO RECOMPUTE))
        (WORDDELETE.TC
            (COND
               ((SETQ CHAR (\CHDEL1))
                (while (\FILLBUFFER.WORDSEPRP CHAR) do     ; first chars are seprs, delete them all
                                                       (\FILLBUFFER.BACKUP CHAR)
                                                       (OR (SETQ CHAR (\CHDEL1))
                                                           (GO RECOMPUTE)))
                (\FILLBUFFER.BACKUP CHAR)
                (OR (SETQ CHAR (\CHDEL1))
                    (GO RECOMPUTE))
                (while (NULL (\FILLBUFFER.WORDSEPRP CHAR)) do (\FILLBUFFER.BACKUP CHAR)
                                                              (OR (SETQ CHAR (\CHDEL1))
                                                                  (GO RECOMPUTE)))
                                                       ; put CHAR back
                (\LINEBUFBOUT \LINEBUF.OFD CHAR)
                (GO RECOMPUTE)))
            (GO NEXT))
        (CTRLV.TC ;; The reasonable thing to do is coerce the character, set TCLASS to NONE.TC, and go REDO.  But on the 10, ctlv
                  ;; disables the immediacy of read-macros.  This is quite bizarre, cause a macro that was suppose to do something in
                  ;; the middle of reading will be done out of context.  We simulate that behavior, however.
                  (COND
                     (PEEKEDECHOED                     ; Has been echoed already, don't echo it again.
                            (SETQ PEEKEDECHOED NIL))
                     (T (\ECHOCHAR CHAR)))             ; Want to echo ^V
                  (\LINEBUFBOUT \LINEBUF.OFD (COND
                                                ([OR (AND (IGEQ (SETQ RAISEDCHAR (\GETCHAR))
                                                                (CHARCODE A))
                                                          (ILEQ RAISEDCHAR (CHARCODE Z)))
                                                     (AND (IGEQ RAISEDCHAR (CHARCODE a))
                                                          (ILEQ RAISEDCHAR (CHARCODE z]
                                                 (LOGAND RAISEDCHAR 31))
                                                (T RAISEDCHAR)))
                  (GO NEXT))
```

```
            (EOL.TC (COND
                        (PEEKEDECHOED                          ; Has been echoed already, don't echo it again.
                               (SETQ PEEKEDECHOED NIL))
                        (T (\ECHOCHAR CHAR)))
                    (\LINEBUFBOUT \LINEBUF.OFD RAISEDCHAR)
                    (GO EXIT))
            NIL)
         (COND
            (PEEKEDECHOED (SETQ PEEKEDECHOED NIL))
            (T (\ECHOCHAR CHAR)))                              ; Here if it isn't a terminal class.  Only echo if it isn't a special
                                                               ; terminal class
         (\LINEBUFBOUT \LINEBUF.OFD RAISEDCHAR)
         (AND (EQ FILLTYPE READC.FT)
              (GO NEXT))
         (COND
            ((EQ ESCAPE.RC (SETQ RSNX (\SYNCODE RTBLSA RAISEDCHAR)))
                                                               ; On Tenex the escape inhibits the action of all terminal
                                                               ; characters except control-V.
             (COND
                ([EQ CTRLV.TC (SETQ TCLASS (fetch TERMCLASS of (\SYNCODE \PRIMTERMSA (SETQ RAISEDCHAR (\GETCHAR]
                 (GO REDO)))
             (\LINEBUFBOUT \LINEBUF.OFD RAISEDCHAR)
             (GO NEXT)))
         (SELECTC FILLTYPE
             (RATOM/RSTRING.FT
                  (COND
                     ((AND CONTROLTON (fetch STOPATOM of RSNX))
                      (GO EXIT))))
             (READ.FT (COND
                         ([AND CONTROLTON (EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                                              0)
                               (EQ (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                                   0)
                               (fetch STOPATOM of RSNX)
                               (SELECTC RSNX
                                    ((LIST LEFTPAREN.RC LEFTBRACKET.RC)
                                        NIL)
                                    (STRINGDELIM.RC
                                         (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD))
                                    (NOT (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD]
                          ;; READ is reading an atom.  Return when atom ends, but also obey bracket/paren exception noted on page 14.33
                          ;; of manual.
                          (GO EXIT)))
                      (COND
                         ((\INCPARENCOUNT RSNX)
                          ;; Parens balance--throw the carriage if the closing paren or bracket character was not a CR, and if FLG argument
                          ;; of READ is NIL.  (We know we are under a READ call because of FILLTYPE)
                          (\CLOSELINE)                         ; \CLOSELINE first so dribble happens before EOL
                          [COND
                             ((AND (NEQ RAISEDCHAR (CHARCODE EOL))
                                   (NOT *READ-NEWLINE-SUPPRESS*))
                              (\OUTCHAR \TERM.OFD (CHARCODE EOL]
                          (RETURN))
                         ((EQ IMMEDIATE.RMW (fetch WAKEUP of RSNX))
                                                               ; Immediate read-macro
                          (GO EXIT))))
             (SHOULDNT))
         (GO NEXT)
      RECOMPUTE
         (AND (EQ FILLTYPE READ.FT)
              (PROGN (UNINTERRUPTABLY
                         (\SETFILEPTR \LINEBUF.OFD 0)
                         (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with RETYPING.LBS)
                         (replace (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD with ILB)
                         (replace (LINEBUFFER INSTRINGP) of \LINEBUF.OFD with ISP)
                         (replace (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD with ILP))
                     [until (\PAGEDEOFP \LINEBUF.OFD) do (SETQ CHAR (\NSIN \LINEBUF.OFD (UNFOLD \NORUNCODE 256))
                                                              )
                                                         (COND
                                                            [(EQ ESCAPE.RC (SETQ RSNX (\SYNCODE RTBLSA CHAR)))
                                                             (OR (\PAGEDEOFP \LINEBUF.OFD)
                                                                 (\NSIN \LINEBUF.OFD (UNFOLD \NORUNCODE 256]
                                                            (T (\INCPARENCOUNT RSNX]
                     (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with FILLING.LBS)))
         (GO NEXT)
      EXIT
         (\CLOSELINE])
```

(\FILLBUFFER.WORDSEPRP

