```
   File created:  20-Mar-2024 11:07:35  {WMEDLEY}<library>tedit>TEDIT-SCREEN.;645

        edit by:  rmk

     changes to:  (FNS \TEDIT.DISPLAYLINE \TEDIT.RAISE.LINES)

  previous date:  20-Mar-2024 07:27:05  {WMEDLEY}<library>tedit>TEDIT-SCREEN.;644

     Read Table:  INTERLISP

        Package:  INTERLISP

         Format:   XCCS


(RPAQQ TEDIT-SCREENCOMS
       ([DECLARE%: EVAL@COMPILE DONTCOPY
               (EXPORT (RECORDS THISLINE LINECACHE)
                      (COMS                                     ; LINEDESCRIPTORS
                            (RECORDS LINEDESCRIPTOR)
                            (I.S.OPRS inlines backlines)
                            (MACROS GETLD FGETLD SETLD FSETLD SETYPOS LINKLD))
                      (MACROS HCSCALE HCUNSCALE)
                      (GLOBALVARS TEDIT.DONT.BREAK.CHARS TEDIT.DONT.LAST.CHARS)
                      (ALISTS (CHARACTERNAMES EM-DASH SOFT-HYPHEN NONBREAKING-HYPHEN NONBREAKING-SPACE))
                      (COMS                                     ; Formatting slots held by THISLINE
                            (RECORDS CHARSLOT)
                            (MACROS CHAR CHARW PREVCHARSLOT PREVCHARSLOT! NEXTCHARSLOT FIRSTCHARSLOT NTHCHARSLOT
                                   LASTCHARSLOT FILLCHARSLOT BACKCHARS PUSHCHAR POPCHAR CHARSLOTP)
                            (CONSTANTS (CELLSPERCHARSLOT 2)
                                   (WORDSPERCHARSLOT (TIMES CELLSPERCHARSLOT WORDSPERCELL))
                                   (MAXCHARSLOTS 256))

                            ;; incharslots can be used only if THISLINE is properly bound in the environment, to provide upperbound checking.
                            ;; Operand can be THISLINE (= FIRSTCHARSLOT) or a within-range slot pointer.  The latter case is not current
                            ;; checked for validity (some \HILOC \LOLOC address calculations?). backcharslots runs backwards.

                            (I.S.OPRS incharslots backcharslots)
                            (MACROS DIACRITICP]
        (FNS \TEDIT.LINEDESCRIPTOR.DEFPRINT)
        (INITRECORDS THISLINE LINEDESCRIPTOR LINECACHE)
        (DECLARE%: EVAL@COMPILE DONTCOPY                        ; Not exported
               (MACROS SPACEBREAK SAVEBREAK DOBREAK FORCEBREAK FORGETHYPHENBREAK FORGETPREVIOUSBREAK)
               (RECORDS PENDINGTAB))
        (INITRECORDS PENDINGTAB)
        (FNS \TEDIT.FORMATLINE \TEDIT.FORMATLINE.SETUP \TEDIT.FORMATLINE.HORIZONTAL \TEDIT.FORMATLINE.VERTICAL
             \TEDIT.FORMATLINE.JUSTIFY \TEDIT.FORMATLINE.TABS \TEDIT.FORMATLINE.SCALETABS
             \TEDIT.FORMATLINE.PURGE.SPACES \TEDIT.FORMATLINE.EMPTY \TEDIT.FORMATLINE.UPDATELOOKS
             \TEDIT.FORMATLINE.LASTLEGAL \TEDIT.LINES.ABOVE)
        (INITVARS (TEDIT.LINELEADING.BELOW NIL))
        (GLOBALVARS TEDIT.LINELEADING.BELOW)
        (FNS \CLEARTHISLINE \TLVALIDATE)
                                                                ; Consistency checking
        (INITVARS *TEDIT-CACHED-FMTSPEC*)
                                                                ; Heuristic for \FORMATLINE
        (GLOBALVARS *TEDIT-CACHED-FMTSPEC*)
        (FNS \TEDIT.DISPLAYLINE \TEDIT.DISPLAYLINE.TABS \TEDIT.LINECACHE \TEDIT.CREATE.LINECACHE \TEDIT.BLTCHAR
             \TEDIT.DIACRITIC.SHIFT)
        (DECLARE%: EVAL@COMPILE DONTCOPY

               ;; Machine independent version of \TEDIT.BLTCHAR

               (MACROS MI-TEDIT.BLTCHAR))
        (FNS \TEDIT.UPDATE.SCREEN \TEDIT.BACKFORMAT \TEDIT.PREVIOUS.LINEBREAK \TEDIT.FILLPANE
             \TEDIT.UPDATE.LINES \TEDIT.CREATEPLINE \TEDIT.FIND.DIRTYCHARS \TEDIT.LINES.BELOW \FORMAT.GAP.LINES
             \TEDIT.LOWER.LINES \TEDIT.RAISE.LINES \TEDIT.VALID.LINES \TEDIT.CLEARPANE.BELOW.LINE
             \TEDIT.INSERTLINE \TEDIT.INSURE.TRAILING.LINE \TEDIT.MARK.LINES.DIRTY \TEDIT.LINE.BOTTOM
             \TEDIT.NCONC.LINES)))

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(DATATYPE THISLINE (;; Cache for line-related character location info, for selection and line-display code to use.

                    (DESC FULLXPOINTER)                         ; Line descriptor for the line this describes now
                    TLSPACEFACTOR                               ; The SPACEFACTOR to be used in printing this line
                    TLFIRSTSPACE                                ; The first space to which SPACEFACTOR is to apply.  This is
                                                                ; used sothat spaces to the left of a TAB have their default width.
                    CHARSLOTS                                   ; Pointer block holdomg char/width slots MAXCHARSLOTS (with
                                                                ; an extra slot so that there is always storage behind
                                                                ; NEXTAVAILABLECHARSLOT
                    NEXTAVAILABLECHARSLOT)                      ; The last used CHARSLOT is at (PREVCHARSLOT
                                                                ; NEXTAVAILABLECHARSLOT)
       CHARSLOTS _ (\ALLOCBLOCK (ITIMES (ADD1 MAXCHARSLOTS)
                                       CELLSPERCHARSLOT)
                          PTRBLOCK.GCT))
```

```
(DATATYPE LINECACHE (;; Image cache for display lines.
                     LCBITMAP                                ; The bitmap that will be used by this instance of the cache
                     (LCNEXTCACHE FULLXPOINTER)              ; The next cache in the chain, for screen updates.

                     ))
)

(/DECLAREDATATYPE 'THISLINE '(FULLXPOINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '10)

(/DECLAREDATATYPE 'LINECACHE '(POINTER FULLXPOINTER)
       ;; ---field descriptor list elided by lister---
       '4)

;; LINEDESCRIPTORS

(DECLARE%: EVAL@COMPILE

(DATATYPE LINEDESCRIPTOR (;; Description of a single line of formatted text, either on the display or for a printed page.
                          YBOT                               ; Y value for the bottom of the line (below the descent)
                          YBASE                              ; Yvalue for the base line the characters sit on
                          LEFTMARGIN                         ; Left margin, in screen points
                          RIGHTMARGIN                        ; Right margin, in screen points
                          LXLIM                              ; X value of right edge of LCHARLIM character on the line (may
                                                             ; exceed right margin, if char is a space.). In natural stream units
                          LX1                                ; X value of the left edge of LCHAR1 from the left margin, in
                                                             ; stream natural units.
                          LHEIGHT                            ; Total height of hte line, Ascent+Descent plus leading
                          ASCENT                             ; Ascent of the line above YBASE, adjusted for line leading
                          DESCENT                            ; How far line descends below YBASE, adjusted for line leading
                          LTRUEDESCENT                       ; The TRUE DESCENT for this line, unadjusted for line leading.
                          LTRUEASCENT                        ; The TRUE ASCENT for this line, unadjusted for pre-paragraph
                                                             ; leading.
                          LCHAR1                             ; CH# of the first character on the line.
                          LCHARLIM                           ; CH# of the last character on the line
                          FORCED-END                         ; NIL or character (EOL, FORM...) that forces a line break
                                                             ; Was CHARTOP: CH# of the character which forced the line
                                                             ; break (may be less than  CHARLIM)
                          NEXTLINE                           ; Next line chain pointer
                          (PREVLINE FULLXPOINTER)            ; Previous line chain pointer
                          LMARK                              ; One of SOLID, GREY, NIL.  Tells what kind of special-line
                                                             ; marker should be put in the left margin for this paragraph.  (For
                                                             ; hardcopy, can also be an indicator for special processing?)
                          LTEXTSTREAM                        ; A cached textstream that this line took its text from.  Filled in by
                                                             ; \TEDIT.FORMATLINE only in hardcopy, used temporarily and
                                                             ; the cleared by \TEDIT.FORMATBOX to avoid the circularity.
                          NIL                                ; Was CACHE: A cached THISLINE, for keeping hardcopy info
                                                             ; around while we crunch with the line descriptors to make things
                                                             ; fit.  Now:  THISLINE comes from TEXTOBJ
                          NIL                                ; Was LDOBJ: The object which lies behind this line of text, for
                                                             ; updating, etc.
                          LFMTSPEC                           ; The format spec for this line's paragraph (eventually)
                          (LDIRTY FLAG)                      ; T if this line has changed since it was last formatted.
                          (NIL FLAG)                         ; Was FORCED-END flag
                          (DELETED FLAG)                     ; T if this line has been completely deleted since it was last
                                                             ; formatted or displayed.  (Used by deletion routines to detect
                                                             ; garbage lines)
                          (LHASPROT FLAG)                    ; This line contains protected text.
                          (LDUMMY FLAG)                      ; This is a dummy line. Was: LHASTABS.  But never fetched and
                                                             ; this descriptions wasn't true: If this line has a tab in it, this is the
                                                             ; line-relative ch# of the final tab.  This is to let us punt properly
                                                             ; with tabs in a line.
                          (1STLN FLAG)                       ; This line is the first line in a paragraph
                          (LSTLN FLAG)                       ; This is the last line in a paragraph

                          )
       (INIT (DEFPRINT 'LINEDESCRIPTOR (FUNCTION \TEDIT.LINEDESCRIPTOR.DEFPRINT)))
       [ACCESSFNS ((YTOP (IPLUS (FGETLD DATUM YBOT)
                                (FGETLD DATUM LHEIGHT)))
                   [LTRUEHEIGHT (IPLUS (FGETLD DATUM LTRUEASCENT (FGETLD DATUM LTRUEDESCENT]
                   (LTRUEYTOP (IPLUS (GETLD DATUM YBOT)
                                     (FGETLD DATUM LTRUEHEIGHT)))
                   (LTRUEYBOT (IDIFFERENCE (FGETLD DATUM YBASE)
                                     (FGETLD DATUM LTRUEDESCENT]
       LHEIGHT _ 0 LTRUEASCENT _ 0 LTRUEDESCENT _ 0 LCHARLIM _ 1000000 NEXTLINE _ NIL PREVLINE _ NIL LDIRTY _
       NIL YBOT _ 0 YBASE _ 0 LEFTMARGIN _ 0 DELETED _ NIL)
)

(/DECLAREDATATYPE 'LINEDESCRIPTOR
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
               POINTER POINTER FULLXPOINTER POINTER POINTER POINTER POINTER POINTER FLAG FLAG FLAG FLAG FLAG
               FLAG FLAG)
```

```
        ;; ---field descriptor list elided by lister---
        '42)

(DEFPRINT 'LINEDESCRIPTOR (FUNCTION \TEDIT.LINEDESCRIPTOR.DEFPRINT))

(DECLARE%: EVAL@COMPILE

[I.S.OPR 'inlines NIL '(bind $$PREVLINE declare (LOCALVARS $$PREVLINE) first (SETQ I.V. (\DTEST (OR BODY
                                                                                                      (GO $$OUT))
                                                                                                  'LINEDESCRIPTOR))
                          by (PROGN (SETQ $$PREVLINE I.V.)
                                    (\DTEST (OR (fetch (LINEDESCRIPTOR NEXTLINE) of I.V.)
                                                (GO $$OUT))
                                           'LINEDESCRIPTOR]

[I.S.OPR 'backlines NIL '(bind $$NEXTLINE declare (LOCALVARS $$NEXTLINE) first (SETQ I.V.
                                                                                     (\DTEST (OR BODY (GO $$OUT))
                                                                                            'LINEDESCRIPTOR))
                            by (PROGN (SETQ $$NEXTLINE I.V.)
                                      (\DTEST (OR (fetch (LINEDESCRIPTOR PREVLINE) of I.V.)
