File created: 18-Oct-93 16:53:42 **{Pele:mv:envos}<LispCore>Sources>CLTL2>TTYIN.;2**

previous date: 3-Sep-91 18:20:56 {Pele:mv:envos}<LispCore>Sources>CLTL2>TTYIN.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1990, 1991, 1993 by Venue & Xerox Corporation.  All rights reserved.

(RPAQQ TTYINCOMS
       [(COMS                                                    ; Main TTYIN editor
              (FNS TTYIN TTYIN.SETUP TTYIN.CLEANUP TTYIN1 TTYIN1RESTART TTYIN.FINISH TTYIN.BALANCE ADDCHAR
                   TTMAKECOMPLEXCHAR ADDNAKEDCHAR TTADDTAB ADJUSTLINE ADJUSTLINE.AND.RESTORE AT.END.OF.SCREEN
                   AT.END.OF.TEXT AUTOCR? BACKSKREAD BACKWARD.DELETE.TO BREAKLINE BUFTAILP CHECK.MARGIN
                   CLEAR.LINE? CURRENT.WORD DELETE.TO.END DELETELINE DELETETO DELETETO1 DO.EDIT.COMMAND
                   DO.EDIT.PP TTDOTABS EDITCOLUMN EDITNUMBERP END.DELETE.MODE ENDREAD? FIND.LINE FIND.LINE.BREAK
                   FIND.MATCHING.QUOTE FIND.NEXT.WORD FIND.NON.SPACE FIND.START.OF.WORD FORWARD.DELETE.TO
                   GO.TO.ADDRESSING GO.TO.FREELINE GO.TO.RELATIVE INIT.CURSOR INSERT.NODE INSERTLINE KILL.LINES
                   KILLSEGMENT L-CASECODE MOVE.BACK.TO MOVE.FORWARD.TO MOVE.TO.LINE MOVE.TO.NEXT.LINE
                   MOVE.TO.START.OF.WORD MOVE.TO.WHEREVER NTH.COLUMN.OF NTH.RELATIVE.COLUMN.OF OVERFLOW?
                   OVERFLOWLINE? PREVLINE PREVWORD PROPERTAILP READFROMBUF RENUMBER.LINES RESTORE.CURSOR
                   RESTOREBUF RETYPE.BUFFER SAVE.CURSOR SCANBACK SCANFORWARD SCRATCHCONS SEGMENT.LENGTH
                   SEGMENT.BIT.LENGTH SETLASTC SETTAIL? SHOW.MATCHING.PAREN SKIP/ZAP START.NEW.LINE
                   START.OF.PARAGRAPH? TTADJUSTWORD TTBIN TTBITWIDTH TTCRLF TTCRLF.ACCOUNT TTDELETECHAR
                   TTDELETELINE TTDELETEWORD TTECHO.TO.FILE TTGIVEHELP TTGIVEHELP1 TTGIVEHELP2 TTLASTLINE
                   TTLOADBUF TTNEXTLINE TTNEXTNODE TTNLEFT TTNTH TTNTHLINE TTPRIN1 TTPRINSPACE TTPRIN1COMMENT
                   TTPRIN2 TTPROMPTCHAR TTRUBOUT TTUNREADBUF TTWAITFORINPUT TTYINSTRING TYPE.BUFFER U-CASECODE
                   U/L-CASE))
        (COMS                                                    ; Internal reading.  These functions all expect caller to have
                                                                 ; bound *READTABLE* correctly (not bound in TTYIN for
                                                                 ; who-line transparency)
              (FNS TTRATOM TTREADLIST TTSKIPSEPR TTSKREAD TTYIN.READ))
        (COMS                                                    ; Escape completion and friends
              (FNS FIND.MATCHING.WORD TTCOMPLETEWORD WORD.MATCHES.BUFFER TTYIN.SHOW.?ALTERNATIVES))
        (COMS                                                    ; ? and ?= handler
              (FNS DO?CMD TTYIN.PRINTARGS TTYIN.READ?=ARGS DO?CMD.ERRORHANDLER))
        (COMS                                                    ; Display handling
              (FNS BEEP BITBLT.DELETE BITBLT.ERASE BITBLT.INSERT DO.CRLF DO.DELETE.LINES DO.INSERT.LINE DO.LF
                   ERASE.TO.END.OF.LINE ERASE.TO.END.OF.PAGE INSERT.TEXT TTDELSECTION TTADJUSTWIDTH
                   TTINSERTSECTION TTSETCURSOR))
        [COMS                                                    ; TTYINBUFFERSTREAM
              (FNS TTYINBUFFERDEVICE TTYINBUFFERSTREAM TTYINBUFFERBIN TTYINBUFFERPEEK TTYINBUFFERREADP
                   TTYINBUFFEREOFP TTYINBUFFERBACKPTR TTYINWORDRDTBL)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (TTYINBUFFERDEVICE (TTYINBUFFERDEVICE))
                                                    (TTYINWORDRDTBL (TTYINWORDRDTBL]
        (COMS                                                    ; Mouse handling
              (FNS DO.MOUSE DO.SHIFTED.SELECTION COPY.SEGMENT DELETE.LONG.SEGMENT DELETE.LONG.SEGMENT1
                   INVERT.LONG.SEGMENT INVERT.SEGMENT BRACKET.CURRENT.WORD TTBEFOREPOS TTNEXTPOS TTRACKMOUSE))
        (COMS ;; Auxiliary fns.  These are outside the TTYIN block, and are provided to aid the outside world in special interfaces to TTYIN
              (FNS SETREADFN TTYINENTRYFN TTYINREADP TTYINREAD TTYINFIX CHARMACRO? TTYINMETA TTYIN.LASTINPUT)
              (FNS TTYINEDIT SIMPLETEXTEDIT SET.TTYINEDIT.WINDOW TTYIN.PPTOFILE)
              (COMS                                              ; New, correct way of getting scratch file
                    (FNS MAKE-TTSCRATCHFILE)
                    (RESOURCES TTSCRATCHFILE))
              (COMS                                              ; Obsolete, but maybe someone calls it
                    (FNS TTYIN.SCRATCHFILE \TTYIN.RPEOF)
                    (INITVARS (TTYINEDIT.SCRATCH)))
              (INITVARS (TTYINEDITWINDOW)
                     (TTYINEDITPROMPT T)
                     (TTYINAUTOCLOSEFLG)
                     (TTYINPRINTFN)
                     (TTYIN?=FN)))
        [COMS                                                    ; Kludge of the week
              (FNS TTYINPROMPTFORWORD)
              (INITVARS (TTYIN.USE.EXACT.CHARS))
              (DECLARE%: DONTEVAL@LOAD DOCOPY                    ; This is so that you can (MOVD 'TTYINPROMPTFORWORD
                                                                 ; 'PROMPTFORWORD) and not die
                     (P (MOVD? 'PROMPTFORWORD 'NON-TTYIN-PROMPTFORWORD NIL T]
        (DECLARE%: DOEVAL@COMPILE DONTCOPY (COMS * TTCOMPILETIME))
        (INITVARS (DORADO.RESTORE.BUF.CODES '(194))
               (TTYIN.RESTORE.BUF.CODES '(516 530))
               (TTYINBUFFER)
               (?ACTIVATEFLG T)
               (EDITPREFIXCHAR)
               (SHOWPARENFLG T)
               (TTYINBSFLG T)
               (\TTYIN.LAST.FONT)
               (\TTYIN.LAST.COMMENTFONT)
               (TTYINFILLDEFAULT T)
               (TTYINCOMPLETEFLG T)
               (TTYINUSERFN)
```

```
               (TYPEAHEADFLG T)
               (null "")
               (DEFAULTPROMPT "** ")
               (TTYJUSTLENGTH -1)
               (\INSIDE.TTYIN)
               (TTYINERRORSETFLG)
               (TTYINRAISEFLG T)
               (TTYINAUTOFILLMARGIN 8)
               (TTYINFIXLIMIT 50)
               (TTYINDEBUGFLG)
               (HISTSTR1 "from file:")
               (TTYINCOMMENTCHAR)
               (\RESTOREBUFCODES))
         (P (MOVD? 'NILL 'GUESTUSER?)
            (MOVD? 'FIXSPELL 'FIXSPELL!!)
            (MOVD? 'HELPSYS 'XHELPSYS)
            [PUTDQ? SPRINTT (LAMBDA (X)
                                    (PRIN1 X]
            (MOVD? 'NILL 'WINDOWWORLD)
            (MOVD? 'LISPXFIX 'NONTTYINLISPXFIX))
         (ADDVARS (TTYINREADMACROS)
               (TTYINRESPONSES)
               (LISPXCOMS (STOP . OK))
               (\SYSTEMCACHEVARS \RESTOREBUFCODES))
         (PROP VARTYPE TTYINREADMACROS)
         [DECLARE%: DONTEVAL@LOAD DOCOPY (P (COND ((CCODEP 'TTYIN)
                                                   (CHANGENAME 'PROMPTCHAR 'LISPXREADP 'TTYINREADP)
                                                   (SETREADFN)
                                                   (MOVD 'TTYINFIX 'LISPXFIX]
         (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                                (NLAML CHARMACRO?)
                                                                                (LAMA])
```

;; Main TTYIN editor

```
(DEFINEQ
```

(**TTYIN**

```
  [LAMBDA (PROMPT SPLST HELP OPTIONS ECHOTOFILE TABS UNREADBUF RDTBL)
                                                                    ; Edited 24-May-91 10:39 by jds
    (DECLARE (SPECVARS SPLST HELP OPTIONS ECHOTOFILE TABS UNREADBUF RDTBL))
```

;;; TTYIN is a general input function. See TTYIN.DOC for details on the arguments and use of this fn. TTYIN was designed and implemented by Bill van Melle in the late 1970's at Stanford.

;;; The bulk of the code here was oriented toward smart use on display terminals from remote machines, and then hacked to get it to work in windows in Interlisp-D. As such, there are a large number of fairly obsolete crocks in the code.

;;; The most important terminal at the time was the datamedia. If on a dm, TTYIN puts the terminal in binary mode so it can read the 200q bit supplied by the EDIT key. Most of the cursor-moving commands from TVEDIT are available or slightly modified, and a few extra are supplied as well.

;;; The text being typed in is represented as a list of character codes, with a data structure on top of it which partitions it by line. Thus, you can view the text as one string, or broken into lines, depending on the function desired. \BUFFER is the pointer to the start of the buffer, \ENDBUFFER points one past the end. TTYIN saves up cons cells between calls and reuses them; \ENDBUFFER points to this list of free cells. TTYINBUFFER is the master record, which keeps assorted global information about where the cursor is, and saves some state info from one call to the next, enabling the restore previous buffer command. One of the fields points to the LINE records which describe the two-dimensional structure of the input. Each record points to the region of the buffer containing the text for one line, and has fields indicating the first and last columns, and a pointer to the next line record. \ARROW always points to the current LINE record --- \CURSOR points to where in the buffer the cursor appears. --- This representation is not terribly space-efficient for large buffers, but it is easily manipulated, and fast. If there is a particularly long input, there will be many cons cells tied up in TTYINBUFFER, so a good thing to do when trying to free up space is reset TTYINBUFFER to NIL to force its regeneration from scratch.

```
    (RESETLST
        (PROG ((\INSIDE.TTYIN T)
               (\AUTOFILL TTYINFILLDEFAULT)
               (\DSP (TTYDISPLAYSTREAM))
               (\FIRSTTIME T)
               (\INITCRLFS 0)
               (\RAISEINPUT (OR TTYINRAISEFLG (fetch RAISEFLG of \PRIMTERMTABLE)))
               (\TTYINSTATE TTYINBUFFER)
               (TYPEAHEAD TYPEAHEADFLG)
               \ARROW \BMARG \BUFFER \CHARHEIGHT \CHARWIDTH \COMMAND \COMMENTFONT \CURSOR \CURSORCOL \CURSORROW
               \DELETING \DESCENT \DONTCOMPLETE \ENDBUFFER \FILLINGBUFFER \FIRSTLINE \FIX \FONT \HOMECOL
               \HOMEROW \INITPOS \LAST.DELETION \LASTAIL \LASTAILCOL \LASTAILROW \LASTCHAR \LISPXREADING \LMARG
               \LOC.ROW.0 \NOFIXSPELL \NOVALUE \PFW.FIRSTTIME \PROMPT1 \PROMPT2 \PROMPTFORWORD \RDTBLSA \READING
               \REPEAT \RMARG \STRINGVALUE \TEXTURE \TTPAGELENGTH \TTYINBUFFERSTREAM VALUE)
              (SETQ TTYINBUFFER)                                    ; Global resource. Any ttyin calls while we are running need to
                                                                    ; create their own
              [OR (LISTP \TTYINSTATE)
                  (SETQ \TTYINSTATE (create TTYINBUFFER
                                            FIRSTLINE _ (create LINE
                                                                START _ (CONS 0)
                                                                ROW _ 0]
              [COND
                 ((AND SPLST (NLISTP SPLST))
                  (SETQ SPLST (CONS SPLST]
              (for OP inside OPTIONS do (SELECTQ OP
                                            ((NOFIXSPELL MUSTAPPROVE CRCOMPLETE)
```

```
                                        (SETQ \NOFIXSPELL (SETQ \DONTCOMPLETE OP)))
                                    (\NOVALUE (SETQ \NOVALUE OP))
                                    (STRING (SETQ \STRINGVALUE OP))
                                    (COMMAND (SETQ \COMMAND OP))
                                    (REPEAT (SETQ \REPEAT OP))
                                    (NORAISE (SETQ \RAISEINPUT))
                                    (RAISE (SETQ \RAISEINPUT T))
                                    (TEXT (SETQ \REPEAT (SETQ \NOVALUE (SETQ \AUTOFILL OP)))
                                          (SETQ \RAISEINPUT))
                                    (FIX (SETQ \FIX OP))
                                    (READ (SETQ \READING (SETQ \AUTOFILL OP)))
                                    (LISPXREAD [SETQ TYPEAHEAD (SETQ \LISPXREADING (SETQ \READING
                                                                                    (SETQ \AUTOFILL OP]
                                               (SETQ \RAISEINPUT (fetch RAISEFLG of \PRIMTERMTABLE)))
                                    (EVALQT                 ; like LISPXREAD, but with added proviso about checking for
                                                            ; EVALQT right-bracket hacks
                                            [SETQ TYPEAHEAD (SETQ \LISPXREADING (SETQ \READING
                                                                                 (SETQ \AUTOFILL OP]
                                            (SETQ \RAISEINPUT (fetch RAISEFLG of \PRIMTERMTABLE)))
                                    (TYPEAHEAD (SETQ TYPEAHEAD OP))
                                    (FILLBUFFER (SETQ \FILLINGBUFFER OP))
                                    (NOPROMPT (SETQ \FIRSTTIME OP))
                                    ((PROMPTFORWORD PROMPTFORWORD-SPACE)
                                                            ; For faking PROMPTFORWORD
                                        (SETQ \PROMPTFORWORD (SETQ \STRINGVALUE OP))
                                        (SETQ \PFW.FIRSTTIME UNREADBUF)
                                                            ; Flag that says to erase the line if user types something other
                                                            ; than <bs>, etc.
                                        (SETQ \RAISEINPUT NIL))
                                    NIL))
        [SETQ \RDTBLSA (fetch READSA of (SETQ RDTBL (COND
                                                      ((AND (NOT \READING)
                                                            (NULL RDTBL))
                                                            ; Use the word table, rather than a Lispish table
                                                       (\DTEST TTYINWORDRDTBL 'READTABLEP))
                                                      (T (\GTREADTABLE RDTBL]
        (TTYIN.SETUP)                                       ; Setup window, including fonts.  Didn't do this til now since it
                                                            ; uses \READING.
        [COND
           ((EQ PROMPT T)
            (SETQ \PROMPT1 (SETQ \PROMPT2)))
           (T [COND
                 ((NOT PROMPT)
                  (SETQ PROMPT DEFAULTPROMPT))
                 [(LISTP PROMPT)
                  (COND
                     ((NLISTP (CDR PROMPT))                 ; User has already supplied us with a dotted pair of prompts
                      (SETQ \PROMPT1 (CAR PROMPT))
                      (SETQ \PROMPT2 (CDR PROMPT)))
                     (T (SETQ PROMPT (SUBSTRING PROMPT 2 -2]
                 ((AND (NOT (STRINGP PROMPT))
                       (NOT (LITATOM PROMPT)))
                  (SETQ PROMPT (MKSTRING PROMPT]
              (COND
                 ((NLISTP PROMPT)                           ; Now create 2 prompts out of one
                  (SETQ \PROMPT1 PROMPT)
                  (SETQ \PROMPT2 (COND
                                    ((OR \LISPXREADING \PROMPTFORWORD)
                                                            ; Don't use a secondary prompt for LISPX or
                                                            ; PROMPTFORWORD
                                     NIL)
                                    ((AND \REPEAT (< (NCHARS PROMPT)
                                                     12))   ; Okay to use this short prompt as a secondary prompt
                                     PROMPT)
                                    (T '|...|]
        (COND
           ((NOT SPLST)
            (SETQ \DONTCOMPLETE T)))
        (COND
           (\READING (SETQ \REPEAT)))
        (COND
           ((NOT TYPEAHEAD)
            (CLEARBUF T)))
    LP  (SETQ VALUE (NLSETQ (TTYIN1)))
        (COND
           ((NOT VALUE)                                     ; NLSETQ aborted.  Try again.
            (COND
               ((OR (NOT TTYINERRORSETFLG)
                    \LISPXREADING)                          ; LISPXREAD is not errorset-protected, so why should this be?
                (COND
                   (\CURSORCOL                              ; If this is NIL, then we haven't initialized enough to go anywhere
                           (GO.TO.FREELINE)))
                (RESTOREMOD)
                (COND
                   ((NEQ \BUFFER \ENDBUFFER)
                    (replace (TTYINBUFFER OLDTAIL) of \TTYINSTATE with \ENDBUFFER)))
                (ERROR!)))
```

```
                (GO LP)))
            (COND
               ((AND (NEQ \BUFFER \ENDBUFFER)
                     (> (add (fetch (TTYINBUFFER STORAGECOUNTER) of \TTYINSTATE)
                             1)
                        10))                                    ; Release some storage, since it seems to accumulate and
                                                                ; fragment
                (replace (TTYINBUFFER STORAGECOUNTER) of \TTYINSTATE with 0)
                (FRPLACD \ENDBUFFER)))
            (SETQ VALUE (CAR VALUE))
            (POSITION T 0)
            [COND
               ((AND CTRLUFLG (NEQ VALUE T))                    ; user typed ^U to edit input
                (SETQ CTRLUFLG)
                (PROG ((\INSIDE.TTYIN))
                      (COND
                         ((OR (LITATOM VALUE)
                              (GUESTUSER?))                     ; guests may not edit

                          )
                         ((LISTP VALUE)
                          (EDITE VALUE))
                         (T (SETQ VALUE (CAR (EDITE (LIST VALUE)
                                                    '(REPACK]
            (COND
               ((AND TTYINMAILFLG (NEQ \READING 'EVALQT))       ; Note time of last user input
                (MWNOTE)))
            (RETURN VALUE)))])
```

(TTYIN.SETUP

```
  [LAMBDA NIL                                                   ; Edited 19-Jan-88 01:51 by bvm
                                                                ; Disable buttons so we can do selection
    [LET ((WINDOW (WFROMDS \DSP T)))
         (COND
            (WINDOW (replace (TTYINBUFFER TTOLDRIGHTFN) of \TTYINSTATE with (WINDOWPROP WINDOW 'RIGHTBUTTONFN
                                                                                        'TOTOPW))
                    (replace (TTYINBUFFER TTOLDBUTTONFN) of \TTYINSTATE with (WINDOWPROP WINDOW 'BUTTONEVENTFN
                                                                                        'TOTOPW))
                    (replace (TTYINBUFFER TTOLDENTRYFN) of \TTYINSTATE with (WINDOWPROP WINDOW 'WINDOWENTRYFN
                                                                                       'TTYINENTRYFN))
                    (replace (TTYINBUFFER TTYINWINDOW) of \TTYINSTATE with WINDOW)
                    (WINDOWPROP WINDOW 'TTYINSTATE (fetch (TTYINBUFFER TTYINWINDOWSTATE) of \TTYINSTATE))
                    (RESETSAVE NIL (LIST (FUNCTION TTYIN.CLEANUP)
                                         \TTYINSTATE]
    (COND
       ((OR (IMAGESTREAMTYPEP \DSP 'TEXT)
            (FMEMB (DSPDESTINATION NIL \DSP)
                   \SCREENBITMAPS))
        (SETQ \CHARWIDTH (CHARWIDTH (CHARCODE A)
                                    \DSP))
        (SETQ \FONT (DSPFONT NIL \DSP))
        (if (EQ \FONT \TTYIN.LAST.FONT)
            then (SETQ \COMMENTFONT \TTYIN.LAST.COMMENTFONT)
          elseif \READING
            then                                                ; Want a "comment" font for ?=
                 [SETQ \COMMENTFONT (SETQ \TTYIN.LAST.COMMENTFONT (FONTCOPY \FONT 'WEIGHT
                                                                           (SELECTQ (FONTPROP \FONT 'WEIGHT)
                                                                                (BOLD 'MEDIUM)
                                                                                'BOLD]
                 (SETQ \TTYIN.LAST.FONT \FONT)
          else (SETQ \COMMENTFONT \FONT))
        (SETQ \CHARHEIGHT (MAX (FONTHEIGHT \FONT)
                               (FONTHEIGHT \COMMENTFONT)))
        (SETQ \DESCENT (FONTPROP \FONT 'DESCENT))               ; How many pixels below the baseline this font goes
        (SETQ \TEXTURE (DSPTEXTURE NIL \DSP))
        (SETQ \TTPAGELENGTH (PAGEHEIGHT NIL \DSP))
        (SETQ \LMARG (DSPLEFTMARGIN NIL \DSP))                  ; bit pos of left margin
        (SETQ \RMARG (DSPRIGHTMARGIN NIL \DSP))                 ; bit pos of right margin, dsp relative
        (SETQ \INITPOS (DSPXPOSITION NIL \DSP])
```

(TTYIN.CLEANUP

```
  [LAMBDA (\TTYINSTATE)                                         ; Edited 24-May-91 10:39 by jds
    (PROG ((WINDOW (fetch (TTYINBUFFER TTYINWINDOW) of \TTYINSTATE)))
          (COND
             (WINDOW (WINDOWPROP WINDOW 'RIGHTBUTTONFN (fetch (TTYINBUFFER TTOLDRIGHTFN) of \TTYINSTATE))
                     (WINDOWPROP WINDOW 'BUTTONEVENTFN (fetch (TTYINBUFFER TTOLDBUTTONFN) of \TTYINSTATE))
                     (WINDOWPROP WINDOW 'WINDOWENTRYFN (fetch (TTYINBUFFER TTOLDENTRYFN) of \TTYINSTATE))
                     (WINDOWPROP WINDOW 'TTYINSTATE NIL)))
          (SETQ TTYINBUFFER \TTYINSTATE])
```

(TTYIN1

```
  [LAMBDA NIL                                                   ; Edited 24-May-91 10:33 by jds
```

;;; The main moby subfn of TTYIN. Is errorset protected in caller

```
(PROG ((DRIBFL (DRIBBLEFILE))
       CHAR MATCHED RESULT STARTOFWORD X TMP WASEDITCHAR SNX)
      (COND
         ((SETQ CHAR (fetch (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD))
                                                          ; Handle peeked char
          [COND
             ((AND (OR (NULL \PROMPT1)
                       (EQ \FIRSTTIME 'NOPROMPT))
                   (OR T (fetch (LINEBUFFER PEEKEDECHOFLG) of \LINEBUF.OFD))
                   (>= CHAR (CHARCODE SPACE)))
              ;; Want to avoid echoing peeked char twice.  Only feasible to do so if we were called with no prompt, implying that there is some
              ;; hope that the preceding char on the line is the peeked char
              (SETQ X (FCHARWIDTH CHAR \FONT))
              (DSPBACKUP X \DSP)
              (SETQ \INITPOS (- \INITPOS X]
          (replace (LINEBUFFER PEEKEDCHAR) of \LINEBUF.OFD with NIL)))
      (SETQ \LASTAIL)
  RESTART
  PROMPT0
      (TTYIN1RESTART)
      (COND
         ((NOT \FIRSTTIME)                                ; Space over to where we started
          (GO.TO.ADDRESSING \INITPOS 0)))
      (SETQ RESULT NIL)
  PROMPT1
      (INIT.CURSOR \INITPOS)
      (COND
         [(AND (EQ \FIRSTTIME 'NOPROMPT)
               \PROMPT1)                                  ; Prompting has already happened;  account for it
          (COND
             ((< (SETQ X (- \INITPOS (STRINGWIDTH \PROMPT1 \FONT)))
                 \LMARG)
              ;; Caller is confused;  prompt couldn't have fit.  Typically happens when LISPXREAD is called by other than LISPX
              (SETQ \PROMPT1))
             (T (SETQ \INITPOS X]
         (T (TTPROMPTCHAR \ARROW)))
      (replace (LINE FIRSTCOL) of \ARROW with (replace (LINE LASTCOL) of \ARROW with \CURSORCOL))
      [COND
         ([OR (NLISTP TABS)
              (NOT (SMALLP (CAR TABS]
          (SETQ TABS))
         ((NOT (> (ITIMES (SUB1 (CAR TABS))
                          \CHARWIDTH)
                  \CURSORCOL))
          ;; Caller specified first tabstop as the position of the first char;  we don't treat that as a tabstop, so peel it off
          (SETQ TABS (CDR TABS]
      (COND
         [UNREADBUF                                       ; something to preload buffer with
                (COND
                   ((FIXP UNREADBUF)
                    (SETQ CHAR UNREADBUF)                 ; interpret number as character code of something to type ahead,
                                                          ; usually altmode
                    (SETQ UNREADBUF NIL)
                    (GO SELECTCHAR))
                   (T (WITH-RESOURCES (TTSCRATCHFILE)
                              (TTLOADBUF (PROG1 (COND
                                                   ((EQMEMB 'PRETTY OPTIONS)
                                                    ;; We were told to pretty-print the FIXed form, so have to use a temp file.
                                                    ;; Pass TTLOADBUF a list (<HISTSTR1> (file start . end)).
                                                    (LIST HISTSTR1 (TTYIN.PPTOFILE
                                                                     (COND
                                                                        ((EQ (CAR (SETQ X (LAST UNREADBUF)))
                                                                             HISTSTR0)
                                                                                          ; knock off the terminating <cr> marker
                                                                         (LDIFF UNREADBUF X))
                                                                        (T UNREADBUF))
                                                                     'PRETTY RDTBL TTSCRATCHFILE)))
                                                   (T                 ; Not pretty printing; just pass TTLOADBUF the form to FIX.
                                                      UNREADBUF))
                                                (SETQ UNREADBUF NIL)
                                                (SETFILEPTR TTSCRATCHFILE 0))]
         (\FIRSTTIME
                ;; (for FORM in AFTERPROMPTCHARFORMS bind REFRESH when (EVAL FORM) do (SETQ REFRESH T) (* User forms
                ;; to do after prompt is printed but before we do anything more.  If one returns T, means it altered the display) finally (COND
                ;; (REFRESH (SETQ \FIRSTTIME) (GO PROMPT1))))
                ))
      (SETQ \FIRSTTIME)
      (COND
         (CHAR (GO SELECTCHAR)))
  CHAR
      (AND CHAR (SETQ \LASTCHAR CHAR))
```

```
        (SETQ CHAR (TTBIN))
      SELECTCHAR
        [COND
           ([AND (SETQ X (FASSOC CHAR TTYINREADMACROS))
                 (OR [NLISTP (SETQ X (CDR (SETQ TMP X]
                     (AND (COND
                             ((EQ (CAR X)
                                  T)
                              (EMPTY.BUFFER))
                             ((LISTP (CAR X))
                              (EVAL (CAR X)))
                             (T                                    ; Old style macros that worked only at start of buffer
                                (SETQ X TMP)
                                (EMPTY.BUFFER)))
                          (OR (NLISTP (SETQ X (CDR X)))
                              (SETQ X (EVAL X]
            ;; Simple read macros: if you type the char on a blank line, and the macro returns something, use it as the value of the READ (or
            ;; whatever)
            (COND
               [(FIXP X)                                           ; Special: means pretend this CHARACTER code was typed
                (SELECTQ X
                     (0                                            ; No action
                        (GO CHAR))
                     (-1                                           ; Means refresh line, because terminal control was taken away
                         (SETQ CHAR NIL)
                         (GO PROMPT1))
                     (COND
                        ((METACHARP (SETQ CHAR X))
                         [COND
                            ((EQ (NONMETACHARBITS X)
                                 0)                                ; another way to get edit prefix
                             (SETQ CHAR (METACHAR (TTBIN T]
                          T]
                ((EMPTY.BUFFER)                                    ; For now I'm not handling funny results in the middle
                 (SETQ RESULT (OR (LISTP X)
                                  (LIST X)))
                 (GO DOCRLF]
        (COND
           ((NOT (METACHARP CHAR))
            (SETQ WASEDITCHAR NIL))
           ([NOT (SETQ CHAR (DO.EDIT.COMMAND (NONMETACHARBITS CHAR]
            (GO CHAR))
           (T                                                      ; Fall thru if edit char gave us something to chomp on
              (SETQ WASEDITCHAR T)))
        [COND
           ((SELECTC (fetch TERMCLASS of (\SYNCODE \PRIMTERMSA CHAR))
                 (CHARDELETE.TC
                      (TTDELETECHAR)
                      T)
                 (LINEDELETE.TC
                      (TTDELETELINE)
                      T)
                 (WORDDELETE.TC
                      (TTDELETEWORD)
                      T)
                 (RETYPE.TC                                        ; ^R retype
                           (SETQ \PFW.FIRSTTIME NIL)
                           [RETYPE.BUFFER (COND
                                             ((OR (ON.FIRST.LINE)
                                                  (NOT (EMPTY.LINE)))
                                              \ARROW)
                                             (T                    ; If sitting on empty line, refresh the previous line
                                                (PREVLINE \ARROW 1]
                           (COND
                              ((EQ CHAR (SETQ CHAR (TTBIN)))       ; two ^R's means retype whole buffer
                               (OR DISPLAYTERMFLG (TTCRLF))        ; set off full retype by double line
                               (RETYPE.BUFFER \FIRSTLINE T))
                              (T (GO SELECTCHAR)))
                           T)
                 NIL)                                              ; Did some routine editing command.  This cancels
                                                                   ; promptforword kill mode
            (SETQ \PFW.FIRSTTIME NIL))
           ((PROGN (SETQ SNX (\SYNCODE \RDTBLSA CHAR))
                   (AND \FILLINGBUFFER (EQ (fetch WAKEUP of SNX)
                                           IMMEDIATE.RMW)
                        (AT.END.OF.TEXT \CURSOR)))                 ; Immediate read macro--return now
            (GO DOCRLF))
           (T (if \PFW.FIRSTTIME
                  then                                             ; The only non-meta characters that accept the input are cr,
                                                                   ; space and the hard-wired editing commands (which we have
                                                                   ; mostly covered already)
                      (SELCHARQ CHAR
                           ((CR SPACE ^X ^A BS RUBOUT ^Q ^U ^W))
                           (PROGN                                  ; Kill the entire input (could be more than one line if long input or
                                                                   ; long prompt)
                                  (MOVE.TO.LINE \FIRSTLINE)
```

```
                        (DELETE.TO.END)))
                (SETQ \PFW.FIRSTTIME NIL))
        (COND
           ((AND (fetch STOPATOM of SNX)
                 (NOT \DONTCOMPLETE))                    ; End of atom, try completion
            (TTCOMPLETEWORD T)))
        (SELECTC SNX
            ((LIST RIGHTPAREN.RC RIGHTBRACKET.RC)
                                ;; Right paren/bracket.  See if it terminates read.  Note that \READING is implicitly true here,
                                ;; since there are no parens in the word rdtbl
                (SETQ STARTOFWORD \CURSOR)
                (ADDCHAR CHAR)
                (COND
                   ((ENDREAD?)
                    (GO DOCRLF))
                   ((AND SHOWPARENFLG T (NOT (TYPEAHEAD?)))
                                                         ; prime conditions for hack to show which paren it matched
                    (SHOW.MATCHING.PAREN STARTOFWORD))))
            (SELECTC CHAR
                ((CHARCODE ESCAPE)
                    [COND
                       (SPLST                            ; try to complete from spelling list
                              (OR (TTCOMPLETEWORD)
                                  (BEEP)))
                       [(AND TTYINCOMPLETEFLG \LISPXREADING)
                                ;; always try to complete
                        (COND
                           ((SETQ STARTOFWORD (CURRENT.WORD))
                            (SETQ MATCHED (FIND.MATCHING.WORD USERWORDS STARTOFWORD)))
                           ((NEQ TTYINCOMPLETEFLG 0)
                                ;; naked escape stands for LASTWORD.
                            (SETQ MATCHED (LIST LASTWORD))
                            LASTWORD))
                        (SETQ CHAR DIDESCAPECODE)        ; Kludge used by ? routine below
                        (COND
                           (MATCHED (OR (TTCOMPLETEWORD NIL NIL MATCHED (OR STARTOFWORD \CURSOR))
                                        (BEEP)))
                           (T (BEEP]
                       (T                                ; no special significance
                          (ADDNAKEDCHAR (CHARCODE ESCAPE])
                ((CHARCODE (%" *))
                    (ADDCHAR CHAR)
                    (TTDOTABS TABS))
                ((CHARCODE TAB)
                    (OR (TTDOTABS TABS)
                        (TTADDTAB)))
                ((CHARCODE SPACE)
                    (if (AND (EQ \PROMPTFORWORD 'PROMPTFORWORD-SPACE)
                             (AT.END.OF.TEXT \CURSOR))
                        then                             ; Space completes
                             (GO DOCRLF))
                    (OR (AUTOCR?)
                        (ADDCHAR CHAR)))
                ((CHARCODE ?)                            ; supply alternative completions
                    (TTYIN.SHOW.?ALTERNATIVES))
                ((CHARCODE CR)                           ; terminate line
                    [COND
                       ((NOT WASEDITCHAR)                ; i.e.  not edit-CR
                                                         ; Check for ?  and ?= macros
                        (PROG ((START (fetch (LINE START) of \ARROW))
                               TAIL)
                              (COND
                                 ((EQ \CURSOR START)
                                  (RETURN)))
                              (SETQ TAIL (NLEFT START 1 \CURSOR))
                                                         ; Look at last char on line
                              (SELCHARQ (CAR TAIL)
                                   (? (COND
                                         ((AND (DEFINEDP 'XHELPSYS)
                                               [OR (EQ TAIL START)
                                                   (BREAK.OR.SEPRP (FIRSTCHAR (NLEFT START 1 TAIL]
                                               (DO?CMD '? TAIL))
                                          (GO CHAR))))
                                   (= (COND
                                         ((AND (NEQ TAIL START)
                                               (EQ (CAR (SETQ TAIL (NLEFT START 1 TAIL)))
                                                   (CHARCODE ?))
                                               [OR (EQ TAIL START)
                                                   (BREAK.OR.SEPRP (FIRSTCHAR (NLEFT START 1 TAIL]
                                               (DO?CMD '?= TAIL))
                                          (GO CHAR))))
                                   NIL))
                        (COND
                           ((NOT (AT.END.OF.TEXT \CURSOR))
```