```
  [LAMBDA (CHAR)                                               (* lmm "17-Jan-86 19:44")
    (OR (EQ WORDSEPR.TC (fetch TERMCLASS of (\SYNCODE \PRIMTERMSA CHAR)))
        (NEQ OTHER.RC (\SYNCODE RTBLSA CHAR])
```

(\FILLBUFFER.BACKUP

```
[LAMBDA (CHAR)                                          (* lmm "10-Jan-86 18:32")
  (DSPBACKUP (CHARWIDTH CHAR \TERM.OFD)
         \TERM.OFD])
```

(\GETCHAR

```
[LAMBDA NIL                                             (* lmm "30-Dec-85 17:25")
  (PROG [(C (BIN (fetch (LINEBUFFER KEYBOARDSTREAM) of \LINEBUF.OFD]
        (\ECHOCHAR C)                                   ; Echo here so raise-echo is correct
        (RETURN (\RAISECHAR C])
```

(\INCPARENCOUNT

```
[LAMBDA (RSNX)                                          (* bvm%: "14-Feb-85 00:30")
  ;; This maintains the paren count as characters are added to the buffer.  RSNX is a readtable syntax code.  Returns T when parens balance.
  (COND
     ((EQ RSNX STRINGDELIM.RC)
      (replace (LINEBUFFER INSTRINGP) of \LINEBUF.OFD with (NOT (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD)))
      NIL)
     ((NOT (fetch (LINEBUFFER INSTRINGP) of \LINEBUF.OFD))
      (SELECTC RSNX
           (LEFTPAREN.RC (AND (EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                                  0)
                              (add (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                                   1))
                         NIL)
           (RIGHTPAREN.RC                               ; NOTE: RP's never match left-brackets, just like on 10
                (AND (EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                         0)
                     (OR (EQ (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                             0)
                         (EQ (add (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                                  -1)
                             0))))
           (LEFTBRACKET.RC
                (add (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                     1)
                NIL)
           (RIGHTBRACKET.RC
                [COND
                   ((EQ (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                        0)
                    (replace (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD with 0))
                   (T (AND (EQ (add (fetch (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD)
                                    -1)
                               0)
                           (EQ (fetch (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD)
                               0])
           NIL])
```

(\RESETLINE

```
[LAMBDA NIL                                             (* jds "10-Apr-85 23:17")
  (UNINTERRUPTABLY
      (replace (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD with FILLING.LBS)
      (\SETFILEPTR \LINEBUF.OFD 0)
      (\SETEOFPTR \LINEBUF.OFD 0))
  (SETQ \CURRENTDISPLAYLINE 0])
```

(\RESETTERMINAL

```
[LAMBDA NIL                                             (* bvm%: "11-Jul-84 23:15")
  (DECLARE (GLOBALVARS \VideoColor))
  ;; Called by CLEARBUF and by RESET and ERROR! when returning to the TOPFRAME on the stack
  (replace (LINEBUFFER LPARCOUNT) of \LINEBUF.OFD with 0)
  (replace (LINEBUFFER LBRKCOUNT) of \LINEBUF.OFD with 0)
  (replace (LINEBUFFER INSTRINGP) of \LINEBUF.OFD with NIL)
  (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with NIL)
  (\RESETLINE)                                          ; Since we aren't immediately filling the buffer, guarantee that the
                                                        ; next read causes an EOF error
  (VIDEOCOLOR \VideoColor])
```

(\SAVELINEBUF

```
[LAMBDA NIL                                             ; Edited  9-Mar-88 11:41 by bvm
  ;; Don't have to set the fileptr to its original place cause we are heading for a \RESETTERMINAL in CLEARBUF
  (SELECTC (fetch (LINEBUFFER LINEBUFSTATE) of \LINEBUF.OFD)
       (FILLING.LBS (\CLOSELINE))
       (RETYPING.LBS                                    ; EOF is valid, but current fileptr isn't
                     (\SETFILEPTR \LINEBUF.OFD 0))
       NIL)
  (COND
     ((NOT (\PAGEDEOFP \LINEBUF.OFD))
```