                                                  (GO $$OUT))
                                             'LINEDESCRIPTOR]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS GETLD MACRO ((L FIELD)
                       (fetch (LINEDESCRIPTOR FIELD) of L)))

(PUTPROPS FGETLD MACRO ((L FIELD)
                        (ffetch (LINEDESCRIPTOR FIELD) of L)))

(PUTPROPS SETLD MACRO ((L FIELD NEWVALUE)
                       (replace (LINEDESCRIPTOR FIELD) of L with NEWVALUE)))

(PUTPROPS FSETLD MACRO ((L FIELD NEWVALUE)
                        (freplace (LINEDESCRIPTOR FIELD) of L with NEWVALUE)))

(PUTPROPS SETYPOS MACRO [OPENLAMBDA (LINE BOTTOM)
                          (FSETLD LINE YBASE (IPLUS (GETLD LINE DESCENT)
                                                    (FSETLD LINE YBOT BOTTOM])

(PUTPROPS LINKLD MACRO (OPENLAMBDA (LINE1 LINE2)
                         (CL:WHEN LINE1 (SETLD LINE1 NEXTLINE LINE2))
                         (CL:WHEN LINE2 (SETLD LINE2 PREVLINE LINE1))))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS HCSCALE MACRO [OPENLAMBDA (SCALE ITEM)
                          (CL:IF (LISTP ITEM)
                              (for I in ITEM collect (FIXR (FTIMES SCALE ITEM)))
                              (FIXR (FTIMES SCALE ITEM)))])

(PUTPROPS HCUNSCALE MACRO [OPENLAMBDA (SCALE ITEM)
                            (CL:IF (LISTP ITEM)
                                (for I in ITEM collect (FIXR (FQUOTIENT I SCALE)))
                                (FIXR (FQUOTIENT ITEM SCALE)))])
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.DONT.BREAK.CHARS TEDIT.DONT.LAST.CHARS)
)

(ADDTOVAR CHARACTERNAMES (EM-DASH "357,045")
                         (SOFT-HYPHEN "357,043")
                         (NONBREAKING-HYPHEN "357,042")
                         (NONBREAKING-SPACE "357,041"))

;; Formatting slots held by THISLINE

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD CHARSLOT (CHAR CHARW                                    ; If CHAR is NIL, then CHARW is CHARLOOKS.
                            ))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS CHAR MACRO ((CSLOT)
                      (ffetch (CHARSLOT CHAR) of CSLOT)))

(PUTPROPS CHARW MACRO ((CSLOT)
                       (ffetch (CHARSLOT CHARW) of CSLOT)))
```

```
(PUTPROPS PREVCHARSLOT MACRO ((CSLOT)
                                (\ADDBASE CSLOT (IMINUS WORDSPERCHARSLOT))))

(PUTPROPS PREVCHARSLOT! MACRO ((CSLOT)
                                 ;; Backs over looks and invisibles to the last character slot

                                 (find CS _ (PREVCHARSLOT CSLOT) by (PREVCHARSLOT CS) while CS
                                    suchthat (CHAR CS))))

(PUTPROPS NEXTCHARSLOT MACRO ((CSLOT)
                                (\ADDBASE CSLOT WORDSPERCHARSLOT)))

(PUTPROPS FIRSTCHARSLOT MACRO ((TLINE)
                                 (fetch (THISLINE CHARSLOTS) of TLINE)))

(PUTPROPS NTHCHARSLOT MACRO ((TLINE N)
                               (\ADDBASE (fetch (THISLINE CHARSLOTS) of TLINE)
                                      (ITIMES N WORDSPERCHARSLOT))))

(PUTPROPS LASTCHARSLOT MACRO ((TLINE)
                                (\ADDBASE (fetch (THISLINE CHARSLOTS) of TLINE)
                                       (TIMES (SUB1 MAXCHARSLOTS)
                                              WORDSPERCHARSLOT))))

(PUTPROPS FILLCHARSLOT MACRO ((CSLOT C W)
                               (freplace (CHARSLOT CHAR) of CSLOT with C)
                               (freplace (CHARSLOT CHARW) of CSLOT with W)))

(PUTPROPS BACKCHARS MACRO ((CSLOTVAR CHARVAR WIDTHVAR)
                             (SETQ CSLOTVAR (PREVCHARSLOT CSLOTVAR))
                             (SETQ CHARVAR (fetch (CHARSLOT CHAR) of CSLOTVAR))
                             (SETQ WIDTHVAR (fetch (CHARSLOT CHARW) of CSLOTVAR))))

(PUTPROPS PUSHCHAR MACRO ((CSLOTVAR C W)
                            (FILLCHARSLOT CSLOTVAR C W)
                            (SETQ CSLOTVAR (NEXTCHARSLOT CSLOTVAR))))

(PUTPROPS POPCHAR MACRO ((CSLOTVAR CHARVAR WIDTHVAR)
                           (SETQ CHARVAR (fetch (CHARSLOT CHAR) of CSLOTVAR))
                           (SETQ WIDTHVAR (fetch (CHARSLOT CHARW) of CSLOTVAR))
                           (SETQ CSLOTVAR (NEXTCHARSLOT CSLOTVAR))))

(PUTPROPS CHARSLOTP MACRO [OPENLAMBDA (X TL)
                              ;; True if TL is a THISLINE and X is a pointer into its CHARSLOTS block.  A tool for consistency assertions.

                              (CL:WHEN (TYPE? THISLINE TL)
                                  [LET ((FIRSTSLOT (FIRSTCHARSLOT TL))
                                        (LASTSLOT (LASTCHARSLOT TL)))
                                       (AND [OR (IGREATERP (\HILOC X)
                                                       (\HILOC FIRSTSLOT))
                                                (AND (EQ (\HILOC X)
                                                         (\HILOC FIRSTSLOT))
                                                     (IGEQ (\LOLOC X)
                                                           (\LOLOC FIRSTSLOT]
                                            (OR (ILESSP (\HILOC X)
                                                       (\HILOC LASTSLOT))
                                                (AND (EQ (\HILOC X)
                                                         (\HILOC LASTSLOT))
                                                     (ILEQ (\LOLOC X)
                                                           (\LOLOC LASTSLOT])])
)

(DECLARE%: EVAL@COMPILE

(RPAQQ CELLSPERCHARSLOT 2)

(RPAQ WORDSPERCHARSLOT (TIMES CELLSPERCHARSLOT WORDSPERCELL))

(RPAQQ MAXCHARSLOTS 256)

(CONSTANTS (CELLSPERCHARSLOT 2)
       (WORDSPERCHARSLOT (TIMES CELLSPERCHARSLOT WORDSPERCELL))
       (MAXCHARSLOTS 256))
)

;; incharslots can be used only if THISLINE is properly bound in the environment, to provide upperbound checking.  Operand can be THISLINE (=
;; FIRSTCHARSLOT) or a within-range slot pointer.  The latter case is not current checked for validity (some \HILOC \LOLOC address calculations?).
;; backcharslots runs backwards.

(DECLARE%: EVAL@COMPILE

(I.S.OPR 'incharslots NIL '[SUBST (GETDUMMYVAR)
                                  '$$STARTSLOT
                                  '(bind $$STARTSLOT _ BODY CHAR CHARW $$CHARSLOTLIMIT
                                       declare (LOCALVARS $$STARTSLOT $$CHARSLOTLIMIT)
                                       first (SETQ I.V. (COND
                                                          ((TYPE? THISLINE $$STARTSLOT)
                                                            (FIRSTCHARSLOT $$STARTSLOT))
                                                           (T $$STARTSLOT)))
                                              (SETQ $$CHARSLOTLIMIT (fetch (THISLINE NEXTAVAILABLECHARSLOT)
                                                                      of THISLINE))
                                         by (NEXTCHARSLOT I.V.) until (EQ I.V. $$CHARSLOTLIMIT)
                                         eachtime (SETQ CHAR (fetch (CHARSLOT CHAR) of I.V.))
                                               (SETQ CHARW (fetch (CHARSLOT CHARW) of I.V.]
       T)

(I.S.OPR 'backcharslots NIL '[SUBST (GETDUMMYVAR)
                                    '$$STARTSLOT
                                    '(bind $$STARTSLOT _ BODY CHAR CHARW $$CHARSLOTLIMIT
                                        declare (LOCALVARS $$STARTSLOT $$CHARSLOTLIMIT)
                                        first (SETQ I.V. (COND
                                                            ((TYPE? THISLINE $$STARTSLOT)
                                                             (PREVCHARSLOT (fetch (THISLINE NEXTAVAILABLECHARSLOT)
                                                                              of THISLINE)))
                                                            (T $$STARTSLOT)))
                                              (SETQ $$CHARSLOTLIMIT (FIRSTCHARSLOT THISLINE))
                                        by (PREVCHARSLOT I.V.) eachtime (SETQ CHAR (fetch (CHARSLOT CHAR)
                                                                                      of I.V.))
                                                                       (SETQ CHARW (fetch (CHARSLOT CHARW)
                                                                                     of I.V.))
                                        repeatuntil (EQ I.V. $$CHARSLOTLIMIT]
       T)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS DIACRITICP MACRO (OPENLAMBDA (CHAR)
                                 ;; An XCCS diacritic
                                 (AND (SMALLP CHAR)
                                      (IGEQ CHAR 192)
                                      (ILEQ CHAR 207))))
)
)

;; END EXPORTED DEFINITIONS

(DEFINEQ

(\TEDIT.LINEDESCRIPTOR.DEFPRINT
  [LAMBDA (LINE STREAM)                                      ; Edited  2-Dec-2023 23:05 by rmk
                                                             ; Edited  4-Oct-2023 21:18 by rmk
                                                             ; Edited  3-Jul-2023 22:02 by rmk
                                                             ; Edited 22-May-2023 14:42 by rmk
                                                             ; Edited 21-May-2023 09:15 by rmk
    (LET (INFO LOC)
         (SETQ INFO (CONCAT (CL:IF (GETLD LINE 1STLN)
                                   "*"
                                   "")
                            (GETLD LINE LCHAR1)
                            "-"
                            (GETLD LINE LCHARLIM)
                            (CL:IF (GETLD LINE LSTLN)
                                   "*"
                                   "")
                            (CL:IF (GETLD LINE FORCED-END)
                                   " FE"
                                   "")))
         (SETQ LOC (LOC LINE))
         (CONS (CONCAT "{L" (CL:IF (GETLD LINE LDIRTY)
                                   "D"
                                   "")
                       ":" INFO " " (CAR LOC)
                       "/"
                       (CDR LOC)
                       "}"])
)

(/DECLAREDATATYPE 'THISLINE '(FULLXPOINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '10)

(/DECLAREDATATYPE 'LINEDESCRIPTOR
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
               POINTER POINTER FULLXPOINTER POINTER POINTER POINTER POINTER POINTER FLAG FLAG FLAG FLAG FLAG
               FLAG FLAG)
       ;; ---field descriptor list elided by lister---
       '42)

(DEFPRINT 'LINEDESCRIPTOR (FUNCTION \TEDIT.LINEDESCRIPTOR.DEFPRINT))
```

```
(/DECLAREDATATYPE 'LINECACHE '(POINTER FULLXPOINTER)
         ;; ---field descriptor list elided by lister---
         '4)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS SPACEBREAK MACRO (NIL ;; TX is the beginning of the first space of a run. Needed for SPACELEFT in DOBREAK. FIRSTWHITEX
                                ;; is updated on the first space after one or more non-spaces.
                                (CL:WHEN INWORD
                                    (FORGETHYPHENBREAK)
                                    (SETQ FIRSTWHITEX TX)        ; The beginning of the space
                                    (SETQ FIRSTWHITESLOT CHARSLOT)
                                    (SETQ INWORD NIL)
                                    (SETQ INSPACES T))))

(PUTPROPS SAVEBREAK MACRO (NIL ;; Values including the character just before a break
                               (SETQ ASCENTB TRUEASCENT)
                               (SETQ DESCENTB TRUEDESCENT)
                               (SETQ CHNOB CHNO)
                               (SETQ CHARSLOTB CHARSLOT)
                               (SETQ TXB TX)))

(PUTPROPS DOBREAK MACRO [(SPACERUN)
                         ;; Back up to the last potential break, if TXB says that there was one.  Otherwise, break here.