```
                    (COND
                       ((OR \REPEAT \READING)   ; Insert a <cr> and continue reading
                        (BREAKLINE EOLCHARCODE)
                        (GO CHAR))
                       (T                       ; <cr> typed here would terminate, so unread what's left
                          (TTUNREADBUF]
               (COND
                  [(NOT (AT.END.OF.BUF))
                   (COND
                      ((ON.LAST.LINE)
                       (SETQ \CURSOR \ENDBUFFER))
                      ((AND \READING (NOT \PROMPT2)
                            (AT.END.OF.TEXT (fetch (LINE END) of \ARROW)))
                       ;; Really the same condition as previous clause: there are lines after this one, but they're blank, so it
                       ;; looks like we're on the last line
                       (MOVE.FORWARD.TO (fetch (LINE END) of \ARROW))
                                                ; have to make the extra stuff go away so the finishing routines
                                                ; are happy
                       (DELETE.TO.END))
                      (T (DO.EDIT.COMMAND (CHARCODE CR))
                                                ; CR on other than last line just means go down one
                         (GO CHAR]
                  ((OR (NOT \DONTCOMPLETE)
                       (EQ \DONTCOMPLETE 'CRCOMPLETE))
                   (TTCOMPLETEWORD T)))
               (COND
                  ((COND
                      (\READING (TTSKREAD \BUFFER))
                      [\REPEAT (AND (ON.FIRST.LINE)
                                    (OR (EQ (CAR \BUFFER)
                                            TTYINCOMMENTCHAR)
                                        (AND \COMMAND (EQ (FIND.NEXT.WORD (FIND.NON.SPACE \BUFFER
                                                                                  ))
                                                          \ENDBUFFER]
                      (T T))                    ; Terminating conditions: no REPEAT, or first line is a comment
                                                ; or has a single command on it
                   (SETQ CTRLVFLG (SETQ RESULT))
                   (SETQ CHAR (CHARCODE EOL))   ; Lisp likes to treat cr as (choke) EOL
                   (GO DOCRLF))
                  (T (START.NEW.LINE EOLCHARCODE))))
         ((CHARCODE ^X)                         ; Maybe exit
               (COND
                  ((COND
                      (\READING                 ; return if parens balance.  If already at end, add enough parens
                                                ; to balance
                            (TTYIN.BALANCE T (AT.END.OF.TEXT \CURSOR)))
                      (T                        ; Taking string input, etc--finish now
                         (MOVE.TO.WHEREVER \ENDBUFFER)
                         T))
                   (SETQ CHAR (CHARCODE EOL))
                   (GO DOCRLF))))
         ((CHARCODE ^V)
               (COND
                  ((NEQ \REPEAT 'TEXT)          ; Means enter control char
                   (SETQ CHAR (TTBIN))
                   (ADDNAKEDCHAR (if (EQ CHAR (CHARCODE ?))
                                     then       ; This is the only way to get a rubout
                                          (SETQ CHAR (CHARCODE RUBOUT))
                                   elseif (>= CHAR (CHARCODE @))
                                     then       ; Change alphabetics to corresponding control char
                                          (SETQ CHAR (LOGAND CHAR 31))
                                   else         ; take exact char
                                        CHAR)))
                  ((AT.END.OF.BUF)              ; terminate multiline input and set special flag
                   (SETQ CTRLVFLG T)
                   (TTBOUT ^ V)
                   (GO DOCRLF))
                  (T (BEEP))))
         ((CHARCODE ^Z)                         ; ^Z terminates multiline input
               (COND
                  ((AND \REPEAT (AT.END.OF.BUF))
                   (TTBOUT ^ Z)
                   (SETQ CTRLVFLG)
                   (GO DOCRLF))
                  (\READING (ADDNAKEDCHAR CHAR))
                  (T (BEEP))))
         ((CHARCODE ^Y)                         ; ^Y invokes user exec
               (COND
                  ((AND \READING (NOT WASEDITCHAR))
                                                ; let ^Y read macro work instead
                   (ADDNAKEDCHAR CHAR))
                  ((GUESTUSER?)
                   (BEEP))
                  (T (SETTAIL?)
                     (SAVE.CURSOR)
                     (GO.TO.FREELINE)
```

```
                              (COND
                                 (DRIBFL                    ; Make typescript understandable
                                        (AND \PROMPT1 (PRIN1 \PROMPT1 DRIBFL))
                                        (PRINT '^Y DRIBFL)))
                              (PRIN1 "lisp:
                                     " T)
                              (COND
                                 (TTYINMAILFLG (MWNOTE)))
                              (RESTOREMOD)
                              (PROG ((\INSIDE.TTYIN))
                                    (USEREXEC '__))
                              (GO RETYPEBUFFER))))
                        (0                                  ; ignore NULL
                           )
                        ((CHARCODE (^A BS RUBOUT))
                             (TTDELETECHAR))
                        ((CHARCODE (^Q ^U))                 ; ^Q delete line;  ^U on tops20
                             (TTDELETELINE))
                        ((CHARCODE ^W)                      ; ^W delete last word
                             (TTDELETEWORD))
                        (COND
                           ([MEMB CHAR (OR \RESTOREBUFCODES (SETQ \RESTOREBUFCODES
                                                              (APPEND (AND (EQ (MACHINETYPE)
                                                                               'DORADO)
                                                                           DORADO.RESTORE.BUF.CODES)
                                                                      TTYIN.RESTORE.BUF.CODES]
                            ;; One of the characters we interpret as "restore last buffer".  Recomputed after exit in case we change machine.
                            ;; The dorado code is a perfectly good charset 0 code, so don't usually want to usurp it.
                            (RESTOREBUF))
                           [(> CHAR 32)                     ; not a control char
                            (ADDCHAR (COND
                                        (\RAISEINPUT (U-CASECODE CHAR))
                                        (T CHAR]
                           (T (ADDNAKEDCHAR CHAR]
                (GO CHAR)
RETYPEBUFFER
                (RETYPE.BUFFER \FIRSTLINE T T)
                (GO CHAR)
DOCRLF

;; Come here when it is time to terminate line
                (COND
                   ((EQ (SETQ RESULT (TTYIN.FINISH CHAR DRIBFL RESULT))
                        'ABORT)                             ; Aborted, try again
                    (SETQ CHAR NIL)
                    (GO PROMPT0))
                   (T (RETURN RESULT])
```

(TTYIN1RESTART

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:39 by jds
    (\RESETLINE)                                            ; clear some terminal-related stuff, including the info about where
                                                            ; to hold scroll
    (\SETEOFPTR \LINEBUF.OFD 0)                             ; Clear the line buffer
    (SETQ \ARROW (SETQ \FIRSTLINE (fetch (TTYINBUFFER FIRSTLINE) of \TTYINSTATE)))
    [replace (LINE END) of \ARROW with (SETQ \CURSOR (SETQ \BUFFER (SETQ \ENDBUFFER (fetch (LINE START)
                                                                                      of \ARROW]
    [PROG ((MORELINES (fetch (LINE NEXTLINE) of \ARROW)))
          (COND
             (MORELINES                                     ; Return old line records to cons pool
                    (replace (LINE NEXTLINE) of \ARROW with NIL)
                    (KILL.LINES MORELINES]
    (SETQ \DELETING])
```

(TTYIN.FINISH

```
  [LAMBDA (FINALCHAR DRIBFL RESULT)                         ; Edited 24-May-91 10:39 by jds
    (PROG ((*READTABLE* RDTBL)
           WORD X ORIGBUFFER)
          (COND
             ((NOT \PROMPTFORWORD)                          ; Go to new line.  Fake promptforword mode doesn't do this.
              (TTCRLF)
              (CLEAR.LINE? T)))
          [COND
             ((EQ FINALCHAR (CHARCODE EOL))
              (bind TAIL (START _ (fetch (LINE START) of \ARROW))
                 while (AND (NEQ START \ENDBUFFER)
                            (EQ (CAR (SETQ TAIL (TTNLEFT \ENDBUFFER 1 START)))
                                (CHARCODE SPACE))
                            (NEQ (\SYNCODE \RDTBLSA (FIRSTCHAR (TTNLEFT TAIL 1 START)))
                                 ESCAPE.RC))
                 do ;; Strip blanks, e.g., resulting from escape completion, so that Lispx does not do its silly ... thing.  Be careful not to strip a
                    ;; quoted space
                    (SETQ \ENDBUFFER TAIL]
```

```
(COND
   (DRIBFL                                                  ; print answer on typescript file
          (TTECHO.TO.FILE DRIBFL T)))
(for X inside ECHOTOFILE do (TTECHO.TO.FILE X))
(COND
   [(EMPTY.BUFFER)                                          ; blank line.  RESULT is NIL unless set above by a read macro
    (COND
       ((OR RESULT (EQ FINALCHAR (CHARCODE EOL)))
        (SETLASTC (CHARCODE EOL))
        (RETURN RESULT]
   ((EQ (CAR \BUFFER)
        TTYINCOMMENTCHAR)                                   ; comment
    (RETURN 'ABORT))
   ((AND (EQ (CDR \BUFFER)
             \ENDBUFFER)
         (EQ (CAR \BUFFER)
             (CHARCODE ?))
         (OR HELP (AND \NOVALUE \REPEAT)))                  ; a bare ?
    (TTGIVEHELP (OR HELP "Terminate text with control-Z."))
    (RETURN 'ABORT))
   (T                                                       ; Save last buffer position for posterity
      (replace (TTYINBUFFER OLDTAIL) of \TTYINSTATE with \ENDBUFFER)))
(SETQ ORIGBUFFER \BUFFER)
[COND
   [\READING (SETQ RESULT (COND
                            (\FILLINGBUFFER (TTYIN.READ FINALCHAR T \LINEBUF.OFD))
                            (T (TTYIN.READ FINALCHAR NIL (TTYIN.SCRATCHFILE]
   ((AND HELP (FIND.MATCHING.WORD '(? HELP)
                  \BUFFER \ENDBUFFER))
    (TTGIVEHELP HELP)                                       ; help handled; now restart
    (RETURN 'ABORT))
   ((AND \STRINGVALUE (NOT \COMMAND))                       ; Return input as string, no other special interpretation
    (SETQ RESULT (TTYINSTRING ORIGBUFFER)))
   (T
    (SETQ WORD (TTRATOM))
    [for RESPONSE in TTYINRESPONSES when (AND (EQMEMB WORD (CAR RESPONSE))
                                              (OR (EQ \BUFFER \ENDBUFFER)
                                                  (CADDR RESPONSE)))
       do ;; Process global user option.  RESPONSE is a triple (commands response-form rest-of-line-arg);  if user gives one of the
          ;; commands, the response form is evaluated with \COMMAND set to the command and LINE set to the remainder of the line;
          ;; the third component says how to compute LINE: as a STRING or as a LIST;  if NIL, means there should be nothing else on
          ;; the line.  If the response form returns the atom IGNORE, the input is not considered to be a special response and the
          ;; normal computation proceeds;  otherwise it is assumed the response has been processed, and we return to the original
          ;; TTYIN prompt for more input.  Response-form may be an atom, in which case it is APPLYed to \COMMAND and LINE.
          (COND
             ((NEQ [PROG [(\COMMAND WORD)
                          (\BUFFER \BUFFER)
                          (LINE (COND
                                   ((EQ \BUFFER \ENDBUFFER)
                                    NIL)
                                   ((EQ (CADDR RESPONSE)
                                        'STRING)
                                    (TTYINSTRING \BUFFER))
                                   (T (TTREADLIST]
                         (DECLARE (SPECVARS \COMMAND \BUFFER LINE))
                         (RETURN (COND
                                    ((LITATOM (CADR RESPONSE))
                                     (APPLY* (CADR RESPONSE)
                                             \COMMAND LINE))
                                    (T (EVAL (CADR RESPONSE]
                   'IGNORE)
              (RETFROM 'TTYIN.FINISH 'ABORT))
             (T ;; That response was ignored.  We could quit the iteration now, but continue in case there is another entry with the
                ;; same command.  I.e.  user can 'redefine' special responses this way, but still let the old definition happen if the
                ;; input looks wrong
                ]
    [SETQ WORD (COND
                  ((TTADJUSTWORD WORD))
                  ((AND (NULL WORD)
                        (NULL SPLST))                       ; NIL is acceptable response, so don't abort!
                   NIL)
                  (T (RETURN 'ABORT]
    [SETQ RESULT (COND
                    [(EQ \BUFFER \ENDBUFFER)
                     (COND
                        (\COMMAND (LIST WORD))
                        (\NOVALUE T)
                        (T (LIST WORD]
                    (\STRINGVALUE                           ; return (command . string).  Note that if \command is false, we
                                                            ; handled it much earlier
                           (CONS WORD (TTYINSTRING \BUFFER)))
                    (\NOVALUE (COND
                                 (\COMMAND (CONS WORD T))
                                 (T T)))
                    (T (SETQ RESULT (TTREADLIST))
                       (COND
```

```
                                    ((OR \COMMAND (NULL SPLST))  ; only check first word typed, or nothing at all
                                     (CONS WORD RESULT))
                                    (T (for TL on RESULT do [RPLACA TL (COND
                                                                          ((TTADJUSTWORD (CAR TL)))
                                                                          ((AND (NULL (CAR TL))
                                                                                (NULL \FIX))
                                                                           ; NIL is acceptable response, so don't abort!
                                                                           NIL)
                                                                          (T (RETURN 'ABORT]
                                                finally (RETURN (CONS WORD RESULT]
                  (SETLASTC FINALCHAR)
                  (PROGN
                   ;; All this nonsense is just to convince prettyprint to keep the indentation down to a reasonable amount
                   (PROGN
                    (PROGN
                     (PROGN
                      (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN NIL]
          ;; We have now processed the line, with the relevant value being RESULT...
               [COND
                  ((AND TTYINUSERFN (LISTP RESULT))
                   (COND
                      ((EQ (SETQ X (APPLY* TTYINUSERFN RESULT))
                           T)                                  ; Special response has been processed;  try again
                       (RETURN 'ABORT))
                      (X                                       ; this is what we should return
                         (RETURN X]
               (SETQ \CURRENTDISPLAYLINE 0)                    ; get scrolling right (again)
                                                               ; see system \CLOSELINE
               (RETURN RESULT])
```

(TTYIN.BALANCE

```
  [LAMBDA (ERRORFLG ADDPARENS)                                 ; Edited 17-Jan-88 16:36 by bvm:
    (LET ((X (TTSKREAD \BUFFER NIL ADDPARENS)))
         (PROG1 (COND
                   [(OR (EQ X \ENDBUFFER)
                        (AND (LISTP X)
                             (EQ (\SYNCODE \RDTBLSA (FIRSTCHAR X))
                                 RIGHTBRACKET.RC)
                             (AT.END.OF.TEXT (CDR X]
                   ((FIXP X)                                   ; Number of parens you'd have to add to balance
                    (MOVE.TO.WHEREVER \ENDBUFFER)
                    (RPTQ X (ADDCHAR (CHARCODE ")")))
                    (SETQ X NIL)
                    T)
                   (T (COND
                         ((AND ERRORFLG (EQ \CURSOR (OR X \ENDBUFFER)))
                                                               ; Only beep if cursor won't move
                          (BEEP)))
                      NIL))
              (MOVE.TO.WHEREVER (OR X \ENDBUFFER)))])
```

(ADDCHAR

```
  [LAMBDA (CHAR)
    (DECLARE (USEDFREE \CURSORCOL \ARROW \RMARG \CURSOR \AUTOFILL))
                                                               ; Edited 24-May-91 11:09 by jds
```

;;; Add CHAR to buffer and print it, advancing cursor position appropriately

```
    (LET ([WIDTH (COND
                    ((COMPLEXCHARP CHAR)
                     (fetch (COMPLEXCHAR CPXWIDTH) of CHAR))
                    (T (TTBITWIDTH CHAR]
          (ENDP (AT.END.OF.LINE)))
         (END.DELETE.MODE)
         (OVERFLOW? WIDTH)
         (COND
            ((NOT ENDP)                                        ; Inserting in middle of line, so make space
             (TTINSERTSECTION WIDTH)))
         (COND
            ((COMPLEXCHARP CHAR)
             (for PC in (fetch (COMPLEXCHAR CPXPRINTCHARS) of CHAR) do (TTBOUT PC)))
            (T (TTBOUT CHAR)))
         (INSERT.NODE \CURSOR)
         (FRPLACA \CURSOR CHAR)
         (SETQ \CURSOR (CDR \CURSOR))
         (add \CURSORCOL WIDTH)
         [COND
            (ENDP (replace (LINE END) of \ARROW with \CURSOR)
                  (replace (LINE LASTCOL) of \ARROW with \CURSORCOL)
                                                               ; If we just advanced past the last column, do autofill stuff
                  (OVERFLOW? 0))
            (T                                                 ; Check to see if line got shoved beyond right margin
               (LET ((OVFL (IDIFFERENCE (add (fetch (LINE LASTCOL) of \ARROW)
```

```
                                          WIDTH)
                                  \RMARG)))
                    (COND
                       ((OR (IGREATERP OVFL 0)
                            (AND (EQ OVFL 0)
                                 \AUTOFILL))
                        (ADJUSTLINE (AND \AUTOFILL T))
                        (MOVE.TO.WHEREVER \CURSOR]
          NIL])
```

(TTMAKECOMPLEXCHAR

```
  [LAMBDA (REALCHAR PRINTCHARS)                                    (* bvm%: "16-Apr-85 16:50")
    (LET ((WIDTH 0)
          (NC 0))
         (for C in PRINTCHARS do (add WIDTH (TTBITWIDTH C))
                                 (add NC 1))
         (create COMPLEXCHAR
                 CPXREALCHAR _ REALCHAR
                 CPXWIDTH _ WIDTH
                 CPXNCHARS _ NC
                 CPXPRINTCHARS _ PRINTCHARS])
```

(ADDNAKEDCHAR

```
  [LAMBDA (CHAR NOAUTOFILL)                                        (* bvm%: "17-Apr-85 19:46")
```

;;; Adds CHAR with no special processing, e.g. most control chars (except cr and lf, which I can't figure out yet) go thru ok.

```
    (COND
       ((AND (IGREATERP CHAR 32)
             (NEQ CHAR 127))
        (ADDCHAR CHAR))
       (T (SELCHARQ CHAR
                 (CR                                               ; CR can be attempted if at end
                     (COND
                        ((AT.END.OF.BUF)
                         (START.NEW.LINE EOLCHARCODE))
                        (T (BEEP))))
                 (SPACE (OR (AND (NOT NOAUTOFILL)
                                 (AUTOCR?))
                            (ADDCHAR (CHARCODE SPACE))))
                 (ESCAPE                                           ; Altmode will echo as $
                         [ADDCHAR (TTMAKECOMPLEXCHAR CHAR (LIST (CHARCODE $])
                 (TAB (TTADDTAB))
                 (ADDCHAR (TTMAKECOMPLEXCHAR CHAR (LIST (CHARCODE ^)
                                                        (COND
                                                           ((EQ CHAR (CHARCODE DEL))
                                                              ; DELETE is represented as ^?
                                                            (CHARCODE ?))
                                                           (T (LOGOR CHAR 64])
```

(TTADDTAB

```
  [LAMBDA NIL                                                      ; Edited 24-May-91 10:33 by jds
```

;; Represent <tab> in buffer as a tab with 128 bit on, followed by the appropriate number of spaces, each with 256 bit on. Tab is always
;; self-inserting, i.e. it never overwrites anything (except itself, as above)

```
    (ADDCHAR (TTMAKECOMPLEXCHAR (CHARCODE TAB)
                     (from (LOGAND (IQUOTIENT (IDIFFERENCE \CURSORCOL (fetch (LINE FIRSTCOL) of \ARROW))
                                              \CHARWIDTH)
                                   7)
                           to 7 collect (CHARCODE SPACE])
```

(ADJUSTLINE

```
  [LAMBDA (JUSTIFYING LINE)                                        ; Edited 24-May-91 10:33 by jds
```

;; Handles patching up lines that are too long or short. Assures that the current line, ARROW, is correct with regard to overflows. If JUSTIFYING is
;; true, it is a number specifying how many lines to 'justify' , by which we mean moving text around so that each line has as many words as possible
;; for the linelength, but does not overflow. We don't do anything very fancy with that, like take care of deleting extra spaces.

```
    (PROG ((IDEALLENGTH (- [COND
                             ((> TTYJUSTLENGTH 0)
                              (IMIN \RMARG (TIMES TTYJUSTLENGTH \CHARWIDTH)))
                             (T                                    ; Relative to right margin
                                (IMAX (- \RMARG (TIMES (- TTYJUSTLENGTH)
                                                       \CHARWIDTH))
                                      (+ (LRSH (- \RMARG \LMARG)
                                               1)
                                         \LMARG]
                           \LMARG))
           BREAK LASTCOL NEWENDLINE NEXTLINE OLDENDLINE OVFL START USECR ROW %#BITS)
          (OR LINE (SETQ LINE \ARROW))
          (SETQ ROW (fetch (LINE ROW) of LINE))
      LP  (SETQ NEXTLINE (fetch (LINE NEXTLINE) of LINE))
          (SETQ OVFL (OVERFLOWLINE? LINE))
          (SETQ %#BITS (- \RMARG (fetch (LINE LASTCOL) of LINE)))
```

```
          (SETQ USECR (SETQ BREAK NIL))
          (SETQ START (fetch (LINE START) of LINE))
          (COND
             ((< %#BITS 0)
              ;; Too much on line;  need to break it somewhere, preferably at a space if permissible.  If justifying, try to break at the appropriate
              ;; length
              (COND
                 ([OR (AND JUSTIFYING (< (+ (fetch (LINE FIRSTCOL) of LINE)
                                            IDEALLENGTH)
                                         \RMARG)
                             (SETQ BREAK (FIND.LINE.BREAK START (NTH.RELATIVE.COLUMN.OF LINE IDEALLENGTH)
                                                           T)))
                      (PROGN (SETQ NEWENDLINE (NTH.COLUMN.OF LINE \RMARG))
                             (AND (OR JUSTIFYING \AUTOFILL)
                                  (SETQ BREAK (FIND.LINE.BREAK START NEWENDLINE T]
                  (SETQ USECR T))
                 (T (SETQ BREAK NEWENDLINE)))
              (GO DOBREAK))
             [(AND OVFL (NEQ %#BITS 0)
                   (NEQ (SETQ NEWENDLINE (NTH.RELATIVE.COLUMN.OF NEXTLINE %#BITS))
                        (fetch (LINE START) of NEXTLINE)))
              ;; Line is too short, but is an overflow line, so text MUST be moved to fill the gap;  alternatively, if we are justifying, we could break the
              ;; line sooner
              ;; NEWENDLINE = where the line should end, based on linelength
              (COND
                 ([OR (EQ (fetch (LINE END) of LINE)
                          NEWENDLINE)
                      (AND (OR \AUTOFILL JUSTIFYING)
                           (SETQ BREAK (FIND.LINE.BREAK (fetch (LINE END) of LINE)
                                                        NEWENDLINE JUSTIFYING))
                           (SETQ NEWENDLINE BREAK))
                      (NOT JUSTIFYING)
                      (NOT (SETQ BREAK (FIND.LINE.BREAK START (fetch (LINE END) of LINE)
                                                        T]
                  (GO DOJOIN))
                 (T (SETQ USECR T)
                    (GO DOBREAK]
             ((NOT JUSTIFYING)
              (RETURN))
             [(OR OVFL (AND (NEQ JUSTIFYING T)
                            (> (- (fetch (LINE LASTCOL) of LINE)
                                  (fetch (LINE FIRSTCOL) of LINE))
                               IDEALLENGTH)))                    ; line is longer than we'd like
              (COND
                 ((SETQ BREAK (FIND.LINE.BREAK START (NTH.RELATIVE.COLUMN.OF LINE IDEALLENGTH)
                                                T))
                  (SETQ USECR T)
                  (GO DOBREAK]
             [[AND (NOT (EMPTY.LINE LINE))
                   (NOT (START.OF.PARAGRAPH? NEXTLINE))
                   (OR (NEQ JUSTIFYING T)
                       (EQ (CAR (fetch (LINE END) of LINE))
                           (CHARCODE SPACE]
              ;; Don't move up text from next line if it is blank or starts with tab -- treat those as paragraph breaks
              ;; Note that we are guaranteed at this point that LINE is not an overflow line, so (fetch END of LINE) points at a space or cr
              (COND
                 ((OR (EQ [SETQ BREAK (NTH.RELATIVE.COLUMN.OF NEXTLINE
                                                  (SUB1 (IMIN (- (+ IDEALLENGTH (fetch (LINE FIRSTCOL) of LINE))
                                                                 (fetch (LINE LASTCOL) of LINE))
                                                              %#BITS]
                          (fetch (LINE END) of NEXTLINE))
                      (SETQ BREAK (FIND.LINE.BREAK (fetch (LINE START) of NEXTLINE)
                                                   BREAK T)))
                  (SETQ NEWENDLINE BREAK)                        ; At least one more word from next line will fit up here
                  (GO DOJOIN))
                 (T                                              ; No text movement, but if line ended in a real <cr>, make it a
                                                                 ; space
                    (FRPLACA (fetch (LINE END) of LINE)
                             (CHARCODE SPACE]
             ((EQ JUSTIFYING T)                                  ; If this line is fine, quit

              ))
          (SETQ LINE NEXTLINE)
          (GO BOTTOM)
       DOJOIN

      ;; Move text from next line up to this one.  NEWENDLINE is where line should end when done.  BREAK=NEWENDLINE if this new end line is a
      ;; pseudo-cr break
          (COND
             ((EQ (SETQ OLDENDLINE (fetch (LINE END) of LINE))
                  NEWENDLINE)
              (SETQ %#BITS 0))
```

```
            (T (GO.TO.RELATIVE (fetch (LINE LASTCOL) of LINE)
                               ROW)
               (SETQ %#BITS (SEGMENT.BIT.LENGTH OLDENDLINE NEWENDLINE))
                                                          ; # chars to delete from next line
               [COND
                  ((NOT OVFL)                             ; Joining toa non-overflow line: turn its cr into a space
                   (FRPLACA OLDENDLINE (CHARCODE SPACE))
                   (while (AND (NEQ (CDR OLDENDLINE)
                                    NEWENDLINE)
                               (EQ (CADR OLDENDLINE)
                                   (CHARCODE SPACE)))
                      do                                  ; strip leading spaces from next line
                         (KILLSEGMENT OLDENDLINE (CDR OLDENDLINE)))
                   (COND
                      ((EQ (CAR (NLEFT (fetch (LINE START) of LINE)
                                       1 OLDENDLINE))
                           (CHARCODE %.))                 ; LINE ends in period, so space twice
                       (FRPLACA (INSERT.NODE OLDENDLINE)
                                (CHARCODE SPACE]
               (TYPE.BUFFER OLDENDLINE NEWENDLINE)
               (replace (LINE END) of LINE with NEWENDLINE)
               (replace (LINE LASTCOL) of LINE with \CURSORCOL)))
         (GO.TO.RELATIVE 'LINE NEXTLINE)
         (replace (LINE START) of NEXTLINE with (COND
                                                   (BREAK (FRPLACA BREAK (CHARCODE SPACE))
                                                          ; In case BREAK was at the CR turn it into space
                                                          (COND
                                                             (OVFL (add %#BITS (TTBITWIDTH (CHARCODE SPACE)))
                                                          ; will delete space also

                                                                   ))
                                                          (CDR NEWENDLINE))
                                                   (T NEWENDLINE)))
         (COND
            ((EQ (fetch (LINE END) of NEXTLINE)
                 NEWENDLINE)
             (DELETELINE NEXTLINE T)                      ; Nothing left here, so kill it
             [COND
                (JUSTIFYING                               ; maybe we can move from next line, too
                       (COND
                          ((AND (NEQ JUSTIFYING T)
                                (NEQ (SUB1VAR JUSTIFYING)
                                     0))
                           (GO LP))
                          (T (RETURN]
             (SETQ LINE (fetch (LINE NEXTLINE) of LINE)))
            (T (TTDELSECTION %#BITS)
               (replace (LINE LASTCOL) of NEXTLINE with (- (fetch (LINE LASTCOL) of NEXTLINE)
                                                           %#BITS))
               (SETQ LINE NEXTLINE)))
         (GO BOTTOM)
      DOBREAK

;; Break line at BREAK, moving excess down to next line or a new line.  USECR is true if break is to act like a cr;  otherwise we are breaking a
;; too-long line at the right margin, so there is no end of line place holder
         [replace (LINE LASTCOL) of LINE with (SETQ LASTCOL (+ (SEGMENT.BIT.LENGTH (fetch (LINE START)
                                                                                       of LINE)
                                                                                   BREAK)
                                                               (fetch (LINE FIRSTCOL) of LINE]
                                                          ; Column where break will occur
         [SETQ %#BITS (SEGMENT.BIT.LENGTH BREAK (SETQ OLDENDLINE (fetch (LINE END) of LINE]
                                                          ; length of segment being moved
         (COND
            ((NEQ LASTCOL \RMARG)
             (GO.TO.RELATIVE LASTCOL ROW)                 ; Go wipe out what was there.  Don't need to do this if the break
                                                          ; is right at the margin
             (ERASE.TO.END.OF.LINE)))
         (replace END of LINE with BREAK)
         [COND
            (USECR                                        ; we have counted one char too many above...
                   [SETQ %#BITS (- %#BITS (TTBITWIDTH (CHARCODE SPACE]
                   (SETQ BREAK (CDR BREAK]
         (COND
            [[AND NEXTLINE (OR OVFL (AND (OR (SMALLP JUSTIFYING)
                                             (AND (EQ (CAR OLDENDLINE)
                                                      (CHARCODE SPACE))
                                                  (< (+ (fetch LASTCOL of NEXTLINE)
                                                        %#BITS)
                                                     \RMARG)))
                                         (NOT (START.OF.PARAGRAPH? NEXTLINE]
             ;; Insert the text on the next line, rather than starting new line, if justifying, overflow (forced), or the text will fit, i.e.  not cause anything
             ;; to be bumped off the next line
             (GO.TO.RELATIVE 'LINE (SETQ LINE NEXTLINE))
             (COND
                ((NOT OVFL)
```

```
                    ;; Turn the terminating <cr> into ordinary space;  this space also needs to be inserted and counted, of course
                    (add %#BITS (TTBITWIDTH (CHARCODE SPACE)))
                    (SETQ OLDENDLINE (CDR (FRPLACA OLDENDLINE (CHARCODE SPACE]
                 (T (SETQ LINE (INSERTLINE LINE))
                    (replace END of LINE with OLDENDLINE)))
              (replace START of LINE with BREAK)
              (INSERT.TEXT BREAK OLDENDLINE (fetch END of LINE))
              (add (fetch LASTCOL of LINE)
                   %#BITS)
        BOTTOM
              (COND
                 (LINE (ADD1VAR ROW)
                       (COND
                          ((AND JUSTIFYING (NEQ JUSTIFYING T)
                                (EQ (SUB1VAR JUSTIFYING)
                                    0))
                           (SETQ JUSTIFYING NIL)))
                       (GO LP])
```

(ADJUSTLINE.AND.RESTORE

```
[LAMBDA (JUSTIFYING)                                          ; Edited 24-May-91 10:33 by jds
  (SAVE.CURSOR)
  (ADJUSTLINE JUSTIFYING)
  (COND
     ((IGREATERP \HOMECOL (fetch (LINE LASTCOL) of \ARROW))   ; Oops, cursor must have moved
      (MOVE.TO.WHEREVER \CURSOR))
     (T (RESTORE.CURSOR])
```

(AT.END.OF.SCREEN

```
[LAMBDA NIL                                                   (* bvm%: "11-Apr-85 14:58")
  (OR (AT.END.OF.LINE)
      (IGREATERP (IPLUS \CURSORCOL (SEGMENT.LENGTH \CURSOR (TTNEXTCHAR \CURSOR))
                        \CHARWIDTH)
            \RMARG])
```

(AT.END.OF.TEXT

```
[LAMBDA (BUF)                                                 (* bvm%: "11-Apr-85 15:00")
  ;; Checks that this is the last printing char in buffer.  Fancier than just checking that BUF = ENDBUFFER, since that would mess up if user deletes
  ;; a line and decides to terminate on previous line
  (for (X _ BUF) by (TTNEXTCHAR X) until (EQ X \ENDBUFFER) always (SPACEP (FIRSTCHAR X])
```

(AUTOCR?

```
[LAMBDA NIL                                                   (* bvm%: "16-Apr-85 18:57")
  ;; Terminates line if near edge of screen and in autofill mode
  (COND
     ((AND \AUTOFILL (IGREATERP (IPLUS \CURSORCOL TTYINAUTOFILLMARGIN)
                                \RMARG))
      [COND
         ((AT.END.OF.LINE)
          (START.NEW.LINE (CHARCODE SPACE)))
         (T (BREAKLINE (CHARCODE SPACE]
      T])
```

(BACKSKREAD

```
[LAMBDA (BUF NOTIFQUOTED)                                     ; Edited  8-Feb-88 12:45 by bvm:
  ;; Returns buffer position of start of list containing cursor position BUF, or start of buffer.  If NOTIFQUOTED is true, then returns NIL if the
  ;; paren/bracket at BUF is quoted with the escape char or is inside a string.  Strategy: start at beginning of buffer and TTSKREAD forward (much
  ;; easier);  if read ends at BUF, we win;  if ends before BUF, then resume reading there (we skipped an internal list);  otherwise if read did not end,
  ;; BUF must be inside a list, so scan ahead for start of an inner list, and repeat
  (PROG ((B \BUFFER)
         (INNERMOSTLIST \BUFFER)
         ESCAPED BRACKETFLG X)
    LP   [COND
            ((EQ B BUF)                                       ; No list in buffer at all
             (RETURN (AND (OR (NOT NOTIFQUOTED)
                              (NOT ESCAPED))
                          INNERMOSTLIST]
         [SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR B))
              ((LIST LEFTPAREN.RC LEFTBRACKET.RC)             ; open paren or bracket.  Try scanning this new internal list
                   [COND
                      (ESCAPED                                ; Inside a multiple escape
                               )
                      ((EQ (SETQ X (TTSKREAD (CDR B)
                                             BUF))
                           BUF)
                       (RETURN (OR BRACKETFLG B)))
                      (X                                      ; Skip over internal list just scanned
                         (SETQ B X))
```

```
                    (T ;; The TTSKREAD failed, so BUF must be at least this deeply nested.  Save pointer here in case we abort inside a
                       ;; string or such
                       (SETQ INNERMOSTLIST B)
                       (COND
                          ((AND (EQ (CAR B)
                                    (CHARCODE %[))
                                (EQ (CAR BUF)
                                    (CHARCODE %])))
                           ;; Brackets may match;  save position of this open bracket.  Otherwise we'll return the innermost list, rather than
                           ;; the start of the bracket expression
                           (SETQ BRACKETFLG B])
              (ESCAPE.RC                                          ;  to quote the next char
                         [COND
                            ((EQ (CDR B)
                                 BUF)
                             ;; The char at BUF is quoted.  This is why TTSKREAD failed here.  Just return the list we're now inside
                             (RETURN (AND (NOT NOTIFQUOTED)
                                          INNERMOSTLIST)))
                            (T                                    ; skip over escape char
                               (SETQ B (CDR B])
              (STRINGDELIM.RC                                     ; double-quote
                   [COND
                      ([AND (NOT ESCAPED)
                            (NOT (SETQ B (FIND.MATCHING.QUOTE (CDR B)
                                                              BUF]            ; Termination analogous to previous case
                       (RETURN (AND (NOT NOTIFQUOTED)
                                    INNERMOSTLIST])
              (MULTIPLE-ESCAPE.RC
                   (SETQ ESCAPED (NOT ESCAPED)))
              (OTHER.RC NIL)
              (PROGN (COND
                        ((AND (EQ (CAR B)
                                  (CHARCODE ;))
                              (READTABLEPROP RDTBL 'COMMONLISP))
                                                                  ; Handle semicolon special
                         (COND
                            ([do (SETQ B (CDR B))
                                 (COND
                                    ((EQ B BUF)
                                     (RETURN T))
                                    ((EQ (FIRSTCHAR B)
                                         (CHARCODE EOL))
                                     (RETURN]                     ; Done inside a comment
                             (RETURN (AND (NOT NOTIFQUOTED)
                                          INNERMOSTLIST]
        (SETQ B (CDR B))
        (GO LP])
```