```
        (LET* [(NBYTES (- (\GETEOFPTR \LINEBUF.OFD)
                          (\GETFILEPTR \LINEBUF.OFD)))
               (NC NBYTES)
               (STR (if (EQ (fetch (STREAM CHARSET) of \LINEBUF.OFD)
                            0)
                        then                                    ; Thin linebuffer
                            (ALLOCSTRING NC)
                      else                                      ; Fat linebuffer.  This should always be the case now
                           (ALLOCSTRING (SETQ NC (FOLDHI NBYTES 2))
                                  NIL NIL T]
              ;; Read chars into string.  Do it this way, rather than thru, say RSTRING, because we want to treat linebuf as an ordinary stream, not
              ;; a terminal stream;  (EOFP T) = NIL would defeat us.
              (\BINS \LINEBUF.OFD (fetch (STRINGP BASE) of STR)
                     0 NBYTES)
              (if (OR (> NC 1)
                      (NEQ (CHCON1 STR)
                           (CHARCODE CR)))
                  then                                          ; Only something to save if it's not a naked eol.
                       STR])
```

(\STOPSCROLL?

```
  [LAMBDA NIL                                                   (* lmm "11-Feb-86 09:56")
```

;; Called whenever a carriage-return is printed on the display. Keeps track of number of lines since last user input. If this one would scroll
;; information off the screen, it calls the users window specific function or the function PAGEFULLFN which waits for the user to type a character.

```
    (DECLARE (GLOBALVARS \STOPSCROLLMESSAGE))                   ; Set \#DISPLAYLINEs to NIL to disable
    (COND
       [(AND (NEQ \CURRENTDISPLAYLINE -1)
             (OR (EQ \#DISPLAYLINES 0)
                 (NOT (SMALLP \#DISPLAYLINES]
       ([OR (EQ \CURRENTDISPLAYLINE -1)
            (EQ \#DISPLAYLINES (SETQ \CURRENTDISPLAYLINE (ADD1 \CURRENTDISPLAYLINE]
        (SETQ \CURRENTDISPLAYLINE 0)
        (LET ([W (AND \WINDOWWORLD (WFROMDS (TTYDISPLAYSTREAM]
              WINDOWFN)
             (COND
                ([AND W (SETQ WINDOWFN (WINDOWPROP W 'PAGEFULLFN]
                 (APPLY* WINDOWFN (TTYDISPLAYSTREAM)))
                (T (PAGEFULLFN (TTYDISPLAYSTREAM])

)

(RPAQQ BCPLDISPLAYCOMS ((FNS \DSCCOUT \INITBCPLDISPLAY)
                        (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\INITBCPLDISPLAY)))
                        (EXPORT (GLOBALVARS \BCPLDISPLAY))))

(DEFINEQ
```

(\DSCCOUT

```
  [LAMBDA (STREAM CHARCODE)                                     (* lmm " 5-OCT-83 18:31")
```

;; The terminal outcharfn, prior for non-displaystream systems. STREAM is always \TERM.OFD, but passed as an argument so that calling
;; structure is the same as the more general display outcharfn, and thus, so that a simple MOVD can be done to install the display world.

```
    (SELECTC (fetch CCECHO of (\SYNCODE \PRIMTERMSA CHARCODE))
         (INDICATE.CCE [PROG ((CC CHARCODE))
                             (add (fetch CHARPOSITION of STREAM)
                                  (IPLUS (COND
                                             ((IGREATERP CC 127)  ; META character
                                              (DSPBOUT (CHARCODE %#))
                                              (SETQ CC (LOGAND CC 127))
                                              1)
                                             (T 0))
                                         (COND
                                             ((ILESSP CC 32)      ; CONTROL character
                                              (DSPBOUT (CHARCODE ^))
                                              (SETQ CC (LOGOR CC 64))
                                              1)
                                             (T 0))
                                         (PROGN (DSPBOUT CC)
                                                1])
         (SIMULATE.CCE (SELCHARQ CHARCODE
                                 (LF (DSPBOUT (CHARCODE EOL))
                                     (RPTQ (fetch CHARPOSITION of STREAM)
                                           (DSPBOUT (CHARCODE SPACE)))
                                     (\STOPSCROLL?))
                                 (EOL (DSPBOUT (CHARCODE EOL))
                                      (\STOPSCROLL?)
                                      (replace CHARPOSITION of STREAM with 0))
                                 (ESCAPE (DSPBOUT (CHARCODE $))     ; change to $
                                         (add (fetch CHARPOSITION of STREAM)
                                              1))
                                 (TAB (FRPTQ (IDIFFERENCE 8 (MOD (fetch CHARPOSITION of STREAM)
                                                                  8))
                                             (DSPBOUT (CHARCODE SPACE))
                                             (add (fetch CHARPOSITION of STREAM)
```