                         ;; SPACERUN if we are backing up to a space run with unexpandable overhang spaces
                         (CL:WHEN TXB
                             (SETQ TRUEASCENT ASCENTB)
                             (SETQ TRUEDESCENT DESCENTB)
                             (SETQ TX TXB)
                             (SETQ CHNO CHNOB)
                             (SETQ CHARSLOT CHARSLOTB))
                         (COND
                            ((AND SPACERUN FIRSTWHITESLOT)       ; Clear/register the overhangs
                             (CL:WHEN PREVSP
                                 (SETQ PREVSP (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP (fetch (CHARSLOT CHAR)
                                                                                      of FIRSTWHITESLOT))))
                             (SETQ SPACELEFT (IDIFFERENCE WIDTH FIRSTWHITEX))
                             (SETQ OVERHANG (IDIFFERENCE TX FIRSTWHITEX)))
                            (T (SETQ SPACELEFT (IDIFFERENCE WIDTH TX))
                               (SETQ OVERHANG 0])

(PUTPROPS FORCEBREAK MACRO [NIL (SETQ PREVSP (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP))
                                                                 ; All spaces are natural
                                ;; If the EOL comes right after a word-character that was preceded by a space run, those earlier spaces
                                ;; don't count in our overhang.  INSPACES tracks that.
                                (add TX DX)
                                (SETQ OVERHANG (CL:IF INSPACES
                                                   (IDIFFERENCE TX FIRSTWHITEX)
                                                   DX))
                                (SETQ SPACELEFT (IDIFFERENCE WIDTH (IDIFFERENCE TX OVERHANG])

(PUTPROPS FORGETHYPHENBREAK MACRO (NIL (CL:WHEN PREVDHYPH     ; Previous soft hyphen becomes invisible
                                           (add TX (IMINUS (CHARW PREVDHYPH)))
                                           (FILLCHARSLOT PREVDHYPH NIL 1))
                                       (SETQ PREVDHYPH (SETQ PREVHYPH NIL))))

(PUTPROPS FORGETPREVIOUSBREAK MACRO (NIL (FORGETHYPHENBREAK) ; Forget hyphens
                                         (SETQ FIRSTWHITEX 0)
                                         (SETQ FIRSTWHITESLOT NIL)))
)

(DECLARE%: EVAL@COMPILE

(DATATYPE PENDINGTAB (;; The data structure for a tab, within the line formatter, that we haven't finished dealing with yet, e.g. a centered tab where
                      ;; you need to wait for AFTER the centered text to do the formatting.
                      PTRESOLVEDWIDTH
                      ;; Width resolved for a prior tab.  This results from the resolution of an old RIGHT, CENTERED, or DECIMAL tab.
                      PTOLDTAB                                 ; The pending tab
                      PTTYPE                                   ; Its tab type
                      PTTABX                                   ; Its nominal X position
                      (PTCHARSLOT FULLXPOINTER)                ; The CHARSLOT that may need to be updated later.  (RMK:  I
                                                               ; don't know why this is a FULLXPOINTER--maybe an issue in
                                                               ; the older THISLINE implementation?)
                      PTOLDTX                                  ; The TX as of when the tab was encountered.

                      ))
)
```

```
(/DECLAREDATATYPE 'PENDINGTAB '(POINTER POINTER POINTER POINTER FULLXPOINTER POINTER)
          ;; ---field descriptor list elided by lister---
          '12)
)

(/DECLAREDATATYPE 'PENDINGTAB '(POINTER POINTER POINTER POINTER FULLXPOINTER POINTER)
          ;; ---field descriptor list elided by lister---
          '12)

(DEFINEQ
```

(\TEDIT.FORMATLINE

```
  [LAMBDA (TEXTOBJ CH#1 LINE REGION IMAGESTREAM FORMATTINGSTATE) ; Edited 17-Mar-2024 00:27 by rmk
                                                                 ; Edited 15-Mar-2024 19:43 by rmk
                                                                 ; Edited 14-Mar-2024 12:53 by rmk
                                                                 ; Edited  2-Mar-2024 07:39 by rmk
                                                                 ; Edited  5-Feb-2024 09:35 by rmk
                                                                 ; Edited 26-Jan-2024 11:01 by rmk
                                                                 ; Edited  3-Dec-2023 16:48 by rmk
                                                                 ; Edited 27-Nov-2023 23:05 by rmk
                                                                 ; Edited 28-Oct-2023 13:14 by rmk
                                                                 ; Edited 24-Jul-2023 23:13 by rmk
                                                                 ; Edited 23-Oct-2022 09:11 by rmk
    (DECLARE (SPECVARS IMAGESTREAM FORMATTINGSTATE))

    ;; Format the next line of text starting at CH#1.  Return the LINEDESCRIPTOR;  reusing LINE if given.

    ;; The SPECVARS are accessed and reset under the subfunctions, particularly \FORMATLINE.UPDATELOOKS.  IMAGESTREAM and
    ;; FORMATTINGSTATE are passed only for hardcopy.

    ;;

    ;; The objective of this body of code is to find

    ;;    LCHAR1:  The CHNO of the first visible character/object of this line.  LCHAR1=0 for empty/dummy line.

    ;;    LCHARLIM: The CHNO of the last character in the line-vector, including final EOL or last of run of spaces that overflows.

    ;;     LXLIM:   The X coordinate of the right edge of character/object LCHARLIM

    ;;    PREVSP:  The slot position in THISLINE of the right most scalable space.

    ;;    SPACELEFT:  How much unoccupied space is to be allocated according to justified, right, center alignments.

    ;;    OVERHANG:  How far beyond the right margin will trailing spaces/EOL occupy

    ;;   THISLINE:  The CHARSLOT vector that contains the actual characters and widths, together with their looks, as abstracted from the piece
    ;; sequences of the underlying text.

    ;;

    ;; At the end, \FORMATLINE.JUSTIFYmodifies LINE and THISLINE to deal with the vagaries of justification. The overhanging right-margin spaces
    ;; don't get fattened even though justifying might fatten earlier spaces on the line.

    ;;

    ;; If a (visible) word crosses the margin |, then the line ends at the space just before the beginning of that word.  For x==yz==ab|cd, LCHARLIM
    ;; goes to the space before a, LXLIM is its right edge. The justifier will leave the spaces between z and a alone, but might fatten the spaces
    ;; between x and y based on the SPACELEFT between z and margin |.  The spaces after z OVERHANG. An EOL or FORM force a line-end and
    ;; also overhang with along with any immediately preceding spaces--they are essentially treated as line-breaking spaces.

    ;;      abc123#45|6 => abc[123]#$|  (456 on next line--leading white space only after EOL)

    (CL:UNLESS LINE

        ;; Not needed until the end, but then we might not get the starting values for WRIGHT and WBOTTOM, if those change from piece to
        ;; piece--check this.

        [SETQ LINE (create LINEDESCRIPTOR
                          YBOT _ (SUB1 (ffetch (TEXTOBJ WBOTTOM) of TEXTOBJ])
    (PROG ((TSTREAM (fetch (TEXTOBJ STREAMHINT) of TEXTOBJ))
           (THISLINE (ffetch (TEXTOBJ THISLINE) of TEXTOBJ))
           (OFFSET 0)
           (TRUEASCENT -1)
           (TRUEDESCENT -1)
           (ASCENTB 0)
           (DESCENTB 0)
           (ASCENTC 0)
           (DESCENTC 0)
           (OVERHANG 0)
           (SPACELEFT 0)
           (TX 0)
           LINETYPE DISPLAYSTREAM WIDTH WMARGIN SCALE FMTSPEC RIGHTMARGIN TABSPEC KERN FIRSTWHITEX
           FIRSTWHITESLOT PC CHARSLOT PREVSP 1STLN PROTECTED CHNOB FORCED-END CHNO LX1 TX TXB FONT CHARSLOTB
           TABPENDING PREVHYPH PREVDHYPH START-OF-PIECE UNBREAKABLE JUSTIFIED)
          (DECLARE (SPECVARS LINETYPE CHARSLOT CHNO OFFSET ASCENTC DESCENTC FONT START-OF-PIECE KERN
                           UNBREAKABLE))

     ;; CHNO = Current character # in the text, CHNOB is the character at the last potential break

     ;; CHARSLOT = Pointer to the next available slot in THISLINE's CHARS.

     ;; DX = width of current char/object

     ;; TX = Right end of current text, TXB is the right end at the last potential break

     ;; PREVSP = CHARPOS of the last space of the most recent space-run
```

```
;; ASCENT, DESCENT = The ascent and descent values of the line at the current character position
;; ASCENTC, DESCENTC = The ascent and descent from the last CLOOKS (including OFFSET)
;; ASCENTB, DESCENTB, CHNOB, TXB, CHARSLOTB = The values at the most recent potential break-point
;; LX1 = theoffset from the true left margin of the first character, in native units, accounting for the first-line indentation.
;;
     (replace (THISLINE NEXTAVAILABLECHARSLOT) of THISLINE with (LASTCHARSLOT THISLINE))
;; Start with LASTCHARSLOT just so STL debugger will show everything before the true end has been determined.
     (SETQ LINETYPE (if IMAGESTREAM
                        then 'TRUEHARDCOPY
                      else (SETQ DISPLAYSTREAM (WINDOWPROP (CAR (FGETTOBJ TEXTOBJ \WINDOW))
                                                          'DSP))
                           (SETQ IMAGESTREAM DISPLAYSTREAM)
                           'TRUEDISPLAY))                    ; DISPLAYSTREAM needed for HARDCOPYDISPLAY objects
     [if (REGIONP REGION)
         then (SETQ WMARGIN (ffetch (REGION LEFT) of REGION))
                                                             ; Presumably hardcopy in different page regions.
              (SETQ WIDTH (ffetch (REGION WIDTH) of REGION))
       else (SETQ WMARGIN \TEDIT.LINEREGION.WIDTH)           ; A little more display margin on both sides
            (SETQ WIDTH (IDIFFERENCE (FGETTOBJ TEXTOBJ WRIGHT)
                               (UNFOLD WMARGIN 2]
;;
     (SETQ PC (\TEDIT.CHTOPC CH#1 TEXTOBJ T))
     (CL:WHEN (OR (NULL PC)
                  (EQ PC (FGETTOBJ TEXTOBJ LASTPIECE)))
         ;; The dummy line is presumably the one that allows for the cursor to blink after a final EOL.
         (RETURN (AND (FGETLD LINE LDUMMY)
                      LINE)))
;;
;; Make sure we have a visible starting piece.
     (CL:UNLESS (VISIBLEPIECEP PC)
         (CL:UNLESS (SETQ PC (\NEXT.VISIBLE.PIECE PC))
             (RETURN (\TEDIT.FORMATLINE.EMPTY TEXTOBJ CH#1 LINE)))
         (SETQ CH#1 (\TEDIT.PCTOCH PC TEXTOBJ))              ; Unusual, simpler than keeping track on the fly
         (SETQ START-OF-PIECE CH#1))
     (SETQ CHNO CH#1)
     (SETQ IMAGESTREAM (\TEDIT.FORMATLINE.SETUP TEXTOBJ PC LINE IMAGESTREAM))
     (SETQ FMTSPEC (FGETLD LINE LFMTSPEC))
;; Display stream could have switched for hardcopy font widths.
     (CL:WHEN (AND (EQ LINETYPE 'TRUEDISPLAY)
                   (ffetch (FMTSPEC FMTHARDCOPY) of FMTSPEC))
         (SETQ LINETYPE 'HARDCOPYDISPLAY))
     (SETQ SCALE (ffetch (FMTSPEC FMTHARDCOPYSCALE) of FMTSPEC))
;; This line starts a paragraph if it starts the document or it is at the beginning of a piece just after a last-paragraph piece. This assumes that only
;; visible pieces matter; otherwise, use PREVPIECE.
     (SETQ JUSTIFIED (EQ 'JUSTIFIED (fetch (FMTSPEC QUAD) of FMTSPEC)))
     [SETQ 1STLN (OR (IEQP CH#1 1)
                     (AND (IEQP CH#1 START-OF-PIECE)
                          (OR (NOT (\PREV.VISIBLE.PIECE PC))
                              (PPARALAST (\PREV.VISIBLE.PIECE PC]
;; Account for first-line indentation from the true left margin (LEFTMAR), in natural units
     (SETQ LX1 (CL:IF 1STLN
                   (ffetch (FMTSPEC 1STLEFTMAR) of FMTSPEC)
                   (ffetch (FMTSPEC LEFTMAR) of FMTSPEC)))
     (SETQ RIGHTMARGIN (if (ZEROP (ffetch (FMTSPEC RIGHTMAR) of FMTSPEC))
                           then ;; RIGHTMAR = 0 => follow the window/region's width
                                WIDTH
                         else (ffetch (FMTSPEC RIGHTMAR) of FMTSPEC)))
     (SETQ WIDTH (IDIFFERENCE RIGHTMARGIN LX1))
     (SETQ TABSPEC (ffetch (FMTSPEC TABSPEC) of FMTSPEC))
     (CL:WHEN (EQ LINETYPE 'HARDCOPYDISPLAY)                 ; Scale points up to hardcopy
         (SETQ LX1 (HCSCALE SCALE LX1))
         (SETQ WIDTH (HCSCALE SCALE WIDTH))
         (SETQ TABSPEC (\TEDIT.FORMATLINE.SCALETABS TABSPEC SCALE)))
;;
;; The unchanging paragraph looks have now been established. Set up starting piece for BINNING characters
;; The LOOKSUPDATEFN will initialize the character looks of the starting piece PC. It is also called at piece boundaries to reset the
;; character-looks variables when BIN (=\TEXTBIN) moves from piece to piece.