(**BACKWARD.DELETE.TO**

```
  [LAMBDA (BUF)                                                   (* bvm%: "19-MAR-81 11:55")
    (FORWARD.DELETE.TO (PROG1 \CURSOR (MOVE.BACK.TO BUF])
```

(**BREAKLINE**

```
  [LAMBDA (USECR STAY)
    (DECLARE (USEDFREE \CURSOR \ARROW \CURSORCOL \CURSOR))        ; Edited 24-May-91 10:33 by jds
```

;;; Break current line at \CURSOR position, inserting a suitable <cr> if USECR is given. If STAY is true, \CURSOR does not move; otherwise cursor
;;; moves to first position of new line.

```
    (PROG ((OLDLINE \ARROW)
           (OLDEND (fetch (LINE END) of \ARROW)))
          (replace (LINE END) of \ARROW with \CURSOR)             ; terminate current line at \CURSOR position
          (replace (LINE LASTCOL) of \ARROW with \CURSORCOL)
          (ERASE.TO.END.OF.LINE)
          (COND
             (STAY (SAVE.CURSOR)))
          (SETQ \ARROW (INSERTLINE \ARROW USECR))
          (COND
             ((NOT STAY)
              (SAVE.CURSOR)))
          (replace (LINE END) of \ARROW with OLDEND)
          [COND
             [(EQ \CURSOR OLDEND)                                 ; cr was inserted at end of line.  Maybe this never happens
              (replace (LINE END) of \ARROW with (SETQ \CURSOR (CDR OLDEND]
             (T (TYPE.BUFFER (SETQ \CURSOR (fetch (LINE START) of \ARROW))
                             OLDEND)                              ; Restore to screen what we erased above
                (replace (LINE LASTCOL) of \ARROW with \CURSORCOL)
                (COND
                   ((OVERFLOWLINE? \ARROW)
                    ;; the previous line overflowed, but when we inserted a cr we added more space on the line, so go fix it up
                    (ADJUSTLINE]
```

```
          [COND
             (STAY                                          ; Oh well, undo what we did to poor \CURSOR
                    (SETQ \CURSOR (fetch (LINE END) of (SETQ \ARROW OLDLINE]
          (RESTORE.CURSOR])
```

(BUFTAILP

```
  [LAMBDA (TAIL START END)                                  (* bvm%: "23-JUN-81 15:48")
    (do (COND
           ((EQ TAIL START)
            (RETURN TAIL))
           ((OR (NOT START)
                (EQ START END))
            (RETURN)))
        (SETQ START (CDR START])
```

(CHECK.MARGIN

```
  [LAMBDA (BUF LINE)                                        ; Edited 24-May-91 10:33 by jds
```

;;; If BUF is the pseudo-cr at the end of this LINE, then back it up one, since you can't let the cursor sit on it

```
    (COND
       ((AND (EQ (fetch (LINE END) of LINE)
                 BUF)
             (OR (EQ (fetch (LINE LASTCOL) of LINE)
                     \RMARG)
                 (EQ (fetch (LINE START) of (fetch (LINE NEXTLINE) of LINE))
                     BUF)))
        (TTNLEFT BUF 1 (fetch (LINE START) of LINE)))
       (T BUF])
```

(CLEAR.LINE?

```
  [LAMBDA (FLG)                                             (* lmm "20-Nov-86 00:27")
                                                            ; If FLG true, erase lots
    (COND
       (FLG (ERASE.TO.END.OF.PAGE))
       (T (ERASE.TO.END.OF.LINE])
```

(CURRENT.WORD

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:34 by jds
```

;; Used by word-completion routines. Returns position in buffer of the start of the current word, or NIL if no word is in progress, or \COMMAND is
;; true and this is not the first word, or the line is a comment. Definition of 'word' here is different from that of WORDSEPRP since we want only
;; words with respect to the reader, not with respect to text

```
    (COND
       ((AND (NOT (AT.START.OF.LINE))
             (NEQ (CAR (fetch (LINE START) of \ARROW))
                  TTYINCOMMENTCHAR))
        (for (X _ (fetch (LINE START) of \ARROW)) by (TTNEXTCHAR X) until (EQ X \CURSOR)
           bind (NEW _ T)
                SNX
           do                                               ; NEW is true after we scan a break character
               (SETQ SNX (\SYNCODE \RDTBLSA (FIRSTCHAR X)))
               (COND
                  ((COND
                      (NEW                                  ; Most ANY funny character at start of word considered sepr
                           (SELECTC SNX
                                ((LIST OTHER.RC ESCAPE.RC MULTIPLE-ESCAPE.RC)
                                                            ; Looks like good start of word
                                    NIL)
                                T))
                      (T                                    ; If in middle of word, only 'terminating macros' stop word
                         (fetch STOPATOM of SNX)))
                   (SETQ NEW T))
                  (NEW                                      ; This is the start of a new word;  note it
                       (COND
                          ((AND $$VAL \COMMAND)             ; Means this is second word
                           (RETURN)))
                       (SETQ $$VAL X)
                       (SETQ NEW NIL)))
           finally (RETURN (AND (NOT NEW)
                                $$VAL])
```

(DELETE.TO.END

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:34 by jds
```

;;; Kills buffer from \CURSOR onward

```
    (SETTAIL? T)
    (COND
       (DISPLAYTERMFLG (ERASE.TO.END.OF.PAGE)))
    (COND
       ((fetch (LINE NEXTLINE) of \ARROW)                   ; There are lines after this, so return them to garbage heap
```

```
        (KILL.LINES (fetch (LINE NEXTLINE) of \ARROW))
        (replace (LINE NEXTLINE) of \ARROW with NIL)))
    (replace (LINE END) of \ARROW with (SETQ \ENDBUFFER \CURSOR))
    (replace (LINE LASTCOL) of \ARROW with \CURSORCOL])
```

(DELETELINE

```
  [LAMBDA (LINE EMPTYLINE?)                                   ; Edited 24-May-91 10:34 by jds
```

;; Deletes this LINE from buffer and screen; assumes cursor is currently positioned somewhere on the line. EMPTYLINE? is true on calls from ;; ADJUSTLINE where the line is naked and hence no text in the buffer needs to be killed.

```
    (PROG ((NEXTLINE (fetch (LINE NEXTLINE) of LINE))
           OLDSTART NEWSTART PREVLINE)
          [COND
             ((AND (EQ LINE \ARROW)
                   (ON.FIRST.LINE))
              (COND
                 ((NOT NEXTLINE)                              ; Can't delete the only line
                  (RETURN (BEEP)))
                 ((NEQ \PROMPT1 \PROMPT2)                     ; tricky to delete first line, since the correct prompt should be
                                                              ; displayed
                  (MOVE.BACK.TO \BUFFER)
                  (RETURN (FORWARD.DELETE.TO (fetch (LINE END) of \ARROW]
          (COND
             (DISPLAYTERMFLG (DO.DELETE.LINES 1)))
          (RENUMBER.LINES NEXTLINE (fetch (LINE ROW) of LINE))
          (replace (LINE NEXTLINE) of (SETQ PREVLINE (PREVLINE LINE 1)) with NEXTLINE)
          [COND
             ((NOT NEXTLINE)                                  ; deleting last line: need to worry about \ENDBUFFER and such
              (SETQ \ENDBUFFER (fetch (LINE END) of PREVLINE)))
             (T (replace (LINE NEXTLINE) of LINE with NIL)    ; in preparation for KILL.LINES below
                (COND
                   ((NOT EMPTYLINE?)
                    (KILLSEGMENT (SETQ OLDSTART (fetch (LINE START) of LINE))
                           (SETQ NEWSTART (fetch (LINE START) of NEXTLINE)))
                                                              ; flush anything on the line.  PREVLINE pointers remain valid
                    (COND
                       ((EQ (fetch (LINE END) of NEXTLINE)
                            NEWSTART)
                        (replace (LINE END) of NEXTLINE with OLDSTART)))
                    (replace (LINE START) of NEXTLINE with OLDSTART]
          (KILL.LINES LINE)                                   ; return to heap
          (COND
             ((EQ \ARROW LINE)                                ; if this is our home position, adjust appropriately
              (SETQ \ARROW (SETQ LINE (OR NEXTLINE PREVLINE)))
              (SETQ \CURSOR (fetch (LINE START) of LINE))
              (GO.TO.RELATIVE 'LINE LINE])
```

(DELETETO

```
  [LAMBDA (TAIL)                                              ; Edited 24-May-91 10:34 by jds
    (SETTAIL?)
    (COND
       ((NEQ \CURSOR \ENDBUFFER)                              ; On other terminals also when Cursor capable
        (BACKWARD.DELETE.TO TAIL))
       (T [COND
             [(NOT DISPLAYTERMFLG)
              (COND
                 ((NOT \DELETING)                             ; prefix deletions with backslash
                  (COND
                     ((NOT TTYINBSFLG)                        ; unless we are going to physically backspace
                      (TTBOUT \)))
                  (SETQ \DELETING 0)))
              (DELETETO1 TAIL)
              (COND
                 ((EQ TAIL \BUFFER)
                  (END.DELETE.MODE]
             (T (PROG ((N (SEGMENT.BIT.LENGTH TAIL \ENDBUFFER)))
                                                              ; need to kill the previous N chars
                 ;; (COND ((CAPABILITY? ERASE.TO.END T) (* Ah, all we need do is go back N and erase to end) (DO.BACK N)
                 ;; (ERASE.TO.END.OF.LINE)) (T (* laborious technique for glass ttys: go back and wipe out each char one at a time) (FRPTQ N
                 ;; (PROGN (DO.BACK 1) (* back up) (TTBOUT SPACE) (* overwrite with space) (DO.BACK 1) (* and back up again)))))
                      (DSPBACKUP N \DSP)
                      (SETQ \CURSORCOL (IDIFFERENCE \CURSORCOL N]
          (replace (LINE END) of \ARROW with (SETQ \CURSOR (SETQ \ENDBUFFER TAIL)))
          (replace (LINE LASTCOL) of \ARROW with \CURSORCOL])
```

(DELETETO1

```
  [LAMBDA (TAIL)                                              ; Edited 24-May-91 11:09 by jds
```

;;; Not used in Interlisp-D

;; on non-DMs: delete chars until we reach TAIL; since we echo deleted chars in reverse order, this is most easily done recursively

```
    [COND
       ((NEQ (CDR TAIL)
```

```
            \ENDBUFFER)
        (DELETETO1 (CDR TAIL]
    (for CH inside (COND
                     ((COMPLEXCHARP (CAR TAIL))
                      (fetch (COMPLEXCHAR CPXPRINTCHARS) of (CAR TAIL)))
                     (T (CAR TAIL)))
       do (SELECTQ TTYINBSFLG
                (NIL (TTBOUT CH))
                (LF ;; physically backspace, crossing out character.  LF means we will do a LF when ENDELETE happens.  If we don't LF, then
                    ;; best not to cross out chars
                    (TTBOUT BS \ BS)
                    (ADD1VAR \DELETING))
                (TTBOUT BS)))                                    ; echo deleted char
    (SETQ \CURSORCOL (SUB1 \CURSORCOL])
```

(DO.EDIT.COMMAND

```
  [LAMBDA (CHAR EDITARG)                                         ; Edited 24-May-91 10:40 by jds
```

;;; Handles the various edit commands, which mostly move the cursor around in the buffer, or kill pieces of it. CHAR is the character stripped of its ;;; editbit. EDITARG is the argument, if any (not set by type-in, but by program asking for a particular edit function). If this routine returns something, it ;;; means process it like ordinary character (this is how we can invoke non-editbit routines)

```
    (PROG (EDITMINUS L X LASTSKIP)
          [COND
             ((NOT EDITARG)
              (SETQ EDITARG 1))
             ((MINUSP EDITARG)
              (SETQ EDITMINUS T)
              (SETQ EDITARG (IMINUS EDITARG]
      LP  [SELCHARQ (SETQ CHAR (U-CASECODE CHAR))
                (CR ;; <edit>CR on empty buffer means get back last buffer;  in the middle of a buffer it is the same as normal CR, but also ends
                    ;; insert mode
                    [COND
                       ((EMPTY.BUFFER)
                        (RESTOREBUF))
                       ((ON.LAST.LINE)
                        (RETURN CHAR))
                       (T (MOVE.TO.LINE (TTNEXTLINE \ARROW EDITARG])
                ((SPACE >)                                       ; move right
                     [COND
                        (EDITMINUS (SETQ CHAR (CHARCODE DEL))    ; backward space is delete
                               (GO NOMINUS))
                        ((AT.END.OF.BUF)
                         (BEEP))
                        ((AT.END.OF.SCREEN)
                         (MOVE.TO.NEXT.LINE))
                        (T (MOVE.FORWARD.TO (TTNTH \CURSOR EDITARG])
                ((DEL ^A BS <)                                   ; back up
                     [COND
                        (EDITMINUS (SETQ CHAR (CHARCODE SPACE))  ; backward delete is space
                               (GO NOMINUS))
                        ((AT.START.OF.BUF)
                         (BEEP))
                        ((AT.START.OF.LINE)
                         (MOVE.TO.LINE (SETQ X (PREVLINE \ARROW 1))
                                (fetch (LINE END) of X)))
                        (T (MOVE.BACK.TO (TTNLEFT \CURSOR EDITARG])
                (%(                                              ; backs up one word
                    [COND
                       (EDITMINUS (SETQ CHAR (CHARCODE %)))
                              (GO NOMINUS))
                       (T (MOVE.BACK.TO (PREVWORD \CURSOR EDITARG])
                (%)                                              ; moves ahead one word
                    [COND
                       (EDITMINUS (SETQ CHAR (CHARCODE %()))
                             (GO NOMINUS))
                       ((AT.END.OF.SCREEN)
                        (BEEP))
                       (T (MOVE.FORWARD.TO (FIND.NEXT.WORD \CURSOR EDITARG])
                (TAB                                             ; go to end of line
                     (MOVE.TO.LINE (SETQ X (TTNEXTLINE \ARROW (SUB1VAR EDITARG)))
                           (fetch (LINE END) of X)))
                (^L                                              ; go to start of line
                    (MOVE.TO.LINE (PREVLINE \ARROW (SUB1VAR EDITARG))))
                ({                                               ; { goes to start of buffer, like infinite FF
                   (MOVE.TO.LINE \FIRSTLINE))
                (}                                               ; } goes to end of buffer, like infinite TAB
                   (MOVE.TO.LINE (SETQ X (TTLASTLINE))
                          (fetch (LINE END) of X)))
                (LF                                              ; moves down
                    [COND
                       (EDITMINUS (SETQ CHAR (CHARCODE ^))
                             (GO NOMINUS))
                       [(ON.LAST.LINE)
```

```
            (COND
               ((EMPTY.BUFFER)                          ; Treat this the same as regular linefeed, i.e. restore buffer
                (RETURN (CHARCODE LF)))
               (T (BEEP]
            (T (MOVE.TO.LINE (SETQ X (TTNEXTLINE \ARROW EDITARG))
                     (NTH.COLUMN.OF X (EDITCOLUMN])
(^                                                      ; moves up
   [COND
      (EDITMINUS (SETQ CHAR (CHARCODE LF))
            (GO NOMINUS))
      ((ON.FIRST.LINE)
       (BEEP))
      (T (MOVE.TO.LINE (SETQ X (PREVLINE \ARROW (IMIN (IPLUS \LOC.ROW.0 \CURSORROW)
                                                      EDITARG)))
                 (NTH.COLUMN.OF X (EDITCOLUMN])
(K                                                      ; kills one char
   [COND
      ((AT.END.OF.LINE)
       (BEEP))
      (T (FORWARD.DELETE.TO (TTNTH \CURSOR EDITARG])
((S Z B)                                                ; various skip or zap commands
     (SKIP/ZAP CHAR (TTBIN T)
           EDITARG EDITMINUS))
(A                                                      ; repeat last S or Z
   (COND
      ((SETQ LASTSKIP (fetch (TTYINBUFFER LASTSKIP) of \TTYINSTATE))
       (SKIP/ZAP LASTSKIP (fetch (TTYINBUFFER LASTSKIPCHAR) of \TTYINSTATE)
             EDITARG EDITMINUS))
      (T (BEEP))))
(L                                                      ; lowercase word
   (U/L-CASE EDITARG))
(U                                                      ; uppercase word
   (U/L-CASE EDITARG T))
(C                                                      ; capitalize word
   (U/L-CASE EDITARG 1))
(G                                                      ; grab a copy of Nth previous line
   (COND
      ((OR (ON.FIRST.LINE)
           (NOT (AT.END.OF.LINE))
           (EQ (SETQ X (NTH.COLUMN.OF (SETQ L (PREVLINE \ARROW EDITARG))
                                 \CURSORCOL))
               (fetch (LINE END) of L)))                ; nothing to copy
       (BEEP))
      (T (READFROMBUF X (fetch (LINE END) of L)
                T))))
(%]                                                     ; Move to end of current expression
    [COND
       ((AT.END.OF.BUF)
        (BEEP))
       (T (MOVE.TO.WHEREVER (OR (TTSKREAD (TTNEXTCHAR \CURSOR))
                                \ENDBUFFER])
(%[                                                     ; Move to start of current list expression
    [COND
       ((AT.START.OF.BUF)
        (BEEP))
       (T (MOVE.TO.WHEREVER (BACKSKREAD \CURSOR])
(^W                                                     ; delete back to start of current word
    (TTDELETEWORD EDITARG))
(D                                                      ; Delete forward to end of word
   (COND
      ((AT.END.OF.LINE)
       (BEEP))
      (T (COND
            ((AND (NEQ (SETQ X (FIND.NEXT.WORD \CURSOR EDITARG T))
                       (fetch (LINE END) of \ARROW))
                  (NOT (AT.START.OF.LINE))
                  [NOT (WORDSEPRP (FIRSTCHAR (TTNLEFT \CURSOR 1]
                  [SPACEP (FIRSTCHAR (SETQ L (TTNLEFT X 1 \CURSOR]
                  (NEQ L \CURSOR))
             ;; Don't want to delete all the way to start of new word, since we'd like a little space in between.  Simulating
             ;; EMACS would probably be easier if we just made FIND.NEXT.WORD stop at the intervening spaces rather than
             ;; at the end
             (SETQ X L)))
         (FORWARD.DELETE.TO X))))
((^Q ^U)                                                ; Delete line;  ^U for tops20 folk
     (COND
        ((EQ EDITARG 1000)
         (DELETE.TO.END))
        (T (DELETELINE \ARROW))))
(^Y                                                     ; gets userexec
    (COND
       ((AND (EQ EDITARG 1000)
             (NEQ \CURSOR \ENDBUFFER))
        (TTUNREADBUF)                                   ; Stuff what's ahead of cursor into input buffer

       ))
```

```
     (RETURN CHAR))
(F                                                    ; accept tvedit's $$F to finish
   (COND
      [(EQ EDITARG 1000)
       (MOVE.TO.WHEREVER \ENDBUFFER)
       (COND
          ((NEQ \CURSOR \ENDBUFFER)                   ; This is because the cursor mover refuses to put me in column
                                                      ; 80 of a line, due to certain anomalies
           (add \CURSORCOL (SEGMENT.BIT.LENGTH \CURSOR \ENDBUFFER))
           (SETQ \CURSOR \ENDBUFFER)
           (OVERFLOW? 0)))
       (RETURN (COND
                  (\REPEAT                            ; End with ^Z
                         (CHARCODE ^Z))
                  (\READING
                         ;; End read with ']' ;  of course, this doesn't always 'finish' , but it's simple enough to remember
                         ;; what this is
                         (CHARCODE %]))
                  (T (CHARCODE CR]
      (T (BEEP))))
(J                                                    ; Justify/fill line
   (ADJUSTLINE.AND.RESTORE EDITARG))
(-                                                    ; minus sign negates arg
   (SETQ EDITARG 0)
   (SETQ EDITMINUS T)
   (GO DONUMBERS))
(ESCAPE                                               ; ESCAPE may modify next command
        [COND
           ((AND (EQ EDITARG 1000)
                 (EQ EDITPREFIXCHAR (CHARCODE ESCAPE)))
            ;; 3 escapes in a row is the way to type a regular Escape when Escape is the edit prefix.  Better ways might be
            ;; forthcoming
            (RETURN (CHARCODE ESCAPE]
        (SETQ EDITARG 1000)                           ; 1000 is an adequate infinity for these purposes
        (SETQ EDITMINUS)
        (SETQ CHAR (TTBIN T))
        (GO LP))
((N ^R)                                               ; refresh n lines, or whole buffer for $$N
     [COND
        ((EQ EDITARG 1000)
         (RETYPE.BUFFER \FIRSTLINE T))
        (EDITMINUS (RETYPE.BUFFER (PREVLINE \ARROW EDITARG)
                                  \ARROW))
        (T (RETYPE.BUFFER \ARROW (TTNEXTLINE \ARROW EDITARG])
(T ;; transpose chars.  If at end of line, do preceding two, else do the ones before and after the cursor.
   [SETQ L (TTNLEFT \CURSOR (SETQ X (COND
                                      ((AT.END.OF.LINE)
                                       2)
                                      (T 1]      ; start of swap
   [COND
      ((OR (EQ L \CURSOR)
           (COMPLEXCHARP (CAR L))
           (AND (EQ X 2)
                (EQ (CDR L)
                    \CURSOR))
           (COMPLEXCHARP (CADR L)))               ; Complain if not enough chars to swap, or one of them is a
                                                  ; funny multiple char (I'm lazy)
       (BEEP))
      (T [GO.TO.RELATIVE (IDIFFERENCE \CURSORCOL (SEGMENT.BIT.LENGTH L (NTH L (ADD1 X]
                                                  ; Back up to start of segment
         [FRPLACA L (PROG1 (CADR L)
                           (FRPLACA (CDR L)
                                    (CAR L)))]    ; Do the swap in the buffer
         (TYPE.BUFFER L (CDDR L))                 ; Fix the display
         (COND
            ((EQ X 1)                             ; Were between two chars, so get back there
             (GO.TO.RELATIVE (IDIFFERENCE \CURSORCOL (TTBITWIDTH (FIRSTCHAR (CDR L])
(O                                                ; Open line, i.e.  insert <cr> but stay here
   (BREAKLINE EOLCHARCODE T))
(_ ;; Special hack: says to add the word before the cursor to USERWORDS, so I can use altmode completion on it
   (COND
      [(AND TTYINCOMPLETEFLG (SETQ X (CURRENT.WORD))
            [SETQ X (PROG ((\BUFFER X))
                          (RETURN (TTRATOM]
            (LITATOM X))
       (COND
          ((EQ EDITARG 0)                         ; Means to remove! I don't know if there's an 'official' way to do
                                                  ; this
           (DREMOVE X USERWORDS))
          (T (ADDSPELL X 0]
      (T (BEEP))))
(P (DO.EDIT.PP))
(COND
```

```
                ((SETQ CHAR (EDITNUMBERP CHAR))
                 (SETQ EDITARG CHAR)
                 (GO DONUMBERS))
                (T (BEEP]
          (SETQ \LASTCHAR CHAR)
          (RETURN)
      NOMINUS
          (SETQ EDITMINUS)
          (GO LP)
      DONUMBERS
```

;; scanning a numeric arg. EDITARG is its magnitude; EDITMINUS set if negative. <edit>escape is treated as 1000, which is probably big
;; enough. Doesn't matter if any of the next chars has edit bit on, since once we start a number, any other digits must be part of it, since numbers
;; aren't themselves commands

```
          (COND
             ([SETQ X (EDITNUMBERP (SETQ CHAR (TTBIN T]
              [SETQ EDITARG (COND
                                ((IGREATERP EDITARG 100)          ; Limit numeric args to 1000 so small number stuff works
                                 1000)
                                (T (IPLUS (ITIMES EDITARG 10)
                                          X]
              (GO DONUMBERS)))
          (COND
             ((AND EDITMINUS (EQ EDITARG 0))                      ; Happens if we get a '-' followed by no number
              (SETQ EDITARG 1)))
          (GO LP])
```

(DO.EDIT.PP

```
  [LAMBDA NIL                                                     ; Edited 24-May-91 10:34 by jds
    (COND
       ((NOT \READING)                                            ; Nothing to prettyprint--just redisplay
        (RETYPE.BUFFER \FIRSTLINE T))
       (T                                                         ; Read what we have, supplying closing parens if suitable, and
                                                                  ; then prettyprint it
          (WITH-RESOURCES (TTSCRATCHFILE)
                 (PROG ((*READTABLE* RDTBL)
                        (\BUFFER \BUFFER)
                        LEFTOVER EXPRS)
                       [COND
                          ((TTYIN.BALANCE NIL T)                  ; Input is now perfectly balanced

                           )
                          ((NEQ \CURSOR \ENDBUFFER)               ; There was extra stuff at end
                           (SETQ LEFTOVER (COPY.SEGMENT \CURSOR \ENDBUFFER))
                           (SETQ \ENDBUFFER \CURSOR))
                          (T                                      ; Didn't balance, so punt
                             (RETURN (BEEP]
                       (SETQ \CURSOR \BUFFER)
                       [COND
                          ((NEQ (TTSKIPSEPR)
                                \ENDBUFFER)
                           (SETQ EXPRS (TTYIN.READ 0 NIL TTSCRATCHFILE]
                       (MOVE.TO.LINE \FIRSTLINE)
                       (ERASE.TO.END.OF.PAGE)
                       (TTYIN1RESTART)
                       (replace (LINE FIRSTCOL) of \ARROW with (replace (LINE LASTCOL) of \ARROW with \CURSORCOL))
                       [COND
                          (EXPRS (TTLOADBUF (LIST HISTSTR1 (TTYIN.PPTOFILE EXPRS NIL NIL TTSCRATCHFILE]
                       (SETFILEPTR TTSCRATCHFILE 0)               ; Leave it nice for next customer
                       (COND
                          (LEFTOVER                               ; Display the stuff that follows the normal read termination
                                 (BREAKLINE (CHARCODE EOL))
                                 (READFROMBUF LEFTOVER])
```

(TTDOTABS

```
  [LAMBDA (TABS)                                                  (* bvm%: "16-Apr-85 17:35")
```

;;; Tab to next tabstop in TABS, if any. Represent pseudotabs as a complex space. Return T if anything done

```
    (COND
       ((AND TABS (AT.END.OF.BUF))
        (for TB in TABS bind SPACES when (AND (SMALLP TB)
                                              (IGREATERP (SETQ SPACES (IDIFFERENCE (ITIMES TB \CHARWIDTH)
                                                                                   \CURSORCOL))
                                                         \CHARWIDTH))
         do                                                       ; Make pseudo-tab and echo as spaces
            [ADDCHAR (TTMAKECOMPLEXCHAR (CHARCODE SPACE)
                                (to (IQUOTIENT SPACES \CHARWIDTH) collect (CHARCODE SPACE]
            (RETURN T])
```

(EDITCOLUMN

```
  [LAMBDA NIL                                                     (* bvm%: "24-AUG-81 23:17")
```

;; If last edit command moved up/down, then return the same column we were using then; else use current cursor column, and record it as the

```
    ;; 'goal' column for any future such commands
    (OR (SELCHARQ \LASTCHAR
              ((LF ^)
                  \HOMECOL)
               NIL)
        (SETQ \HOMECOL \CURSORCOL])
```

(EDITNUMBERP

```
  [LAMBDA (CHAR)                                                    (* bvm%: "11-MAR-81 22:05")
    (AND [NOT (MINUSP (SETQ CHAR (IDIFFERENCE CHAR (CONSTANT (CHCON1 0]
         (NOT (IGREATERP CHAR 9))
         CHAR])
```

(END.DELETE.MODE

```
  [LAMBDA NIL                                                       (* bvm%: "19-MAR-81 11:59")
    (COND
       (\DELETING (SELECTQ TTYINBSFLG
                      (NIL (TTBOUT \))
                      (LF (COND
                             ((IGREATERP \DELETING 1)               ; if more than one char x'd out, lf to new line
                              (DO.LF))))
                      NIL)
             (SETQ \DELETING NIL])
```

(ENDREAD?

```
  [LAMBDA NIL                                                       (* bvm%: "10-Apr-86 14:21")
    ;; Return true if the paren/bracket just typed terminates the input.  It does if the right paren (or even one earlier in buffer) is in excess, i.e
    ;; unbalanced, or just balances and this is the only list on the line, or we are doing a LISPX input and the input is in EVALQT form, with no space
    ;; after the first atom
    (LET
     (X)
     (AND
      (AT.END.OF.TEXT \CURSOR)
      (SETQ X (TTSKREAD \BUFFER))
      (OR
       (NEQ X \ENDBUFFER)
       (AND
        [SELCHARQ (CAR (SETQ X (FIND.NON.SPACE \BUFFER)))
             ((%( %[)                                               ; OK, line started with paren/bracket
              T)
             (AND (EQ \READING 'EVALQT)
                  (NEQ \PROMPT1 '*)
                  (while (NEQ X \ENDBUFFER) bind ESCAPED
                     do ;; Skip over this first atom, to see if input is in EVALQT form.  Prompt check is so we don't do this in the editor
                        (SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR X))
                             (SEPRCHAR.RC                           ; Space, etc: probably wants more on line
                                          (OR ESCAPED (RETURN NIL)))
                             ((LIST LEFTPAREN.RC LEFTBRACKET.RC)
                                                                    ; Open paren/bracket: looks good
                                  [OR ESCAPED (RETURN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN (PROGN
                                                                    ; Prettyprint sucks again!
                                                                                                    T])
                             (ESCAPE.RC                             ; Skip over escape char
                                        (SETQ X (CDR X)))
                             (MULTIPLE-ESCAPE.RC                    ; Multiple escape
                                  (SETQ ESCAPED (NOT ESCAPED)))
                             NIL)
                        (SETQ X (TTNEXTCHAR X]
        (EQ (CDR (TTSKREAD (CDR X)))
            \CURSOR])
```

(FIND.LINE

```
  [LAMBDA (BUF)
    (DECLARE (USEDFREE \FIRSTLINE))                                 ; Edited 24-May-91 10:34 by jds
```

;;; Returns the buffer LINE on which BUF, a cursor position, occurs

```
    (for (LINE _ (PROGN \FIRSTLINE)) do (COND
                                           [(EQ BUF (fetch (LINE END) of LINE))
                                                                    ; Check this separately so next BUFTAILP doesn't catch it
                                            (RETURN (COND
                                                       ((OVERFLOWLINE? LINE)
                                                        (fetch (LINE NEXTLINE) of LINE))
                                                       (T LINE]
                                           ((BUFTAILP BUF (fetch (LINE START) of LINE)
                                                      (fetch (LINE END) of LINE))
                                            (RETURN LINE)))
                                          (OR (SETQ LINE (fetch (LINE NEXTLINE) of LINE))
                                              (SHOULDNT])
```

(FIND.LINE.BREAK

```
  [LAMBDA (START END USELAST)                                    (* bvm%: "20-FEB-82 22:35")
```

;;; Locates a place between START and END where line can be broken. If USELAST is true, returns last such place, else first

```
    (while (NEQ START END) do [COND
                                  ((EQ (CAR START)
                                       (CHARCODE SPACE))
                                   (COND
                                      (USELAST (SETQ $$VAL START))
                                      (T (RETURN START]
                                (SETQ START (TTNEXTCHAR START])
```

(FIND.MATCHING.QUOTE

```
  [LAMBDA (BUF END)                                              (* bvm%: "16-Apr-86 15:07")
```

;;; Searches BUF until END for a closing double-quote

```
    (while (NEQ BUF END) do (SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR BUF))
                                (STRINGDELIM.RC
                                     (RETURN BUF))
                                (ESCAPE.RC                       ; '%' quotes next char
                                           (COND
                                              ((EQ (SETQ BUF (CDR BUF))
                                                   END)
                                               (RETURN))))
                                NIL)
                            (SETQ BUF (CDR BUF])
```

(FIND.NEXT.WORD

```
  [LAMBDA (BUFTAIL N BACKUPFLG)                                  ; Edited 24-May-91 10:34 by jds
```

;;; Return start of Nth word after BUFTAIL, or end of line if none. BACKUPFLG means if you cross a paren getting to the Nth word, return the paren
;;; rather than the word (used for smart word-delete)

```
    (PROG ((END (fetch (LINE END) of \ARROW)))
          (COND
             ((EQ BUFTAIL END)
              (RETURN END)))
          (SETQ BUFTAIL (CDR BUFTAIL))
      LP  [COND
             ((EQ BUFTAIL END)
              (RETURN END))
             ((WORDSEPRP (FIRSTCHAR BUFTAIL))                    ; Found a space. Now scan for first non-space, and return there
              [COND
                 (BACKUPFLG (SETQ BUFTAIL (SETQ BACKUPFLG (FIND.START.OF.WORD BUFTAIL END]
              (while (AND (NEQ BUFTAIL END)
                          (WORDSEPRP (FIRSTCHAR BUFTAIL)))
                 do (SETQ BUFTAIL (TTNEXTCHAR BUFTAIL)))
              (COND
                 ((OR (NOT N)
                      (EQ (SUB1VAR N)
                          0)
                      (EQ BUFTAIL END))
                  (RETURN (OR BACKUPFLG BUFTAIL]
          (SETQ BUFTAIL (TTNEXTCHAR BUFTAIL))
          (GO LP])
```