```
                                        1)))
                                (PROGN (DSPBOUT CHARCODE)
                                       (add (fetch CHARPOSITION of STREAM)
                                            1))))
          (REAL.CCE (DSPBOUT CHARCODE)
                    (COND
                       ((EQ CHARCODE (CHARCODE EOL))
                        (\STOPSCROLL?)
                        (replace CHARPOSITION of STREAM with 0))
                       (T (add (fetch CHARPOSITION of STREAM)
                               1))))
          (IGNORE.CCE)
          (SHOULDNT])
```

(\INITBCPLDISPLAY

```
  [LAMBDA NIL                                                    ; Edited 17-Jan-87 16:08 by bvm:
    (SETQ \BCPLDISPLAY (create STREAM
                               DEVICE _ (create FDEV
                                                BOUT _ (FUNCTION \DSCCOUT))
                               ACCESS _ 'OUTPUT
                               LINELENGTH _ 72
                               USERCLOSEABLE _ NIL
                               USERVISIBLE _ NIL
                               OUTCHARFN _ (FUNCTION \DSCCOUT)))
    (OR (STREAMP \TERM.OFD)
        (SETQ \TERM.OFD \BCPLDISPLAY))
    (OR (STREAMP *STANDARD-OUTPUT*)
        (SETQ *STANDARD-OUTPUT* \BCPLDISPLAY])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\INITBCPLDISPLAY)
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \BCPLDISPLAY)
)

;; END EXPORTED DEFINITIONS

(DEFINEQ
```

(\VIDEOCOLOR

```
  [LAMBDA NARGS                                                  ; Edited 29-Jul-88 15:30 by drc:
    (DECLARE (GLOBALVARS \VideoColor))
    ;; sets the interpretation of bits that are displayed on the screen so that 1 is black {NIL} or 1 is white {anything else}.
    (PROG1 \VideoColor
        [COND
           ((NEQ NARGS 0)
            (SETQ \VideoColor (AND (ARG NARGS 1)
                                   T))
            (SELECTC \MACHINETYPE
                 (\MAIKO (SETQ \VideoColor (SUBRCALL DSP-VIDEOCOLOR \VideoColor)))
                 (\DANDELION [replace DLDISPCONTROL of \IOPAGE with (COND
                                                                       (\VideoColor
                                                                    ; Inverse video
                                                                        (LOGOR 2048 (fetch DLDISPCONTROL
                                                                                       of \IOPAGE)))
                                                                       (T (LOGAND (LOGXOR 2048 MAX.SMALLP)
                                                                                  (fetch DLDISPCONTROL of \IOPAGE])
                 (\DAYBREAK (DOVE.XOR.CURSOR \DoveDisplay.XorCursor))
                 (SETSCREENCOLOR \VideoColor])])

)

(RPAQQ \VideoColor NIL)

(PUTPROPS VIDEOCOLOR ARGNAMES (BLACKFLG))

(DECLARE%: DOCOPY DONTEVAL@LOAD

(MOVD? 'NILL 'SETDISPLAYHEIGHT)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \RAISECHAR MACRO (OPENLAMBDA (C)
```

```
                                (COND
                                   ((AND (fetch RAISEFLG of \PRIMTERMTABLE)
                                         (IGEQ C (CHARCODE a))
                                         (ILEQ C (CHARCODE z)))
                                    (IDIFFERENCE C 32))
                                   (T C))))

(PUTPROPS \LINEBUFBOUT MACRO (OPENLAMBDA (STRM CHAR)
                                 (\BOUT STRM (\CHARSET CHAR))
                                 (\BOUT STRM (\CHAR8CODE CHAR))))
)
)