     (freplace (TEXTSTREAM LOOKSUPDATEFN) of TSTREAM with (FUNCTION \TEDIT.FORMATLINE.UPDATELOOKS))
     (freplace (TEXTSTREAM CURRENTLOOKS) of TSTREAM with NIL)
     (SETQ CHARSLOT (FIRSTCHARSLOT THISLINE))
     (\TEDIT.INSTALL.PIECE TSTREAM PC (- CH#1 START-OF-PIECE))
;;
;; Note:  the character looks of the first piece establish the initial FONT, ASCENTC, DESCENTC in anticipation of  the first as yet unseen
;; character, and these are reset when the PLOOKS of each piece change.  These character ASCENTC and DESCENTC values apply only to
;; actual characters, not to image objects, which have their own intrinsic values.  The character values and image values together determine the
;; ASCENT and DESCENT for the line.  But importantly: the initial character-looks or the looks at each piece-transition don't affect the line values
;; until at least one character with those looks has been seen. That's why the line values are computed for each BIN, using character or object
;; values as appropriate..

;;

;; TEXTLEN anticipates the EOL error.  Wouldn't need it if we reset the ENDOFSTREAMOP.

;;  INWORD=T if we haven't just seen a space, INSPACES=T if we are in the middle of a space run.

      (SETQ FIRSTWHITEX TX)
      (bind CH DX BOX INSPACES (INWORD _ T)
            (LASTCHARSLOT _ (LASTCHARSLOT THISLINE))
            (TEXTLEN _ (TEXTLEN TEXTOBJ)) for old CHNO by 1 while (ILEQ CHNO TEXTLEN)
         while (SETQ CH (BIN TSTREAM))
         do ;; Get CH's X width and maintain line ascent and descent.

            [SETQ DX (COND
                        ((SMALLP CH)                          ; CH is a character
                         (SELCHARQ CH
                              ((EOL LF CR FORM Meta,EOL)      ; The reader should coerce LF/CR to EOL

                                   ;;  Force an end to the line. BIN shouldn't produce CR or LF. Should FORM do morein display
                                   ;; mode?

                                 ;; If the EOL is the only character on the line, we want to use the current font's ascent/descent.  But if
                                 ;; only preceded by objects, use the objects values.

                                   ;; The minimum width (N?) is so that the terminator can be selected

                                   [SETQ DX (IMAX (\FGETCHARWIDTH FONT (CHARCODE N))
                                                  (\FGETCHARWIDTH FONT (CHARCODE EOL]
                                   (FILLCHARSLOT CHARSLOT (CL:IF (EQ CH (CHARCODE FORM))
                                                                 (CHARCODE FORM)
                                                                 (CHARCODE EOL))
                                          DX)
                                   (CL:UNLESS (EQ CH (CHARCODE Meta,EOL))
                                       (SETQ FORCED-END (CL:IF (MEMB CH (CHARCODE (LF CR)))
                                                               (CHARCODE EOL)
                                                               CH)))
                                                              ; Remember whether EOL, FORM, but not
                                   (FORCEBREAK)

                                 ;; The break does not set the ascent/descent, the rest of the line does that.  If the line is empty except
                                 ;; for an EOL, the font's ASCENTC is stuck in at the end. This is important for hardcopydisplay.

                                   (RETURN))
                              NIL)
                         (SETQ TRUEASCENT (IMAX TRUEASCENT (IPLUS ASCENTC OFFSET)))
                         (SETQ TRUEDESCENT (IMAX TRUEDESCENT (IDIFFERENCE DESCENTC OFFSET)))
                         (\FGETCHARWIDTH FONT CH))
                        (T                                    ; CH is an object, get its size.

                           ;; If this isn't TRUEHARDCOPY, we want to do the imageobject in the displaystream with displaystream
                           ;; coordinates, because we don't know what internal size computations the imageobject might make based on
                           ;; its displaystream and fonts.  But we do have to down-scale WIDTH (right margin) back to the units of the
                           ;; display stream.

                           (SETQ BOX (APPLY* (IMAGEOBJPROP CH 'IMAGEBOXFN)
                                            CH
                                            (CL:IF (EQ LINETYPE 'TRUEHARDCOPY)
                                                IMAGESTREAM
                                                DISPLAYSTREAM)
                                            TX
                                            (CL:IF (EQ LINETYPE 'HARDCOPYDISPLAY)
                                                (HCUNSCALE SCALE WIDTH)
                                                WIDTH)))
                           (IMAGEOBJPROP CH 'BOUNDBOX BOX)
                           (SETQ TRUEASCENT (IMAX TRUEASCENT (IPLUS (IDIFFERENCE (fetch (IMAGEBOX YSIZE)
                                                                                    of BOX)
                                                                             (fetch (IMAGEBOX YDESC)
                                                                                of BOX))
                                                                    OFFSET)))
                           (SETQ TRUEDESCENT (IMAX TRUEDESCENT (IDIFFERENCE (fetch (IMAGEBOX YDESC)
                                                                               of BOX)
                                                                      OFFSET)))
                           (SETQ DX (IPLUS (fetch (IMAGEBOX XSIZE) of BOX)
                                           (fetch (IMAGEBOX XKERN) of BOX)))

                           ;; The external DX has to be upscaled from its displaystream coordinates.

                           (CL:IF (EQ LINETYPE 'HARDCOPYDISPLAY)
                               (HCSCALE SCALE DX)
                               DX)]
            (CL:WHEN KERN                                     ; Unlikely for display
                (add DX KERN))
            [SELCHARQ CH
                 (SPACE ;; White space and EOL can overhang the right margin, but no visible character can. The only white-space leading
                        ;; a line must follow an [EOL]

                        ;;    123abc456xy|z  =>  123abc|[456]$xyz  Line break in front of x, 456 overhangs margin

                        (if UNBREAKABLE
                                then  (add TX DX)
                                      ;; Not including this space in the justifying chain, so it won't expand.  If that looks odd, let it fall through
                                      ;; to the PUSHCHAR below.
                                      (PUSHCHAR CHARSLOT CH DX)
                              else  (SPACEBREAK)
                                    (add TX DX)
                                    (SAVEBREAK)
                                    ;; CHAR will be the slot of the previous space, not this space character, CHARW is the natural width of
                                    ;; this space. PREVSP is the new chain-header.
                                    (PUSHCHAR CHARSLOT (CL:IF JUSTIFIED
                                                              (PROG1 PREVSP (SETQ PREVSP CHARSLOT))
                                                              CH)
                                           DX)))
                 (TAB  ;; Try to be reasonable with tabs.  This will create trouble when doing fast-case insert/delete, but Pah! for now.
                      ;; Remove all prior candidate break points and expandable spaces
                      (FORGETPREVIOUSBREAK)
                      (SETQ PREVSP (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP))
                      ;; Now for this tab:                              ; Start with 0 width, then set up the next tab
                      (FILLCHARSLOT CHARSLOT CH 0)
                      (SETQ TABPENDING (\TEDIT.FORMATLINE.TABS TEXTOBJ TABSPEC SCALE CHARSLOT LX1 TX
                                              TABPENDING))      ; Proper width is already in CHARSLOT
                      (SETQ DX (CL:IF (FIXP TABPENDING)
                                   (PROG1 TABPENDING (SETQ TABPENDING NIL))
                                   (fetch (PENDINGTAB PTRESOLVEDWIDTH) of TABPENDING)))
                      (add TX DX)
                      (CL:WHEN (IGREATERP TX WIDTH)              ; Tab pushed beyond the margin
                          (SETQ OVERHANG (IDIFFERENCE TX WIDTH))
                          (SETQ SPACELEFT 0)
                          (RETURN))
                      (SETQ CHARSLOT (NEXTCHARSLOT CHARSLOT)))
                 (PROGN  ;; Not an EOL, space, or tab character.
                        (SETQ INWORD T)                         ; Space run has ended
                        (SETQ INSPACES NIL)
                        (CL:UNLESS (DIACRITICP CH)
                            ;; Assume that diacritics have zero width.  \DISPLAYLINE and \TEDIT.HARDCOPY.DISPLAYLINE adjust
                            ;; their alignment, centering on the next character.  However, if a diacritic is wider than the the next character,
                            ;; here the next character should be assigned the diacritic's width.
                            (add TX DX))
                        (CL:WHEN (IGREATERP TX WIDTH)
                            ;; Overflow: If there's a previous break, go back to it.  If this character won't fit no matter what, just put it here
                            ;; and let it run out into the margin (or off the page).
                            (CL:WHEN FIRSTWHITESLOT                ; Back to previous space run
                                (DOBREAK T)
                                (RETURN))
                            (CL:WHEN (OR PREVHYPH PREVDHYPH)
                                ;; A good break-point not followed by spaces. NOTE: Even pending tabs go on the next line.
                                (CL:UNLESS TXB (FILLCHARSLOT CH DX))
                                (DOBREAK)
                                (RETURN))
                            (CL:WHEN (IGREATERP DX WIDTH)
                                ;; This character will never fit (e.g. a large image object). Move it to next line, by itself, if this line isn't
                                ;; empty.  Otherwise, dump it here by itself.
                                (if (IGREATERP CHNO CH#1)
                                    then
                                  ;; Move the offender to the next line, by itself.  For this line it essentially acts like an EOL wrt breaking
                                  ;; and justifying, except that it doesn't get tacked on to the end.  There was no good earlier break,
                                  ;; otherwise we would have done it.
                                         (add TX (IMINUS DX))
                                         (add CHNO -1)          ; back up to preceding character
                                         (SETQ CHARSLOT (PREVCHARSLOT! CHARSLOT))
                                         (SETQ CH (CHAR CHARSLOT))
                                         (SETQ DX (CHARW CHARSLOT))
                                         ;; ASCENT/DESCENT for the previous CLOOKS. BUT: if the previous character is an
                                         ;; object, it has to back out its box parameters
                                         (SETQ TRUEASCENT ASCENTC)
                                         (SETQ TRUEDESCENT DESCENTC)
                                  else  ;; Dump it here
                                       (FILLCHARSLOT CHARSLOT CH DX))
                                (SETQ OVERHANG 0)
                                (SETQ SPACELEFT 0)
                                (RETURN))
                            (CL:WHEN (IGREATERP CHNO CH#1)
                                ;; We've seen at least one real character, line is not empty, but no good candidate break point. Back up
                                ;; to the last legal break (or add a real hyphenator).
```

```
                                        (CL:UNLESS (\TEDIT.FORMATLINE.LASTLEGAL)
                                            ;; Didn't find one, the offender protrudes on this line
                                            (FILLCHARSLOT CHARSLOT CH DX))
                                        (RETURN))
                                    ;; Don't break: can't split before the first thing on the line!
                                    (PUSHCHAR CHARSLOT CH DX)
                                    (RETURN))
                                ;;
                                ;; Not past the rightmargin yet. Save the character and width, then maybe adjust.
                                (SELCHARQ CH
                                     (%.                                ; Check for decimal tabs, immediately after TAB
                                         (PUSHCHAR CHARSLOT CH DX)
                                         (CL:WHEN (AND TABPENDING (EQ (fetch PTTYPE of TABPENDING)
                                                                      'DECIMAL))
                                                                        ; Figure out which tab stop to use, and what we need to do to get
                                                                        ; there.
                                             (add (fetch (PENDINGTAB PTTABX) of TABPENDING)
                                                  DX)                   ; Adjust the tab stop's X value so that the LEFT edge of the
                                                                        ; decimal point goes there.
                                             (SETQ TABPENDING (\TEDIT.FORMATLINE.TABS TEXTOBJ TABSPEC SCALE CHARSLOT
                                                                     LX1 TX TABPENDING T))
                                                                        ; Tab over to the LEFT side of the decimal point.