(FIND.NON.SPACE

```
  [LAMBDA (BUF END)                                              (* bvm%: "11-Apr-85 15:07")
    (OR END (SETQ END \ENDBUFFER))
    (while (AND (NEQ BUF END)
                (SPACEP (FIRSTCHAR BUF)))
       do (SETQ BUF (TTNEXTCHAR BUF)))
    BUF])
```

(FIND.START.OF.WORD

```
  [LAMBDA (BUF END)                                              (* bvm%: "11-Apr-85 15:07")
```

;;; Returns position of first word, i.e. non-space, in BUF before END

```
    (OR END (SETQ END \ENDBUFFER))
    (while (AND (NEQ BUF END)
                (WORDSEPRP (FIRSTCHAR BUF)))
       do (SETQ BUF (TTNEXTCHAR BUF)))
    BUF])
```

(FORWARD.DELETE.TO

```
  [LAMBDA (BUFTAIL)                                              ; Edited 24-May-91 10:34 by jds
```

;;; Delete from \CURSOR to BUFTAIL. Cursor does not move

```
    [COND
       ((EQ BUFTAIL \CURSOR)                                    ; Nothing to do

        )
       ((EQ BUFTAIL \ENDBUFFER)                                 ; deleting to end is simple
        (ERASE.TO.END.OF.LINE)
        (replace (LINE END) of \ARROW with (SETQ \ENDBUFFER \CURSOR))
        (replace (LINE LASTCOL) of \ARROW with \CURSORCOL))
       (T (PROG ((DELETEDWIDTH (SEGMENT.BIT.LENGTH \CURSOR BUFTAIL))
                 L)
                (COND
                   ((EQ BUFTAIL (fetch (LINE END) of \ARROW))   ; End pointer is about to disappear into free list, so move it back
                                                                ; here
                    (replace (LINE END) of \ARROW with \CURSOR)
                    [COND
                       ((EQ (fetch (LINE START) of (SETQ L (fetch (LINE NEXTLINE) of \ARROW)))
                            BUFTAIL)
                        (replace (LINE START) of L with \CURSOR)
                        (COND
                           ((EQ (fetch (LINE END) of L)
                                BUFTAIL)
                            (replace (LINE END) of L with \CURSOR]
                    (ERASE.TO.END.OF.LINE))
                   (T (TTDELSECTION DELETEDWIDTH)))
                (KILLSEGMENT \CURSOR BUFTAIL)
                (replace (LINE LASTCOL) of \ARROW with (IDIFFERENCE (fetch (LINE LASTCOL) of \ARROW)
                                                                    DELETEDWIDTH))
                (COND
                   ((OVERFLOWLINE? \ARROW)
                    (ADJUSTLINE.AND.RESTORE]
    \CURSOR])
```

(GO.TO.ADDRESSING

```
  [LAMBDA (COL ROW)                                             (* bvm%: "20-Mar-84 14:50")
                                                                ; Regardless of where we are now, go to logical position
                                                                ; COL,ROW using cursor addressing
    (PROG ((ABSROW (IPLUS \LOC.ROW.0 ROW)))
          (TTSETCURSOR COL ABSROW)
     ;; Used to prohibit going above top, but that is ugly.  Better to go up there and be clipped out of existence by the display code.  Formerly: (COND
     ;; ((ILESSP ABSROW 0) (* trying to go beyond top of screen;  ideally we should scroll, but for now just forbid it) (SETQ ROW (IDIFFERENCE
     ;; ROW ABSROW)) 0) ((NOT (ILESSP ABSROW \TTPAGELENGTH)) (* This shouldn't happen at all until we can scroll!) (SETQ ROW (IPLUS
     ;; (IDIFFERENCE ROW ABSROW) \TTPAGELENGTH -1)) (SUB1 \TTPAGELENGTH)) (T ABSROW))
          (SETQ \CURSORROW ROW)
          (SETQ \CURSORCOL COL])
```

(GO.TO.FREELINE

```
  [LAMBDA NIL                                                   ; Edited 24-May-91 10:34 by jds
```

;;; Moves cursor to the first free line after the buffer, and clears it

```
    (GO.TO.RELATIVE NIL (fetch (LINE ROW) of (TTLASTLINE)))     ; Put the cursor on the last row of buffer
    (TTCRLF)                                                    ; And down one more
    (ERASE.TO.END.OF.PAGE])
```

(GO.TO.RELATIVE

```
  [LAMBDA (COL ROW)                                             ; Edited 24-May-91 10:34 by jds
```

;;; Moves cursor to indicated row/col. ROW arg may be omitted if the movement is on the same row. If COL=LINE then ROW is interpreted as a LINE
;;; record, and destination is the start of that line

```
    (COND
       ((EQ COL 'LINE)
        (SETQ COL (fetch (LINE FIRSTCOL) of ROW))
        (SETQ ROW (fetch (LINE ROW) of ROW)))
       ((NOT COL)
        (SETQ COL \CURSORCOL))
       ((NOT ROW)
        (SETQ ROW \CURSORROW)))
    (MOVETO COL (+ (TIMES (SUB1 (- \TTPAGELENGTH (+ \LOC.ROW.0 ROW)))
                          \CHARHEIGHT)
                   \BMARG)
          \DSP)
    (SETQ \CURSORROW ROW)
    (SETQ \CURSORCOL COL])
```

(INIT.CURSOR

```
  [LAMBDA (COL)                                                 ; Edited 18-Jan-88 15:12 by bvm
```

;;; Initializes cursor accounting; in Interlisp-10, this assumed/forced the cursor to be in column COL of the bottom row of the screen

```
    (PROG ((YBOT (fetch (REGION BOTTOM) of (DSPCLIPPINGREGION NIL \DSP)))
           INITY)
```

```
          (SETQ INITY (- (DSPYPOSITION NIL \DSP)
                         YBOT))
          (SETQ \LOC.ROW.0 (- \TTPAGELENGTH (IQUOTIENT INITY \CHARHEIGHT)
                              1))
```

;; \LOC.ROW.0 is the number of the 'line' of the first line of text, counting from the top of the window. Instead, we really should count from the
;; bottom and fix everyone who cares

```
          (SETQ \BMARG (+ YBOT (IREMAINDER INITY \CHARHEIGHT)))
          (SETQ \CURSORROW 0)
          (SETQ \CURSORCOL COL])
```

(INSERT.NODE

```
  [LAMBDA (BUF)                                                    (* bvm%: "20-FEB-82 22:34")
```

;;; Effectively does (ATTACH garbage BUF), but reuses from the garbage heap

```
    (COND
       ((EQ BUF \ENDBUFFER)                                        ; Already at end, just push pointer
        (SETQ \ENDBUFFER (TTNEXTNODE \ENDBUFFER)))
       (T (FRPLACD BUF (FRPLNODE2 (SCRATCHCONS)
                                  BUF])
```

(INSERTLINE

```
  [LAMBDA (OLDLINE USECR)                                          ; Edited 24-May-91 10:34 by jds
```

;; Inserts a new line between OLDLINE and the next line, whose START is the END of LINE; caller must fill in END if line is non-empty (defaults to
;; start); USECR, if supplied, is the <cr> char to end the previous line with

```
    (PROG ((OLDEND (fetch (LINE END) of OLDLINE))
           (ROW (ADD1 (fetch (LINE ROW) of OLDLINE)))
           X NEWLINE)
          [COND
             (USECR (INSERT.NODE OLDEND)
                    (FRPLACA OLDEND USECR)
                    (SETQ OLDEND (CDR OLDEND]
          (TTCRLF)
          (COND
             ((NEQ OLDEND \ENDBUFFER)                              ; Not last line, so insert a line on screen.
              (DO.INSERT.LINE 1)))
          (TTPROMPTCHAR)
          [replace (LINE NEXTLINE) of OLDLINE with (SETQ NEWLINE
                                                    (create LINE
                                                            START _ OLDEND
                                                            END _ OLDEND
                                                            FIRSTCOL _ (SETQ X \CURSORCOL)
                                                            LASTCOL _ X
                                                            ROW _ ROW
                                                            NEXTLINE _ (fetch (LINE NEXTLINE) of OLDLINE]
          (RENUMBER.LINES NEWLINE ROW)
          (RETURN NEWLINE])
```

(KILL.LINES

```
  [LAMBDA (FIRSTLINE)                                              (* bvm%: " 2-JUN-82 15:46")
```

;;; Returns line records from FIRSTLINE onward to the heap

```
    [PROG NIL
      LP  (COND
             (FIRSTLINE (SETQ FIRSTLINE (CDR (FRPLACA FIRSTLINE 0)))
                                                                   ; Remove some of the circularity in the buffer
                    (GO LP]
    (FRPLACD (FLAST \ENDBUFFER)
             FIRSTLINE])
```

(KILLSEGMENT

```
  [LAMBDA (START END)                                              ; Edited 24-May-91 10:40 by jds
```

;;; Removes segment from START up to, but not including END. When done, START contains the contents of former cell END. I.e. any pointer to
;;; START is still valid; any pointer to END should be reset to START.

```
    (COND
       ((EQ END \ENDBUFFER)
        (SETQ \ENDBUFFER START))
       (T (replace (TTYINBUFFER OLDTAIL) of \TTYINSTATE with (SETQ \LASTAIL))
                                                                   ; kill last buffer markers, as they may be trashed
          (FRPLNODE START (CAR END)
                 (PROG1 (CDR END)
                     (FRPLACD END (CDR \ENDBUFFER))                ; Cell at END will point to free list
                     (FRPLACD \ENDBUFFER (CDR START))              ; And this segment now is start of free list

                     )])
```

(L-CASECODE

```
  [LAMBDA (CHAR)                                                    (* lmm "16-Nov-86 13:24")
    (CL:CHAR-INT (CL:CHAR-DOWNCASE (CL:INT-CHAR CHAR])
```

(MOVE.BACK.TO

```
  [LAMBDA (BUFTAIL)                                                 (* bvm%: " 1-JUN-82 18:10")
    (GO.TO.RELATIVE (IDIFFERENCE \CURSORCOL (SEGMENT.BIT.LENGTH BUFTAIL \CURSOR)))
    (SETQ \CURSOR BUFTAIL])
```

(MOVE.FORWARD.TO

```
  [LAMBDA (BUFTAIL)                                                 (* bvm%: " 1-JUN-82 18:03")
    [GO.TO.RELATIVE (IPLUS \CURSORCOL (SEGMENT.BIT.LENGTH \CURSOR (SETQ BUFTAIL (CHECK.MARGIN BUFTAIL \ARROW]
    (SETQ \CURSOR BUFTAIL])
```

(MOVE.TO.LINE

```
  [LAMBDA (NEWLINE BUFTAIL)                                         ; Edited 24-May-91 10:35 by jds
```

;;; Moves to indicated line at indicate buffer position (default is START), resetting \ARROW etc appropriately.

```
    (PROG ((RELATIVE.POSITION 0))
          [COND
             [BUFTAIL (SETQ RELATIVE.POSITION (SEGMENT.BIT.LENGTH (fetch (LINE START) of NEWLINE)
                                                                  (SETQ BUFTAIL (CHECK.MARGIN BUFTAIL NEWLINE]
             (T (SETQ BUFTAIL (fetch (LINE START) of NEWLINE]
          (GO.TO.RELATIVE (IPLUS (fetch (LINE FIRSTCOL) of NEWLINE)
                                 RELATIVE.POSITION)
                 (fetch (LINE ROW) of NEWLINE))
          (SETQ \CURSOR BUFTAIL)
          (RETURN (SETQ \ARROW NEWLINE])
```

(MOVE.TO.NEXT.LINE

```
  [LAMBDA NIL                                                       ; Edited 24-May-91 10:35 by jds
    (GO.TO.RELATIVE 'LINE (SETQ \ARROW (fetch (LINE NEXTLINE) of \ARROW)))
    (SETQ \CURSOR (fetch (LINE START) of \ARROW])
```

(MOVE.TO.START.OF.WORD

```
  [LAMBDA NIL                                                       (* bvm%: "20-FEB-82 22:34")
    [COND
       ((AT.END.OF.LINE)
        (MOVE.BACK.TO (PREVWORD \CURSOR)))
       ((SELCHARQ (CAR \CURSOR)
             ((%( %[)
                  NIL)
             T)
        ;; Do nothing if sitting under an open paren/bracket, since otherwise the PREVWORD below will go to the previous word, rather than selecting
        ;; the 'word' which begins with the paren;  in all other cases the PREVWORD will do the right thing: if under the word, goes to its start
        ;; (ignoring parens), or if under a space goes to the start of the word before the space
        (MOVE.BACK.TO (PREVWORD (TTNEXTCHAR \CURSOR]
    NIL])
```

(MOVE.TO.WHEREVER

```
  [LAMBDA (BUF)                                                     (* bvm%: "24-Feb-80 00:28")
```

;;; Moves to BUF, wherever it may be.

```
    (MOVE.TO.LINE (FIND.LINE BUF)
          BUF])
```

(NTH.COLUMN.OF

```
  [LAMBDA (LINE N)                                                  ; Edited 24-May-91 10:35 by jds
```

;;; Returns buffer tail of LINE record which best approximates the Nth printing column of that line

```
    (NTH.RELATIVE.COLUMN.OF LINE (IDIFFERENCE N (fetch (LINE FIRSTCOL) of LINE])
```

(NTH.RELATIVE.COLUMN.OF

```
  [LAMBDA (LINE N)                                                  ; Edited 24-May-91 11:10 by jds
    ;; Returns buffer tail in LINE which represents the Nth printing character on the line.  Returns start or end of buffer if out of range.  If the nth char is
    ;; a pad char, returns the start of the pad char sequence
    (COND
       ((NOT (IGREATERP N 0))
        (fetch (LINE START) of LINE))
       (T (for WIDTH CH (BUF _ (fetch (LINE START) of LINE))
             (END _ (fetch (LINE END) of LINE)) do [COND
                                                      ((EQ BUF END)
                                                                ; Ran off the end, so quit
                                                       (RETURN END))
                                                      (T (COND
```

```
                                                    ([ILESSP N (SETQ WIDTH (COND
                                                                   ((COMPLEXCHARP
                                                                     (SETQ CH (CAR BUF)))
                                                                    (fetch (COMPLEXCHAR
                                                                                CPXWIDTH)
                                                                       of CH))
                                                                   (T (TTBITWIDTH CH]
                                                       (RETURN BUF)))
                                                  (SETQ N (IDIFFERENCE N WIDTH]
                                                (SETQ BUF (CDR BUF])
```

(OVERFLOW?

```
[LAMBDA (WIDTH)                                                    ; Edited 24-May-91 10:35 by jds
```

;; If typing WIDTH more chars would cause this line to overflow, starts new line (or simply goes to next line when N=0)

```
(COND
   ((NOT (ILESSP (IPLUS \CURSORCOL WIDTH)
                 \RMARG))
    (COND
       [(AT.END.OF.LINE)
        (PROG ((OLDLINE \ARROW))
              (START.NEW.LINE)
              (COND
                 ((AND \AUTOFILL DISPLAYTERMFLG)                   ; Hit the margin in the middle of a word.  Try to move that word
                                                                   ; intact to the new line
                  (ADJUSTLINE 1 OLDLINE)
                  (GO.TO.RELATIVE (fetch (LINE LASTCOL) of \ARROW)
                        (fetch (LINE ROW) of \ARROW]
       ((EQ WIDTH 0)
        (MOVE.TO.NEXT.LINE))
       (T (BREAKLINE])
```

(OVERFLOWLINE?

```
[LAMBDA (LINE)                                                     ; Edited 24-May-91 10:35 by jds
```

;;; True if LINE overflows into next line, rather than ending in a cr

```
   (EQ (fetch (LINE END) of LINE)
       (fetch (LINE START) of (fetch (LINE NEXTLINE) of LINE])
```

(PREVLINE

```
[LAMBDA (LINE N)                                                   ; Edited 24-May-91 10:35 by jds
```

;;; Backs up N lines in buffer before LINE, as far as start of buffer. i.e. an NLEFT on line records.

```
   (PROG ((X \FIRSTLINE)
          (L \FIRSTLINE))
     LP                                                            ; Advance X by N chars
         (COND
            ((EQ N 0)
             (GO LP1))
            ((OR (EQ X LINE)
                 (NULL X))                                         ; The NULL case should never happen, but better be safe
             (RETURN L)))
         (SETQ X (fetch (LINE NEXTLINE) of X))
         (SUB1VAR N)
         (GO LP)
     LP1                                                           ; Now advance X and L in parallel until X reaches LINE, at which
                                                                   ; point L is N before it
         (COND
            ((OR (EQ X LINE)
                 (NULL X))
             (RETURN L)))
         (SETQ X (fetch (LINE NEXTLINE) of X))
         (SETQ L (fetch (LINE NEXTLINE) of L))
         (GO LP1])
```

(PREVWORD

```
[LAMBDA (BUF N START)                                              ; Edited 24-May-91 10:35 by jds
  (OR START (SETQ START (fetch (LINE START) of \ARROW)))
  (for (X _ START)
       (NEW _ T)
       (%#HITS _ 0) by (TTNEXTCHAR X) until (EQ X BUF) do
```

;; Return start of the Nth word in line before BUF, or beginning of line if no such word

```
(COND
   ((WORDSEPRP (FIRSTCHAR X))
       ; Space between words
    (SETQ NEW T))
   (NEW (SETQ $$VAL X)
       ; Start of new word
        (SETQ NEW NIL)
        (ADD1VAR %#HITS)))
```

```
    finally (RETURN (COND
                    ((OR (NOT N)
                         (EQ N 1)
                         (EQ %#HITS 0))
                     (OR $$VAL START))
                    ((ILESSP (SETQ N (IDIFFERENCE %#HITS N))
                            0)                              ; N was greater than #words in buffer
                     START)
                    ((EQ N 0)
                     (FIND.START.OF.WORD START))
                    (T (FIND.NEXT.WORD (FIND.START.OF.WORD START)
                                       N])
```

(PROPERTAILP

```
  [LAMBDA (X Y)                                             (* bvm%: " 4-Aug-78 12:03")
```

;;; true if X is a PROPER tail of Y

```
    (AND X (NEQ X Y)
         (BUFTAILP X Y])
```

(READFROMBUF

```
  [LAMBDA (START END COPYFLG)                               ; Edited 24-May-91 11:10 by jds
```

;; Unreads the chars in the buffer from START to END. The cells are returned to the free pool as they are used to reduce the storage demands on ;; large unreads. Multichar sequences in buffer are unread as just their 'real' characters

```
    (PROG (FIXUP CH)
          [COND
             ([AND (NOT (AT.END.OF.LINE))
                   (for (BUF _ START) by (CDR BUF) until (EQ BUF END) thereis (EQ (CAR BUF)
                                                                                  (CHARCODE EOL]
```

;; An insertion that contains a cr. This will look awful if we have to keep shoving text in front of us, so break the line first, then unbreak ;; it at end

```
              (BREAKLINE (CHARCODE SPACE)
                     (SETQ FIXUP T]
          (until (EQ START END) do [COND
                                      ((COMPLEXCHARP (SETQ CH (CAR START)))
                                       (SETQ CH (fetch (COMPLEXCHAR CPXREALCHAR) of CH]
                                    (COND
                                       ((NEQ CH EOLCHARCODE)
                                        (ADDNAKEDCHAR CH T))
                                       ((NOT (AT.END.OF.LINE))      ; Insert EOL in middle of line
                                        (BREAKLINE EOLCHARCODE))
                                       ((OR (NEQ (CDR START)
                                                 END)
                                            (NOT (AT.END.OF.TEXT \CURSOR)))
                                                                    ; EOL. Start new line. Ignore it if this is a terminating eol
                                        (START.NEW.LINE EOLCHARCODE)))
                                    (SETQ START (CDR START)))
          (COND
             (FIXUP                                             ; Kill the cr we inserted
                    (MOVE.TO.WHEREVER (PROG1 \CURSOR
                                             (DELETE.LONG.SEGMENT1 \ARROW \CURSOR (fetch (LINE NEXTLINE)
                                                                                         of \ARROW)
                                                                   (TTNEXTCHAR \CURSOR)))])
```

(RENUMBER.LINES

```
  [LAMBDA (LINE ROW)                                        ; Edited 24-May-91 10:35 by jds
```

;;; Renumbers lines from LINE onward, giving LINE the value ROW

```
    (while LINE do (replace (LINE ROW) of LINE with ROW)
                   (ADD1VAR ROW)
                   (SETQ LINE (fetch (LINE NEXTLINE) of LINE])
```

(RESTORE.CURSOR

```
  [LAMBDA NIL                                               (* lmm "20-Nov-86 00:27")
    (GO.TO.RELATIVE \HOMECOL \HOMEROW])
```

(RESTOREBUF

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:41 by jds
```

;; recover previous buffer, which extends to either our current LASTAIL, if user has done deletions on this line, or previous LASTAIL, stored in the ;; front of the buffer. If neither, then recover last thing zapped with the mouse

```
    (PROG (TAIL)
          (COND
             ([AND (AT.END.OF.BUF)
                   (SETQ TAIL (OR (AND \LASTAIL (IGEQ \LASTAILROW (fetch (LINE ROW) of \ARROW))
                                       (OR (IGREATERP \LASTAILCOL \CURSORCOL)
                                           (IGREATERP \LASTAILROW (fetch (LINE ROW) of \ARROW)))
                                       (PROPERTAILP \LASTAIL \ENDBUFFER))
```

```
                                (PROPERTAILP (fetch (TTYINBUFFER OLDTAIL) of \TTYINSTATE)
                                      \ENDBUFFER]
            (END.DELETE.MODE)
            (READFROMBUF [CONS (CAR \ENDBUFFER)
                               (PROG1 (CDR \ENDBUFFER)       ; now detach buffer from here to TAIL to avoid conflict
                                   (FRPLNODE \ENDBUFFER 0 (CDR TAIL)))]
                   TAIL)
            (SETQ \LASTAIL \ENDBUFFER)
            (SETQ \LASTAILCOL \CURSORCOL)
            (SETQ \LASTAILROW (fetch (LINE ROW) of \ARROW))
            (replace (TTYINBUFFER OLDTAIL) of \TTYINSTATE with NIL))
           (\LAST.DELETION (READFROMBUF \LAST.DELETION NIL T)
                  (ADJUSTLINE.AND.RESTORE T))
            (T                                               ; Can't find where buffer ended;  perhaps we have written past it
               (BEEP])
```

(RETYPE.BUFFER

```
[LAMBDA (LINE LASTLINE FROM.HERE)                            ; Edited 24-May-91 10:35 by jds
```

;; Refreshes buffer starting with LINE for one line, or going to LASTLINE, where LASTLINE=T means end of buffer. Moves cursor to start of LINE
;; (based on where we think we might be now) unless FROM.HERE is set. FROM.HERE is set when retyping whole buffer with the current cursor
;; position defined as 0,0; in this case, the cursor is restored on completion to wherever it was last saved, rather than its current position

```
(PROG* ((ROW (fetch (LINE ROW) of LINE))
        (COL0 (if (EQ ROW 0)
                  then \INITPOS
                else \LMARG))
        L)
       (SETQ \DELETING)
       (BINARY.MODE)
       [COND
          (FROM.HERE (INIT.CURSOR COL0))
          (T (SAVE.CURSOR)
             (PROGN                                          ; position cursor at start of line
                    (CANCEL.MODES)                           ; in case an funny terminal setting occurred, say because of
                                                             ; noise
                    (if (EQ ROW 0)
                        then                                 ; If reprinting from the top, restore \LOC.ROW.0 to its original
                                                             ; value
                             (SETQ \LOC.ROW.0 (- \LOC.ROW.0 \INITCRLFS)))
                    (GO.TO.ADDRESSING COL0 ROW]
   LP  (TTPROMPTCHAR LINE)
       (TYPE.BUFFER (fetch (LINE START) of LINE)
            (fetch (LINE END) of LINE))
       (COND
          ((AND LASTLINE (SETQ L (fetch (LINE NEXTLINE) of LINE))
                (NEQ L LASTLINE))
           (SETQ LINE L)
           (TTCRLF)
           (ADD1VAR ROW)
           (GO LP)))
       (COND
          ((EQ LASTLINE T)                                   ; kill any text that might be below bottom line
           (ERASE.TO.END.OF.PAGE)))
       (RESTORE.CURSOR])
```

(SAVE.CURSOR

```
[LAMBDA NIL                                                  (* bvm%: "11-MAR-81 21:40")
  (SETQ \HOMEROW \CURSORROW)
  (SETQ \HOMECOL \CURSORCOL])
```

(SCANBACK

```
[LAMBDA (CHAR BUF N START)                                   ; Edited 24-May-91 10:35 by jds
```

;;; Searches back for Nth previous occurrence of CHAR in buffer before BUF, returning NIL if there are no occurrences. Scan terminates at START,
;;; default is start of line; default N is 1; if there are fewer than N occurrences, returns the earliest one it can

```
   (for [X _ (OR START (SETQ START (fetch (LINE START) of \ARROW]
       (%#HITS _ 0) by (TTNEXTCHAR X) until (EQ X BUF) do (COND
                                                             ((EQ (U-CASECODE (FIRSTCHAR X))
                                                                  CHAR)
                                                              (SETQ $$VAL X)
                                                              (ADD1VAR %#HITS)))
      finally (RETURN (COND
                         ((OR (NOT N)
                              (EQ N 1)
                              (EQ %#HITS 0)
                              (EQ %#HITS 1))
                          $$VAL)
                         (T                                  ; There are #HITS occurrences of CHAR, and we want the Nth
                                                             ; from the end
                            (SCANFORWARD CHAR START (ADD1 (IMAX (IDIFFERENCE %#HITS N)
                                                                0))
                                  BUF])
```

(SCANFORWARD

```
[LAMBDA (CHAR BUF N END)                                    ; Edited 24-May-91 10:35 by jds
```

;;; Finds Nth occurrence of CHAR in BUF before END. Default END is end of current line; default N is 1; CHAR should be uppercase if a letter

```
    (OR N (SETQ N 1))
    (OR END (SETQ END (fetch (LINE END) of \ARROW)))
    (while (NEQ BUF END) do [COND
                               ((EQ (U-CASECODE (FIRSTCHAR BUF))
                                    CHAR)
                                (COND
                                   ((EQ (SUB1VAR N)
                                        0)
                                    (RETURN BUF))
                                   (T (SETQ $$VAL BUF]
                              (SETQ BUF (TTNEXTCHAR BUF])
```

(SCRATCHCONS

```
[LAMBDA NIL                                                 ; Edited 24-May-91 10:41 by jds
```

;;; Returns a garbage cons from the heap at the end of the buffer, or a fresh cons if none available

```
    (replace (TTYINBUFFER OLDTAIL) of \TTYINSTATE with (SETQ \LASTAIL))
                                                            ; Wipe out last buffer ptrs, as this may trash them
    (PROG1 (OR (CDR \ENDBUFFER)
               (CONS))
        (FRPLACD \ENDBUFFER (CDDR \ENDBUFFER)))])
```

(SEGMENT.LENGTH

```
[LAMBDA (START END)                                         ; Edited 24-May-91 11:11 by jds
```

;;; Returns number of print positions in buffer from START to END

```
    (PROG ((N 0))
      LP  (COND
             ((EQ START END)
              (RETURN N)))
          (add N (COND
                    ((COMPLEXCHARP (CAR START))
                     (fetch (COMPLEXCHAR CPXNCHARS) of (CAR START)))
                    (T 1)))
          (SETQ START (CDR START))
          (GO LP])
```

(SEGMENT.BIT.LENGTH

```
[LAMBDA (START END)                                         ; Edited 24-May-91 11:11 by jds
```

;;; Returns number of print positions in bits in buffer from START to END

```
    (PROG ((N 0))
      LP  (COND
             ((EQ START END)
              (RETURN N)))
          [add N (COND
                    ((COMPLEXCHARP (CAR START))
                     (fetch (COMPLEXCHAR CPXWIDTH) of (CAR START)))
                    (T (FCHARWIDTH (CAR START)
                                   \FONT]
          (SETQ START (CDR START))
          (GO LP])
```

(SETLASTC

```
[LAMBDA (CHAR)                                              (* bvm%: "10-APR-81 23:28")
```

;; Makes CHAR be LASTC for T. This is a kludge; I should be interfacing better with \LINEBUF.OFD at a more fundamental level.

```
(\BOUT \LINEBUF.OFD CHAR])
```

(SETTAIL?

```
[LAMBDA (EVEN.IF.NOT.THERE)                                 ; Edited 24-May-91 10:35 by jds
```

;; If \ENDBUFFER is farther than we've been before, save this position on LASTAIL. If EVEN.IF.NOT.THERE is set, do this even if cursor is not
;; currently at the end

```
(COND
   ([AND (NOT \DELETING)
         (NOT (EMPTY.BUFFER))
         (OR EVEN.IF.NOT.THERE (EQ \CURSOR \ENDBUFFER))
         (OR (NOT \LASTAIL)
             (OR (ILESSP \LASTAILROW (fetch (LINE ROW) of \ARROW))
                 (AND (ILESSP \LASTAILCOL \CURSORCOL)
                      (ILEQ \LASTAILROW (fetch (LINE ROW) of \ARROW]
    (SETQ \LASTAIL \ENDBUFFER)
```

```
        (SETQ \LASTAILCOL \CURSORCOL)
        (SETQ \LASTAILROW (fetch (LINE ROW) of \ARROW])
```

(SHOW.MATCHING.PAREN

```
  [LAMBDA (BUF)                                          ; Edited 24-May-91 10:36 by jds
```

;;; Indicates parenthesis nesting by briefly moving the cursor to the paren that matches the paren at BUF, if that position is still on the screen. The cursor ;;; stays there for SHOWPARENFLG seconds, or until there is input from the user. Assumes terminal has cursor addressability

```
    (PROG ((MATCHING (BACKSKREAD BUF T))
           LINE ROW COL)                                 ; MATCHING is the buffer position that matches BUF, or NIL if
                                                         ; this paren was quoted somehow.
          (OR MATCHING (RETURN))
          (SETQ LINE (FIND.LINE MATCHING))               ; The buffer LINE on which it appears
          (COND
             ((< (+ (SETQ ROW (fetch (LINE ROW) of LINE))
                    \LOC.ROW.0)
                 0)                                      ; Not on screen, so forget it
              (RETURN)))
          (SETQ COL (+ (SEGMENT.BIT.LENGTH (fetch (LINE START) of LINE)
                                           MATCHING)
                       (fetch (LINE FIRSTCOL) of LINE)))  ; The absolute column position
          (COND
             ((TYPEAHEAD?)
              ;; After all this computation, there is now input waiting, so don't do anything.  Didn't do this earlier, since the SIBE itself takes time,
              ;; and is likely to fail when done immediately after reading the closing paren
              (RETURN)))
          (SAVE.CURSOR)
          (GO.TO.ADDRESSING COL ROW)                     ; Go to absolute coordinates of matching paren
          (TTWAITFORINPUT (COND
                             ((FIXP SHOWPARENFLG)
                              (TIMES SHOWPARENFLG 1000))
                             (T 1000)))                  ; Wait a while to let user see it
          (\CHECKCARET \DSP)                             ; Tell background we moved the cursor
          (RESTORE.CURSOR)                               ; Put cursor back where it belongs

      ])
```

(SKIP/ZAP

```
  [LAMBDA (CMD CHAR N MINUS)                             ; Edited 24-May-91 10:41 by jds
```

;; Performs <edit>S or <edit>Z, i.e. skip or zap to character. CMD is S, Z, B, or -Z (latter two are backward versions of the first two); CHAR is the ;; target character, N is a repeat arg and MINUS is its sign. Last such operation is saved on LASTSKIP so that <edit>A can repeat it

```
    (SETQ CHAR (U-CASECODE CHAR))                        ; Ignore case differences
    [COND
       (MINUS                                            ; invert command
              (SETQ CMD (SELECTC CMD
                           ((CHARCODE S)
                                (CHARCODE B))
                           ((CHARCODE B)
                                (CHARCODE S))
                           ((CHARCODE Z)
                                (IMINUS (CHARCODE Z)))
                           ((IMINUS (CHARCODE Z))
                                (CHARCODE Z))
                           (SHOULDNT]
    (COND
       ([SETQ N (SELECTC CMD
                   ((CHARCODE B)
                       (SCANBACK CHAR \CURSOR N))
                   ((IMINUS (CHARCODE Z))
                       (SCANBACK CHAR (TTNLEFT \CURSOR 1)
                              N))
                   (AND (NOT (AT.END.OF.LINE))
                        (SCANFORWARD CHAR (TTNEXTCHAR \CURSOR)
                               N]
        (SELECTC CMD
             ((CHARCODE S)                               ; S
                  (MOVE.FORWARD.TO N))
             ((CHARCODE Z)                               ; Z
                  (FORWARD.DELETE.TO N))
             ((CHARCODE B)                               ; B
                  (MOVE.BACK.TO N))
             ((IMINUS (CHARCODE Z))                      ; -Z
                  [FORWARD.DELETE.TO (PROG1 (COND
                                               ((AT.END.OF.LINE)
                                                \CURSOR)
                                               (T (TTNEXTCHAR \CURSOR)))
                                            (MOVE.BACK.TO (TTNEXTCHAR N)))])
             (SHOULDNT)))
       (T (BEEP)))
    (replace (TTYINBUFFER LASTSKIP) of \TTYINSTATE with CMD)
    (replace (TTYINBUFFER LASTSKIPCHAR) of \TTYINSTATE with CHAR])
```

(START.NEW.LINE

```
  [LAMBDA (USECR)                                          ; Edited 24-May-91 10:36 by jds
```

;;; Handles moving to new line. USECR, if set, is the <cr> character that should terminate current line

```
    (SETQ \CURSOR (fetch (LINE START) of (SETQ \ARROW (INSERTLINE \ARROW USECR])
```

(START.OF.PARAGRAPH?

```
  [LAMBDA (LINE)                                           ; Edited 24-May-91 11:11 by jds
    (OR (EQ (fetch (LINE END) of LINE)
            (SETQ LINE (fetch (LINE START) of LINE)))
        (AND (COMPLEXCHARP (CAR LINE))
             (EQ (fetch (COMPLEXCHAR CPXREALCHAR) of (CAR LINE))
                 (CHARCODE TAB])
```

(TTADJUSTWORD

```
  [LAMBDA (WORD)                                           ; Edited 20-Jan-88 12:33 by bvm
```

;;; Returns WORD, possibly corrected, according to the spelling list, if any. Returns NIL if FIX was specified and the word fails.