(DEFINEQ
```

(\PEEKREFILL

```
  [LAMBDA NIL                                                    ; Edited 15-Jun-87 16:26 by jds
    ;; Called from \ENDOFFILE via \RefillBufferFn when the linebuffer is empty
    (PROG (C)
          [COND
             ((SETQ C (fetch (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD))
              ;; Saved char, just return it.  Ideally we might want to pay attention to echo state, but Interlisp-10 doesn't, so be compatible
              ;; Following code is a major crock.  Problem is that the eof interface is at the BIN level, but terminal deals in characters, which take
              ;; two BINs.  The code here assumes that the main way we get called is via \NSPEEK macro, which does a \PEEKBIN to start, then a
              ;; BIN of that character to get it out of the way, then another \PEEKBIN followed by \BACKFILEPTR.  We're assuming the
              ;; \BACKFILEPTR on the buffer stream is a no-op and that it is always called after the second \PEEKBIN.  We keep track of whether
              ;; we're peeking at the left or right half of the character by negating it after the first half is consumed, then making it normal again after
              ;; the second one (so that a subsequent PEEKC would still return the character).  It is also possible to be called here from
              ;; SKIPSEPR[CODE]S, in which case there might be a real BIN to consume the char if it's a sepr.
              ;; Anyway, this code should be reworked someday using the READ-CHAR, PEEK-CHAR stream interface.
              (RETURN (SELECTQ (STKNAME '(\BIN \PEEKBIN))
                               (\BIN ;; He is doing a \BIN -- remember for later calls that we have passed over the left half of the character
                                     [COND
                                        ((IGREATERP C 0)             ; We're looking at the left half.  Return it and move to right half.
                                         (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with (IMINUS C))
                                         (\CHARSET (\RAISECHAR C)))
                                        (T                           ; We looked at the left half before.  Now return the right half and
                                                                     ; char is now totally consumed.
                                           (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with NIL)
                                           (\CHAR8CODE (\RAISECHAR (IMINUS C])
                               (COND
                                  ((IGREATERP C 0)                   ; We're still looking at the left half.  Return it.
                                   (\CHARSET (\RAISECHAR C)))
                                  (T                                 ; We looked at the left half before.  Now look at the right half.
                                     (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with (IMINUS C))
                                     (\CHAR8CODE (\RAISECHAR (IMINUS C]
      ;; Echo the character, and remember whether we echoed it or not:
            [replace (LINEBUFFER PEEKEDECHOFLG) of \LINEBUF.OFD with (\ECHOCHAR (SETQ C (BIN (fetch (LINEBUFFER
                                                                                                    KEYBOARDSTREAM)
                                                                                                of \LINEBUF.OFD]
                                                                     ; First time thru this: Get a key, and echo it.
            (\RESETLINE)                                             ; Clear the line buffer.
      ;; Save the peeked character OUTSIDE the line buffer, to avoid problems if the guy later types ^E before the character is really read.
            (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with C)
            (RETURN (\CHARSET (\RAISECHAR C])
```

(\READREFILL

```
  [LAMBDA NIL                                                    (* AJB "15-Jan-86 14:52")
    ;; Called from \ENDOFFILE via \RefillBufferFn when the linebuffer is empty
    (DECLARE (USEDFREE \LINEBUF.OFD))
    ;; If the LINEBUFFER has a REFILLBUFFERFN use it, otherwise call \FILLBUFFER as default
    (COND
       ((STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
        (APPLY* (STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
                READ.FT))
       (T (\FILLBUFFER READ.FT)))
    (CL:FUNCALL (STKNAME '(\BIN \PEEKBIN))
                \LINEBUF.OFD])
```

(\RATOM/RSTRING-REFILL

```
  [LAMBDA NIL                                                    (* AJB "15-Jan-86 14:53")
    ;; Called from \ENDOFFILE via \RefillBufferFn when the linebuffer is empty
    (DECLARE (USEDFREE \LINEBUF.OFD))
    ;; If the LINEBUFFER has a REFILLBUFFERFN use it, otherwise call \FILLBUFFER as default
    (COND
```

```
       ((STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
        (APPLY* (STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
               RATOM/RSTRING.FT))
       (T (\FILLBUFFER RATOM/RSTRING.FT)))
    (\BIN \LINEBUF.OFD])
```

(\READCREFILL

```
  [LAMBDA NIL                                                    (* AJB "15-Jan-86 14:53")
```

;; Called from \ENDOFFILE via \RefillBufferFn when the linebuffer is empty

```
    (DECLARE (USEDFREE \LINEBUF.OFD))
```

;; If the LINEBUFFER has a REFILLBUFFERFN use it, otherwise call \FILLBUFFER as default

```
    (COND
       ((STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
        (APPLY* (STREAMPROP \LINEBUF.OFD 'REFILLBUFFERFN)
               READC.FT))
       (T (\FILLBUFFER READC.FT)))
    (\BIN \LINEBUF.OFD])

)

(DEFINEQ
```

(DRIBBLE

```
  [LAMBDA (FILE APPENDFLG)                                       ; Edited 16-Jan-87 17:03 by hdj
```

;; Turn on/off dribbling for this process

;;

;; Dribbling is on if the special variable *dribble-output* is bound to a stream.

```
    (LET ((OLD-DRIBBLE-STREAM (DRIBBLEFILE))
          (NEW-DRIBBLE-STREAM NIL))
```

;;; Turn off dribbling.

```
         (if OLD-DRIBBLE-STREAM
             then
```

;; disable dribbling to old dribble stream

```
                  (SETQ *DRIBBLE-OUTPUT* NIL)
                  (replace (STREAM USERCLOSEABLE) of OLD-DRIBBLE-STREAM with T)
                  (replace (STREAM USERVISIBLE) of OLD-DRIBBLE-STREAM with T)
                  (CLOSEF OLD-DRIBBLE-STREAM))
```

;;; Turn on dribbling.

```
         (if (AND FILE (NEQ FILE T))
             then [SETQ NEW-DRIBBLE-STREAM (OPENSTREAM FILE (COND
                                                              (APPENDFLG 'APPEND)
                                                              (T 'OUTPUT]
                  (UNINTERRUPTABLY
                      (replace (STREAM USERCLOSEABLE) of NEW-DRIBBLE-STREAM with NIL)
                      (replace (STREAM USERVISIBLE) of NEW-DRIBBLE-STREAM with NIL)
```

;; Start dribbling to new-dribble-stream.