                                             (add TX (CL:IF (FIXP TABPENDING)
                                                             (PROG1 TABPENDING (SETQ TABPENDING NIL))
                                                             (fetch (PENDINGTAB PTRESOLVEDWIDTH) of TABPENDING)))
                                             (SETQ PREVSP (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP))
                                                                        ; Spaces before a tab don't take part in later justification.
                                             (SAVEBREAK)))
                                     ((- EM-DASH SOFT-HYPHEN)           ; Hyphen, M-dash, discretionary hyphen
                                          (CL:UNLESS UNBREAKABLE
                                              (FORGETPREVIOUSBREAK)
                                              (SETQ PREVHYPH CHARSLOT)
                                              (CL:WHEN (EQ CH (CHARCODE SOFT-HYPHEN))
                                                  (SETQ PREVDHYPH CHARSLOT)
                                                                        ; Discretionary hyphen may become invisible
                                                  (SETQ CH (CHARCODE))
                                                                        ; Otherwise, it shows as a real hyphen
                                                  (SETQ DX (\FGETCHARWIDTH FONT (CHARCODE "-"))))
                                              (SAVEBREAK))              ; Save the hyphen slot, then fill it
                                          (PUSHCHAR CHARSLOT CH DX))
                                     (NONBREAKING-HYPHEN ;; Switch the character code and width in case font doesn't have a glyph??
                                          (PUSHCHAR CHARSLOT (CHARCODE -)
                                                 (\FGETCHARWIDTH FONT (CHARCODE "-"))))
                                     (NONBREAKING-SPACE                 ; This will eventually convert to SPACE
                                          (PUSHCHAR CHARSLOT (PROG1 PREVSP (SETQ PREVSP CHARSLOT))
                                                 DX))
                                     (PUSHCHAR CHARSLOT CH DX]
                        ;; BOUNDS CHECKING!
                        (CL:WHEN (EQ CHARSLOT LASTCHARSLOT)
                            ;; If too long, we let it roll over to the next line.  Should we put something in the margin??
                            (TEDIT.PROMPTPRINT TEXTOBJ "Line too long to format." T)
                            (RETURN))
                  finally ;; Ran out of TEXTLEN (and paragraph). Back up and force a break. Are ASCENT/DESCENT correct?
                        (CL:WHEN (AND (EQ PREVSP (PREVCHARSLOT CHARSLOT))
                                      (NULL (CHAR PREVSP)))
                            ;; The line ended in a space that needs to be resolved.  If we coded the end of a space-chain as (CHARCODE SPACE)
                            ;; instead of NIL, maybe this wouldn't be necessary.
                            (FILLCHARSLOT PREVSP (CHARCODE SPACE)
                                   (CHARW PREVSP))
                            (SETQ PREVSP NIL))
                        (SETQ CHARSLOT (PREVCHARSLOT! CHARSLOT))
                        (add CHNO -1)
                        (SETQ DX 0)                                     ; TX is already correct
                        (FORCEBREAK))
          ;; End of character loop.
               (freplace (THISLINE NEXTAVAILABLECHARSLOT) of THISLINE with (NEXTCHARSLOT CHARSLOT))
               (freplace (TEXTSTREAM LOOKSUPDATEFN) of TSTREAM with NIL)
          ;; Fix up last tab?
               (CL:WHEN TABPENDING
                   (SETQ PREVSP (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP))
                                                                        ; Don't justify spaces before tabs
                   (add TX (\TEDIT.FORMATLINE.TABS TEXTOBJ TABSPEC SCALE (FETCH (PENDINGTAB PTCHARSLOT) OF
                                                                                                   TABPENDING
                                                                                  )
                                   LX1
                                   (IDIFFERENCE TX OVERHANG)
                                   TABPENDING T)))
```

```
    ;;
    ;; All the line information is now in our variables.  Migrate to the LINE and THISLINE fields.
         (FSETLD LINE LCHAR1 CH#1)
         (FSETLD LINE LCHARLIM CHNO)
         (FSETLD LINE LX1 LX1)                                    ; Still maybe scaled for hardcopy display
         (FSETLD LINE LXLIM (IPLUS LX1 TX))
         (FSETLD LINE 1STLN 1STLN)                                ; First line of a paragraph
         [FSETLD LINE LSTLN (AND FORCED-END (PPARALAST (\TEDIT.CHTOPC CHNO TEXTOBJ]
                                                                  ; Last line of a paragraph
    ;; For display, the value of LMARK (GREY) just causes the little grey box to show up in the left margin, but is not interpreted in any other way. The
    ;; hardcopy code uses this field for other purposes.
         (FSETLD LINE LMARK (CL:WHEN [AND 1STLN (NEQ LINETYPE 'TRUEHARDCOPY)
                                          (OR (EQ (fetch FMTPARATYPE of FMTSPEC)
                                                  'PAGEHEADING)
                                              (fetch FMTNEWPAGEBEFORE of FMTSPEC)
                                              (fetch FMTNEWPAGEAFTER of FMTSPEC)
                                              [AND (fetch FMTSPECIALX of FMTSPEC)
                                                   (NOT (ZEROP (fetch FMTSPECIALX of FMTSPEC]
                                              (AND (fetch FMTSPECIALY of FMTSPEC)
                                                   (NOT (ZEROP (fetch FMTSPECIALY of FMTSPEC]
                                 'GREY))
         (FSETLD LINE FORCED-END FORCED-END)
         (FSETLD LINE LHASPROT PROTECTED)
         (FSETLD LINE LEFTMARGIN (CL:IF 1STLN
                                      (fetch (FMTSPEC 1STLEFTMAR) of FMTSPEC)
                                      (fetch (FMTSPEC LEFTMAR) of FMTSPEC)))
         (FSETLD LINE RIGHTMARGIN RIGHTMARGIN)
         (CL:UNLESS FONT
             ;; Use TEXTOBJ defaults if empty charlooks. Maybe this never happens?
             (SETQ FONT (FONTCOPY (OR (AND (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS)
                                           (fetch CLFONT of (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS)))
                                      DEFAULTFONT)
                               'DEVICE IMAGESTREAM)))
         (CL:WHEN (EQ -1 TRUEASCENT)                              ; Blank or only
             (SETQ TRUEASCENT ASCENTC)
             (SETQ TRUEDESCENT DESCENTC))
         (FSETLD LINE LTRUEASCENT TRUEASCENT)                     ; |FORMATLINE.ALIGNED adjusts ASCENT, DESCENT,
                                                                  ; LHEIGHT
         (FSETLD LINE LTRUEDESCENT TRUEDESCENT)
    ;;
         (FSETLD LINE LFMTSPEC FMTSPEC)
         (CL:WHEN (EQ LINETYPE 'TRUEHARDCOPY)
             ;; Used temporarily and cleared by \TEDIT.FORMATBOX; not an XPOINTER
             (FSETLD LINE LTEXTSTREAM TSTREAM))
         (freplace (THISLINE DESC) of THISLINE with LINE)
         (\TEDIT.FORMATLINE.VERTICAL LINE TSTREAM)
         (\TEDIT.FORMATLINE.HORIZONTAL LINE THISLINE PREVSP SPACELEFT OVERHANG LINETYPE)
    ;; Finally translate to the left edge, perhsps a specialx if true hardcopy.
         (CL:WHEN [AND (EQ LINETYPE 'TRUEHARDCOPY)
                       (fetch (FMTSPEC FMTSPECIALX) of FMTSPEC)
                       (NOT (ZEROP (fetch (FMTSPEC FMTSPECIALX) of FMTSPEC]
             ;; Maybe SETQ instead of add ??
             (add WMARGIN (ffetch (FMTSPEC FMTSPECIALX) of FMTSPEC)))
         (add (FGETLD LINE LEFTMARGIN)
              WMARGIN)
         (add (FGETLD LINE RIGHTMARGIN)
              WMARGIN)
         (add (FGETLD LINE LX1)
              WMARGIN)
         (add (FGETLD LINE LXLIM)
              WMARGIN)
         (RETURN LINE])
```

(\TEDIT.FORMATLINE.SETUP

```
  [LAMBDA (TEXTOBJ PC LINE IMAGESTREAM)                           ; Edited 16-Dec-2023 23:34 by rmk
                                                                  ; Edited 14-Jun-2023 16:43 by rmk
                                                                  ; Edited  8-Mar-2023 22:15 by rmk
                                                                  ; Edited  7-Mar-2023 16:52 by rmk
                                                                  ; Edited  6-Mar-2023 00:25 by rmk
                                                                  ; Edited  2-Mar-2023 12:06 by rmk
   ;; The paragraph looks of a line are the same for every piece of every line in a paragraph, only the character looks can change from piece to piece.
   ;; We retrieve the para looks from the starting piece, or the stream's default.
   ;; The global variable *TEDIT-CACHED-FMTSPEC* is a heuristic optimization
   ;; In hardcopy-display mode, the verticals (lineleading etc.) are in screen points, only the horizontals are upscaled according to the
   ;; points-to-hardcopy scalefactor installed in the retrieved FMTSPEC.
   ;; See comments in TEDIT-LOOKSCOMS about the style-cache variables.  Probably not completely or correctly coordinated with this code.
```

```
    ;; The global variable *TEDIT-CACHED-FMTSPEC* offers a heuristic optimization to speed up construction of the FMTSPEC for successive lines
    ;; in the same paragraph (or maybe even in a sequence of same-format paragraphs.

    (LET [(FMTSPEC (OR (AND PC (PPARALOOKS PC))
                       (GETTOBJ TEXTOBJ FMTSPEC]
         (SETQ FMTSPEC (\TEDIT.APPLY.PARASTYLES FMTSPEC PC TEXTOBJ))
         (if (NOT (DISPLAYSTREAMP IMAGESTREAM))
             then (SETQ FMTSPEC (\TEDIT.HCPYFMTSPEC FMTSPEC IMAGESTREAM))
           elseif (fetch (FMTSPEC FMTHARDCOPY) of FMTSPEC)
             then ;; Coerce the image stream and FMTSPEC for chracter-width scaling.

                  [SETQ IMAGESTREAM (OR (FGETTOBJ TEXTOBJ DISPLAYHCPYDS)
                                        (FSETTOBJ TEXTOBJ DISPLAYHCPYDS (OPENIMAGESTREAM '{NODIRCORE}
                                                                                 'POSTSCRIPT]
                  (SETQ FMTSPEC (create FMTSPEC using FMTSPEC FMTHARDCOPYSCALE _ (DSPSCALE NIL IMAGESTREAM)))
           elseif (NULL (fetch (FMTSPEC FMTHARDCOPYSCALE) of FMTSPEC))
             then                                               ; Should be done at create
                  (replace (FMTSPEC FMTHARDCOPYSCALE) of FMTSPEC with 1))
         (CL:UNLESS (OR (EQ FMTSPEC *TEDIT-CACHED-FMTSPEC*)
                        (NOT (fetch (FMTSPEC FMTCHARSTYLES) of FMTSPEC)))

             ;; The cache of styles for the current paragraph is invalid; flush it, and note the new paragraph to cache for.

             (SETQ *TEDIT-CURRENTPARA-CACHE* NIL)
             (SETQ *TEDIT-CACHED-FMTSPEC* FMTSPEC))
         (SETLD LINE LFMTSPEC FMTSPEC)
         IMAGESTREAM])
```

(\TEDIT.FORMATLINE.HORIZONTAL

```
  [LAMBDA (LINE THISLINE PREVSP SPACELEFT OVERHANG LINETYPE)    ; Edited 15-Mar-2024 19:35 by rmk
                                                                ; Edited  3-Dec-2023 16:49 by rmk
                                                                ; Edited 29-Oct-2023 18:24 by rmk
                                                                ; Edited  2-Jul-2023 15:15 by rmk
                                                                ; Edited  6-Apr-2023 10:13 by rmk
                                                                ; Edited  8-Mar-2023 12:45 by rmk

    ;; Do the formatting work for justified, centered, etc. lines. We calculate how much space between LX0 and right margin is not occupied by the
    ;; natural widths of the characters cached in THISLINE. For this calculation we back out spaces at the end of the line. They are present for later
    ;; display and selection, but are ignored for purposes of right, centered, and justified alignment.

    ;;

    ;; In HARDCOPYDISPLAY, LX1, LXLIM, SPACELEFT, and OVERHANG are all in scaled units, otherwise in natural stream units.
    ;; SPACELEFT+LXLIM-OVERHANG should be the right margin.

    ;;

    ;; The display-alignment is controlled by LX0 (offset from LEFTMARGIN) and LXLIM. At entry, LXLIM is the natural width of the line-characters.
    ;; LXLIM may embrace the extra spaces, but they are out in the right margin or beyond the window, invisible unless selected

    ;; SPACELEFT is what it takes to push the last visible character out to the right margin. This is done by expanding spaces. OVERHANG is what
    ;; gets added to LXLIM because of white space after the last visible. The OVERHANG white space is not expanded.