```
    (LET (X)
         (COND
            ((OR (NULL SPLST)
                 (FMEMB WORD '(%( %) %[ %] %" %,))
                 (FMEMB WORD SPLST))
             WORD)
            ((AND WORD (SETQ X (FASSOC WORD SPLST)))           ; Is synonym. FASSOC assumes car of atom is NIL
             (CDR X))
            ([AND SPLST (LITATOM WORD)
                  (NEQ \NOFIXSPELL 'NOFIXSPELL)
                  (SETQ X (FIXSPELL WORD 70 SPLST (AND \NOFIXSPELL T]
                                                           ; respelled okay
             X)
            (\FIX (TTPRIN1 WORD)
                  (TTPRIN1 '?)
                  (COND
                     (HELP (TTGIVEHELP HELP))
                     (T (TTPRIN1 "  please try again.")))
                  (TTCRLF)
                  NIL)
            (T WORD])
```

(TTBIN

```
  [LAMBDA (NOMETA)                                         ; Edited 18-Jan-88 15:13 by bvm
```

;;; Read the next char from terminal, return its character code. Sets \EDITBIT true or false according to whether char is meta. If NOMETA is true, the
;;; meta bit is discarded

```
    (PROG ((CHAR (TTWAITFORINPUT NIL T)))
          [COND
             ((EQ CHAR EDITPREFIXCHAR)                     ; edit prefix
              (SETQ CHAR (\GETKEY))
              [COND
                 ((EQ CHAR EDITPREFIXCHAR)                 ; Two edits in a row = Edit-Escape
                  (SETQ CHAR (CHARCODE ESCAPE]
              (SETQ CHAR (METACHAR CHAR]
          [COND
             ((AND NOMETA (METACHARP CHAR))                ; Had meta key down, remove bit.  This is useful for inside Edit
                                                           ; commands
              (SETQ CHAR (NONMETACHARBITS CHAR]
          (\CHECKCARET \DSP)                               ; Turn off the caret, since we will probably move
          (RETURN CHAR])
```

(TTBITWIDTH

```
  [LAMBDA (CHAR)                                           ; Edited 17-Jan-88 16:04 by bvm:
    (FCHARWIDTH CHAR \FONT])
```

(TTCRLF

```
  [LAMBDA NIL                                              (* lmm "16-Nov-86 04:13")
```

;;; Prints a crlf, updating cursor appropriately

```
    (DO.CRLF)
    (TTCRLF.ACCOUNT])
```

(TTCRLF.ACCOUNT

```
  [LAMBDA NIL                                              ; Edited 18-Jan-88 15:41 by bvm
    (SETQ \CURSORROW (ADD1 \CURSORROW))
    [COND
       ((EQ (+ \LOC.ROW.0 \CURSORROW)
```

```
            \TTPAGELENGTH)                                  ; This crlf glitched the screen, so row 0 has moved up one
         (SETQ \LOC.ROW.0 (SUB1 \LOC.ROW.0))                ; We are also now guaranteed to be on the bottom row of the
                                                            ; window
         (SETQ \BMARG (DSPYPOSITION NIL \DSP]
      (SETQ \CURSORCOL \LMARG])
```

(TTDELETECHAR

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:36 by jds
    (COND
       ((AT.START.OF.BUF)
        (BEEP))
       [(AT.END.OF.LINE)
        (COND
           [(AT.START.OF.LINE)                              ; empty line: need to delete to previous line
            (PROG ((PREV (PREVLINE \ARROW 1))
                   DODELETE)
                  (SETQ DODELETE (OVERFLOWLINE? PREV))
                  (DELETELINE \ARROW)                       ; get rid of this line
                  (MOVE.TO.LINE PREV (fetch (LINE END) of PREV))  ; go to end of previous line
                  (COND
                     (DODELETE                              ; We were on overflow line, so have to delete the last char, too
                              (DELETETO (TTNLEFT \CURSOR 1]
           (T (DELETETO (TTNLEFT \CURSOR 1]
       (T (TTRUBOUT])
```

(TTDELETELINE

```
  [LAMBDA NIL                                               ; Edited 24-May-91 10:36 by jds
    (COND
       ((EMPTY.BUFFER)
        (BEEP))
       [(EMPTY.LINE)                                        ; Empty line: delete previous line if at end
        (COND
           ((AT.END.OF.BUF)
            (MOVE.TO.LINE (PREVLINE \ARROW 1))
            (COND
               ((NOT DISPLAYTERMFLG)
                (TTBOUT _)
                (DO.CRLF)))
            (DELETE.TO.END))
           (T (BEEP]
       (T (SETTAIL? T)
          (COND
             ((NOT DISPLAYTERMFLG)
              (TTBOUT %# %#)                                ; On non-display just print ## and return to initial position
              [replace (LINE END) of \ARROW with (SETQ \CURSOR (SETQ \ENDBUFFER (fetch (LINE START) of \ARROW]
              (replace (LINE LASTCOL) of \ARROW with (fetch (LINE FIRSTCOL) of \ARROW))
              (RETYPE.BUFFER \ARROW))
             ((AT.END.OF.LINE)                              ; kill back to start of line.  This can work on glass tty, too,
                                                            ; whereas next clause doesn't
              (DELETETO (fetch (LINE START) of \ARROW)))
             (T                                             ; We're inside line, so go back to start and then zap whole line
                (MOVE.BACK.TO (fetch (LINE START) of \ARROW))
                (FORWARD.DELETE.TO (fetch (LINE END) of \ARROW])
```

(TTDELETEWORD

```
  [LAMBDA (N)                                               ; Edited 24-May-91 10:36 by jds
    (COND
       ((AT.START.OF.BUF)
        (BEEP))
       (T (LET ((TAIL (PREVWORD \CURSOR N))
                PREVL START)
               (SETTAIL?)
               (COND
                  ((EQ TAIL \CURSOR)
                   (DELETE.LONG.SEGMENT1 (SETQ PREVL (PREVLINE \ARROW 1))
                          (SETQ START (PREVWORD \CURSOR N (fetch (LINE START) of PREVL)))
                          \ARROW \CURSOR)
                   (MOVE.TO.WHEREVER START))
                  (T (BACKWARD.DELETE.TO TAIL])
```

(TTECHO.TO.FILE

```
  [LAMBDA (FILE DRIBBLING)                                  ; Edited 24-May-91 11:11 by jds
```

;;; Echos input to FILE. If DRIBBLING is true, the prompts are also echoed

```
    (for (STREAM _ (GETSTREAM FILE 'OUTPUT))
         (LINE _ \FIRSTLINE)
         (FIRSTIME _ T)
         X CH END do (COND
                        ([AND DRIBBLING (SETQ X (COND
                                                   (FIRSTIME     ; Print the first prompt
                                                          (SETQ FIRSTIME NIL)
                                                          (AND \PROMPT1 (NOT (EQMEMB 'NOPROMPT OPTIONS))
```

```
                                                    \PROMPT1))
                                            (T \PROMPT2]
                    (PRIN1 X FILE)))
                (SETQ END (fetch (LINE END) of LINE))
                (SETQ X (fetch (LINE START) of LINE))
                (until (EQ X END) do [COND
                                        ([NOT (COMPLEXCHARP (SETQ CH (CAR X]
                                         (BOUTCCODE STREAM CH))
                                        [(EQ (fetch (COMPLEXCHAR CPXREALCHAR) of CH)
                                             (CHARCODE SPACE))
                                         ;; pseudo-tab kludge: instead of printing the 'real' character, ignore it and print only its
                                         ;; padding spaces
                                         (FRPTQ (fetch (COMPLEXCHAR CPXNCHARS) of CH)
                                                (BOUTCCODE STREAM (CHARCODE SPACE]
                                        (T (BOUTCCODE STREAM (fetch (COMPLEXCHAR CPXREALCHAR) of CH]
                                    (SETQ X (TTNEXTCHAR X)))
                (SETQ LINE (fetch (LINE NEXTLINE) of LINE))
                (COND
                   ((AND (OR DRIBBLING (NEQ (fetch (LINE START) of LINE)
                                            END))
                         (NOT \PROMPTFORWORD))
                    ;; Don't terpri on overflow line, since user didn't;  except always do it to dribblefile, since that's what's on the screen.
                    ;; Promptforword-style input doesn't have terminating cr.
                    (TERPRI FILE)))
     repeatwhile (AND LINE (OR (EQ END \ENDBUFFER)
                               (PROGN                                    ; Avoid echoing the terminating empty line, except when it is an
                                                                         ; empty overflow line
                                      (NEQ (fetch (LINE START) of LINE)
                                           \ENDBUFFER])
```

(TTGIVEHELP

```
[LAMBDA (HELPKEY)                                                        ; Edited 19-Jan-88 19:09 by bvm
  (PROG ((*STANDARD-OUTPUT* \DSP))
        (TERPRI)
        (COND
           ((EQ HELPKEY T)
            (TTGIVEHELP1))
           [(LISTP HELPKEY)
            (COND
               ((EQ (CAR HELPKEY)
                    T)                                                   ; List SPLST first, then subsequent blurb
                (TTGIVEHELP1 T)
                (PRIN1 '% )
                (TTGIVEHELP2 (CDR HELPKEY)
                     T))
               ((EQ (CDR HELPKEY)
                    T)                                                   ; Similar, but blurb first
                (TTGIVEHELP2 (CAR HELPKEY)
                     T)
                [COND
                   ((NEQ (POSITION)
                         0)
                    (PRIN1 '% ]
                (TTGIVEHELP1 T T))
               (T (TTGIVEHELP2 HELPKEY]
           (T (TTGIVEHELP2 HELPKEY)))
        (COND
           ((NEQ (POSITION)
                 0)
            (TERPRI)))
        (TERPRI)
        (RETURN T])
```

(TTGIVEHELP1

```
[LAMBDA (NO.OTHER NO.INTRO)                                              (* bvm%: "11-MAR-81 21:36")
  (COND
     (SPLST (OR NO.INTRO (PRIN1 "Please select from among "))
            (for X on SPLST unless (OR (EQ X SPELLSTR1)
                                       (EQ X SPELLSTR2))
               do (PRIN1 (INPART (CAR X)))
                  (AND (CDR X)
                       (PRIN1 ", ")))
            (COND
               ((NOT NO.OTHER)
                (OR \FIX (PRIN1 ", or other"))
                (TERPRI])
```

(TTGIVEHELP2

```
[LAMBDA (HELPKEY MIXED)                                                  (* bvm%: " 8-Aug-80 00:14")
  (COND
     [[OR (LITATOM HELPKEY)
          (AND (STRINGP HELPKEY)
```

```
                (NOT (STRPOS '%  HELPKEY]                          ; Atom or spaceless string is a hashfile key
         (COND
            ((NOT (DISPLAYHELP HELPKEY))
             (OR MIXED (PRIN1 "Sorry, no help available."]
        (T (SPRINTT HELPKEY (COND
                                (MIXED                             ; no extra space
                                       0)
                                (T 4))
                   4 0])
```

(TTLASTLINE

```
  [LAMBDA NIL                                                      ; Edited 24-May-91 10:36 by jds
```

;;; Returns last LINE record in buffer

```
    (PROG ((LINE \FIRSTLINE)
           L)
      LP   (COND
              ((SETQ L (fetch (LINE NEXTLINE) of LINE))
               (SETQ LINE L)
               (GO LP)))
           (RETURN LINE])
```

(TTLOADBUF

```
  [LAMBDA (BUF)                                                    ; Edited  8-Feb-88 12:54 by bvm:
```

;; BUF is a list, a la READBUF, which is loaded into our character buffer, using DCHCON to convert the s-expressions therein to char codes. If we ;; are READING, then uses PRIN2 pnames, i.e. includes escape chars and such stuff. Alternatively, BUF may be a string, in which case its ;; contents are also loaded into the buffer, a la BKSYSBUF, and the setting of \READING is irrelevant

```
    (COND
       [(EQ (CAR (LISTP BUF))
            HISTSTR1)                                              ; read from file.  BUF is (<histstr1> (file start  . end))
        (SETQ BUF (CADR BUF))
        (SETFILEPTR (CAR BUF)
              (CADR BUF))
        (bind CHAR NEXTCH (STREAM _ (GETSTREAM (CAR BUF)
                                               'INPUT))
             (END _ (CDDR BUF)) while (< (GETFILEPTR STREAM)
                                         END)
           do                                                      ; Read another character.  Unfortunately, we have to go by file
                                                                   ; pointer to determine end, since stream could have ns chars in it
            (COND
               ((NEQ (SETQ CHAR (BINCCODE STREAM))
                     (CHARCODE CR))
                (ADDNAKEDCHAR CHAR T))
               (T                                                  ; eat up lf after the cr
                  (COND
                     ([OR (>= (GETFILEPTR STREAM)
                              END)
                          (PROGN (SETQ NEXTCH (BINCCODE STREAM))
                                 (AND (EQ NEXTCH (CHARCODE LF))
                                      (>= (GETFILEPTR STREAM)
                                          END]                     ; Ignore final CR
                      (RETURN)))
                  (ADDNAKEDCHAR CHAR)
                  (COND
                     ((NEQ NEXTCH (CHARCODE LF))
                      (ADDNAKEDCHAR NEXTCH]
       (T (PROG (START END)
                [COND
                   ((AND (LISTP BUF)
                         (SETQ START (FMEMB HISTSTR0 BUF)))        ; HISTSTR0 is a marker used by lispx to denote end of line
                    (FRPLACD (NLEFT BUF 1 START]
                (SETQ START (DCHCON BUF (CDR \ENDBUFFER)
                                    (AND (LISTP BUF)
                                         \READING)
                                    RDTBL))                        ; Use our own buffer as a scratchlist for DCHCON as long as it's
                                                                   ; lying around anyway.
                [COND
                   ((LISTP BUF)                                    ; Remove the surrounding parens from the outer list
                    (SETQ END (NLEFT (SETQ START (CDR START))
                                     1]
```

;; now detach the result from our buffer to avoid conflict of interest. If DCHCON found our scratchlist inadequate, START will not be a tail ;; of \ENDBUFFER so the NLEFT below comes out NIL, which is also fine

```
                (FRPLACD (NLEFT \ENDBUFFER 1 START))               ; Now unread the CHCON list.
                (READFROMBUF START END])
```

(TTNEXTLINE

```
  [LAMBDA (LINE N)                                                 ; Edited 24-May-91 10:36 by jds
    (bind L while (AND (NEQ N 0)
                       (SETQ L (fetch (LINE NEXTLINE) of LINE)))
       do (SETQ LINE L)
          (SUB1VAR N)
```

```
      finally (RETURN LINE])
```

(TTNEXTNODE

```
  [LAMBDA (BUF)                                                    (* bvm%: " 2-JUN-82 15:44")

;;; Returns cdr of BUF, tacking on a new cons if the cdr was NIL

    (OR (CDR BUF)
        (CDR (FRPLACD BUF (CONS 0])
```

(TTNLEFT

```
  [LAMBDA (BUF N START)                                            ; Edited 24-May-91 10:36 by jds

;;; Backs up N real characters in this line before BUF as far as START, default being the current start of the line.  Assumes BUF is a tail of line and N is
;;; small

    (OR START (SETQ START (fetch (LINE START) of \ARROW)))
    (PROG ((X START)
           (B START))
      LP                                                           ; Advance X by N chars
          (COND
             ((EQ N 0)
              (GO LP1))
             ((OR (EQ X BUF)
                  (NULL X))                                        ; The NULL case should never happen, but better be safe
              (RETURN B)))
          (SETQ X (TTNEXTCHAR X))
          (SUB1VAR N)
          (GO LP)
      LP1                                                          ; Now advance X and B in parallel until X reaches BUF, at which
                                                                   ; point B is N before it
          (COND
             ((OR (EQ X BUF)
                  (NULL X))
              (RETURN B)))
          (SETQ X (TTNEXTCHAR X))
          (SETQ B (TTNEXTCHAR B))
          (GO LP1])
```

(TTNTH

```
  [LAMBDA (BUF N)                                                  ; Edited 24-May-91 10:36 by jds

;;; Advances N real characters in BUF as far as the end of the line

    (bind (END _ (fetch (LINE END) of \ARROW)) while (AND (NEQ N 0)
                                                          (NEQ BUF END))
       do (SETQ BUF (TTNEXTCHAR BUF))
          (SUB1VAR N)
       finally (RETURN BUF])
```

(TTNTHLINE

```
  [LAMBDA (N)
    (DECLARE (USEDFREE \FIRSTLINE))                                ; Edited 24-May-91 10:36 by jds
    (for (LINE _ \FIRSTLINE) do (COND
                                   ((ILEQ N 0)
                                    (RETURN LINE))
                                   (T (SETQ N (SUB1 N))
                                      (SETQ LINE (OR (fetch (LINE NEXTLINE) of LINE)
                                                     (RETURN LINE])
```

(TTPRIN1

```
  [LAMBDA (STR DOWNCASE INITP)                                     ; Edited 20-Jan-88 10:52 by bvm

;;; PRIN1 of STR, atom or string, directly to the terminal, bypassing any dribble file.  Returns the number of crlfs it did.

    (if (AND DOWNCASE (NOT (U-CASEP STR)))
        then (SETQ DOWNCASE NIL))
    (PROG ((CRLFCOUNT 0)
           CH WIDTH)
          (if (OR INITP (EQ \CURSORCOL \LMARG))
              then                                                 ; If starting at left margin, we might as well start printing.  This
                                                                   ; handles the otherwise unpleasant case of STR being wider
                                                                   ; than the window
                   (GO ONE.AT.A.TIME))
     ;; See if we have space first
          (COND
             ((>= [+ \CURSORCOL (SETQ WIDTH (for I from 1 while (SETQ CH (NTHCHARCODE STR I))
                                               sum (if (EQ CH (CHARCODE CR))
                                                       then                ; I don't know how to handle strings with cr in them.  Punt...
                                                            (GO ONE.AT.A.TIME))
                                                   (CHARWIDTH (if DOWNCASE
```

```
                                                                    then (L-CASECODE CH)
                                                                 else CH)
                                                           \DSP]
                \RMARG)                                                    ; We would go past the right margin
           (if (> WIDTH (- \RMARG \LMARG))
               then                                                        ; It wouldn't fit even at the left, so go start printing
                    (GO ONE.AT.A.TIME))
           (add CRLFCOUNT 1)
           (TTCRLF)))
      (for I from 1 while (SETQ CH (NTHCHARCODE STR I)) do (TTBOUT (if DOWNCASE
                                                                       then (L-CASECODE CH)
                                                                    else CH)))
      (add \CURSORCOL WIDTH)
      (GO DONE)
  ONE.AT.A.TIME
```

;; Print chars one at a time. This handles initial prompts, as well as strings that are wider than the window.

```
      (for I from 1 while (SETQ CH (NTHCHARCODE STR I))
         do (if (EQ CH (CHARCODE CR))
                then (TTCRLF)
                     (add CRLFCOUNT 1)
              else (if (> (add \CURSORCOL (CHARWIDTH (SETQ CH (if DOWNCASE
                                                                 then (L-CASECODE CH)
                                                              else CH))
                                                     \DSP))
                          \RMARG)
                       then                                             ; Out of space
                            (TTCRLF)
                            (add CRLFCOUNT 1)
                            (add \CURSORCOL (CHARWIDTH CH \DSP)))
                   (TTBOUT CH)))
  DONE
      (RETURN CRLFCOUNT])
```

(TTPRINSPACE

```
[LAMBDA (N)                                                             ; Edited 18-Jan-88 23:57 by bvm:
  (OR N (SETQ N 1))
  (if (>= (+ \CURSORCOL N)
          \RMARG)
      then (TTCRLF)
    else (RPTQ N (TTBOUT SPACE))
         (add \CURSORCOL (TIMES N (CHARWIDTH (CHARCODE SPACE)
                                             \DSP])
```

(TTPRIN1COMMENT

```
[LAMBDA (STR DOWNCASE)                                                  ; Edited 16-Jan-88 16:55 by bvm:
```

;; TTPRIN1 of STR in the comment, rather than default, font.

```
  (DSPFONT (PROG1 (DSPFONT \COMMENTFONT T)
                  (TTPRIN1 STR DOWNCASE))
          T])
```

(TTPRIN2

```
[LAMBDA (EXPR CARLVL CDRLVL)                                            ; Edited 16-Jan-88 18:01 by bvm:
  (CL:TYPECASE EXPR
      (LISTP
         (OR CARLVL (SETQ CARLVL 10))
         (OR CDRLVL (SETQ CDRLVL 10))
         [LET (FIRST WRAPPER)
              (COND
                 ((<= CARLVL 0)
                  (TTPRIN1 '%#))
                 ((AND (LITATOM (SETQ FIRST (CAR EXPR)))
                       (SETQ WRAPPER (GET FIRST 'PRETTYWRAPPER))
                       (LISTP (CDR EXPR))
                       (NULL (CDDR EXPR))
                       (SETQ WRAPPER (CL:FUNCALL WRAPPER EXPR)))
                                                                        ; This handles quote and friends
                  (TTPRIN1 WRAPPER)
                  (TTPRIN2 (CADR EXPR)
                           CARLVL CDRLVL))
                 (T (TTPRIN1 '%()
                    [do (TTPRIN2 (CAR EXPR)
                                 (SUB1 CARLVL)
                                 (SUB1 CDRLVL))
                        (COND
                           ((NLISTP (SETQ EXPR (CDR EXPR)))
                            (COND
                               (EXPR (TTPRIN1 " . ")
                                     (TTPRIN2 EXPR)))
                            (RETURN))
                           (T (TTPRIN1 '% )
                              (COND
```

```
                                    ((<= (SETQ CDRLVL (SUB1 CDRLVL))
                                         0)
                                     (TTPRIN1 "...")
                                     (RETURN]
                          (TTPRIN1 '%)])
        (T (TTPRIN1 (MKSTRING EXPR T *READTABLE*)))))])
```

(TTPROMPTCHAR

```
  [LAMBDA (LINE)                                          ; Edited 20-Jan-88 11:33 by bvm
```

;;; Prints the prompt for indicated LINE

```
    (CLEAR.LINE?)
    (LET ((PROMPT (COND
                      ((EQ LINE \FIRSTLINE)
                       \PROMPT1)
                      (T \PROMPT2)))
          CRLFS)
         (COND
            (PROMPT (SETQ CRLFS (TTPRIN1 PROMPT NIL T))
                    (if (EQ LINE \FIRSTLINE)
                        then
```

;; If the prompt took more than one line, account for being down a bit farther (normally CRLFS is 0), but insist that ;; \cursorrow is still zero (it was bumped by crlf). (I don't know what to do if an internal prompt is wider). But then

```
                        (add \LOC.ROW.0 (SETQ \INITCRLFS CRLFS))
                        (SETQ \CURSORROW 0])
```

(TTRUBOUT

```
  [LAMBDA NIL                                             ; Edited 24-May-91 10:36 by jds
```

;;; Delete the previous character -- this is the interpretation of DELETE while inserting

```
    (COND
       ((NOT (AT.START.OF.LINE))
        (BACKWARD.DELETE.TO (TTNLEFT \CURSOR 1)))
       ((AT.START.OF.BUF)
        (BEEP))
       (T
```

;; At start of line, backspace deletes previous cr or char at end of previous overflow line, so have to compute more here

```
          (LET ((PREVL (PREVLINE \ARROW 1))
                START)
               (DELETE.LONG.SEGMENT1 PREVL (SETQ START (TTNLEFT \CURSOR 1 (fetch (LINE START) of PREVL)))
                      \ARROW \CURSOR)
               (MOVE.TO.WHEREVER START])
```

(TTUNREADBUF

```
  [LAMBDA NIL
    (DECLARE (USEDFREE \CURSOR \ENDBUFFER))                (* bvm%: "11-Apr-85 15:13")
```

;;; Takes contents of buffer from \CURSOR onward and 'unreads' it, i.e. erases it and simulates terminal input, a la BKSYSBUF

```
    (for (X _ \CURSOR) by (TTNEXTCHAR X) until (EQ X \ENDBUFFER) do (BKSYSCHARCODE (FIRSTCHAR X)))
    (DELETE.TO.END])
```

(TTWAITFORINPUT

```
  [LAMBDA (MSECS RETKEYFLG)                               ; Edited 19-Jan-88 01:00 by bvm
```

;; Waits for mouse or keystroke. If MSECS is non-NIL, waits a maximum of that many milliseconds. If RETKEYFLG is true, returns the input (if ;; there is some), otherwise just T without reading input. Mouse buttons are returned as funny codes

```
    (PROG ((TIMER (AND MSECS (SETUPTIMER MSECS)))
           (REG (DSPCLIPPINGREGION NIL \DSP))
           W X Y FN ABSY NEWMARG)
      LP   [COND
              ((\SYSBUFP)
               (RETURN (COND
                           (RETKEYFLG (\GETKEY))
                           (T T]
           (WAIT.FOR.TTY)
           (GETMOUSESTATE)
           [COND
              ((AND (LASTMOUSESTATE (OR RED YELLOW BLUE))
                    (>= (SETQ X (LASTMOUSEX \DSP))
                        0)
                    (< X (fetch (REGION WIDTH) of REG))
                    (>= (SETQ Y (- (SETQ ABSY (LASTMOUSEY \DSP))
                                   (fetch (REGION BOTTOM) of REG)))
                        0)
                    (< Y (+ (fetch (REGION HEIGHT) of REG)
                            \CHARHEIGHT))
                    (SETQ W (WHICHW LASTMOUSEX LASTMOUSEY))
                    (EQ (WINDOWPROP W 'DSP)
                        \DSP))                            ; Bugged inside this window
```

;; The IPLUS is a grotesque kludge to include the title bar. Problem is that REG needs to be the clipping region, not the window

```
        ;; region, because we get mouse coordinates in DSP terms, not window terms.  Damn Dedit typein buffer
        ;; The WHICHW test is so that we don't fight the scrollbar handler, or anyone else who happens to be on top of this window.  Really
        ;; should have monitorlock on mouse
        (COND
           [(AND (NOT (EMPTY.BUFFER))
                 (< ABSY (+ \BMARG (TIMES (- \TTPAGELENGTH \LOC.ROW.0)
                                          \CHARHEIGHT)))
                 (< Y (fetch (REGION HEIGHT) of REG))
                 (>= Y (- (ITIMES (- \TTPAGELENGTH (+ \LOC.ROW.0 (fetch (LINE ROW) of (TTLASTLINE))
                                                      1))
                                  \CHARHEIGHT)
                          4)))
            ;; Pointing inside text region.  The second ILESSP is in case the text region overflows the window, we still want title bar to be for
            ;; menu
            (COND
               ((NOT RETKEYFLG)
                (RETURN T))
               (T (DO.MOUSE)
                  (SETQ \PFW.FIRSTTIME NIL)
                  (GO LP]
           ([AND \WINDOWWORLD (SETQ FN (COND
                                          ((LASTMOUSESTATE (ONLY BLUE))
                                           (OR (fetch (TTYINBUFFER TTOLDRIGHTFN) of \TTYINSTATE)
                                               (FUNCTION DOWINDOWCOM)))
                                          (T (fetch (TTYINBUFFER TTOLDBUTTONFN) of \TTYINSTATE]
                                                                ; Pointing in our window, but outside text--do regular button stuff
            (\PROTECTED.APPLY FN (WHICHW))
            (COND
               ((NEQ \RMARG (SETQ NEWMARG (DSPRIGHTMARGIN NIL \DSP)))
                                                                ; Window was reshaped
                (COND
                   ((> \RMARG (SETQ \RMARG NEWMARG))            ; Window got narrower, so reprint
                    (DO.EDIT.PP)))
                (SETQ REG (DSPCLIPPINGREGION NIL \DSP]
     (COND
        ((AND TIMER (TIMEREXPIRED? TIMER))
         (RETURN NIL)))
     (\TTYBACKGROUND)
     (GO LP])
```

(TTYINSTRING

```
  [LAMBDA (BUF TAIL)                                            ; Edited 27-Jan-88 16:00 by bvm
```

;;; Returns a string consisting of the 'real' chars in buffer from BUF to TAIL or end of buffer. If BUF = TAIL returns a null string

```
    (OR TAIL (SETQ TAIL \ENDBUFFER))
    (LET ((NC 0)
          FATP RESULT)
         (for (X _ BUF) by (TTNEXTCHAR X) until (EQ X TAIL) do      ; First scan to see how long string needs to be
                                                        (COND
                                                           ((\FATCHARCODEP (FIRSTCHAR X))
                                                            (SETQ FATP T)))
                                                        (add NC 1))
         (SETQ RESULT (ALLOCSTRING NC NIL NIL FATP))
         (for (X _ BUF) by (TTNEXTCHAR X) until (EQ X TAIL) as I from 1 do (RPLCHARCODE RESULT I (FIRSTCHAR X)))
         RESULT])
```

(TYPE.BUFFER

```
  [LAMBDA (START END)                                           ; Edited 24-May-91 11:12 by jds
```

;;; Types buffer from START to END, returning number of chars typed. Assumes no CR's

```
    (bind ($$VAL _ 0)
          WIDTH CH while (NEQ START END) do [SETQ WIDTH (COND
                                                          ((COMPLEXCHARP (SETQ CH (CAR START)))
                                                           (for PC in (fetch (COMPLEXCHAR CPXPRINTCHARS)
                                                                             of CH)
                                                              do (TTBOUT PC))
                                                           (fetch (COMPLEXCHAR CPXWIDTH) of CH))
                                                          (T (TTBOUT CH)
                                                             (TTBITWIDTH CH]
                                            (add \CURSORCOL WIDTH)
                                            (add $$VAL WIDTH)
                                            (SETQ START (CDR START])
```

(U-CASECODE

```
  [LAMBDA (CHAR)                                                (* lmm "16-Nov-86 13:24")
    (CL:CHAR-INT (CL:CHAR-UPCASE (CL:INT-CHAR CHAR])
```

(U/L-CASE

```
  [LAMBDA (N CAPFLG)
```

```
    (DECLARE (USEDFREE \CURSOR \ARROW))                          ; Edited 24-May-91 10:37 by jds

;;; UPPER or lower-case N words.  CAPFLG=T for uppercase;  CAPFLG=1 for just capitalization

    (COND
       ((AND (EQ N 1000)
              (AT.END.OF.LINE))
        ;; $U or $L at end of line means do it to the whole line.  This handles the common situation where you have typed several words in the wrong
        ;; case and want to fix them without backing up to the beginning
        (MOVE.BACK.TO (fetch (LINE START) of \ARROW)))
       (T (MOVE.TO.START.OF.WORD)))                              ; Go to start of current word
    (PROG ((NEXTWD (CHECK.MARGIN (FIND.NEXT.WORD \CURSOR N)))
           NEEDADJUST OLDLENGTH)
          (SETQ OLDLENGTH (SEGMENT.BIT.LENGTH \CURSOR NEXTWD))   ; Notice how long it is now
          (for (BUF _ (PROGN \CURSOR))
               CHAR until (EQ BUF NEXTWD) do [COND
                                                 ((AND [NOT (COMPLEXCHARP (SETQ CHAR (CAR BUF]
                                                       (>= CHAR (CHARCODE A)))
                                                  (RPLACA BUF (COND
                                                                 (CAPFLG (COND
                                                                            ((EQ CAPFLG 1)
                                                                             ; only raise first char of word
                                                                             (SETQ CAPFLG NIL)))
                                                                         (U-CASECODE CHAR))
                                                                 (T (L-CASECODE CHAR]
                                               (SETQ BUF (TTNEXTCHAR BUF)))
          (SETQ NEEDADJUST (TTADJUSTWIDTH (- (SEGMENT.BIT.LENGTH \CURSOR NEXTWD)
                                             OLDLENGTH)
                                          NEXTWD))
          (TYPE.BUFFER \CURSOR (SETQ \CURSOR NEXTWD))
          (COND
             (NEEDADJUST (ADJUSTLINE.AND.RESTORE])

)

;; Internal reading.  These functions all expect caller to have bound *READTABLE* correctly (not bound in TTYIN for who-line transparency)

(DEFINEQ

(TTRATOM
  [LAMBDA NIL                                                    ; Edited 24-May-91 11:18 by jds

;;; Reads next atom from BUFFER, advancing it suitably

    (COND
       ((EQ (TTSKIPSEPR)
            \ENDBUFFER)
        null)
       (T (LET ((STRM (TTYINBUFFERSTREAM \BUFFER)))
               (PROG1 (RATOM STRM)
                      (SETQ \BUFFER (fetch (TTYINBUFFERSTREAM TTYINPUT) of STRM)))])

(TTREADLIST
  [LAMBDA NIL                                                    ; Edited 16-Jan-88 18:01 by bvm:

;;; Read a list of elements.  OPENCHAR is the character that started the list (paren or bracket) or NIL if none.

    (LET ((STRM (TTYINBUFFERSTREAM \BUFFER \ENDBUFFER)))
         (while (SKIPSEPRS STRM) collect (READ STRM])

(TTSKIPSEPR
  [LAMBDA (END)                                                  (* bvm%: "11-Apr-85 15:13")

;;; Skip \BUFFER over any separator chars, returning new value

    (while (AND (NEQ \BUFFER \ENDBUFFER)
                (NEQ \BUFFER END)
                (SPACEP (FIRSTCHAR \BUFFER)))
       do (SETQ \BUFFER (TTNEXTCHAR \BUFFER)))
    \BUFFER])

(TTSKREAD
  [LAMBDA (BUF END PARENCOUNT)                                   ; Edited  8-Feb-88 12:46 by bvm:
    ;; Simulates READLINE starting at BUF, returning tail of BUF where the read would terminate, or NIL if the read does not terminate before END
    ;; (default \ENDBUFFER).  If PARENCOUNT is true and the read does not terminate on account of unmatched parens, then returns the excess
    ;; paren count instead of NIL.
    (OR END (SETQ END \ENDBUFFER))
    (bind X while (NEQ BUF END)
       do [SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR BUF))
                   ((LIST LEFTPAREN.RC LEFTBRACKET.RC)           ; open paren/bracket
                          (SETQ X (CAR BUF))
                          (COND
```

```
                    ((NOT (SETQ BUF (TTSKREAD (CDR BUF)
                                              END PARENCOUNT)))   ; Failed to match string quotes, etc
                     (RETURN NIL))
                    ((EQ BUF END)                                 ; Everything after the paren worked out ok, but we have no
                                                                  ; closing paren
                     (RETURN (AND PARENCOUNT 1)))
                    ((FIXP BUF)                                   ; Unmatched parens
                     (RETURN (ADD1 BUF)))
                    ((AND (EQ (CAR BUF)
                              (CHARCODE %]))
                          (NEQ X (CHARCODE %[)))                  ; left paren terminated by right bracket
                     (RETURN BUF))))
               ((LIST RIGHTPAREN.RC RIGHTBRACKET.RC)              ; closing paren/bracket
                    (RETURN BUF))
               (STRINGDELIM.RC
                    (COND
                       ((NOT (SETQ BUF (FIND.MATCHING.QUOTE (CDR BUF)
                                              END)))
                        (RETURN NIL))))
               (ESCAPE.RC                                         ;  skip over without looking
                          (COND
                             ((EQ (SETQ BUF (CDR BUF))
                                  END)                            ; Last char was escape
                              (RETURN NIL))))
               (MULTIPLE-ESCAPE.RC                                ; Look for matching multiple escape, respecting only single
                                                                  ; escapes along the way
                    (OR (while (NEQ (SETQ BUF (CDR BUF))
                                     END)
                           do (SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR BUF))
                                  (ESCAPE.RC (COND
                                                ((EQ (SETQ BUF (CDR BUF))
                                                     END)
                                                 (RETURN))))
                                  (MULTIPLE-ESCAPE.RC
                                       (RETURN BUF))
                                  NIL))
                        (RETURN NIL)))
               (OTHER.RC NIL)
               (PROGN                                             ; Some sort of macro.  Most we don't care about, but semicolon
                                                                  ; is nasty
                      (COND
                         ((AND (EQ (FIRSTCHAR BUF)
                                   (CHARCODE ;))
                               (READTABLEPROP RDTBL 'COMMONLISP))
                                                                  ; Skip ahead to end of line
                          (COND
                             ([do (COND
                                     ((EQ (SETQ BUF (CDR BUF))
                                          END)
                                      (RETURN T))
                                     ((EQ (FIRSTCHAR BUF)
                                          (CHARCODE EOL))
                                      (RETURN]                    ; Ended inside this comment, so not complete -- just as if
                                                                  ; escaped
                           (RETURN NIL]
          (SETQ BUF (CDR BUF))
      finally (RETURN BUF])
```