```
                      (SETQ *DRIBBLE-OUTPUT* NEW-DRIBBLE-STREAM)))
         (AND OLD-DRIBBLE-STREAM (fetch (STREAM FULLNAME) of OLD-DRIBBLE-STREAM])
```

(DRIBBLEFILE

```
  [LAMBDA NIL                                                    ; Edited 16-Jan-87 16:06 by hdj
```

;; return the stream that this process is dribbling to.

```
    *DRIBBLE-OUTPUT*])

)

(DEFINEQ
```

(\SETUP.DEFAULT.LINEBUF

```
  [LAMBDA NIL                                                    ; Edited 13-Apr-87 17:07 by bvm:
```

;; Line buffer initialization. First create the line buffer device.

```
    (LET [(DEV (\NODIRCOREFDEV 'LINEBUFFER]
         (replace (FDEV READP) of DEV with (FUNCTION \LINEBUF.READP))
                                                                   ; Readp has to look at both keyboard stream and buffer
         (replace (FDEV EOFP) of DEV with (FUNCTION NILL))         ; EOFP is always false from terminal.  May want this to be
                                                                   ; different for network terminals
         (replace (FDEV PEEKBIN) of DEV with (FUNCTION \LINEBUF.PEEKBIN))
                                                                   ; PEEKBIN method has implicit EOFP test, so have to supply
                                                                   ; that ourselves, too.
         (replace (FDEV GETNEXTBUFFER) of DEV with (FUNCTION \LINEBUFFER.GETNEXTBUFFER))
                                                                   ; JRB - LINEBUFFER needs its own GETNEXTBUFFER to avoid
                                                                   ; overloading the ENDOFSTREAMOP slot; Common Lisp uses it
                                                                   ; to implement EOF-ERROR-P, and LINEBUFFER used to use it
                                                                   ; for refilling its buffer.  The latter use is bogus in my humble
                                                                   ; opinion, BvM's impelmentation aside...
```

```
      )
```

;; create a line buffer device which creates a line buffer the first time one is needed.

```
(PROG [(STREAM (\OPENFILE '{LINEBUFFER} 'BOTH 'NEW]
      (replace FULLFILENAME of STREAM with T)
```

;; No-one cares about the true file-name after this, so we make it convenient for code that wants to give a name back to the user

```
      (replace LINEBUFSTATE of STREAM with READING.LBS)
      (replace USERCLOSEABLE of STREAM with NIL)
      (replace USERVISIBLE of STREAM with NIL)                    ; Other linebuffer fields default properly
      [replace ENDOFSTREAMOP of STREAM with (FUNCTION (LAMBDA (STREAM)
                                                                  ; create a TTY window and make it the tty stream.  This also sets
                                                                  ; up a line buffer.
                                                        (\CREATE.TTYDISPLAYSTREAM)
                                                        (STREAMOP 'ENDOFSTREAMOP \LINEBUF.OFD \LINEBUF.OFD]
      (RETURN STREAM])
```

(**CREATELINEBUFFER**

```
[LAMBDA (TERMINAL.STREAM)                                         ; Edited 13-Apr-87 22:57 by bvm:
```

;; Create a new stream that buffers the raw input from TERMINAL.STREAM (default is the keyboard).

```
(LET* ([STREAM (\OPENFILE '{LINEBUFFER} 'BOTH 'NEW '((CHARSET T]
       (DEV (fetch (STREAM DEVICE) of STREAM))
       EOFMETHOD)
      (replace LINEBUFSTATE of STREAM with READING.LBS)
      (replace (LINEBUFFER KEYBOARDSTREAM) of STREAM with (OR TERMINAL.STREAM \KEYBOARD.STREAM))
      (replace USERCLOSEABLE of STREAM with NIL)
      (replace USERVISIBLE of STREAM with NIL)                    ; Other linebuffer fields default properly
      [replace ENDOFSTREAMOP of STREAM with (FUNCTION (LAMBDA (STREAM)
                                                        (CL:FUNCALL \RefillBufferFn]
      (if (AND TERMINAL.STREAM (NEQ (SETQ EOFMETHOD (fetch (FDEV EOFP) of TERMINAL.STREAM))
                                    'NILL))
          then ;; Need to install an eof method for the buffered stream that looks at TERMINAL.STREAM when the buffer runs out.  This is
               ;; optimized away for the normal keyboard case, which never runs out.
               (replace (STREAM DEVICE) of STREAM with (SETQ DEV (NCREATE 'FDEV DEV)))
                                                                  ; Copy the basic linebuffer device
               (replace (FDEV EOFP) of DEV with EOFMETHOD))
      STREAM])
```

(**LINEBUF.READP**

```
[LAMBDA (STREAM FLG)                                              ; Edited 13-Apr-87 22:05 by bvm:
  (LET ((KEYSTREAM (fetch (LINEBUFFER KEYBOARDSTREAM) of STREAM)))
       (OR (AND KEYSTREAM (READP KEYSTREAM))
           (fetch (LINEBUFFER PEEKEDCHAR) of STREAM)
           (\PAGEDREADP STREAM FLG])
```

(**LINEBUF.EOFP**