    ;;

    ;; Also for HARDCOPYDISPLAY the horizontal positions (margins and character widths) are in hardcopy units. At the end we scale them back to
    ;; screen points.

    (LET* ((FMTSPEC (FGETLD LINE LFMTSPEC))
           (SCALE (ffetch (FMTSPEC FMTHARDCOPYSCALE) of FMTSPEC)))

          ;; Distribute SPACELEFT according to QUAD.

          (freplace (THISLINE TLSPACEFACTOR) of THISLINE with 1)
          (CL:WHEN (EQ 'JUSTIFIED (fetch (FMTSPEC QUAD) of FMTSPEC))
                 (\TEDIT.FORMATLINE.JUSTIFY LINE THISLINE PREVSP SPACELEFT LINETYPE))
          (\TEDIT.FORMATLINE.PURGE.SPACES PREVSP)

          ;;

          ;; Done with spaces, expanded or not. Down scale if hard-copy display mode

          (CL:WHEN (EQ LINETYPE 'HARDCOPYDISPLAY)
              (change (FGETLD LINE LX1)
                     (HCUNSCALE SCALE DATUM))
              (change (FGETLD LINE LXLIM)
                     (HCUNSCALE SCALE DATUM))
              (SETQ SPACELEFT (HCUNSCALE SCALE SPACELEFT))
              (SETQ OVERHANG (HCUNSCALE SCALE OVERHANG))

              ;; Scale the character widths to points, propagating rounding error along the way. LOST starts at .5 pt so that rounding doesn't clip the
              ;; last character

              (for CHARSLOT REDUCED (LOST _ 0.5)
                  incharslots THISLINE when CHAR do (SETQ REDUCED (FPLUS LOST (FQUOTIENT CHARW SCALE)))
                                                                ; Include the previously lost point-fraction
                                                    [SETQ LOST (FDIFFERENCE REDUCED (SETQ REDUCED (FIX REDUCED]
                                                    (replace (CHARSLOT CHARW) of CHARSLOT with REDUCED)))

          ;;

          (SELECTQ (ffetch (FMTSPEC QUAD) of FMTSPEC)
               (RIGHT                                           ; Move over to the right margin
                      (add (FGETLD LINE LX1 LINE)
                           SPACELEFT)
                      (add (FGETLD LINE LXLIM)
                           SPACELEFT))
              (CENTERED                                        ; Split the difference
                     (add (FGETLD LINE LX1)
                          (FOLDLO SPACELEFT 2))
                     (add (FGETLD LINE LXLIM)
                          (FOLDLO SPACELEFT 2)))
              NIL])
```

(\TEDIT.FORMATLINE.VERTICAL

```
  [LAMBDA (LINE TEXTOBJ)                                       ; Edited 20-Mar-2024 07:26 by rmk
                                                               ; Edited 17-Dec-2023 00:43 by rmk
                                                               ; Edited  6-Dec-2023 20:13 by rmk
                                                               ; Edited  4-Dec-2023 12:13 by rmk

    ;; Sets up vertical-alignment parameters taking into account the line and paragraph leading specifications. The vertical parameters (line-leading
    ;; etc.) have not been up-scaled and don't need to be down-scaled. For other modes the vertical dimensions are already appropriately scaled.

    ;; This calculates vertical sizes based on inherent line/paragraph parameters.It cannot deal with base-to-base positioning because that is context
    ;; dependent, involving the position and descent of the previous line (\TEDIT.LINE.BOTTOM).

    (LET ((FMTSPEC (FGETLD LINE LFMTSPEC))
          (ASCENT (FGETLD LINE LTRUEASCENT))
          (DESCENT (FGETLD LINE LTRUEDESCENT)))
         (CL:WHEN (FGETLD LINE 1STLN LINE)                     ; Set pre-paragraph leading
             (add ASCENT (ffetch (FMTSPEC LEADBEFORE) of FMTSPEC)))
         (CL:WHEN (FGETLD LINE LSTLN)                          ; Set post-paragraph leading
             (add DESCENT (ffetch (FMTSPEC LEADAFTER) of FMTSPEC)))

         ;; Documentation says that lineleading goes above, which automatically makes for reasonable selection marking.  It went below in the
         ;; original implementation, selections were very odd for large line leadings.  This flag is set to T when recently created files are loaded, we try
         ;; to preserve the old (bad) behavior for older files.

         (CL:IF (FGETTOBJ TEXTOBJ TXTLINELEADINGABOVE)
             (add ASCENT (fetch (FMTSPEC LINELEAD) of FMTSPEC))
             (add DESCENT (fetch (FMTSPEC LINELEAD) of FMTSPEC)))
         (FSETLD LINE ASCENT ASCENT)
         (FSETLD LINE DESCENT DESCENT)
         (FSETLD LINE LHEIGHT (IPLUS ASCENT DESCENT])
```

(\TEDIT.FORMATLINE.JUSTIFY

```
  [LAMBDA (LINE THISLINE PREVSP SPACELEFT LINETYPE)            ; Edited  7-Mar-2023 18:01 by rmk
                                                               ; Edited  2-Mar-2023 22:45 by rmk
                                                               ; Edited 22-Oct-2022 00:06 by rmk
                                                               ; Edited 29-Mar-94 12:36 by jds

    ;; The spaces in this line are to be expanded to eat up SPACELEFT so that the last visible character will align at the right margin.  SPACELELEFT
    ;; may be in hardcopy-display scaled units.

    (CL:WHEN (AND PREVSP (IGREATERP SPACELEFT 0))
        (LET (NATURALWIDTHS COMMONWIDTH)
             [if (EQ LINETYPE 'TRUEHARDCOPY)
                 then
                      ;; Original code removed overhanging spaces, so that LXLIM and the last charslot of THISLINE are consistent, and
                      ;; SPACELEFT is backed off.  But now, SPACELEFT only measures out to the margin, so doesn't need to be further
                      ;; adjusted (OVERHANG deals with that).  So, if the hardcopy stream doesn't mind printing extra spaces, we don't have to
                      ;; pull things back.  Here we just have to measure the sum of the natural widths, to do the space factor.

                      [SETQ NATURALWIDTHS (for (SPSLOT _ PREVSP) by (CHAR SPSLOT) while SPSLOT
                                             sum (PROG1 (CHARW SPSLOT)
                                                        (CL:UNLESS (CHAR SPSLOT)
                                                               ; Some early spaces may not expand
                                                            (replace (THISLINE TLFIRSTSPACE) of THISLINE with SPSLOT)
))]
               else
                    ;; Typically all the spaces on the line have the same natural width and we can avoid floating point below.

                    ;; NB we operate in 32 x value form, for rounding ease and accuracy on screen-point display streams.  .

                    [SETQ NATURALWIDTHS (for (SPSLOT _ PREVSP)
                                             CHARW FIRSTWIDTH (NSPACES _ 0)
                                             (ALLSAME _ T) by (CHAR SPSLOT) first (SETQ FIRSTWIDTH (CHARW SPSLOT))
                                           while SPSLOT sum (SETQ CHARW (CHARW SPSLOT))
                                                            (add NSPACES 1)
                                                            (CL:UNLESS (IEQP CHARW FIRSTWIDTH)
                                                                   (SETQ ALLSAME NIL))
                                                            CHARW
                                           finally (CL:WHEN ALLSAME
                                                       (SETQ COMMONWIDTH (IPLUS (UNFOLD FIRSTWIDTH 32)
                                                                                (IQUOTIENT (UNFOLD SPACELEFT 32)
                                                                                       NSPACES))))]
                    (if COMMONWIDTH
                        then
                             ;; Fast loop for the more common case where all the spaces on a line are of the same width. Multiply by 32 to keep
                             ;; rounding precision. Avoids floating point allocation.

                             (for (SPSLOT _ PREVSP)
                                  EXPANDED
                                  (LOST _ 0) by (CHAR SPSLOT) while SPSLOT do (SETQ EXPANDED (IPLUS LOST
                                                                                                    COMMONWIDTH))
                                                                             (replace (CHARSLOT CHARW) of SPSLOT
                                                                                with (FOLDLO EXPANDED 32))
                                                                             (SETQ LOST (IMOD EXPANDED 32)))
                      else
                           ;; The slow loop is for spaces of difference sizes. It allocates 3 floating point numbers per space.
```

```
                                   (for (SPSLOT _ PREVSP)
                                       EXPANDED NEWW (LOST _ 0.0)
                                       (MULTIPLIER _ (FPLUS 1.0 (FQUOTIENT SPACELEFT NATURALWIDTHS)))
                                      by (CHAR SPSLOT) while SPSLOT do

                                               ;; Spaces are in different fonts with different widths. What we lose in rounding at one space we add
                                               ;; back in the next, until we finally get resynchronized.  The effect is that a later loss may ripple to a
                                               ;; few earlier spaces.

                                                                         (SETQ EXPANDED (FPLUS LOST (FTIMES (CHARW SPSLOT)
                                                                                                            MULTIPLIER)))
                                                                         (SETQ NEWW (FIXR EXPANDED))
                                                                         (freplace (CHARSLOT CHARW) of SPSLOT with NEWW)
                                                                         (SETQ LOST (FDIFFERENCE EXPANDED NEWW]

                     ;; The \DISPLAYLINE for displaystreams does its own (Maiko) BLTCHAR, so the TLSPACEFACTOR  isn't actually used for display,
                     ;; but hardcopy streams make use of it.

                     (add (ffetch (LINEDESCRIPTOR LXLIM) of LINE)
                          SPACELEFT)
                     (freplace (THISLINE TLSPACEFACTOR) of THISLINE with (FQUOTIENT (IPLUS NATURALWIDTHS SPACELEFT)
                                                                                   NATURALWIDTHS))))])
```

(\TEDIT.FORMATLINE.TABS

```
  [LAMBDA (TEXTOBJ TABSPEC SCALE CHARSLOT LX1 TX PRIORTAB CLEANINGUP)
                                                                          ; Edited 17-Dec-2023 12:46 by rmk
                                                                          ; Edited  9-Mar-2023 23:25 by rmk
                                                                          ; Edited  5-Mar-2023 22:54 by rmk
                                                                          ; Edited  4-Mar-2023 18:28 by rmk
                                                                          ; Do the formatting work for a tab.

    ;; PRIORTAB is the outstanding tab, if any, that has to be resolved.  This will be a centered or flush right tab.

    ;; Specific tabs are relative to the true leftmargin; in that coordinate system the current position is LX1+TX (in properly scaled units. The TX entries
    ;; in the prior tab are also in the scaled margin coordinate system.  TABSPEC is also properly scaled.

    ;;

    ;; If CLEANINGUP is non-NIL, then we're at the end of the line, and only need to resolve the outstanding tab.

    ;; This assumes that every thing except the constants is already hardcopy-scaled

    ;;

    ;; The return provides the number of (scaled) width-units that must be added to the TX in \FORMATLINE..  This includes resolving (and updating
    ;; THISLINE) for the prior tab's now-known width, and adding the width for this tab if it can be resolved.  If it can't be resolved, the returned
    ;; PENDINGTAB includes the prior width, so that can be discharged into \FORMATLINE's TX.

    ;;

    ;; GRAIN is the granularity of the tab spacing; anything within GRAIN will slop over to the next tab.  This is to finesse rounding problems when
    ;; going among various devices.

    ;;

    (add TX LX1)                                                          ; Margin relative
    (PROG (NEXTTAB NEXTTABTYPE NEXTTABX DFLTTABX GRAIN (PRIORTABWIDTH 0)
                 (THISTABWIDTH 0))
          (CL:WHEN PRIORTAB

              ;; If there is a prior tab to resolve, do that first--it affects the perceived current X value, which affects later tabs

              ;; TX - OLDTX = W, the width of the segment after the prior tab. The target X (right tab)  is TABX - W

              [SETQ PRIORTABWIDTH (IMAX (ITIMES SCALE 3)
                                        (IDIFFERENCE (IDIFFERENCE (fetch (PENDINGTAB PTTABX) of PRIORTAB)
                                                                  (SELECTQ (fetch (PENDINGTAB PTTYPE) of PRIORTAB)
                                                                      ((CENTERED DOTTEDCENTERED)
                                                                          ; Centered around the tab X
                                                                           (FOLDLO (IDIFFERENCE TX (fetch (PENDINGTAB
                                                                                                            PTOLDTX)
                                                                                                     of PRIORTAB))
                                                                                   2))
                                                                      ((RIGHT DOTTEDRIGHT DECIMAL DOTTEDDECIMAL)
                                                                          ; Snug up against the tab X
                                                                           (IDIFFERENCE TX (fetch (PENDINGTAB PTOLDTX)
                                                                                              of PRIORTAB)))
                                                                      (SHOULDNT)))
                                                     (fetch (PENDINGTAB PTOLDTX) of PRIORTAB]
              (replace (CHARSLOT CHARW) of (fetch (PENDINGTAB PTCHARSLOT) of PRIORTAB) with PRIORTABWIDTH)
              (add TX PRIORTABWIDTH))                                     ; Done with the past
          (CL:WHEN CLEANINGUP                                             ; Cleaning up at end of line.