(TTYIN.READ

```
  [LAMBDA (FINALCHAR DONTREAD STREAM)                             ; Edited 16-Jan-88 17:58 by bvm:
```

;;; Process buffer for reading. FINALCHAR is what prompted us to terminate the call to TTYIN and is not in the buffer. If DONTREAD is true, then
;;; STREAM is the line buffer and we are acting as \FILLBUFFER -- otherwise, STREAM is our own scratch stream, with an eof fn that returns right
;;; paren; we read the buffer and return a list of expressions

```
    (LET (LASTC BUTLASTC)
         (while (NEQ \BUFFER \ENDBUFFER) do (SETQ BUTLASTC LASTC)
                                                                  ; Fill the buffer
                                        (BOUTCCODE STREAM (SETQ LASTC (FIRSTCHAR \BUFFER)))
                                        (SETQ \BUFFER (TTNEXTCHAR \BUFFER)))
         (COND
            ((AND DONTREAD (SELCHARQ FINALCHAR
                                 (EOL (SELECTC (\SYNCODE \RDTBLSA LASTC)
                                          ((LIST RIGHTPAREN.RC RIGHTBRACKET.RC)
                                               (COND
                                                  ((OR (NULL BUTLASTC)
                                                       (EQ (\SYNCODE \RDTBLSA BUTLASTC)
                                                           ESCAPE.RC))
                                                                  ; If it ended in a quoted right paren, then it's just like any other
                                                                  ; character
                                                   T)
                                                  ((EQ (\SYNCODE \RDTBLSA (CHARCODE %]))
                                                       RIGHTBRACKET.RC)
                                                                  ;; Line ended in paren.  Change to right bracket so READLINE doesn't get confused.
                                                                  ;; Only do this if ] really is right bracket!
```

```
                                                    (\SETFILEPTR STREAM (IDIFFERENCE (\GETFILEPTR STREAM)
                                                                                     (STREAMBYTESPERCHAR STREAM)))
                                                    (BOUTCCODE STREAM (CHARCODE %]))
                                                   NIL)))
                                        T))
                                  ((%) %])
                                       NIL)
                                  T))                                    ; Print FINALCHAR unless terminator was EOL and line already
                                                                         ; ended in a closing paren or bracket
            (BOUTCCODE STREAM FINALCHAR)))
      (\SETEOFPTR STREAM (\GETFILEPTR STREAM))
      (\SETFILEPTR STREAM 0)
      (COND
         (DONTREAD                                                       ; STREAM = \LINEBUF.OFD and caller will take care of reading
                                                                         ; buf
                (AND (EQ STREAM \LINEBUF.OFD)
                     (replace (LINEBUFFER LINEBUFSTATE) of STREAM with READING.LBS))
                T)
         (T                                                              ; Read from buffer until it's empty
            (PROG1 (bind TERM while [AND (SKIPSEPRS STREAM)
                                         (SETQ TERM (NLSETQ (READ STREAM]
                         collect (CAR TERM))
                   (\SETFILEPTR STREAM 0)                                ; Now clear the stream so nobody reads extra garbage after us
                   (\SETEOFPTR STREAM 0))])

)

;; Escape completion and friends

(DEFINEQ

(FIND.MATCHING.WORD
  [LAMBDA (WORDS START BUFTAIL)                                          (* lmm "14-Nov-86 17:09")
    ;; Find the first word in spelling list WORDS which matches the characters in the buffer from START to BUFTAIL (or current cursor position), and
    ;; return the corresponding tail of WORDS
    (OR BUFTAIL (SETQ BUFTAIL \CURSOR))
    (find TAIL on WORDS suchthat (WORD.MATCHES.BUFFER (INPART (CAR TAIL))
                                      START BUFTAIL])

(TTCOMPLETEWORD
  [LAMBDA (CAUTIOUS MUST.BE.UNIQUE FIRSTMATCH START)                     ; Edited 20-Jan-88 12:32 by bvm
    ;; Tries to complete the current word from members of SPLST.  Does nothing if no word in progress, or this is a comment line.  Returns true if some
    ;; completion done.  If CAUTIOUS, only complete if can do so uniquely and caller permits fixspell;  if MUST.BE.UNIQUE set, only do unique
    ;; completion.  FIRSTMATCH, if supplied, is the first match in SPLST, and START the start of the current word being worked on
    (LET
     ((UNIQUE T)
      TAIL FIRSTMATCHCHARS SUFFIXCHARS LASTCHAR NEXTCHAR I WORD CH)
     (COND
        ([AND [OR START (SETQ START (COND
                                       ((AT.START.OF.BUF)                ; Empty buffer.  Allow altmode completion on one-word splst
                                                                         ; here
                                        (AND (NOT CAUTIOUS)
                                             \BUFFER))
                                       (T (CURRENT.WORD]
              (OR FIRSTMATCH (SETQ FIRSTMATCH (FIND.MATCHING.WORD SPLST START]
          ;; Completion may be possible.  (CAR FIRSTMATCH) is the first match in SPLST;  START is buffer tail where current word starts;
          ;; NEXTCHAR is the relative position of cursor in current word, i.e. #chars in word + 1;  LASTCHAR is the last char position in common
          ;; among all words which match.  Both NEXTCHAR and LASTCHAR are in terms of the actual characters of the symbol, rather than its
          ;; printed representation, so as to ignore questions of how the words might be escaped.
          [SETQ NEXTCHAR (ADD1 (for (TAIL _ START) by (TTNEXTCHAR TAIL) until (EQ TAIL \CURSOR)
                                  sum (SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR TAIL))
                                               (MULTIPLE-ESCAPE.RC       ; ignore
                                                   0)
                                               (ESCAPE.RC                ; Ignore the escape, but count the next char
                                                          (if (EQ (SETQ TAIL (TTNEXTCHAR TAIL))
                                                                  \CURSOR)
                                                              then         ; Shouldn't happen--FIND.MATCHING.WORD would have failed
                                                                   (RETURN $$VAL)
                                                            else 1))
                                               1]
          [SETQ LASTCHAR (NCHARS (SETQ FIRSTMATCH (INPART (CAR (SETQ TAIL FIRSTMATCH]
          (COND
             ((OR CAUTIOUS (EQ (SUB1 NEXTCHAR)
                               LASTCHAR))
              ;; The latter case happens if the current word is exactly MATCH.  In this case, if there are any other matches they are with words
              ;; containing MATCH as initial substring, and thus no further completion is possible
              (SETQ MUST.BE.UNIQUE T)))
          ;; Now run through all other possible matches with the current word, reducing LASTCHAR to indicate the largest segment in common.
          (while (SETQ TAIL (FIND.MATCHING.WORD (CDR TAIL)
                                   START))
             do (COND
```

```
      (MUST.BE.UNIQUE (RETURN)))
   (SETQ UNIQUE NIL)                                    ; No longer a unique match
   (SETQ WORD (INPART (CAR TAIL)))
   [COND
      ([find old I from NEXTCHAR to LASTCHAR as REFERENCE in (OR SUFFIXCHARS (SETQ SUFFIXCHARS
                                                                                (FNTH (SETQ
                                                                                       FIRSTMATCHCHARS
                                                                                       (CHCON FIRSTMATCH)
                                                                                       )
                                                                                      NEXTCHAR)))
          suchthat (AND (NEQ (SETQ CH (NTHCHARCODE WORD I))
                             REFERENCE)
                        (NOT (AND CH (EQ (LOGXOR CH 32)
                                         REFERENCE)
                                  (IGEQ CH (CHARCODE A))
                                  (ILEQ CH (CHARCODE z]
       (COND
          ((EQ I NEXTCHAR)                              ; Tails are completely different, i.e., we have found two words
                                                        ; that match the prefix so far, but they have no further characters
                                                        ; in common, so give up
           (RETURN))
          (T                                            ; reset LASTCHAR to last common character
             (SETQ LASTCHAR (SUB1 I]
finally ;; chars from NEXTCHAR to LASTCHAR are uniquely determined by prefix so far
      [PROG ((BUF START)
             (OLDLENGTH 0)
             RETYPEBUF RETYPETARGET RETYPELENGTH J NEEDADJUST ESCAPED)
            (END.DELETE.MODE)
            [SETQ FIRSTMATCHCHARS (if (NOT (LITATOM FIRSTMATCH))
                                      then              ; Don't bother with prin2 stuff
                                           (CHCON FIRSTMATCH)
                                    else                ; We want to get the case and escaping right for completion, but
                                                        ; we don't know how to handle packages yet, so get a pname
                                                        ; unlikely to have a package
                                         (LET ((*PACKAGE* (OR (CL:SYMBOL-PACKAGE FIRSTMATCH)
                                                              *PACKAGE*)))
                                              (CHCON FIRSTMATCH T]
            (SETQ I 1)
            (until (EQ I NEXTCHAR)
               do ;; Scan old part of string (part user has typed already) to make sure case is correct
                  (SETQ CH (CAR FIRSTMATCHCHARS))
                  (if RETYPEBUF
                      then (add RETYPELENGTH 1)
                    elseif (OR (NEQ CH (CAR BUF))
                               (EQ BUF \CURSOR))
                      then                              ; The real spelling is different from what's in buf, so we'll want to
                                                        ; fix it.  The (eq buf \\cursor) test is just in case somehow the
                                                        ; buffer has fewer chars than target, but the first n are identical.
                                                        ; (Can you think of an example??)
                           (SETQ RETYPEBUF BUF)
                           (SETQ RETYPETARGET FIRSTMATCHCHARS)
                           (SETQ RETYPELENGTH 1))
                  (if (NOT (if ESCAPED
                               then                     ; Previous char was escape
                                    (SETQ ESCAPED NIL)
                             else (OR (FIXP CH)
                                      (HELP CH))
                                  (SELECTC (\SYNCODE \RDTBLSA CH)
                                       (ESCAPE.RC (SETQ ESCAPED T))
                                       (MULTIPLE-ESCAPE.RC
                                          T)
                                       NIL)))
                      then                              ; Count real chars as they go by
                           (add I 1))
                  (SETQ FIRSTMATCHCHARS (CDR FIRSTMATCHCHARS))
                  (SETQ BUF (CDR BUF)))
            [if RETYPEBUF
                then                                    ; We found a difference, so smash old contents and retype as
                                                        ; needed
                     [if (EQ (SETQ BUF RETYPEBUF)
                             \CURSOR)
                         then                           ; RETYPEBUF = \CURSOR when the word we want to type has
                                                        ; MORE characters than buffer does, yet the characters in buffer
                                                        ; match identically.  I don't think this can happen.
                              (HELP "More chars in match than source?")
                       else (for old J from 1 to RETYPELENGTH until (EQ BUF \CURSOR)
                               do                       ; Replace existing buf chars until we either get to the current
                                                        ; cursor position or we have used up the scanned chars of the
                                                        ; match
                                  (add OLDLENGTH (TTBITWIDTH (CAR BUF)))
                                                        ; OLDLENGTH computes old distance from RETYPEBUF to
                                                        ; BUF
                                  (RPLACA BUF (CAR RETYPETARGET))
                                  (SETQ BUF (CDR BUF))
                                  (SETQ RETYPETARGET (CDR RETYPETARGET)))
```

```
                                [GO.TO.RELATIVE (- \CURSORCOL (+ OLDLENGTH (PROGN
                                                                    ; If the new word is shorter than old, we haven't yet counted the
                                                                    ; bits from old BUF to the cursor
                                                                                  (SEGMENT.BIT.LENGTH
                                                                                   BUF \CURSOR]
                                                                    ; Go to start of changes
                                (SETQ NEEDADJUST (TTADJUSTWIDTH (- (SEGMENT.BIT.LENGTH RETYPEBUF BUF)
                                                                   OLDLENGTH)
                                                                BUF))
                                (TYPE.BUFFER RETYPEBUF BUF)
                                                                    ; Retype with new contents.
                                (COND
                                   (NEEDADJUST (ADJUSTLINE.AND.RESTORE)))
                                (if (NEQ BUF \CURSOR)
                                    then
                                ;; There are more chars in buf than target, so have to delete (this can happen if buffer contains
                                ;; escape characters not deemed necessary in the print name).  We could optimize movement by
                                ;; overtyping some of FIRSTMATCHCHARS instead of doing ADDCHAR's below, but the logic gets
                                ;; way messier than is seemly
                                        (FORWARD.DELETE.TO (PROG1 \CURSOR (MOVE.TO.WHEREVER BUF]
                            (until (EQ RETYPETARGET FIRSTMATCHCHARS) do
                                                                    ; The match has more characters than the buffer, e.g., when
                                                                    ; there were mixed-case chars needing escaping, so add the rest
                                                                    ; of target that we've already scanned.
                                                                        (ADDCHAR (pop RETYPETARGET]
                    ;; Now do second half, the completion part: add new chars from NEXTCHAR thru LASTCHAR
                        (if UNIQUE
                            then                                    ; Just add all the chars, including a possible final vertical bar
                                 (while FIRSTMATCHCHARS do (ADDCHAR (pop FIRSTMATCHCHARS)))
                                 [COND
                                    ((NOT CAUTIOUS)                 ; delimit as well
                                     (ADDCHAR (CHARCODE SPACE))
                                     (COND
                                        ((AND (NEQ NEXTCHAR 1)
                                              (MEMB SPELLSTR1 (OR SPLST USERWORDS)))
                                         ;; Spelling list maintenance: user completed on this word, so move to front of spelling list, assuming
                                         ;; this is a real spelling list.  Don't do it in the trivial case of filling in the entire word uniquely (as when
                                         ;; doing LASTWORD)
                                            (MOVETOP FIRSTMATCH (OR SPLST USERWORDS]
                          else (until (> I LASTCHAR)
                                  do (ADDCHAR (SETQ CH (pop FIRSTMATCHCHARS)))
                                     (if (NOT (if ESCAPED
                                                  then              ; Previous char was escape
                                                       (SETQ ESCAPED NIL)
                                                  else (SELECTC (\SYNCODE \RDTBLSA CH)
                                                                (ESCAPE.RC (SETQ ESCAPED T))
                                                                (MULTIPLE-ESCAPE.RC
                                                                    T)
                                                                NIL)))
                                         then                       ; Count real chars as they go by
                                              (add I 1]
                    (RETURN (OR (AND UNIQUE FIRSTMATCH)
                                T])
```

(WORD.MATCHES.BUFFER

```
  [LAMBDA (WORD START BUFTAIL)                                      ; Edited 17-Jan-88 18:07 by bvm:
```

;;; True if WORD matches case-insensitively chars in buffer from START to BUFTAIL

```
    (for (I _ 0) as (BTAIL _ START) by (TTNEXTCHAR BTAIL) bind CHAR BUFCH until (EQ BTAIL BUFTAIL)
       always (OR (SELECTC (\SYNCODE \RDTBLSA (SETQ BUFCH (FIRSTCHAR BTAIL)))
                           (ESCAPE.RC                               ; Skip to next character
                                      (if (EQ (SETQ BTAIL (TTNEXTCHAR BTAIL))
                                              BUFTAIL)
                                          then                      ; Last character was escape.  How can we match anything?
                                               (RETURN NIL))
                                      (SETQ BUFCH (FIRSTCHAR BTAIL))
                                      NIL)
                           (MULTIPLE-ESCAPE.RC                      ; Just ignore multiple escape--it doesn't affect single escape, and
                                                                    ; so what if we match some things that aren't quite the right
                                                                    ; case?
                               T)
                           NIL)
                      [EQ BUFCH (SETQ CHAR (NTHCHARCODE WORD (add I 1]
                      (AND CHAR (EQ (LOGXOR CHAR 32)
                                    BUFCH)
                           (IGEQ CHAR (CHARCODE A))
                           (ILEQ CHAR (CHARCODE z])
```

(TTYIN.SHOW.?ALTERNATIVES

```
  [LAMBDA NIL                                                       ; Edited  8-Feb-88 12:47 by bvm:
```

;; Called when ? is typed, to indicate alternative completions of current word

```
    (LET (X MATCHED STARTOFWORD DOWNCASE)
         (COND
            ((OR (PROGN                                          ; Global flag controls all of this
                       (NOT ?ACTIVATEFLG))
                 (CL:UNLESS (EQ \LASTCHAR DIDESCAPECODE)         ; If the immediately preceding typein was not an attempt at
                                                                 ; escape completion, don't answer ? if there's no spelling list or
                                                                 ; we're not at the end of the input
                     (OR (NOT SPLST)
                         (NOT (AT.END.OF.BUF))))
                 [PROGN                                          ; There needs to be a word in progress
                        (NOT (SETQ STARTOFWORD (CURRENT.WORD]
                 (PROGN                                          ; If previous char is ?, let it alone (allows ?? etc).
                        (EQ (SETQ X (CAR (NLEFT STARTOFWORD 1 \ENDBUFFER)))
                            (CHARCODE ?)))
                 (SELECTC (\SYNCODE \RDTBLSA X)
                      ((LIST MULTIPLE-ESCAPE.RC ESCAPE.RC)       ; Preceded by an escape character.  This isn't quite right, since
                                                                 ; the escape could be escaped, but it's close
                          T)
                      NIL)
                 (PROGN (FRPLACA \ENDBUFFER (CHARCODE ?))        ; This is pretty random--i.e., if we decide to do something, first
                                                                 ; stick a ? beyond the end of the buffer
                        NIL))
             ;; All sorts of cases where we want to just treat the ? as a normal character
             (ADDCHAR (CHARCODE ?)))
            [(NOT (SETQ MATCHED (FIND.MATCHING.WORD (OR SPLST USERWORDS)
                                    STARTOFWORD)))
             (BEEP)                                              ; No match.  Ring the bell, but accept the ? as is
             (OR (EQ \LASTCHAR DIDESCAPECODE)
                 (ADDCHAR (CHARCODE ?]
            ((TTCOMPLETEWORD NIL T MATCHED STARTOFWORD))
            (T                                                   ; There was more than one completion, so display them (if there
                                                                 ; was a unique one, TTCOMPLETEWORD filled it in)
               (SAVE.CURSOR)
               (GO.TO.FREELINE)
               (if (AND (NEQ *PRINT-CASE* :UPCASE)
                        (READTABLEPROP RDTBL 'CASEINSENSITIVE))
                   then                                          ; Normally would print things in lower case, so try to do that here,
                                                                 ; too.
                        (SETQ DOWNCASE T))
               (TTPRIN1COMMENT "one of ")
               [do (TTPRIN1 (INPART (CAR MATCHED))
                            DOWNCASE)
                   (COND
                      ((SETQ MATCHED (FIND.MATCHING.WORD (CDR MATCHED)
                                                         STARTOFWORD))
                       (TTPRIN1COMMENT ", "))
                      (T (RETURN]
               (RESTORE.CURSOR])

)

;; ? and ?= handler

(DEFINEQ

(DO?CMD
  [LAMBDA (CMD \?TAIL)
    (DECLARE (SPECVARS \?TAIL \?PARAMS \BUFFER \STARTED))        ; Edited  8-Feb-88 12:47 by bvm:

;;; Handles 'read macros' ?  and ?=.  CMD is one of those.  Returns NIL if thinks it isn't.  Saves current cursor location for later restoration

    (\CARET.DOWN)
    (PROG ((*READTABLE* RDTBL)
           (\BUFFER \BUFFER)
           (\?PARAMS null)
           (\STARTED NIL)
           (START (BACKSKREAD \CURSOR))
          STUFF FN FNSTART FNEND SPTAIL SAVE)
          [HANDLER-BIND ((CL:ERROR (FUNCTION DO?CMD.ERRORHANDLER)))
                                                                 ; This handler is in case there is an error while reading the
                                                                 ; symbol we're trying to get information about.
          (SELECTC (\SYNCODE \RDTBLSA (FIRSTCHAR START))
               ((LIST LEFTPAREN.RC LEFTBRACKET.RC)
                   (COND
                      ([AND (EQ (SCANFORWARD (CAR START)
                                             (SETQ FNSTART \BUFFER))
                                START)
                            (PROGN                               ; START is the first paren in buffer, so check and see if there's
                                                                 ; an atom before it, and that the atom is not an exec command
                                   (SETQ FN (TTRATOM))
                                   (SETQ FNEND \BUFFER)
                                   (AND (EQ (TTSKIPSEPR)
                                            START)
                                        (NOT (GETHASH FN *EXEC-COMMAND-TABLE*]
                                                                 ; This is first list on line, preceded by FN in evalqt format
```

```
                          )
                         (T (SETQ FNSTART (SETQ \BUFFER (CDR START)))
                                                          ; EVAL form: read fn
                            (COND
                               ((EQ (SETQ FN (TTRATOM))
                                    CMD)                  ; Hasn't typed the fn name yet!
                                (RETURN)))
                            (SETQ FNEND \BUFFER))))
                     (PROGN                               ; Not inside a list now, so no macro
                            (RETURN)))
                ;; Have to do it this way so that specials get set above to prepare for deletion of ?=
                (SAVE.CURSOR)
                (COND
                   ((EQ CMD '?)                           ; Want verbose description of fn
                    (XHELPSYS FN))
                   (T (GO.TO.FREELINE)
                      (SETQ \STARTED T)                   ; Tells error handler we've begun to work
                      [COND
                         ((EQ \BUFFER START)               ; Apply format, skip over paren
                          (SETQ \BUFFER (CDR START]
                      (COND
                         ([OR (NOT TTYIN?=FN)
                              (NOT (SETQ STUFF (CL:FUNCALL TTYIN?=FN FN]
                                                          ; Default: get the arglist and interpret it
                          (if [NULL (SETQ STUFF (NLSETQ (SMARTARGLIST FN T (SETQ SPTAIL (CONS FN]
                              then                        ; Error occurred getting args, probably not a function
                                   (TTPRIN1COMMENT "Couldn't find args for ")
                                   (TTPRIN2 FN)
                                   (SETQ SPTAIL NIL)
                            else (COND
                                    ((NEQ FN (SETQ FN (CAR SPTAIL)))
                                                          ; Fn was spelling corrected, so There was an extra crlf involved
                                                          ; in printing the correction
                                     (TTCRLF.ACCOUNT))
                                    (T (SETQ SPTAIL NIL)))
                                 (TTYIN.PRINTARGS FN (CAR STUFF)
                                     T)))
                          ((EQ (CAR (LISTP STUFF))
                               'ARGS)
                           (TTYIN.PRINTARGS FN (CDR STUFF)
                                  T))
                          ((LISTP STUFF)
                           (TTPRIN2 STUFF))
                          ((NEQ STUFF T)
                           (TTPRIN1 STUFF]
        (SELECTQ CMD
             (?                                           ; now delete the ?
                (TTRUBOUT))
             (?= (RESTORE.CURSOR)
                 (BACKWARD.DELETE.TO \?TAIL)
                 (COND
                    (SPTAIL
                           ;; Fn was spelling corrected, so replace it.  There was also an extra crlf involved in printing the correction
                           (SETQ SAVE \CURSOR)
                           (MOVE.TO.WHEREVER FNEND)
                           (BACKWARD.DELETE.TO FNSTART)
                           (READFROMBUF (CHCON FN T *READTABLE*))
                           (MOVE.TO.WHEREVER SAVE))))
             NIL)
        (RETURN T])
```

(TTYIN.PRINTARGS

```
  [LAMBDA (FN ARGS ACTUALS ARGTYPE)                       ; Edited 19-Jan-88 01:37 by bvm
    ;; Prints args to fn, matching up with ACTUALS, if supplied.  Do this in a way that lets us keep track of where we are.
    (PROG ((EQUALS " = ")
           (SPACE " ")
           NEXTARG KEY TYPE REMARGS DOWNCASE)
          (\CARET.DOWN)
          (TTPRIN1 "(")
          (TTPRIN2 FN)
          (if (AND ARGS (NEQ *PRINT-CASE* :UPCASE)
                   (READTABLEPROP *READTABLE* 'CASEINSENSITIVE))
              then                                        ; Normally would print things in lower case, so try to do that here,
                                                          ; too.
                   (SETQ DOWNCASE T))
          [COND
             [(LISTP ARGS)                                ; Something interesting to print here
              [COND
                 ((CL:CHARACTERP (CAR ARGS))              ; Forget about actuals
                  (SETQ ACTUALS NIL))
                 ((COND
                     ((EQ ACTUALS T)                      ; Means to compute the actuals
```

```
      (SETQ ACTUALS (TTYIN.READ?=ARGS)))
    (T ACTUALS))                                    ; We have some actuals to match up to args
(COND
    ((CDR ACTUALS)                                  ; More than one actual, so let's put each one on its own line for
                                                    ; legibility
     (TTCRLF))
    (T                                              ; Start on the same line
       (TTPRINSPACE)))
(while ACTUALS
   do ;; This loop will somehow print all the actual args from the user's input
      (COND
         ((NULL ARGS)                               ; More actuals than allowed
          (TTPRIN1COMMENT "+ ... "))
         ((NLISTP ARGS)                             ; Last arg is a "&rest" arg, but indicated as a dotted tail
          (TTPRIN1COMMENT " . ")
          (TTPRIN1COMMENT ARGS DOWNCASE)
          (SETQ ARGS NIL)
          (RETURN))
         ((CL:CHARACTERP (SETQ NEXTARG (CAR ARGS)))
                                                    ; We've gotten to the part where it's reduced to a syntax
                                                    ; description.  I don't plan to match actuals to that.
          (SETQ ACTUALS NIL)
          (RETURN))
         (T                                         ; Some argument name or lambda keyword to show
            (SETQ ARGS (CDR ARGS))
            (TTPRIN1COMMENT NEXTARG DOWNCASE)
            (SELECTQ NEXTARG
                 (&OPTIONAL                         ; We've printed &optional, now print the first name
                            (TTPRINSPACE)
                            (TTPRIN1COMMENT (pop ARGS)
                                   DOWNCASE))
                 ((&REST &BODY)                     ; This will consume all remaining args
                      (TTPRINSPACE)
                      (TTPRIN1COMMENT (pop ARGS)
                             DOWNCASE)
                      (RETURN))
                 (&KEY                              ; Parse actuals into keyword pairs
                       (LET ((ALLOW-OTHER-KEYS (MEMB '&ALLOW-OTHER-KEYS ARGS))
                             USEDKEYS KEY)
                            (while ACTUALS do (TTCRLF)
                                              (SETQ KEY (pop ACTUALS))
                                              (if (OR ALLOW-OTHER-KEYS (MEMB KEY ARGS)
                                                      (EQ KEY :ALLOW-OTHER-KEYS))
                                                  then
                                                  ; Good keyword
                                                      (TTPRIN2 KEY)
                                                      (push USEDKEYS KEY)
                                                  else
                                                  ; Something random--indicate skepticism
                                                      (TTPRIN1COMMENT "[")
                                                      (TTPRIN2 KEY 2 4)
                                                      (TTPRIN1COMMENT "]"))
                                              (TTPRIN1COMMENT EQUALS)
                                              (if ACTUALS
                                                  then (TTPRIN2 (pop ACTUALS)
                                                                2 4)))
                            (if (SETQ ARGS (CL:SET-DIFFERENCE ARGS USEDKEYS))
                                then                ; there is more to print
                                     (TTCRLF))
                            (RETURN)))
                 (&ALLOW-OTHER-KEYS
                      (TTCRLF)
                      (GO $$ITERATE))
                 NIL)))
      (TTPRIN1COMMENT EQUALS)
      (TTPRIN2 (CAR ACTUALS)
             2 4)
      (SETQ ACTUALS (CDR ACTUALS))
      (TTCRLF))
   ;; At this point, if there are any ACTUALS left, it means we had a &REST or dotted tail.  Just print everything that's left
   (if ACTUALS
       then (TTPRIN1COMMENT EQUALS)
            (do (TTPRIN2 (pop ACTUALS)
                         2 4)
                (if ACTUALS
                    then (TTPRINSPACE)
                  else                              ; Finished
                       (RETURN)))
            (if ARGS
                then                                ; More to say yet
                     (TTCRLF]
;; We've now printed all the actuals.  Are there any more args to print?
(while ARGS bind (DOSPACE _ T)
   do (if (NLISTP ARGS)
```

```
                    then (TTPRIN1COMMENT " . ")
                         (TTPRIN1COMMENT ARGS DOWNCASE)
                         (RETURN))
               (SETQ NEXTARG (pop ARGS))
               (SETQ DOSPACE (if (CL:CHARACTERP NEXTARG)
                                 then                          ; Funny syntax description.  Nicer if we handle spacing better
                                                               ; than the default
                                      (CASE NEXTARG
                                          ((#\) #\] #\} #\*)
                                                               ; Don't space before these (but do after)
                                           )
                                          (T (if (AND DOSPACE (NEQ \CURSORCOL \LMARG))
                                                 then (TTPRINSPACE))))
                                      (CASE NEXTARG
                                          ((#\( #\))           ; Parens are part of written syntax, so they come out in regular
                                                               ; font
                                             (TTPRIN1 NEXTARG))
                                          (T                   ; Others are comment
                                             (TTPRIN1COMMENT NEXTARG)))
                                      (CASE NEXTARG
                                          ((#\( #\[ #\{)       ; Don't space after these
                                             NIL)
                                          (T T))
                                 else (if (AND DOSPACE (NEQ \CURSORCOL \LMARG))
                                          then (TTPRINSPACE))
                                      (if (CL:KEYWORDP NEXTARG)
                                          then                 ; Nice to print colon in front of keywords
                                               (TTPRIN1COMMENT ":"))
                                      (TTPRIN1COMMENT NEXTARG DOWNCASE]
          ((NOT (NULL ARGS))                                   ; Atomic arglist--some sort of nospread
           (TTPRIN1COMMENT " ...")
           (if (NEQ ARGS 'U)
               then                                            ; The canonical nospread has arglist U, which is hopelessly
                                                               ; uninformative, so don't even bother printing it
                    (TTPRIN1COMMENT ARGS)
                    (TTPRIN1COMMENT "..."]
     (TTPRIN1 ")")
     (COND
        ((SETQ TYPE (SELECTQ (OR ARGTYPE (ARGTYPE FN))
                             (1 'NL)
                             (3 'NL*)
                            NIL))                              ; indicate arg type
         (TTPRIN1COMMENT (CONCAT " {" TYPE "}"])
```

(TTYIN.READ?=ARGS

```
  [LAMBDA NIL                                                  ; Edited 17-Jan-88 15:20 by bvm:
    ;; Read the actual args for ?= from current input.  Assumes \BUFFER has been positioned at start of args and \?TAIL at ?=.  Caches args in special
    ;; var \?PARAMS so that repeated calls do not recompute.
    (COND
       [(EQ \?PARAMS null)
        (SETQ \?PARAMS (AND (NEQ (TTSKIPSEPR \?TAIL)
                                 \?TAIL)
                            (WITH-RESOURCES (TTSCRATCHFILE)
                                   (LET ((\BUFFER \BUFFER)
                                         (\ENDBUFFER \?TAIL))
                                        (TTYIN.READ NIL NIL TTSCRATCHFILE]
       (T (LISTP \?PARAMS])
```

(DO?CMD.ERRORHANDLER

```
  [LAMBDA (CONDITION)                                          ; Edited 19-Jan-88 20:16 by bvm
    ;; Called by a condition handler underneath ?= handler -- display the condition and abort
    (if (NOT \STARTED)
        then                                                   ; Cursor still after the ?=
             (SAVE.CURSOR)
             (GO.TO.FREELINE))
    (TTPRIN1COMMENT (MKSTRING CONDITION))
    (RESTORE.CURSOR)
    (BACKWARD.DELETE.TO \?TAIL)                                ; Finally, go back and erase the ?=, then return T from DO?CMD
                                                               ; to indicate that we did something.
    (RETFROM (FUNCTION DO?CMD)
           T])

)

;; Display handling

(DEFINEQ

(BEEP
  [LAMBDA (DS)                                                 (* bvm%: "27-JUL-83 23:20")
    (RESETFORM (VIDEOCOLOR (NOT (VIDEOCOLOR)))
           (DISMISS 200])
```

(BITBLT.DELETE

```
  [LAMBDA (X Y WIDTH)                                          ; Edited 18-Jan-88 15:16 by bvm
    (PROG ((MOVEDWIDTH (- \RMARG X WIDTH)))                    ; First move everything from the right over to cursor pos
          (BITBLT \DSP (+ X WIDTH)
                 Y \DSP X Y MOVEDWIDTH \CHARHEIGHT 'INPUT 'REPLACE)
```

;; then delete the last WIDTH positions on the line. May be unnecessary if they were already blank, might want to check LASTCOL

```
          (BITBLT.ERASE (+ X MOVEDWIDTH)
                 Y WIDTH \CHARHEIGHT])
```

(BITBLT.ERASE

```
  [LAMBDA (LEFT BOTTOM WIDTH HEIGHT)                           ; Edited 18-Jan-88 15:18 by bvm
    (BLTSHADE \TEXTURE \DSP LEFT BOTTOM WIDTH HEIGHT 'REPLACE])
```

(BITBLT.INSERT

```
  [LAMBDA (X Y WIDTH)                                          ; Edited 18-Jan-88 15:18 by bvm
    (BITBLT \DSP X Y \DSP (+ X WIDTH)
           Y
           (- \RMARG X WIDTH)
           \CHARHEIGHT
           'INPUT
           'REPLACE)
    (BITBLT.ERASE X Y WIDTH \CHARHEIGHT])
```

(DO.CRLF

```
  [LAMBDA NIL                                                  ; Edited 18-Jan-88 15:19 by bvm
    (SETQ \CURRENTDISPLAYLINE 0)                               ; Avoid stop scroll nonsense
    (DSPLINEFEED (- \CHARHEIGHT)
           \DSP)
    (\DSPPRINTCR/LF (CHARCODE CR)
           \DSP])
```

(DO.DELETE.LINES

```
  [LAMBDA (%#LINES)                                            ; Edited 19-Jan-88 16:35 by bvm
    (PROG ((TOTALHEIGHT (+ (- (DSPYPOSITION NIL \DSP)
                              \BMARG)
                           \CHARHEIGHT))
           (WIDTH (- \RMARG \LMARG))
           (BOTTOM (- \BMARG \DESCENT))
           (DELHEIGHT (TIMES %#LINES \CHARHEIGHT)))
```