```
[LAMBDA (STREAM)                                                  ; Edited 13-Apr-87 18:09 by bvm:
```

;; End of file for linebuffer: true if both the buffer and the source of characters are empty

```
(AND (\PAGEDEOFP STREAM)
     (\EOFP (fetch (LINEBUFFER KEYBOARDSTREAM) of STREAM])
```

(**LINEBUF.PEEKBIN**

```
[LAMBDA (STREAM NOERROR)                                          ; Edited 13-Apr-87 16:53 by bvm:
  (OR (\BUFFERED.PEEKBIN STREAM T)
      (CL:FUNCALL \RefillBufferFn STREAM])
```

(**LINEBUFFER.GETNEXTBUFFER**

```
[LAMBDA (STREAM WHATFOR NOERRORFLG)
```

;; Code stolen from \CORE.GETNEXTBUFFER and slightly modified. Old LINEBUFFER relied on refilling its buffer from the ENDOFSTREAMOP
;; function; this is bogus and was causing Common Lisp READ with EOF-ERROR-P off to fail, as it also uses the ENDOFSTREAMOP function

```
(PROG ((CPAGE# (fetch CPAGE of STREAM))
       (COFF (fetch COFFSET of STREAM))
       EPAGE# COREBUF)
      [COND
         ((NOT (OPENED STREAM))
          (LISPERROR "FILE NOT OPEN" (fetch (STREAM FULLNAME) of STREAM]
      (if (AND (ILESSP COFF (SELECTQ WHATFOR
                                 (READ (fetch CBUFSIZE of STREAM))
                                 BYTESPERPAGE))
               (fetch CBUFPTR of STREAM))
          then                                                    ; all OK, why were we called?
               (SHOULDNT)
               (RETURN T))
```

;; Buffer exhausted or empty, prepare new one

```
      (UNINTERRUPTABLY                                            ; Clean up current page
          (replace CBUFSIZE of STREAM with 0)
```

```
            (replace CBUFPTR of STREAM with NIL)
            (if (EQ COFF BYTESPERPAGE)
                then                                          ; Change to be first byte of next page instead of beyond last byte
                                                              ; of previous page
                     (replace COFFSET of STREAM with (SETQ COFF 0))
                     (replace CPAGE of STREAM with (add CPAGE# 1))))
        [COND
           ([AND (IGEQ CPAGE# (SETQ EPAGE# (fetch EPAGE of STREAM)))
                 (OR (NEQ CPAGE# EPAGE#)
                     (IGEQ COFF (fetch EOFFSET of STREAM]      ; Current file pointer is at or past end of file
            (SELECTQ WHATFOR
                 (READ
                       ;; This used to do (\EOF.ACTION STREAM)
                       (RETURN (AND (NULL NOERRORFLG)
                                    (CL:FUNCALL \RefillBufferFn))))
                 (WRITE (UNINTERRUPTABLY
                            (replace EPAGE of STREAM with (SETQ EPAGE# CPAGE#))
                            (replace EOFFSET of STREAM with COFF)))
                 (\ILLEGAL.ARG WHATFOR]
    ;; Now fill the buffer -- map in current page
        (SETQ COREBUF (\CORE.FINDPAGE STREAM CPAGE#))         ; This is interruptable
        (UNINTERRUPTABLY                                      ; But these two fields must be set uninterruptably for benefit of
                                                              ; ucode
            (replace CBUFSIZE of STREAM with (COND
                                               ((ILESSP CPAGE# EPAGE#)
                                                              ; Full page
                                                BYTESPERPAGE)
                                               ((EQ EPAGE# CPAGE#)
                                                              ; Last page
                                                (fetch EOFFSET of STREAM))
                                               (T             ; Beyond EOF so no data
                                                  0)))
            (replace CBUFPTR of STREAM with COREBUF))
        (COND
           (\INTERRUPTABLE (BLOCK)))                          ; Let someone else run.  Useful for those long writings of scratch
                                                              ; files.
        (RETURN T])
```

(\OPENLINEBUF

```
  [LAMBDA NIL                                                 ; Edited 17-Jan-87 16:08 by bvm:
    ;; Don't assume that \LINEBUF.OFD or \TERM.OFD have been initialized.  That way, they won't get smashed if ATERM is reloaded.
    (DECLARE (GLOBALVARS DisplayFDEV))
    (PROG (STREAM)                                            ; Output parameters
                                                              ; Input parameters
        [COND
           ((OR (NOT (type? STREAM \LINEBUF.OFD))
                (EQ \LINEBUF.OFD \DEFAULTLINEBUF))
            (SETQ \LINEBUF.OFD (\CREATELINEBUFFER))
            (OR (AND (type? STREAM *STANDARD-INPUT*)
                     (NEQ *STANDARD-INPUT* \DEFAULTLINEBUF))
                (SETQ *STANDARD-INPUT* \LINEBUF.OFD]
        (\RESETTERMINAL])

)

;; User entries to make up for fact that (EOFP T) = NIL now.