              (RETURN PRIORTABWIDTH))                                     ; Default Tab width, if there aren't any real tabs to use
          (SETQ NEXTTAB (find TAB in (CDR TABSPEC) suchthat (IGREATERP (fetch TABX of TAB)
                                                                          TX)))
                                                                          ; The next tab on this line, if any
          (SETQ NEXTTABTYPE (OR (AND NEXTTAB (fetch TABKIND of NEXTTAB))
                                'LEFT))                                   ; The type of the next tab  is LEFT if we use the default spacing
          [SETQ NEXTTABX (COND
                             (NEXTTAB                                     ; There is a real tab to go to; use its location.
                                    (fetch TABX of NEXTTAB))
                             (T (SETQ DFLTTABX (OR (FIXP (CAR TABSPEC))
                                                   DEFAULTTAB))
                                (SETQ GRAIN (FOLDLO SCALE 2))
```

```
                                                ;; No real tab; use the next multiple of the default spacing.
                                                (ITIMES DFLTTABX (ADD1 (IQUOTIENT (IPLUS GRAIN TX)
                                                                                  DFLTTABX]
                                                                                       ; The next tab's X value
        (CL:WHEN (FMEMB NEXTTABTYPE '(DOTTEDLEFT DOTTEDCENTERED DOTTEDRIGHT DOTTEDDECIMAL))

            ;; Change a dotted-leader tab to Meta,TAB, so the line displayers can recognize that they need to do special output that can't be
            ;; precomputed here.  By the same token, we could replace the resolved tab with a widened space, since we know that
            ;; space-expansion is suppressed when a tab is seen.

            (replace (CHARSLOT CHAR) of CHARSLOT with (CHARCODE Meta,TAB)))
        (RETURN (if (FMEMB NEXTTABTYPE '(LEFT DOTTEDLEFT))
                    then ;; Prior and LEFT tabs are both resolved.  At least 1 scaled point for display-selection?

                         (SETQ THISTABWIDTH (IMAX SCALE (IDIFFERENCE NEXTTABX TX)))
                         (replace (CHARSLOT CHARW) of CHARSLOT with THISTABWIDTH)
                         (IPLUS PRIORTABWIDTH THISTABWIDTH)
                  else (replace (CHARSLOT CHARW) of CHARSLOT with 0)
                                                                                       ; All others:  wait for this width

                       ;; PTOLDTX and PTTABX in absolute coordinates for future comparisons (on the same line with same LX1).

                       (create PENDINGTAB
                               PTRESOLVEDWIDTH _ (IPLUS PRIORTABWIDTH THISTABWIDTH)
                               PTTYPE _ NEXTTABTYPE
                               PTTABX _ NEXTTABX
                               PTCHARSLOT _ CHARSLOT
                               PTOLDTX _ TX])
```

(\TEDIT.FORMATLINE.SCALETABS

```
  [LAMBDA (TABSPEC SCALE)                                              ; Edited  7-Mar-2023 21:06 by rmk
                                                                       ; Edited  5-Mar-2023 20:39 by rmk

    ;; Scales tab stops to hardcopy units (possibly hardcopy display)

    (CL:WHEN (type? FMTSPEC TABSPEC)
        (SETQ TABSPEC (ffetch (FMTSPEC TABSPEC) of TABSPEC)))
    (CL:UNLESS (CAR TABSPEC)
        (SETQ TABSPEC (CONS DEFAULTTAB (CDR TABSPEC))))
    (if (EQ SCALE 1)
        then TABSPEC
      else (CONS (HCSCALE SCALE (CAR TABSPEC))
                 (for TAB in (CDR TABSPEC) collect (create TAB using TAB TABX _ (HCSCALE SCALE (fetch (TAB TABX)
                                                                                                      of TAB])
```

(\TEDIT.FORMATLINE.PURGE.SPACES

```
  [LAMBDA (PREVSP UNTILSP)                                             ; Edited 29-Oct-2023 19:11 by rmk
                                                                       ; Edited 21-Mar-2023 11:28 by rmk
                                                                       ; Edited 10-Mar-2023 12:28 by rmk
                                                                       (* jds " 9-NOV-83 17:12")

    ;; Walks PREVSP back through the chain until it reaches UNTILSP, either NIL or a back up point.  Each of the slots it passes over is reverted to a
    ;; space, return is the slot of early expandable spaces, if any.

    (CL:WHEN PREVSP
        (bind OPREVSP until (EQ PREVSP UNTILSP) do (SETQ OPREVSP PREVSP)
                                                   (SETQ PREVSP (CHAR OPREVSP))
                                                   (CL:WHEN (SMALLP PREVSP)
                                                                       ; Sanity check--shouldn't be 32
                                                       (HELP 'PURGE PREVSP))
                                                   (replace (CHARSLOT CHAR) of OPREVSP with (CHARCODE SPACE))))
    PREVSP])
```

(\TEDIT.FORMATLINE.EMPTY

```
  [LAMBDA (TEXTOBJ CH#1 LINE)                                          ; Edited 15-Mar-2024 22:00 by rmk
                                                                       ; Edited 26-Jan-2024 11:08 by rmk
                                                                       ; Edited  6-Dec-2023 20:15 by rmk
                                                                       ; Edited  3-Dec-2023 19:41 by rmk
                                                                       ; Edited 26-Sep-2023 17:32 by rmk
                                                                       ; Edited 15-Jul-2023 13:52 by rmk
                                                                       ; Edited  2-Jul-2023 15:20 by rmk
                                                                       ; Edited  7-Mar-2023 23:11 by rmk
                                                                       ; Edited  5-Mar-2023 22:57 by rmk
                                                                       ; Edited  4-Mar-2023 21:40 by rmk

    ;; CH#1 is presumably beyond the end.   This returns an empty line descriptor that is set up correctly wrt leading and font.  This is used by
    ;; \FILLPANE to create the dummy line at end of document when you hit an EOL there.  (For safety, \FORMATLINE also calls this if CH#1 doesn't
    ;; pick out a real piece.)                                         ; .

    ;; NOTE: this follows the original in not distinguishing hardcopy-display mode. Presumably empty is empty, even thought the
    ;; ASCENT/DESCENT/LHEIGHT are not scaled.

    ;; Original code asked for the piece at TEXTLEN (last piece?) to get its looks, but those looks would be the TEXTOBJ default looks anyway.

    (CL:UNLESS LINE
        [SETQ LINE (create LINEDESCRIPTOR
                          RIGHTMARGIN _ (FGETTOBJ TEXTOBJ WRIGHT)
                          YBOT _ (SUB1 (FGETTOBJ TEXTOBJ WBOTTOM])
    (\DTEST LINE 'LINEDESCRIPTOR)
    (LET (CHARSLOT FONT TRUEASCENT TRUEDESCENT LM FMTSPEC (THISLINE (FGETTOBJ TEXTOBJ THISLINE)))
         (\TEDIT.FORMATLINE.SETUP TEXTOBJ NIL LINE (WINDOWPROP (CAR (FGETTOBJ TEXTOBJ \WINDOW))
                                                          'DSP))
         (SETQ FMTSPEC (FGETLD LINE LFMTSPEC))
         (SETQ CHARSLOT (FIRSTCHARSLOT THISLINE))
         (replace (THISLINE NEXTAVAILABLECHARSLOT) of THISLINE with (NEXTCHARSLOT CHARSLOT))
         (freplace (THISLINE DESC) of THISLINE with LINE)

         ;; Get looks from the TSTREAM, so that \DISPLAYLINE works.

         (FILLCHARSLOT CHARSLOT NIL (OR (fetch (TEXTSTREAM CURRENTLOOKS) of (FGETTOBJ TEXTOBJ STREAMHINT))
                                        (FGETTOBJ TEXTOBJ CARETLOOKS)
                                        (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS)))

         ;; Not sure what might break if even an emptyTHISLINE doesn't start with charlooks.

         ;;  Font preferences: the caret looks, else the default for this text, else the system default

         (SETQ FONT (CL:IF (CHARW CHARSLOT)
                           (fetch CLFONT of (CHARW CHARSLOT))
                           DEFAULTFONT))
         (SETQ TRUEASCENT (FONTPROP FONT 'ASCENT))
         (SETQ TRUEDESCENT (FONTPROP FONT 'DESCENT))
         (SETQ LM (IPLUS \TEDIT.LINEREGION.WIDTH (FGETTOBJ TEXTOBJ WLEFT)
                         (fetch 1STLEFTMAR of FMTSPEC)))
         (with LINEDESCRIPTOR LINE (SETQ LDUMMY T)
               (SETQ LCHAR1 CH#1)
               (SETQ LCHARLIM CH#1)
               (SETQ 1STLN T)
               (SETQ LSTLN T)
               (SETQ LMARK NIL)
               (SETQ LX1 LM)
               (SETQ LXLIM LM)
               (SETQ FORCED-END (CHARCODE EOL))
               (SETQ LDIRTY NIL)
               (SETQ LHASPROT NIL)
               (SETQ LFMTSPEC FMTSPEC)
               (SETQ LEFTMARGIN LM)
               (SETQ RIGHTMARGIN (CL:IF (ZEROP (fetch RIGHTMAR of FMTSPEC))
                                        (IDIFFERENCE (FGETTOBJ TEXTOBJ WRIGHT)
                                               \TEDIT.LINEREGION.WIDTH)
                                        (fetch RIGHTMAR of FMTSPEC)))
               (SETQ LTRUEASCENT TRUEASCENT)
               (SETQ LTRUEDESCENT TRUEDESCENT)
               (SETQ LHEIGHT (IPLUS TRUEASCENT TRUEDESCENT)))

         ;; Just to initialize the rest of the fields--no intended transformations.

         (\TEDIT.FORMATLINE.VERTICAL LINE TEXTOBJ)
         (\TEDIT.FORMATLINE.HORIZONTAL LINE THISLINE NIL 0 0)
         LINE])
```

(\TEDIT.FORMATLINE.UPDATELOOKS

```
  [LAMBDA (TSTREAM PC)                                           ; Edited 17-Mar-2024 11:08 by rmk
                                                                 ; Edited 15-Mar-2024 19:34 by rmk
                                                                 ; Edited 24-Dec-2023 22:54 by rmk
                                                                 ; Edited 23-Dec-2023 20:37 by rmk
                                                                 ; Edited 22-Aug-2023 16:46 by rmk
                                                                 ; Edited 24-Jul-2023 16:39 by rmk
                                                                 ; Edited  7-Mar-2023 20:54 by rmk
                                                                 ; Edited 30-May-91 21:47 by jds

;;; Called from \TEDIT.INSTALL.PIECE under \FORMATLINE only when the new piece has different looks than the previous piece. This updates the
;;; formatting fields such as ASCENTC, DESCENTC, etc.  This assumes that the \INSTALL.PIECE caller has passed over any invisible pieces, and that
;;; TSTREAM is set up consistently with looks that match PC

    ;; RMK: Storing the  looks in theTEXTSTREAM here seems to be an attempt to avoid calls to the \TEDIT.APPLY.STYLES function in the transition
    ;; from piece to piece.  Presumably, the looks of each piece may be incomplete, and missing fields are filled in from the current (sequence of?)
    ;; styles.  If the style is changed dynamically, then (also presumably) all of the currently displayed pieces should be upgraded.  But that doesn't
    ;; appear to happen.

    ;; A simpler implementation, whether dynamic or not, would be to expand the looks when the piece is created or the style changes, so that each
    ;; piece is always references its completed looks.  But the piece also needs to keep track of its partial looks, for restyling and for saving.

    ;; Style sheets are undocumented, I suspect that this was never really thought through.