;; TOTALHEIGHT is distance from top of current line to bottom of window. DELHEIGHT is height of lines being removed.

```
          [COND
             ((> DELHEIGHT TOTALHEIGHT)                        ; Delete everything from here down
              (SETQ DELHEIGHT TOTALHEIGHT))
             (T (BITBLT \DSP \LMARG BOTTOM \DSP \LMARG (+ BOTTOM DELHEIGHT)
                        WIDTH
                        (- TOTALHEIGHT DELHEIGHT)
                        'INPUT
                        'REPLACE]
          (BITBLT.ERASE \LMARG BOTTOM WIDTH DELHEIGHT])
```

(DO.INSERT.LINE

```
  [LAMBDA NIL                                                  ; Edited 24-May-91 10:37 by jds
```

;;; Inserts a new line on screen in front of current cursor row. The trickiness here is that unless there are some blank lines at the bottom of the screen,
;;; we actually have to scroll upwards before we can insert downwards, lest we lose the bottom line. Leaves cursor at start of new blank line.

```
    (PROG ((DY (- (DSPYPOSITION NIL \DSP)
                  \DESCENT))
           (WIDTH (- \RMARG \LMARG)))
          [COND
             ((EQ (+ \LOC.ROW.0 (fetch (LINE ROW) of (TTLASTLINE))
                      1)
                  \TTPAGELENGTH)                               ; Bottom line is occupied, so scroll stuff above us upward
              (add DY \CHARHEIGHT)
              (MOVETO (DSPXPOSITION NIL \DSP)
                      (+ DY \DESCENT)
                      \DSP)
              (BITBLT \DSP \LMARG DY \DSP \LMARG (+ DY \CHARHEIGHT)
                      WIDTH
                      (- (fetch (REGION TOP) of (DSPCLIPPINGREGION NIL \DSP))
                         (+ DY \CHARHEIGHT))
                      'INPUT
                      'REPLACE)
              (SETQ \LOC.ROW.0 (SUB1 \LOC.ROW.0))              ; Top line of buffer has moved up one

              )
             (T                                                ; Shove everything at or below current line down one
                (BITBLT \DSP \LMARG (+ \BMARG \CHARHEIGHT)
                        \DSP \LMARG \BMARG WIDTH (- DY \BMARG)
```

```
                    'INPUT
                    'REPLACE]                                    ; and clear this line
          (BITBLT.ERASE \LMARG DY WIDTH \CHARHEIGHT])
```

(DO.LF

```
  [LAMBDA NIL                                                    ; Edited 18-Jan-88 15:26 by bvm
    (\DSPPRINTCR/LF (CHARCODE LF)
           \DSP])
```

(ERASE.TO.END.OF.LINE

```
  [LAMBDA NIL                                                    ; Edited 18-Jan-88 15:27 by bvm
    (LET ((X (DSPXPOSITION NIL \DSP)))
         (BITBLT.ERASE X (- (DSPYPOSITION NIL \DSP)
                            \DESCENT)
                (- \RMARG X)
                \CHARHEIGHT])
```

(ERASE.TO.END.OF.PAGE

```
  [LAMBDA NIL                                                    ; Edited 18-Jan-88 22:41 by bvm:
```

;;; Erases from current cursor position to end of page.

```
    (ERASE.TO.END.OF.LINE)
    (LET ((BELOW (- (DSPYPOSITION NIL \DSP)
                    \BMARG)))
```

;; Y-descent is the bottom of current line. \BMARG-descent is bottom of window. Is there anything there?

```
         (COND
            ((> BELOW 0)
             (BITBLT.ERASE \LMARG (- \BMARG \DESCENT)
                    (- \RMARG \LMARG)
                    BELOW])
```

(INSERT.TEXT

```
  [LAMBDA (START END ENDOFLINE)                                  (* bvm%: " 4-JUN-82 13:43")
```

;;; Inserts on screen the contents of buffer from START to END. Text from END to ENDOFLINE is the remainder of the line, in case it's more
;;; economical to just retype the line than do the insertion

```
    (COND
       ((EQ END ENDOFLINE)
        (TYPE.BUFFER START ENDOFLINE))
       (T (TTINSERTSECTION (SEGMENT.BIT.LENGTH START END))
          (TYPE.BUFFER START END])
```

(TTDELSECTION

```
  [LAMBDA (WIDTH)                                                ; Edited 18-Jan-88 15:28 by bvm
```

;;; Deletes WIDTH bits at current pos

```
    (BITBLT.DELETE (DSPXPOSITION NIL \DSP)
           (- (DSPYPOSITION NIL \DSP)
              \DESCENT)
           WIDTH])
```

(TTADJUSTWIDTH

```
  [LAMBDA (DELTA END)                                            ; Edited 24-May-91 10:37 by jds
```

;; Expand or shrink line at current cursorpos by DELTA. END, if supplied, is the end of the section being adjusted; if it is the end of the current line,
;; then it is assumed that expansion is cheap. Returns true if anything was done

```
    (COND
       ((NEQ DELTA 0)
        (COND
           ((ILESSP DELTA 0)                                     ; Line has shrunk
            (TTDELSECTION (IMINUS DELTA)))
           ((NEQ END (fetch (LINE END) of \ARROW))               ; Line has expanded, so need to spread it if not at the end
            (TTINSERTSECTION DELTA)))
        (add (fetch (LINE LASTCOL) of \ARROW)
             DELTA)
        T])
```

(TTINSERTSECTION

```
  [LAMBDA (WIDTH)                                                ; Edited 18-Jan-88 15:29 by bvm
```

;;; Inserts WIDTH character positions, leaving cursor at start of insertion

```
    (BITBLT.INSERT (DSPXPOSITION NIL \DSP)
           (- (DSPYPOSITION NIL \DSP)
              \DESCENT)
           WIDTH])
```

(TTSETCURSOR

```
  [LAMBDA (COL ROW)                                          ; Edited 18-Jan-88 15:29 by bvm
```

;;; Sets cursor to absolute screen position COL,ROW

```
    (MOVETO COL (+ (TIMES (- \TTPAGELENGTH ROW 1)
                          \CHARHEIGHT)
                   \BMARG)
           \DSP])

)

;; TTYINBUFFERSTREAM

(DEFINEQ
```

(TTYINBUFFERDEVICE

```
  [LAMBDA NIL                                                (* bvm%: "11-Apr-86 11:43")
```

;;; Defines a device for streams that read from the ttyin buffer. Modeled after the null device except for the interesting parts

```
    (create FDEV
           DEVICENAME _ 'TTYIN
           RANDOMACCESSP _ NIL
           NODIRECTORIES _ T
           CLOSEFILE _ (FUNCTION NILL)
           DELETEFILE _ (FUNCTION NILL)
           OPENFILE _ (FUNCTION \NULL.OPENFILE)
           REOPENFILE _ (FUNCTION \NULL.OPENFILE)
           BIN _ (FUNCTION TTYINBUFFERBIN)
           BOUT _ (FUNCTION NILL)
           PEEKBIN _ (FUNCTION TTYINBUFFERPEEK)
           READP _ (FUNCTION TTYINBUFFERREADP)
           BACKFILEPTR _ (FUNCTION TTYINBUFFERBACKPTR)
           EOFP _ (FUNCTION TTYINBUFFEREOFP)
           RENAMEFILE _ (FUNCTION NILL)
           GETFILENAME _ (FUNCTION NILL)
           EVENTFN _ (FUNCTION NILL)
           BLOCKIN _ (FUNCTION \EOF.ACTION)
           BLOCKOUT _ (FUNCTION NILL)
           GENERATEFILES _ (FUNCTION \NULLFILEGENERATOR)
           GETFILEPTR _ (FUNCTION ZERO)
           GETEOFPTR _ (FUNCTION ZERO)
           SETFILEPTR _ (FUNCTION NILL)
           GETFILEINFO _ (FUNCTION NILL)
           SETFILEINFO _ (FUNCTION NILL)
           SETEOFPTR _ (FUNCTION NILL])
```

(TTYINBUFFERSTREAM

```
  [LAMBDA (BUF END EOFACTION)                                ; Edited 24-May-91 11:19 by jds
    (LET [(STRM (OR \TTYINBUFFERSTREAM (SETQ \TTYINBUFFERSTREAM (create STREAM
                                                                       DEVICE _ TTYINBUFFERDEVICE
                                                                       ACCESS _ 'INPUT]
          (replace (TTYINBUFFERSTREAM TTYINPUT) of STRM with BUF)
          (replace (TTYINBUFFERSTREAM TTYEOF) of STRM with (OR END \ENDBUFFER))
          (replace (TTYINBUFFERSTREAM TTYEOFACTION) of STRM with EOFACTION)
          (replace (TTYINBUFFERSTREAM TTYORIGINPUT) of STRM with BUF)
         STRM])
```

(TTYINBUFFERBIN

```
  [LAMBDA (STRM)                                             ; Edited 24-May-91 11:19 by jds
    (LET ((BUF (fetch (TTYINBUFFERSTREAM TTYINPUT) of STRM)))
         (COND
            ((EQ BUF (fetch (TTYINBUFFERSTREAM TTYEOF) of STRM)) ; Eof
             (\EOF.ACTION STRM))
            (T (PROG1 (FIRSTCHAR BUF)
                      (replace (TTYINBUFFERSTREAM TTYINPUT) of STRM with (CDR BUF)))])
```

(TTYINBUFFERPEEK

```
  [LAMBDA (STREAM NOERRORFLG)                                ; Edited 24-May-91 11:19 by jds
    (LET ((BUF (fetch (TTYINBUFFERSTREAM TTYINPUT) of STREAM)))
         (COND
            ((EQ BUF (fetch (TTYINBUFFERSTREAM TTYEOF) of STREAM))
                                                             ; Eof
             (AND (NOT NOERRORFLG)
                  (\EOF.ACTION STREAM)))
            (T (FIRSTCHAR BUF])
```

(TTYINBUFFERREADP

```
  [LAMBDA (STRM)                                             ; Edited 24-May-91 11:19 by jds
    (NEQ (fetch (TTYINBUFFERSTREAM TTYINPUT) of STRM)
```

```
          (fetch (TTYINBUFFERSTREAM TTYEOF) of STRM])
```

(TTYINBUFFEREOFP

```
  [LAMBDA (STRM)                                          ; Edited 24-May-91 11:19 by jds
    (EQ (fetch (TTYINBUFFERSTREAM TTYINPUT) of STRM)
        (fetch (TTYINBUFFERSTREAM TTYEOF) of STRM])
```

(TTYINBUFFERBACKPTR

```
  [LAMBDA (STRM)                                          ; Edited 24-May-91 11:19 by jds
```

;;; Back up STRM one. Needed because of top-level READ. What a kludge

```
    (replace (TTYINBUFFERSTREAM TTYINPUT) of STRM with (OR (NLEFT (fetch (TTYINBUFFERSTREAM TTYORIGINPUT)
                                                                    of STRM)
                                                                  1
                                                                  (fetch (TTYINBUFFERSTREAM TTYINPUT)
                                                                    of STRM))
                                                           (fetch (TTYINBUFFERSTREAM TTYINPUT) of STRM])
```

(TTYINWORDRDTBL

```
  [LAMBDA NIL                                             ; Edited 20-Jan-88 22:01 by bvm
```

;;; Makes a table in which normal Lisp syntax characters are just break characters. Additionally, comma is a break

```
    (LET [(TBL (COPYREADTABLE 'ORIG))
          (BREAKS (CHARCODE (%( %) %[ %] %" %,]
         (SETSEPR (CHARCODE (SPACE TAB CR))
                NIL TBL)
         (SETSEPR BREAKS 1 TBL)                           ; Have to disable their regular meanings before making them
                                                          ; pure break chars
         (SETBRK BREAKS NIL TBL)
         (SETSYNTAX (CHARCODE %%)
                'OTHER TBL)                               ; No escape char
         (READTABLEPROP TBL 'NAME "TtyinText")
         TBL])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQ TTYINBUFFERDEVICE (TTYINBUFFERDEVICE))

(RPAQ TTYINWORDRDTBL (TTYINWORDRDTBL))
)
```

;; Mouse handling

```
(DEFINEQ
```

(DO.MOUSE

```
  [LAMBDA NIL                                             ; Edited 24-May-91 11:07 by jds
```

;; Called when mouse is clicked down inside of our region; performs it as an edit command, returning T, or returns NIL if it is not a legal mouse call.
;; The commands that actually change something display their intent while the button is down and are not actually executed until button is released.

```
    (COND
       ((OR (KEYDOWNP 'LSHIFT)
            (KEYDOWNP 'RSHIFT)
            (KEYDOWNP 'CTRL)
            (KEYDOWNP 'MOVE)
            (KEYDOWNP 'COPY))
        (DO.SHIFTED.SELECTION))
       [(LASTMOUSESTATE (ONLY RED))                       ; Position cursor
        (bind ROW/COL while (SETQ ROW/COL (TTRACKMOUSE ROW/COL)) when (LISTP ROW/COL)
           do (\CHECKCARET \DSP)
              (MOVE.TO.LINE (CAR ROW/COL)
                     (CDR ROW/COL]
       [(LASTMOUSESTATE (ONLY YELLOW))                    ; Position cursor by word
        (bind NEWPOS BUF COL LINE while (SETQ NEWPOS (TTRACKMOUSE NEWPOS)) when (LISTP NEWPOS)
           do (\CHECKCARET \DSP)
              [SETQ BUF (BRACKET.CURRENT.WORD (SETQ LINE (fetch (MOUSEPOS ROWPOS) of NEWPOS))
                              (SETQ COL (fetch (MOUSEPOS COLPOS) of NEWPOS]
              (MOVE.TO.LINE LINE (COND
                                    ((> (SEGMENT.BIT.LENGTH (CAR BUF)
                                                            COL)
                                        (SEGMENT.BIT.LENGTH COL (CDR BUF)))
                                     (CDR BUF))
                                    (T (CAR BUF]
       ((LASTMOUSESTATE (ONLY BLUE))                      ; zap from cursor to mouse location.
        (DO.SHIFTED.SELECTION 'DELETE])
```

(DO.SHIFTED.SELECTION

```
  [LAMBDA (INITMODE)                                      ; Edited 24-May-91 11:07 by jds
    (bind START END SAVE EXTENDING MODE NEWSTART NEWEND COL NEWROW NEWMODE BUF NEWPOS WORDLEVEL ENDLINE
```

```
      while (OR [SETQ NEWMODE (COND
                                ((KEYDOWNP 'MOVE)
                                 'MOVE)
                                ((KEYDOWNP 'COPY)
                                 'COPY)
                                [(OR (KEYDOWNP 'LSHIFT)
                                     (KEYDOWNP 'RSHIFT))
                                 (COND
                                    ((KEYDOWNP 'CTRL)
                                     'MOVE)
                                    (T 'COPY]
                                ((KEYDOWNP 'CTRL)
                                 'DELETE]
                (LASTMOUSESTATE (NOT UP)))
      do (SETQ NEWPOS (TTRACKMOUSE NEWPOS))
         (\TTYBACKGROUND)                                    ; Flash caret
         (COND
            [(LASTMOUSESTATE (OR RED YELLOW))                ; Start new selection
             (COND
                [(AND (LISTP NEWPOS)
                      (NEQ (SETQ COL (fetch (MOUSEPOS COLPOS) of NEWPOS))
                           \ENDBUFFER))                      ; There is a selection
                 (SETQ NEWSTART (create MOUSEPOS using NEWPOS))
                 (SETQ NEWROW (fetch (MOUSEPOS ROWPOS) of NEWPOS))
                 (COND
                    ((OR (LASTMOUSESTATE (ONLY RED))
                         (EQ COL (fetch (LINE END) of NEWROW)))  ; Selection extends to next char
                     (SETQ NEWEND (TTNEXTPOS NEWROW COL))
                     (SETQ WORDLEVEL NIL))
                    (T                                       ; Selection is current 'word'
                       (SETQ BUF (BRACKET.CURRENT.WORD NEWROW (fetch (MOUSEPOS COLPOS) of NEWSTART)))
                       (replace (MOUSEPOS COLPOS) of NEWSTART with (CAR BUF))
                                                             ; Start of previous word
                       (SETQ NEWEND (create MOUSEPOS
                                            ROWPOS _ NEWROW
                                            COLPOS _ (CDR BUF)))
                       (SETQ WORDLEVEL T]
                (T (SETQ NEWSTART NIL)))
             (COND
                ((OR (NEQPOS START NEWSTART)
                     (NEQPOS END NEWEND)
                     (NEQ MODE NEWMODE))
                 (COND
                    (START                                   ; turn off old selection
                           (INVERT.LONG.SEGMENT START END MODE)))
                 (COND
                    ((SETQ START NEWSTART)
                     (INVERT.LONG.SEGMENT START (SETQ END NEWEND)
                            (SETQ MODE NEWMODE]
            [(LASTMOUSESTATE (ONLY BLUE))                    ; Extend selection
             [COND
                ((NOT START)                                 ; No selection, extend from cursor
                 [SETQ NEWSTART (SETQ NEWEND (SETQ START (SETQ END (create MOUSEPOS
                                                                          ROWPOS _ \ARROW
                                                                          COLPOS _ \CURSOR]
                 (SETQ WORDLEVEL (SETQ EXTENDING NIL))
                 (COND
                    (INITMODE (SETQ MODE INITMODE)           ; E.g. in DO.MOUSE on BLUE
                           (SETQ INITMODE))
                    (T (SETQ MODE NEWMODE]
             (SETQ NEWROW (fetch (MOUSEPOS ROWPOS) of NEWPOS))
             (COND
                [(NLISTP NEWPOS)                             ; No selection; cancel any existing extension
                 (COND
                    (EXTENDING (COND
                                  ((NEQPOS START NEWSTART)
                                   (INVERT.LONG.SEGMENT NEWSTART START MODE)
                                   (SETQ NEWSTART START))
                                  ((NEQPOS END NEWEND)
                                   (INVERT.LONG.SEGMENT END NEWEND MODE)
                                   (SETQ NEWEND END)))
                               (SETQ EXTENDING NIL]
                (T (COND
                      ((TTBEFOREPOS NEWPOS START)            ; Extending to left of original selection
                       (COND
                          ((AND EXTENDING (NEQPOS END NEWEND))  ; We were extending to right, so switch
                           (INVERT.LONG.SEGMENT END NEWEND MODE)
                           (SETQ NEWEND END)))
                       (INVERT.LONG.SEGMENT NEWSTART (SETQ NEWSTART (create MOUSEPOS using NEWPOS))
                              MODE))
                      (T                                     ; Extending to right
                         (COND
                            ((AND EXTENDING (NEQPOS START NEWSTART))
                                                             ; We were extending to left, so switch
                             (INVERT.LONG.SEGMENT START NEWSTART MODE)
                             (SETQ NEWSTART START)))
                         (INVERT.LONG.SEGMENT NEWEND (SETQ NEWEND (TTNEXTPOS NEWROW (fetch (MOUSEPOS COLPOS)
```

```
                                                                    of NEWPOS)))
                              MODE)))
                    (SETQ EXTENDING T]
               (EXTENDING                                           ; End of extension, make NEWSTART/END permanent
                    (SETQ START NEWSTART)
                    (SETQ END NEWEND)
                    (SETQ EXTENDING NIL)))
      finally (COND
                 (START                                             ; There is a selection, so do it
                        (\CHECKCARET \DSP)
                        (PROG ((STARTBUF (fetch (MOUSEPOS COLPOS) of START))
                               (ENDBUF (fetch (MOUSEPOS COLPOS) of END)))
                              (COND
                                 [(EQ MODE 'COPY)
                                  (INVERT.LONG.SEGMENT START END MODE)
                                                                    ; Take it as typein
                                  (COND
                                     ((BEFOREBUF STARTBUF \CURSOR ENDBUF)
                                                                    ; Can't just unread, because structure will change as we do so
                                      (READFROMBUF (COPY.SEGMENT STARTBUF ENDBUF)))
                                     (T (READFROMBUF STARTBUF ENDBUF T]
                                 ((AND (EQ MODE 'MOVE)
                                       (BEFOREBUF STARTBUF \CURSOR ENDBUF))
                                                                    ; Action overlaps cursor, so effect is just to move cursor
                                  (INVERT.LONG.SEGMENT START END MODE)
                                  (MOVE.TO.LINE (fetch (MOUSEPOS ROWPOS) of END)
                                        ENDBUF))
                                 (T ;; Delete or move selection, insert it as typein at cursor for the latter.  We save away the selection in
                                    ;; \LAST.DELETION to be restored later if desired
                                    (SETQ SAVE (COND
                                                  ((BEFOREBUF STARTBUF \CURSOR ENDBUF)
                                                                    ; The delete will move \CURSOR into trash heap
                                                   STARTBUF)
                                                  (T \CURSOR)))
                                    (SETQ \LAST.DELETION (SETQ BUF (COPY.SEGMENT STARTBUF ENDBUF)))
                                                                    ; Save selection
                                    (DELETE.LONG.SEGMENT START END)
                                    (MOVE.TO.WHEREVER SAVE)         ; Come back to where cursor is (may have moved)
                                    (AND (EQ MODE 'MOVE)
                                         (READFROMBUF BUF NIL T])
```

(COPY.SEGMENT

```
[LAMBDA (START END)                                                 (* bvm%: " 4-DEC-81 17:04")
  (for (TAIL _ START) by (CDR TAIL) until (EQ TAIL END) collect (CAR TAIL])
```

(DELETE.LONG.SEGMENT

```
[LAMBDA (START END)                                                 ; Edited 24-May-91 11:07 by jds
  (DELETE.LONG.SEGMENT1 (fetch (MOUSEPOS ROWPOS) of START)
        (fetch (MOUSEPOS COLPOS) of START)
        (fetch (MOUSEPOS ROWPOS) of END)
        (fetch (MOUSEPOS COLPOS) of END])
```

(DELETE.LONG.SEGMENT1

```
[LAMBDA (STARTLINE STARTCOL ENDLINE ENDCOL)                         ; Edited 24-May-91 10:38 by jds
  (PROG (FIRSTLINE NEXTLINE NROWS)
        (COND
           ((EQ (SETQ NROWS (IDIFFERENCE (fetch (LINE ROW) of ENDLINE)
                                         (fetch (LINE ROW) of STARTLINE)))
                0)                                                  ; All on one line
            (MOVE.TO.LINE STARTLINE STARTCOL)
            (FORWARD.DELETE.TO ENDCOL))
           (T (MOVE.TO.LINE (SETQ FIRSTLINE (fetch (LINE NEXTLINE) of STARTLINE)))
              (DO.DELETE.LINES NROWS)                               ; Delete excess lines
              (SETQ NEXTLINE (fetch (LINE NEXTLINE) of ENDLINE))
              (replace (LINE NEXTLINE) of STARTLINE with NEXTLINE)
              (RENUMBER.LINES NEXTLINE (ADD1 (fetch (LINE ROW) of STARTLINE)))
              [add (fetch (LINE LASTCOL) of STARTLINE)
                   (IDIFFERENCE (SEGMENT.BIT.LENGTH ENDCOL (fetch (LINE END) of ENDLINE))
                                (SEGMENT.BIT.LENGTH STARTCOL (fetch (LINE END) of STARTLINE]
              (replace (LINE END) of STARTLINE with (fetch (LINE END) of ENDLINE))
              (COND
                 ((EQ ENDCOL (fetch (LINE END) of STARTLINE))
                  (replace (LINE END) of STARTLINE with STARTCOL)))
              (KILLSEGMENT STARTCOL ENDCOL)
              (replace (LINE NEXTLINE) of ENDLINE with NIL)
              (KILL.LINES FIRSTLINE)
              (MOVE.TO.LINE STARTLINE STARTCOL)
              (ERASE.TO.END.OF.LINE)
              (COND
                 ((ILESSP (fetch (LINE LASTCOL) of STARTLINE)
                          \RMARG)
                  (TYPE.BUFFER STARTCOL (fetch (LINE END) of STARTLINE)))
                 (T (TYPE.BUFFER STARTCOL (NTH.COLUMN.OF STARTLINE \RMARG))
```

```
                    (ADJUSTLINE NIL STARTLINE])
```

(INVERT.LONG.SEGMENT

```
[LAMBDA (START END MODE)                                    ; Edited 24-May-91 11:07 by jds
  (COND
     ((NOT (EQPOS START END))
      (OR (TTBEFOREPOS START END)
          (swap START END))
      (PROG ((COL (fetch (MOUSEPOS COLPOS) of START))
             (ROW (fetch (MOUSEPOS ROWPOS) of START)))
            (while (NEQ ROW (fetch (MOUSEPOS ROWPOS) of END)) do (INVERT.SEGMENT COL (fetch (LINE START)
                                                                                       of (fetch (LINE NEXTLINE)
                                                                                             of ROW))
                                                                          ROW MODE T)
                                                                (SETQ ROW (fetch (LINE NEXTLINE) of ROW))
                                                                (SETQ COL (fetch (LINE START) of ROW)))
            (INVERT.SEGMENT COL (fetch (MOUSEPOS COLPOS) of END)
                   ROW MODE T])
```

(INVERT.SEGMENT

```
[LAMBDA (START END LINE MODE NOSWAP)
  (DECLARE (USEDFREE \ARROW \CHARWIDTH \LOC.ROW.0 \CHARHEIGHT \BMARG \LMARG))
                                                            ; Edited 24-May-91 10:38 by jds
  (COND
     ((NEQ START END)
      (OR LINE (SETQ LINE \ARROW))
      (OR MODE (SETQ MODE 'DELETE))
      (OR NOSWAP (BEFOREBUF START END (fetch (LINE END) of LINE))
          (swap START END))
      (PROG ((LEFT (+ (fetch (LINE FIRSTCOL) of LINE)
                      (SEGMENT.BIT.LENGTH (fetch (LINE START) of LINE)
                             START)))
            (BOTTOM (+ (ITIMES (- \TTPAGELENGTH \LOC.ROW.0 (fetch (LINE ROW) of LINE)
                                  1)
                               \CHARHEIGHT)
                       \BMARG
                       (- \DESCENT)))
            (WIDTH (SEGMENT.BIT.LENGTH START END)))
           (BLTSHADE (COND
                        ((NEQ MODE 'COPY)
                         BLACKSHADE)
                        (T DOTSHADE))
                 \DSP LEFT BOTTOM WIDTH (COND
                                           ((NEQ MODE 'COPY)
                                            \CHARHEIGHT)
                                           (T 2))
                 'INVERT)
           (COND
              ((EQ MODE 'MOVE)
               (BLTSHADE DOTSHADE \DSP LEFT BOTTOM WIDTH 2 'INVERT])
```

(BRACKET.CURRENT.WORD

```
[LAMBDA (LINE COL)                                          ; Edited 24-May-91 10:38 by jds
```

;;; Return dotted pair of columns indicating start and end of 'word' containing buffer position COL of LINE

```
  (PROG ((INSPACES T)
         (ENDLINE (fetch (LINE END) of LINE))
         (WSTART (fetch (LINE START) of LINE))
         FIRSTCOL LASTCOL)
        (for (BUF _ WSTART) by (TTNEXTCHAR BUF) until (EQ BUF ENDLINE)
           do [COND
                 ([NEQ INSPACES (SETQ INSPACES (WORDSEPRP (FIRSTCHAR BUF]
                                                            ; Change of state
                  (COND
                     (FIRSTCOL                              ; Done
                            (RETURN (SETQ LASTCOL BUF)))
                     (T                                     ; Still looking for COL, note start of word
                        (SETQ WSTART BUF]
              (COND
                 ((EQ BUF COL)
                  (SETQ FIRSTCOL WSTART)))
           finally                                          ; Got to end before word ended
                 (SETQ LASTCOL ENDLINE)
                 (OR FIRSTCOL (SETQ FIRSTCOL LASTCOL)))
        (OR (BEFOREBUF FIRSTCOL COL LASTCOL)
            (HELP))
        (RETURN (CONS FIRSTCOL LASTCOL])
```

(TTBEFOREPOS

```
[LAMBDA (X Y)                                               ; Edited 24-May-91 11:08 by jds
  (COND
     [(EQ (fetch (MOUSEPOS ROWPOS) of X)
```

```
          (fetch (MOUSEPOS ROWPOS) of Y))
       (AND (NEQ (fetch (MOUSEPOS COLPOS) of X)
                 (fetch (MOUSEPOS COLPOS) of Y))
            (BEFOREBUF (fetch (MOUSEPOS COLPOS) of X)
                       (fetch (MOUSEPOS COLPOS) of Y)
                       (fetch (LINE END) of (fetch (MOUSEPOS ROWPOS) of X]
       (T (ILESSP (fetch (LINE ROW) of (fetch (MOUSEPOS ROWPOS) of X))
                  (fetch (LINE ROW) of (fetch (MOUSEPOS ROWPOS) of Y])
```

(TTNEXTPOS

```
  [LAMBDA (LINE COL)                                        ; Edited 24-May-91 10:38 by jds
```

;;; Makes a MOUSEPOS out of the position, if any, immediately after COL of LINE

```
    (COND
       ((AND (EQ COL (fetch (LINE END) of LINE))
             (NEQ COL \ENDBUFFER))
        (create MOUSEPOS
                ROWPOS _ (SETQ LINE (fetch (LINE NEXTLINE) of LINE))
                COLPOS _ (fetch (LINE START) of LINE)))
       (T (create MOUSEPOS
                  ROWPOS _ LINE
                  COLPOS _ (COND
                              ((EQ COL \ENDBUFFER)
                               COL)
                              (T (TTNEXTCHAR COL])
```

(TTRACKMOUSE

```
  [LAMBDA (OLDROW/COL)                                      ; Edited 24-May-91 11:14 by jds
    (DECLARE (USEDFREE \TTPAGELENGTH \LOC.ROW.0 \BMARG \CHARHEIGHT \LMARG \RMARG \FONT))
```

;; Follows the mouse, returning whenever its row/col changes or the user lets up the mouse buttons. Converts mouse coordinates into a dotted pair (LINE . BUFPOS) indicating what char is being pointed at, or T if outside range of text. Returns NIL when user lets go. OLDROW/COL is the previous value of this routine, which we may smash.

```
    (PROG (OLDX OLDY ROW COL OLDROW OLDCOL CURSORPOS)
          [COND
             ((LISTP OLDROW/COL)
              (SETQ OLDROW (CAR OLDROW/COL))
              (SETQ OLDCOL (CDR OLDROW/COL]
      LP  (COND
             ((MOUSESTATE UP)                               ; everything up
              (RETURN)))
          (SETQ CURSORPOS (CURSORPOSITION NIL \DSP CURSORPOS))
          [COND
             ((OR (NEQ (CAR CURSORPOS)
                       OLDX)
                  (NEQ (CDR CURSORPOS)
                       OLDY))                               ; Cursor moved
              (SETQ ROW (- \TTPAGELENGTH \LOC.ROW.0 (IQUOTIENT (- (SETQ OLDY (CDR CURSORPOS))
                                                                  \BMARG)
                                                               \CHARHEIGHT)
                           1))
              (SETQ OLDX (CAR CURSORPOS))
              (COND
                 [(AND (>= OLDX \LMARG)
                       (< OLDX \RMARG)
                       (>= ROW 0))
                  (SETQ ROW (TTNTHLINE ROW))
                  (SETQ COL (- OLDX (fetch (LINE FIRSTCOL) of ROW)))
                  (SETQ COL (bind WIDTH CH (BUF _ (fetch (LINE START) of ROW))
                                  (END _ (fetch (LINE END) of ROW)) while (NEQ BUF END)
                               do                           ; Scan row for the specific character we're pointing at by adding
                                                            ; widths as we go
                                  [SETQ WIDTH (COND
                                                 ((COMPLEXCHARP (SETQ CH (CAR BUF)))
                                                  (fetch (COMPLEXCHAR CPXWIDTH) of CH))
                                                 (T (FCHARWIDTH CH \FONT]
                                  (COND
                                     ((< COL (LRSH WIDTH 1))
                                      (RETURN BUF)))
                                  (SETQ COL (- COL WIDTH))
                                  (SETQ BUF (TTNEXTCHAR BUF))
                               finally (RETURN BUF)))
                  (COND
                     ((OR (NEQ ROW OLDROW)
                          (NEQ COL OLDCOL))                 ; We moved
                      (RETURN (COND
                                 ((LISTP OLDROW/COL)
                                  (FRPLNODE OLDROW/COL ROW COL))
                                 (T (CONS ROW COL]
                  (T (COND
                        ((NEQ OLDROW/COL T)
                         (RETURN T]
          (\TTYBACKGROUND)
          (GO LP])
```

)

;; Auxiliary fns. These are outside the TTYIN block, and are provided to aid the outside world in special interfaces to TTYIN

```
(DEFINEQ
```

(SETREADFN

```
  [LAMBDA (FLG)                                                    (* bvm%: "10-MAR-83 21:46")
    (/SETATOMVAL 'LISPXREADFN (COND
                                  ((AND (NEQ FLG 'READ)
                                        (OR FLG TTYINBSFLG (DISPLAYTERMP))
                                        (FGETD 'TTYINREAD)
                                        (DISPLAYSTARTEDP))
                                   'TTYINREAD)
                                  (T 'READ])
```

(TTYINENTRYFN

```
  [LAMBDA (WINDOW)                                                 (* bvm%: "24-Aug-84 16:31")
    (COND
       ((LASTMOUSESTATE (ONLY RIGHT))
        (PROG [(STATE (WINDOWPROP WINDOW 'TTYINSTATE]
              (APPLY* (OR (AND STATE (fetch (TTYINWINDOWSTATE TTOLDRIGHTFN) of STATE))
                          (FUNCTION DOWINDOWCOM))
                      WINDOW)))
       (T (GIVE.TTY.PROCESS WINDOW])
```

(TTYINREADP

```
  [LAMBDA (FLG)                                                    ; Edited 14-Apr-87 00:25 by bvm:
```

;;; Intended to replace LISPXREADP. Does the right thing when READBUF has just a <cr> in it

```
    (COND
       (READBUF (OR (NEQ (CAR READBUF)
                         HISTSTR0)
                    FLG))
       ((NOT (LINEBUFFER-EOFP \LINEBUF.OFD))
        (OR FLG (NEQ (PEEKBINCCODE \LINEBUF.OFD)
                     (CHARCODE EOL])
```

(TTYINREAD

```
  [LAMBDA (FILE RDTBL)                                             ; Edited 10-Dec-87 17:57 by raf
    (COND
       ([OR (AND TTYINDEBUGFLG \INSIDE.TTYIN)
            (NOT (DISPLAYSTREAMP (GETSTREAM T 'OUTPUT]             ; If debugging and TTYIN breaks, don't want to die
        (READ FILE RDTBL))
       (T (PROG (X)
                (RETURN (COND
                           ((OR (LINEBUFFER-SKIPSEPRS \LINEBUF.OFD RDTBL)
                                (EQ (SETQ X (TTYIN LISPXID NIL NIL '(EVALQT FILLBUFFER NOPROMPT)
                                                   NIL NIL NIL RDTBL))
                                    T))                            ; Don't call TTYIN if there's something significant already in
                                                                   ; buffer
                            ;; SKIPSEPRS used to be (do (COND ((EOFP \LINEBUF.OFD) (* Nothing in buffer) (RETURN)) ((NEQ
                            ;; (PEEKBINCCODE \LINEBUF.OFD) (CHARCODE EOL)) (* significant stuff) (RETURN T)) (T (BINCCODE
                            ;; \LINEBUF.OFD))))
                            (READ \LINEBUF.OFD RDTBL))
                           (T                                      ; indicate null input
                              (SETQ READBUF (NCONC1 (CDR X)
                                                    HISTSTR0))
                              (CAR X])
```