(DEFINEQ

(LINEBUFFER-EOFP
  [LAMBDA (STREAM)                                            ; Edited 13-Apr-87 17:12 by bvm:
    ;; Public interface to "old functionality" of (EOFP T) -- returns true if there is no buffered input waiting on stream.  If stream is not terminal input, is
    ;; same as EOFP.
    (LET [(S (\GETSTREAM STREAM 'INPUT]
         (if (EQ S \LINEBUF.OFD)
             then (\PAGEDEOFP S)
           else (\EOFP S])

(LINEBUFFER-SKIPSEPRS
  [LAMBDA (STREAM RDTBL)                                      ; Edited 13-Apr-87 22:05 by bvm:
    ;; SKIPSEPRS applied to the terminal input linebuffer.  If run out of buffer, return NIL.
    (LET ((S (\GETSTREAM STREAM 'INPUT))
          (*READTABLE* (\GTREADTABLE RDTBL))
          CH)
         (if (EQ S \LINEBUF.OFD)
             then [until (\PAGEDEOFP S) do (if (SYNTAXP (SETQ CH (PEEKCCODE S))
                                                        'SEPRCHAR)
                                               then (READCCODE S)
                                             else (RETURN (CHARACTER CH]
           else (SKIPSEPRS S *READTABLE*])
```

```
)

(DECLARE%: DOCOPY DONTEVAL@LOAD

(RPAQQ \#DISPLAYLINES 58)

(RPAQQ \DISPLAYLINELENGTH 82)

(RPAQQ \CURRENTDISPLAYLINE 0)

(RPAQ \STOPSCROLLMESSAGE "---MORE---")

(RPAQQ \SYSBUF NIL)

(RPAQQ \LINBUF NIL)

(MOVD? '\OPENLINEBUF '\CREATE.TTYDISPLAYSTREAM)

(RPAQ \DEFAULTLINEBUF (\SETUP.DEFAULT.LINEBUF))

(\OPENLINEBUF)
)

(DEFINEQ

(\INTERMP
  [LAMBDA (OFD)                                          (* rrb "21-JUL-83 16:33")
    (EQ OFD \LINEBUF.OFD])


(\OUTTERMP
  [LAMBDA (OFD)                                          (* rrb "21-JUL-83 07:23")
    (EQ OFD \TERM.OFD])

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[ACCESSFNS LINEBUFFER ((LPARCOUNT (fetch FW6 of DATUM)
                                  (replace FW6 of DATUM with NEWVALUE))
                       (LBRKCOUNT (fetch FW7 of DATUM)
                                  (replace FW7 of DATUM with NEWVALUE))
                       (LINEBUFSTATE (fetch F5 of DATUM)
                                  (replace F5 of DATUM with NEWVALUE))
                       (KEYBOARDSTREAM (fetch F2 of DATUM)
                                  (replace F2 of DATUM with NEWVALUE))
                       (PEEKEDCHAR (fetch F3 of DATUM)
                                  (replace F3 of DATUM with NEWVALUE))
                       (LBFLAGS (fetch FW9 of DATUM)
                                  (replace FW9 of DATUM with NEWVALUE)))
       (ACCESSFNS LINEBUFFER [(LBFLAGBASE (LOCF (fetch LBFLAGS of DATUM]
              (BLOCKRECORD LBFLAGBASE ((PEEKEDECHOFLG FLAG)
                                       (INSTRINGP FLAG]
)

(RPAQQ LINEBUFFERSTATES (FILLING.LBS READING.LBS RETYPING.LBS))

(DECLARE%: EVAL@COMPILE

(RPAQQ FILLING.LBS 0)

(RPAQQ READING.LBS 1)

(RPAQQ RETYPING.LBS 2)

(CONSTANTS FILLING.LBS READING.LBS RETYPING.LBS)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \INTERMP MACRO ((OFD)
                          (EQ OFD \LINEBUF.OFD)))

(PUTPROPS \OUTTERMP MACRO ((OFD)
                           (EQ OFD \TERM.OFD)))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \DEFAULTLINEBUF)
)
)
```

```
;; END EXPORTED DEFINITIONS

(DECLARE%: DONTCOPY

(RPAQQ FILLTYPES ((READ.FT 0)
                  (RATOM/RSTRING.FT 1)
                  (READC.FT 2)))

(DECLARE%: EVAL@COMPILE

(RPAQQ READ.FT 0)

(RPAQQ RATOM/RSTRING.FT 1)

(RPAQQ READC.FT 2)

(CONSTANTS (READ.FT 0)
       (RATOM/RSTRING.FT 1)
       (READC.FT 2))
)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA VIDEOCOLOR TERMINAL-OUTPUT TERMINAL-INPUT)
)

(PUTPROPS ATERM FILETYPE BCOMPL)

(PUTPROPS ATERM COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1992))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

MACRO INDEX

PROPERTY INDEX

RECORD INDEX