(TTYINFIX

```
  [LAMBDA (INPUT COMS)                                             ; Edited 20-Jan-88 12:13 by bvm
    (LET (TAIL)
         (COND
            ([OR COMS (NEQ LISPXREADFN 'TTYINREAD)
                 (>= (count INPUT)
                     TTYINFIXLIMIT)
                 (CDR (SETQ TAIL (MEMB HISTSTR0 INPUT]
             (NONTTYINLISPXFIX INPUT COMS))
            (T (TTYIN LISPXID NIL NIL (COND
                                         ((for X in [COND
                                                       ((EQ TAIL (CDR INPUT))
                                                        (CAR INPUT))
                                                       (T (OR (LISTP (CADR INPUT))
                                                              (CDR INPUT]
                                             thereis (LISTP X))    ; Something worth prettyprinting
                                         '(PRETTY EVALQT))
                                         (T 'EVALQT))
                      NIL NIL INPUT T])
```

(CHARMACRO?

```
  [NLAMBDA (MACRO)
    (DECLARE (USEDFREE \READING LISPXID))                    (* bvm%: "19-MAR-81 12:15")
```

;;; For use in a TTYINREADMACRO. If we are reading inside the editor, clear the output buffer and return MACRO

```
    (COND
       ((AND (EQ \READING 'EVALQT)
             (EQ LISPXID '*))
        (COND
           ((LISTP MACRO)                                    ; a list of edit commands; we'd better copy
            (APPEND MACRO))
           (T MACRO])
```

(TTYINMETA

```
  [LAMBDA (FLG)                                              (* bvm%: " 2-May-85 14:27")
    (METASHIFT FLG])
```

(TTYIN.LASTINPUT

```
  [LAMBDA NIL                                                ; Edited 24-May-91 10:44 by jds
    (PROG [(BUF (AND (LISTP TTYINBUFFER)
                     (fetch (TTYINBUFFER OLDTAIL) of TTYINBUFFER]
          (RETURN (AND BUF (TTYINSTRING (fetch (LINE START) of (fetch (TTYINBUFFER FIRSTLINE) of TTYINBUFFER))
                                        BUF])

)

(DEFINEQ
```

(TTYINEDIT

```
  [LAMBDA (EXPRS WINDOW PRINTFN PROMPT RDTBL)                ; Edited 19-Jan-88 17:13 by bvm
    (OR PRINTFN (SETQ PRINTFN TTYINPRINTFN))
    (RESETLST
        (SET.TTYINEDIT.WINDOW WINDOW)
        (RESETSAVE (CURSOR T))                               ; Make sure we have something to point with
        (WITH-RESOURCES (TTSCRATCHFILE)
               (PROG1 (TTYIN (OR PROMPT TTYINEDITPROMPT)
                             NIL NIL 'LISPXREAD NIL NIL [COND
                                                          ([OR (EQ PRINTFN T)
                                                               (AND (NULL PRINTFN)
                                                                    (NULL (CDR EXPRS))
                                                                    (STRINGP (CAR EXPRS]
                                                               ; Don't prettyprint it
                                                           EXPRS)
                                                          (T (LIST HISTSTR1
                                                                   (LET* ((\DSP (GETSTREAM WINDOW))
                                                                          (\INITPOS (DSPXPOSITION NIL \DSP))
                                                                          (\RMARG (DSPRIGHTMARGIN NIL \DSP))
                                                                          (\PROMPT1 (OR PROMPT TTYINEDITPROMPT)
                                                                                    ))
                                                                   ; Set up these vars for TTYIN.PPTOFILE to understand the
                                                                   ; environment better
                                                                          (if (EQ \PROMPT1 T)
                                                                              then (SETQ \PROMPT1 NIL))
                                                                          (TTYIN.PPTOFILE EXPRS PRINTFN RDTBL
                                                                                 TTSCRATCHFILE]
                                     RDTBL)
                      (COND
                         ((AND TTYINAUTOCLOSEFLG WINDOW)
                          (CLOSEW WINDOW)))
                      (SETFILEPTR TTSCRATCHFILE 0)))])
```

(SIMPLETEXTEDIT

```
  [LAMBDA (FILE WINDOW)                                      ; Edited 17-Jan-88 15:29 by bvm:
    (RESETLST
        [WITH-RESOURCES (TTSCRATCHFILE)
               (LET (INSTREAM MAINOUTPUT)
                    (SET.TTYINEDIT.WINDOW WINDOW)
                    (COND
                       ([TTYIN TTYINEDITPROMPT NIL NIL '(TEXT NOVALUE)
                               TTSCRATCHFILE NIL (COND
                                                   (FILE        ; User specified a file to edit
                                                         [RESETSAVE NIL (LIST 'CLOSEF (SETQ INSTREAM
                                                                                            (OPENSTREAM FILE
                                                                                                  'INPUT]
                                                         `(,HISTSTR1 (,INSTREAM 0 ,@(GETEOFPTR INSTREAM]
                        (repeatuntil [AND [SETQ MAINOUTPUT (COND
                                                             (INSTREAM
                                                             ; Default output to new version of input
                                                               (PROG1 (PACKFILENAME.STRING 'VERSION NIL
                                                                                           'BODY INSTREAM)
                                                                      (SETQ INSTREAM)))
                                                             [(TTYIN "Output file: " NIL "Name of file for edited
```

```
                                                        output" '(NORAISE STRING]
                                                    ((CL:Y-OR-N-P "Abort edit? ")
                                                        ; Really didn't want to write it anywhere
                                                     (RETURN NIL]
                                          (NLSETQ (SETQ MAINOUTPUT (OPENSTREAM MAINOUTPUT 'OUTPUT]
                                  finally                 ; Copy from scratch file to real output file
                                        (COPYBYTES TTSCRATCHFILE MAINOUTPUT 0 (GETFILEPTR TTSCRATCHFILE))
                                        (SETFILEPTR TTSCRATCHFILE 0)    ; Leave scratch file in good shape
                                        (RETURN (CLOSEF MAINOUTPUT])])
```

(SET.TTYINEDIT.WINDOW

```
  [LAMBDA (WINDOW)                                          (* lmm "14-Nov-86 17:04")
```

;; Changes output to WINDOW, or the TTYIN edit window, returning the resulting WINDOW, or NIL if there is no window change. Caller must have ;; RESETLST

```
    (COND
       ((EQ WINDOW T)                                       ; Use current window
        NIL)
       (T [OR WINDOW (SETQ WINDOW (OR TTYINEDITWINDOW (SETQ TTYINEDITWINDOW (CREATEW NIL "Edit Work Area"]
          (CLEARW WINDOW)
          [PROG [(OFFSET (IREMAINDER (WINDOWPROP WINDOW 'HEIGHT)
                                     (IMINUS (DSPLINEFEED NIL WINDOW]
                (COND
                   ((NEQ OFFSET 0)
```

;; Window is not an integral number of lines, so start down a little, so that bottom line will be correctly aligned (we count from ;; bottom of screen)

```
                    (RELMOVETO 0 (IMINUS OFFSET)
                           WINDOW]
          (RESETSAVE (TTYDISPLAYSTREAM WINDOW))
          WINDOW])
```

(TTYIN.PPTOFILE

```
  [LAMBDA (EXPRS PRINTFN RDTBL STREAM)                      ; Edited 19-Jan-88 17:19 by bvm
```

;;; Prettyprint each of EXPRS to a scratch file, returning (scratchfile start . end), as TTYIN would like. If STREAM is supplied, it is a scratch stream

```
    (LET ([*STANDARD-OUTPUT* (OR STREAM (OPENSTREAM '{NODIRCORE} 'BOTH]
          (*READTABLE* (\GTREADTABLE RDTBL T))
          (*PRINT-ARRAY* T)
          (*PRINT-STRUCTURE* T)
          (FONTCHANGEFLG NIL))
         (DECLARE (CL:SPECIAL FONTCHANGEFLG))               ; The others already are
         (SETFILEPTR *STANDARD-OUTPUT* 0)
         (LINELENGTH (- (IQUOTIENT (- \RMARG \INITPOS)
                                   (CHARWIDTH (CHARCODE X)
                                              \DSP))
                        (if \PROMPT1
                            then (NCHARS \PROMPT1)
                          else 0))
                *STANDARD-OUTPUT*)                          ; Prettyprint to a linelength that accounts for available space,
                                                            ; excluding margins and prompt
         (COND
            ((AND PRINTFN (NEQ PRINTFN 'PRETTY))
             (CL:FUNCALL (COND
                            ((EQ PRINTFN T)
                             'PRINT)
                            (T PRINTFN))
                    EXPRS *STANDARD-OUTPUT*))
            ((AND (CDR EXPRS)
                  (NLISTP (CAR EXPRS)))                     ; Be careful not to separate exec command or apply fn from its
                                                            ; args
             (PRIN2 (CAR EXPRS)
                    *STANDARD-OUTPUT*)
             (SPACES 1 *STANDARD-OUTPUT*)
             (PRINTDEF (CDR EXPRS)
                    (POSITION)
                    T T NIL *STANDARD-OUTPUT*))
            (T (PRINTDEF EXPRS NIL T T NIL *STANDARD-OUTPUT*)))
         (CONS *STANDARD-OUTPUT* (CONS 0 (GETFILEPTR *STANDARD-OUTPUT*])

)

;; New, correct way of getting scratch file

(DEFINEQ

(MAKE-TTSCRATCHFILE
  [LAMBDA NIL                                               ; Edited 17-Jan-88 15:16 by bvm:
    (OPENSTREAM '{NODIRCORE} 'BOTH])

)

(DECLARE%: EVAL@COMPILE
```

```
[PUTDEF 'TTSCRATCHFILE 'RESOURCES '(NEW (MAKE-TTSCRATCHFILE]
)

;; Obsolete, but maybe someone calls it

(DEFINEQ

(TTYIN.SCRATCHFILE
  [LAMBDA NIL
    (DECLARE (GLOBALVARS TTYINEDIT.SCRATCH))                    (* lmm "14-Nov-86 17:05")
    [COND
       ([OR (NOT TTYINEDIT.SCRATCH)
            (NOT (OPENP TTYINEDIT.SCRATCH 'BOTH]
        (SETQ TTYINEDIT.SCRATCH (OPENSTREAM '{NODIRCORE} 'BOTH 'OLD/NEW NIL
                                       (CONSTANT (LIST (LIST 'ENDOFSTREAMOP (FUNCTION \TTYIN.RPEOF]
    (SETFILEPTR TTYINEDIT.SCRATCH 0)
    TTYINEDIT.SCRATCH])

(\TTYIN.RPEOF
  [LAMBDA (STREAM)                                              (* lmm "14-Nov-86 17:06")
                                                                ; End of stream op for ttyin scratch file -- supplies as many
                                                                ; closing parens as needed
    (CHARCODE ")"])

)

(RPAQ? TTYINEDIT.SCRATCH )

(RPAQ? TTYINEDITWINDOW )

(RPAQ? TTYINEDITPROMPT T)

(RPAQ? TTYINAUTOCLOSEFLG )

(RPAQ? TTYINPRINTFN )

(RPAQ? TTYIN?=FN )

;; Kludge of the week

(DEFINEQ

(TTYINPROMPTFORWORD
  [LAMBDA (PROMPT.STR CANDIDATE.STR GENERATE?LIST.FN ECHO.CHANNEL DONTECHOTYPEIN.FLG URGENCY.OPTION
                TERMINCHARS.LST KEYBD.CHANNEL)                  ; Edited  8-Feb-88 14:26 by bvm:
    ;; Attempt at a plug-compatible replacement for common cases of PROMPTFORWORD -- lets you use your mouse and other editing commands.
    (LET ((TYPE 'PROMPTFORWORD))                                ; Default uses space or cr to terminate
         (if [OR DONTECHOTYPEIN.FLG KEYBD.CHANNEL [if (NULL TERMINCHARS.LST)
                                                      then (SETQ TYPE 'PROMPTFORWORD-SPACE)
                                                                ; Default is CR SPACE
                                                           NIL
                                                 else (for C in TERMINCHARS.LST
                                                         do (SELCHARQ C
                                                                (SPACE (SETQ TYPE 'PROMPTFORWORD-SPACE))
                                                                ((CR ^X)
                                                                ; ok, ttyin uses these by default

                                                                    )
                                                                (if TTYIN.USE.EXACT.CHARS
                                                                    then
                                                                ; A terminator we can't handle
                                                                      (RETURN T]
                 (AND ECHO.CHANNEL (NOT (DISPLAYSTREAMP (SETQ ECHO.CHANNEL (GETSTREAM ECHO.CHANNEL 'OUTPUT]
             then                                               ; Sorry, can't help
                  (NON-TTYIN-PROMPTFORWORD PROMPT.STR CANDIDATE.STR GENERATE?LIST.FN ECHO.CHANNEL
                        DONTECHOTYPEIN.FLG URGENCY.OPTION TERMINCHARS.LST KEYBD.CHANNEL)
           else (RESETLST
                    [if (AND (EQ URGENCY.OPTION 'TTY)
                             (NOT (TTY.PROCESSP)))
                        then                                    ; Caller wants to grab tty
                             (RESETSAVE (TTY.PROCESS (THIS.PROCESS]
                    (if (AND ECHO.CHANNEL (NEQ ECHO.CHANNEL (TTYDISPLAYSTREAM)))
                        then (RESETSAVE (TTYDISPLAYSTREAM ECHO.CHANNEL)))
                    (TTYIN (COND
                              ((NOT PROMPT.STR)
                               T)
                              ((EQ (NTHCHARCODE PROMPT.STR -1)
                                   (CHARCODE SPACE))
                               PROMPT.STR)
                              (T                                ; Promptforword spaces after prompt
                                 (CONCAT PROMPT.STR " ")))
                           NIL
                           (STRINGP GENERATE?LIST.FN)
                           TYPE NIL NIL (if (FIXP CANDIDATE.STR)
```

```
                                                    then                    ; Coerce integer to string, or otherwise ttyin will interpret it as a
                                                                            ; character code
                                                         (MKSTRING CANDIDATE.STR)
                                                    else CANDIDATE.STR)))])

)

(RPAQ? TTYIN.USE.EXACT.CHARS )

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD? 'PROMPTFORWORD 'NON-TTYIN-PROMPTFORWORD NIL T)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(RPAQQ TTCOMPILETIME
       [(VARS TTYINBLOCKS)
        (LOCALVARS . T)
        (SPECVARS HELP RDTBL SPLST TABS OPTIONS ECHOTOFILE \ARROW \AUTOFILL \BMARG \BUFFER \CHARHEIGHT
               \CHARWIDTH \COMMAND \CURSOR \CURSORCOL \CURSORROW \DELETING \DESCENT \ENDBUFFER \FIRSTLINE \FIX
               \HOMECOL \HOMEROW \INITPOS \LASTAIL \LASTCHAR \LMARG \LOC.ROW.0 \NOFIXSPELL \PROMPT1 \PROMPT2
               \READING \REPEAT \RMARG \INSIDE.TTYIN \TTYINSTATE \TTPAGELENGTH \RAISEINPUT \FIRSTTIME
               \DONTCOMPLETE \NOVALUE \STRINGVALUE \LISPXREADING \FILLINGBUFFER \RDTBLSA \LAST.DELETION \FONT
               \TEXTURE \LASTAILROW \LASTAILCOL \TTYINBUFFERSTREAM \PROMPTFORWORD \PFW.FIRSTTIME \DSP \INITCRLFS
               \COMMENTFONT)
        (GLOBALVARS ?ACTIVATEFLG CTRLUFLG CTRLVFLG EDITPREFIXCHAR EOLCHARCODE HISTSTR0 HISTSTR1 SPELLSTR1
               LASTMOUSEBUTTONS LASTWORD LISPXREADFN SHOWPARENFLG SPELLSTR1 SPELLSTR2 TTYINAUTOCLOSEFLG
               TTYINBSFLG TTYINBUFFER TTYINCOMMENTCHAR TTYINCOMPLETEFLG TTYINEDITPROMPT TTYINEDITWINDOW
               TTYINERRORSETFLG TTYINRAISEFLG TTYINREADMACROS TTYINRESPONSES TTYINUSERFN TTYJUSTLENGTH
               TYPEAHEADFLG USERWORDS null TTYINAUTOFILLMARGIN TTYINPRINTFN TTYIN?=FN TTYINFIXLIMIT
               TTYINDEBUGFLG DORADO.RESTORE.BUF.CODES TTYIN.RESTORE.BUF.CODES \RESTOREBUFCODES)
        (MACROS * TTYINMACROS)
        (RECORDS LINE TTYINBUFFER TTYINWINDOWSTATE MOUSEPOS COMPLEXCHAR TTYINBUFFERSTREAM)
        (VARS DMCHARCODES TTSUPPORTFNS)
        (ADDVARS (DONTCOMPILEFNS DELETETO1))
        (CONSTANTS (DISPLAYTERMFLG T)
               (TTYINMAILFLG)
               (DIDESCAPECODE 283)
               DOTSHADE)
        (VARS TTNILFNS)
        (MACROS * TTNILFNS)
        (DECLARE%: DONTEVAL@COMPILE (TEMPLATES TTBOUT TTBOUTN)
               DONTEVAL@LOAD EVAL@COMPILE (VARS (DONTCOMPILEFNS (UNION (UNION TTYINMACROS TTSUPPORTFNS)
                                                                        DONTCOMPILEFNS])

(RPAQQ TTYINBLOCKS
       ((TTYIN TTYIN TTBIN TTCRLF TTCRLF.ACCOUNT SCANFORWARD TTNLEFT TTNTH TTPRIN1 TTPROMPTCHAR TTRATOM TTREAD
               TTREADLIST TTSKIPSEPR TTSKREAD TTYINSTRING ADDCHAR ADDNAKEDCHAR AUTOCR? BACKWARD.DELETE.TO BEEP
               BUFTAILP CLEAR.LINE? CREATE.LINE DELETE.TO.END DELETETO DELETETO1 DELNCHARS TTECHO.TO.FILE
               END.DELETE.MODE ENDREAD? AT.END.OF.TEXT FIND.START.OF.WORD TTADJUSTWORD FORWARD.DELETE.TO
               GO.TO.RELATIVE GO.TO.ADDRESSING GO.TO.FREELINE INIT.CURSOR INSERT.CHAR.IN.BUF ADDCHARS.INSERTING
               INSERT.NODE TTRUBOUT KILL.LINES KILLSEGMENT MOVE.BACK.TO MOVE.FORWARD.TO MOVE.TO.NEXT.LINE
               START.NEW.LINE TTNEXTCHAR TTNEXTNODE OVERFLOW? PROPERTAILP RESTORE.CURSOR SAVE.CURSOR SCRATCHCONS
               SETLASTC SETTAIL? SPACE/PARENP DO.EDIT.COMMAND ADDSILENTCHAR TTADDTAB AT.END.OF.SCREEN SCANBACK
               BACKSKREAD BREAKLINE SEGMENT.LENGTH CHECK.MARGIN TTCOMPLETEWORD FIND.MATCHING.WORD NTHCHARCODE
               DELETELINE DO?CMD TTDOTABS EDITCOLUMN FIND.LINE FIND.LINE.BREAK ADJUSTLINE START.OF.PARAGRAPH?
               ADJUSTLINE.AND.RESTORE TTGIVEHELP TTGIVEHELP1 TTGIVEHELP2 INSERTLINE TTLASTLINE TTLOADBUF
               MOVE.TO.LINE MOVE.TO.START.OF.WORD MOVE.TO.WHEREVER TTNEXTLINE FIND.MATCHING.QUOTE FIND.NEXT.WORD
               NTH.COLUMN.OF NTH.RELATIVE.COLUMN.OF OVERFLOWLINE? PREVLINE PREVWORD READFROMBUF RENUMBER.LINES
               RESTOREBUF RETYPE.BUFFER SHOW.MATCHING.PAREN SKIP/ZAP SLEEP CURRENT.WORD TYPE.BUFFER U/L-CASE
               TTUNREADBUF DO.BACK DO.DELETE.LINES DO.DOWN DO.FORWARD DO.INSERT.LINE DO.UP ERASE.SCREEN
               ERASE.TO.END.OF.LINE ERASE.TO.END.OF.PAGE INSERT.TEXT INSERTNCHARS TTSETCURSOR
               (LOCALFREEVARS ARROW AUTOFILL BUFFER COMMAND CURSOR DELETING EDITBIT ENDBUFFER INITPOS INSERTING
                      NOFIXSPELL READING REPEAT)
               (SPECVARS CTRLO!)
               (LINKFNS . T)
               (NOLINKFNS DISPLAYHELP DISPLAYTERMP EDITE ERROR! FIXSPELL!! GRIPE GUESTUSER? MAILWATCH MWNOTE
                      SETBACKSPACE SHOULDNT SMARTARGLIST SPRINTT STKEVAL STRPOS USEREXEC XHELPSYS)
               (BLKLIBRARY NLEFT))
        (NIL TTYINREAD (LOCALVARS . T)
             (LINKFNS TTYIN))
        (NIL DISPLAYTERMP SETREADFN TTECHOMODE TTED TTYINPEEKC TTYINREADP TTYINREADPREP CHARMACRO? (LOCALVARS
                                                                                                    . T)
             )))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS HELP RDTBL SPLST TABS OPTIONS ECHOTOFILE \ARROW \AUTOFILL \BMARG \BUFFER \CHARHEIGHT \CHARWIDTH
       \COMMAND \CURSOR \CURSORCOL \CURSORROW \DELETING \DESCENT \ENDBUFFER \FIRSTLINE \FIX \HOMECOL \HOMEROW
       \INITPOS \LASTAIL \LASTCHAR \LMARG \LOC.ROW.0 \NOFIXSPELL \PROMPT1 \PROMPT2 \READING \REPEAT \RMARG
       \INSIDE.TTYIN \TTYINSTATE \TTPAGELENGTH \RAISEINPUT \FIRSTTIME \DONTCOMPLETE \NOVALUE \STRINGVALUE
```

```
       \LISPXREADING \FILLINGBUFFER \RDTBLSA \LAST.DELETION \FONT \TEXTURE \LASTAILROW \LASTAILCOL
       \TTYINBUFFERSTREAM \PROMPTFORWORD \PFW.FIRSTTIME \DSP \INITCRLFS \COMMENTFONT)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ?ACTIVATEFLG CTRLUFLG CTRLVFLG EDITPREFIXCHAR EOLCHARCODE HISTSTR0 HISTSTR1 SPELLSTR1
       LASTMOUSEBUTTONS LASTWORD LISPXREADFN SHOWPARENFLG SPELLSTR1 SPELLSTR2 TTYINAUTOCLOSEFLG TTYINBSFLG
       TTYINBUFFER TTYINCOMMENTCHAR TTYINCOMPLETEFLG TTYINEDITPROMPT TTYINEDITWINDOW TTYINERRORSETFLG
       TTYINRAISEFLG TTYINREADMACROS TTYINRESPONSES TTYINUSERFN TTYJUSTLENGTH TYPEAHEADFLG USERWORDS null
       TTYINAUTOFILLMARGIN TTYINPRINTFN TTYIN?=FN TTYINFIXLIMIT TTYINDEBUGFLG DORADO.RESTORE.BUF.CODES
       TTYIN.RESTORE.BUF.CODES \RESTOREBUFCODES)
)

(RPAQQ TTYINMACROS (TYPEAHEAD? AT.END.OF.BUF AT.END.OF.LINE AT.START.OF.BUF AT.START.OF.LINE BEFOREBUF
                           BREAK.OR.SEPRP DISPLAYTERMP EMPTY.BUFFER EMPTY.LINE EQPOS NEQPOS INPART ON.FIRST.LINE
                           ON.LAST.LINE METACHARP NONMETACHARBITS METACHAR COMPLEXCHARP STREAMBYTESPERCHAR
                           SPACEP TTBOUT TTNEXTCHAR BOUTCCODE PEEKBINCCODE BINCCODE WORDSEPRP FCHARWIDTH
                           FIRSTCHAR))

(DECLARE%: EVAL@COMPILE

(PUTPROPS TYPEAHEAD? MACRO (NIL (\SYSBUFP)))

(PUTPROPS AT.END.OF.BUF MACRO (NIL (EQ \CURSOR \ENDBUFFER)))

(PUTPROPS AT.END.OF.LINE MACRO (NIL (EQ (fetch END of \ARROW)
                                        \CURSOR)))

(PUTPROPS AT.START.OF.BUF MACRO (NIL (EQ \CURSOR \BUFFER)))

(PUTPROPS AT.START.OF.LINE MACRO (NIL (EQ (fetch START of \ARROW)
                                          \CURSOR)))

(PUTPROPS BEFOREBUF MACRO ((THIS THAT END)
                           (BUFTAILP THAT THIS END)))

(PUTPROPS BREAK.OR.SEPRP MACRO ((C)
                                (fetch STOPATOM of (\SYNCODE \RDTBLSA C))))

(PUTPROPS DISPLAYTERMP ALTOMACRO (NIL T))

(PUTPROPS EMPTY.BUFFER MACRO (NIL (EQ \BUFFER \ENDBUFFER)))

(PUTPROPS EMPTY.LINE MACRO [X (SUBST (OR (CAR X)
                                         '\ARROW)
                                     '\ARROW
                                     '(EQ (fetch START of \ARROW)
                                          (fetch END of \ARROW])

(PUTPROPS EQPOS MACRO [(X Y)
                       (AND (EQ (fetch COLPOS of X)
                                (fetch COLPOS of Y))
                            (EQ (fetch ROWPOS of X)
                                (fetch ROWPOS of Y])

(PUTPROPS NEQPOS MACRO ((X Y)
                        (NOT (EQPOS X Y))))

(PUTPROPS INPART MACRO (OPENLAMBDA (X)
                         (COND
                            ((LISTP X)
                             (CAR X))
                            (T X))))

(PUTPROPS ON.FIRST.LINE MACRO (NIL (EQ \FIRSTLINE \ARROW)))

(PUTPROPS ON.LAST.LINE MACRO (NIL (EQ (fetch END of \ARROW)
                                      \ENDBUFFER)))

(PUTPROPS METACHARP MACRO ((C)
                           (EQ (LRSH C 8)
                               1)))

(PUTPROPS NONMETACHARBITS MACRO ((C)
                                 (LOGAND C 255)))

(PUTPROPS METACHAR MACRO ((C)
                          (LOGOR C 256)))

(PUTPROPS COMPLEXCHARP MACRO (= . LISTP))

(PUTPROPS STREAMBYTESPERCHAR MACRO ((STREAM)
                                    (COND
                                       ((\RUNCODED STREAM)
                                        1)
                                       (T 2))))
```

```
(PUTPROPS SPACEP MACRO [(CHAR)
                        (FMEMB CHAR (CHARCODE (SPACE TAB CR])

(PUTPROPS TTBOUT MACRO [X (CONS 'PROGN (for ARG in X collect (LIST 'BLTCHAR (OR (FIXP ARG)
                                                                                (CDR (ASSOC ARG DMCHARCODES))
                                                                                (AND (EQ (NCHARS ARG)
                                                                                         1)
                                                                                     (CHCON1 ARG))
                                                                                ARG)
                                                                    '(TTYDISPLAYSTREAM])

(PUTPROPS TTNEXTCHAR MACRO (= . CDR))

(PUTPROPS BOUTCCODE MACRO (OPENLAMBDA (STREAM CHAR)
                            (PRINTCCODE CHAR STREAM)))

(PUTPROPS PEEKBINCCODE MACRO (= . PEEKCCODE))

(PUTPROPS BINCCODE MACRO (= . READCCODE))

(PUTPROPS WORDSEPRP DMACRO [OPENLAMBDA (X)
                             (OR (EQ (\SYNCODE \PRIMTERMSA X)
                                     WORDSEPR.TC)
                                 (fetch STOPATOM of (\SYNCODE \RDTBLSA X])

(PUTPROPS FCHARWIDTH MACRO (= . CHARWIDTH))

(PUTPROPS FIRSTCHAR MACRO ((BUF)
                           ([LAMBDA (CH)
                              (DECLARE (LOCALVARS CH))
                              (COND
                                 ((COMPLEXCHARP CH)
                                  (fetch CPXREALCHAR of CH))
                                 (T CH]
                            (CAR BUF))))
)

(DECLARE%: EVAL@COMPILE

(RECORD LINE (START END FIRSTCOL LASTCOL ROW . NEXTLINE))

(RECORD TTYINBUFFER (FIRSTLINE OLDTAIL LASTSKIP LASTSKIPCHAR STORAGECOUNTER TTYINWINDOW . TTYINWINDOWSTATE)
       (SUBRECORD TTYINWINDOWSTATE)
       STORAGECOUNTER _ 0)

(RECORD TTYINWINDOWSTATE (TTOLDBUTTONFN TTOLDRIGHTFN TTOLDENTRYFN))

(RECORD MOUSEPOS (ROWPOS . COLPOS))

(RECORD COMPLEXCHAR (CPXREALCHAR CPXWIDTH CPXNCHARS . CPXPRINTCHARS))

[ACCESSFNS TTYINBUFFERSTREAM ((TTYINPUT (fetch (STREAM F1) of DATUM)
                                       (replace (STREAM F1) of DATUM with NEWVALUE))
                             (TTYEOF (fetch (STREAM F2) of DATUM)
                                       (replace (STREAM F2) of DATUM with NEWVALUE))
                             (TTYEOFACTION (fetch (STREAM F3) of DATUM)
                                       (replace (STREAM F3) of DATUM with NEWVALUE))
                             (TTYORIGINPUT (fetch (STREAM F4) of DATUM)
                                       (replace (STREAM F4) of DATUM with NEWVALUE]
)

(RPAQQ DMCHARCODES
       ((HOME . 2)
        (BELL . 7)
        (DELCH . 8)
        (BS . 8)
        (DOWN . 10)
        (INSERT.LINE . 10)
        (LF . 10)
        (ADDR . 12)
        (CR . 13)
        (BLINKON . 14)
        (INSERT/DELETE . 16)
        (DLE . 16)
        (ERASE.TO.END . 23)
        (CANCEL . 24)
        (UP . 26)
        (DELETE.LINE . 26)
        (ESC . 27)
        (FORWARD . 28)
        (ROLL . 29)
        (ERASE . 30)
        (CLEAR . 30)
        (US . 31)
        (SPACE . 32)))
```

```
(RPAQQ TTSUPPORTFNS NIL)

(ADDTOVAR DONTCOMPILEFNS DELETETO1)

(DECLARE%: EVAL@COMPILE

(RPAQQ DISPLAYTERMFLG T)

(RPAQQ TTYINMAILFLG NIL)

(RPAQQ DIDESCAPECODE 283)

(RPAQQ DOTSHADE 13260)

(CONSTANTS (DISPLAYTERMFLG T)
       (TTYINMAILFLG)
       (DIDESCAPECODE 283)
       DOTSHADE)
)

(RPAQQ TTNILFNS (BINARY.MODE RESTOREMOD CANCEL.MODES GUESTUSER?))

(RPAQQ TTNILFNS (BINARY.MODE RESTOREMOD CANCEL.MODES GUESTUSER?))

(DECLARE%: EVAL@COMPILE

(PUTPROPS BINARY.MODE MACRO (NIL NIL))

(PUTPROPS RESTOREMOD MACRO (NIL NIL))

(PUTPROPS CANCEL.MODES MACRO (NIL NIL))

(PUTPROPS GUESTUSER? MACRO (NIL NIL))
)

(DECLARE%: DONTEVAL@COMPILE

[SETTEMPLATE 'TTBOUT '(CALL |..| (IF [OR (LISTP EXPR)
                                         (AND (NTHCHAR EXPR 2)
                                              (NOT (ASSOC EXPR DMCHARCODES]
                                    EVAL NIL]

[SETTEMPLATE 'TTBOUTN '(MACRO (X . Y)
                              (FRPTQ X (TTBOUT . Y]
EVAL@COMPILE

(RPAQ DONTCOMPILEFNS (UNION (UNION TTYINMACROS TTSUPPORTFNS)
                            DONTCOMPILEFNS))
)
)

(RPAQ? DORADO.RESTORE.BUF.CODES '(194))

(RPAQ? TTYIN.RESTORE.BUF.CODES '(516 530))

(RPAQ? TTYINBUFFER )

(RPAQ? ?ACTIVATEFLG T)

(RPAQ? EDITPREFIXCHAR )

(RPAQ? SHOWPARENFLG T)

(RPAQ? TTYINBSFLG T)

(RPAQ? \TTYIN.LAST.FONT )

(RPAQ? \TTYIN.LAST.COMMENTFONT )

(RPAQ? TTYINFILLDEFAULT T)

(RPAQ? TTYINCOMPLETEFLG T)

(RPAQ? TTYINUSERFN )

(RPAQ? TYPEAHEADFLG T)

(RPAQ? null "")

(RPAQ? DEFAULTPROMPT "** ")

(RPAQ? TTYJUSTLENGTH -1)

(RPAQ? \INSIDE.TTYIN )

(RPAQ? TTYINERRORSETFLG )
```

```
(RPAQ? TTYINRAISEFLG T)

(RPAQ? TTYINAUTOFILLMARGIN 8)

(RPAQ? TTYINFIXLIMIT 50)

(RPAQ? TTYINDEBUGFLG )

(RPAQ? HISTSTR1 "from file:")

(RPAQ? TTYINCOMMENTCHAR )

(RPAQ? \RESTOREBUFCODES )

(MOVD? 'NILL 'GUESTUSER?)

(MOVD? 'FIXSPELL 'FIXSPELL!!)

(MOVD? 'HELPSYS 'XHELPSYS)

[PUTDQ? SPRINTT (LAMBDA (X)
                  (PRIN1 X]

(MOVD? 'NILL 'WINDOWWORLD)

(MOVD? 'LISPXFIX 'NONTTYINLISPXFIX)

(ADDTOVAR TTYINREADMACROS )

(ADDTOVAR TTYINRESPONSES )

(ADDTOVAR LISPXCOMS (STOP . OK))

(ADDTOVAR \SYSTEMCACHEVARS \RESTOREBUFCODES)

(PUTPROPS TTYINREADMACROS VARTYPE ALIST)

(DECLARE%: DONTEVAL@LOAD DOCOPY

[COND
   ((CCODEP 'TTYIN)
    (CHANGENAME 'PROMPTCHAR 'LISPXREADP 'TTYINREADP)
    (SETREADFN)
    (MOVD 'TTYINFIX 'LISPXFIX]
)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML CHARMACRO?)

(ADDTOVAR LAMA )
)

(PUTPROPS TTYIN COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1991 1993))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

RECORD INDEX

CONSTANT INDEX

TEMPLATE INDEX

PROPERTY INDEX