    (DECLARE (USEDFREE LINETYPE CHARSLOT CHNO PROTECTED OFFSET ASCENTC DESCENTC FONT IMAGESTREAM KERN
                    UNBREAKABLE))
    (CL:UNLESS PC                                                ; Ran off the end ? Skips the ENDOFSTREAMOP
        (RETFROM (FUNCTION \TEDIT.TEXTBIN)
               NIL))
    (LET (PLOOKS INVISIBLERUN SCALE CLOFFSET)

         ;;

         ;; We have to adjust the CHNO to pass over invisible pieces, and to record the number of characters we passed over in THISLINE's
         ;; character vector.  This maintains the correspondence between the indexing of actual characters in the vector and characters positions in
         ;; the stream.  This information isn't need for display, but TEDIT.SCAN.LINE requires that mapping.

         ;; Invisible runs are coded in a character slot, like other non-character entries (looks, objects) by putting a NIL in the CHAR field of a slot and
         ;; putting the non-character information in the CHARWIDTH field.  Thus, an invisible run is represented as a pair (NIL,runlength).

         (SETQ INVISIBLERUN (for old PC inpieces PC until (VISIBLEPIECEP PC) sum (PLEN PC)))
         (if (EQ 0 INVISIBLERUN)
                then ;; If the looks are the same as current looks, we don't need to change anything.  APPLY STYLES AT PIECE CREATION??
                     (SETQ PLOOKS (PLOOKS PC))
                     (CL:UNLESS (EQ PLOOKS (ffetch (TEXTSTREAM CURRENTLOOKS) of TSTREAM))
                         (freplace (TEXTSTREAM CURRENTLOOKS) of TSTREAM with PLOOKS)
                         ;;
                         (SETQ OFFSET (OR (ffetch (CHARLOOKS CLOFFSET) of PLOOKS)
                                          0))
                         (SETQ FONT (fetch (CHARLOOKS CLFONT) of PLOOKS))
                                                                    ; CLFONT is a display font or a class
                         [if (EQ LINETYPE 'TRUEHARDCOPY)
                             then (SETQ FONT (FONTCOPY FONT 'DEVICE IMAGESTREAM))
                                                                    ; Hardcopy widths and verticals
                                  (SETQ ASCENTC (ffetch \SFAscent of FONT))
                                  (SETQ DESCENTC (ffetch \SFDescent of FONT))
                                  (CL:UNLESS (EQ OFFSET 0)
                                      (SETQ OFFSET (HCSCALE (DSPSCALE NIL IMAGESTREAM)
                                                            OFFSET)))
                           else (CL:WHEN (type? FONTCLASS FONT)     ; Display widths and verticals
                                    (SETQ FONT (FONTCOPY FONT 'DEVICE 'DISPLAY)))
                                (SETQ ASCENTC (ffetch \SFAscent of FONT))
                                (SETQ DESCENTC (ffetch \SFDescent of FONT))
                                (CL:WHEN (EQ LINETYPE 'HARDCOPYDISPLAY)
                                                                    ; Switch widths to hardcopy
                                    (SETQ FONT (FONTCOPY FONT 'DEVICE IMAGESTREAM)))]
                         ;;
                         (SETQ UNBREAKABLE (fetch (CHARLOOKS CLUNBREAKABLE) of PLOOKS))
                         (SETQ KERN (LISTGET (ffetch (CHARLOOKS CLUSERINFO) of PLOOKS)
                                           'KERN))
                         ;; Apparently, KERN's are given in display points, which seems odd.  So here we scale up. Is there just a single kern value?
                         ;; Very strange.
                         (CL:WHEN KERN
                             (SETQ KERN (HCSCALE (DSPSCALE NIL IMAGESTREAM)
                                               KERN)))
                         (STREAMPROP TSTREAM 'KERN KERN)
                         (CL:WHEN (ffetch (CHARLOOKS CLPROTECTED) of PLOOKS)
                                                                    ; Mark the line as containing protected text
                             (SETQ PROTECTED T))
                         (PUSHCHAR CHARSLOT NIL PLOOKS))
                     (CL:UNLESS T
                         ;; This (with higher spevars for FMTSPEC and TABSPEC) would allow tabspecs to change across a paragraph.  But then
                         ;; what should the paragraph-looks menu show?
                         (EQ FMTSPEC (PPARALOOKS PC))
                         (SETQ FMTSPEC (PPARALOOKS PC))
                         (SETQ TABSPEC (ffetch (FMTSPEC TABSPEC) of FMTSPEC))
                         (CL:WHEN (EQ LINETYPE 'TRUEHARDCOPY)
                             (SETQ TABSPEC (\TEDIT.FORMATLINE.SCALETABS TABSPEC (DSPSCALE NIL IMAGESTREAM)))))
              else (add CHNO INVISIBLERUN)
                   (\TEDIT.INSTALL.PIECE TSTREAM PC 0))
          PC])
```

(\TEDIT.FORMATLINE.LASTLEGAL

```
  [LAMBDA NIL                                                       ; Edited  1-Feb-2024 16:51 by rmk
                                                                    ; Edited  2-Jul-2023 14:39 by rmk
                                                                    ; Edited 17-Mar-2023 05:36 by rmk

    ;; An overflowing line without the kind of break point we are looking for (spaces, explicit hyphens).

    ;; Find the last legal break point, given the global TEDIT control variables TEDIT.DONT.BREAK.CHARS and TEDIT.DONT.LAST.CHARS.

    ;; If we run back to the beginning without finding a good break, we just take the original overflowed line. (Or, we could just chop at the end, and
    ;; push the residue to the next line?

    ;; Once we find the break point, we have to sweep through from the beginning in order to accurately know the lines ascent and descent at the break
    ;; point.

    (DECLARE (USEDFREE THISLINE TX CHNO CHARSLOT TRUEASCENT TRUEDESCENT LINETYPE IMAGESTREAM TABPENDING))
    (LET [(BESTSLOT (find SLOT PCS backcharslots (PREVCHARSLOT! CHARSLOT)
                       suchthat (CL:WHEN (AND TABPENDING (EQ SLOT (fetch (PENDINGTAB PTCHARSLOT) of TABPENDING)))
                                       (SETQ TABPENDING NIL))
                             (OR (MEMB CHAR TEDIT.DONT.BREAK.CHARS)
                                 (AND (SETQ PCS (PREVCHARSLOT! SLOT))
                                      (MEMB (CHAR PCS)
                                            TEDIT.DONT.LAST.CHARS]

         ;; BESTSLOT is our last legal  break. Replay to figure out TX, CHNO, ASCENT, DESCENT

         (CL:WHEN BESTSLOT
             (SETQ TX (SETQ TRUEASCENT (SETQ TRUEDESCENT 0)))
             (SETQ CHNO (SUB1 CH#1))
             (for old CHARSLOT FONT OFFSET incharslots THISLINE
                do [if CHAR
                       then (add CHNO 1)
                            (add TX CHARW)
                     else                                           ; Must be looks
                         (SETQ OFFSET (OR (fetch (CHARLOOKS CLOFFSET) of CHARW)
                                            0))
                         (SETQ FONT (fetch (CHARLOOKS CLFONT) of CHARW))
                         [SETQ FONT (if (EQ LINETYPE 'TRUEHARDCOPY)
                                        then (SETQ OFFSET (HCSCALE (DSPSCALE NIL IMAGESTREAM)
                                                                    OFFSET))
                                              (FONTCOPY FONT 'DEVICE IMAGESTREAM)
                                       else (FONTCOPY FONT 'DEVICE 'DISPLAY]
                         (SETQ TRUEASCENT (IMAX TRUEASCENT (IDIFFERENCE (ffetch \SFAscent of FONT)
                                                                    OFFSET)))
                         (SETQ TRUEDESCENT (IMAX TRUEDESCENT (IDIFFERENCE (ffetch \SFDescent of FONT)
                                                                     OFFSET]
                repeatuntil (EQ CHARSLOT BESTSLOT))
             T)])
```

(\TEDIT.LINES.ABOVE

```
  [LAMBDA (TEXTOBJ CHN YBOTN)                                    ; Edited 15-Mar-2024 19:22 by rmk
                                                                 ; Edited  5-Apr-2023 09:13 by rmk
                                                                 ; Edited  1-Apr-2023 12:02 by rmk
                                                                 ; Edited 30-May-91 23:02 by jds

    ;; Produces a chain of formatted lines where  LCHAR1 of the first one either starts a paragraph or comes immediately after a forced end. LN, the
    ;; last line of the chain includes CHN.  The LCHAR's and X positions are good, and their Y positions are set relative to YBOTN, the intended YBOT
    ;; of LN.

    ;; We assume this is not called on an empty text (TEXTLEN = 0), since we wouldn't know what to return. Caller should check that.

    (CL:WHEN (IGREATERP CHN (TEXTLEN TEXTOBJ))
        (SETQ CHN (TEXTLEN TEXTOBJ)))
    (CL:UNLESS YBOTN (SETQ YBOTN 0))
    (bind L1 LN LINE HEIGHT (CHNO _ (\TEDIT.PREVIOUS.LINEBREAK TEXTOBJ CHN))
       first (SETQ L1 (\TEDIT.FORMATLINE TEXTOBJ CHNO))         ; CHNO is the first char of the top line
            (SETQ LN L1)
            (SETQ CHNO (ADD1 (GETLD L1 LCHARLIM)))
       until (IGREATERP CHNO CHN) do (SETQ LINE (\TEDIT.FORMATLINE TEXTOBJ CHNO))
                                                                 ; The line immediately after a preceding known break
                                     (LINKLD LN LINE)
                                     (SETQ LN LINE)
                                     (SETQ CHNO (ADD1 (GETLD LINE LCHARLIM)))
       finally ;; Fill in the YBOT's, given that YBOTN is the YBOT of LN.

             (for L (YB _ YBOTN)
                 backlines LN do (SETYPOS L YB)
                                 (add YB (GETLD L LHEIGHT)))
             (RETURN (LIST L1 LN])
)

(RPAQ? TEDIT.LINELEADING.BELOW NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.LINELEADING.BELOW)
)

(DEFINEQ
```

(\CLEARTHISLINE

```
  [LAMBDA (THISLINE)                                             ; Edited  7-Nov-2022 10:09 by rmk

    ;; This sets it up for a consistency checker to determine that something has gone wrong.  Only called in an assertion.

    (create THISLINE smashing THISLINE DESC _ 'NODESC TLSPACEFACTOR _ 'NOSPACEFACTOR TLFIRSTSPACE _
                             'NOTLFIRSTSPACE NEXTAVAILABLECHARSLOT _ (FIRSTCHARSLOT THISLINE))
    (for CHARSLOT _ (FIRSTCHARSLOT THISLINE)
        (LASTCHARSLOT _ (LASTCHARSLOT THISLINE)) until (EQ CHARSLOT LASTCHARSLOT)
       do (PUSHCHAR CHARSLOT 'BADCHAR 'BADCHARW))
    THISLINE])
```

(\TLVALIDATE

```
  [LAMBDA (THISLINE)                                             ; Edited 15-Mar-2024 19:33 by rmk
                                                                 ; Edited  7-Nov-2022 10:16 by rmk

    ;; Check validity of THISLINE, either just before or anytime after \TEDIT.FORMATLINE.JUSTIFY

    [with THISLINE THISLINE (CL:WHEN (EQ DESC 'NODESC)
                                  (HELP "INVALID THISLINE" DESC))
          (CL:WHEN (EQ TLSPACEFACTOR 'NOSPACEFACTOR)
                 (HELP "INVALID THISLINE" TLSPACEFACTOR))
          (CL:WHEN (EQ TLFIRSTSPACE 'NOTLFIRSTSPACE)
                 (HELP "INVALID THISLINE" TLFIRSTSPACE))
          (CL:UNLESS (CHARSLOTP NEXTAVAILABLECHARSLOT THISLINE)
              (HELP "INVALID THISLINE" 'NEXTAVAILABLE))]
    (for CHARSLOT incharslots THISLINE do (if CHAR
                                              then (CL:UNLESS (OR (SMALLP CHAR)
                                                                  (CHARSLOTP CHAR THISLINE))

                                                       ;; CHARSLOTP if spaces haven't been instantiated

                                                       (HELP "INVALID THISLINE" 'BADCHAR))
                                                   (CL:UNLESS (SMALLP CHARW)
                                                              (HELP "INVALID THISLINE" 'BADCHARW))
                                                    elseif (OR (SMALLP CHARW)
                                                               (type? CHARLOOKS CHARW))
                                                    else (HELP "INVALID THISLINE" 'BADCHARW])

)

;; Consistency checking

(RPAQ? *TEDIT-CACHED-FMTSPEC* NIL)

;; Heuristic for \FORMATLINE

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *TEDIT-CACHED-FMTSPEC*)
)

(DEFINEQ

(\TEDIT.DISPLAYLINE
  [LAMBDA (TEXTOBJ LINE PANE)                                    ; Edited 20-Mar-2024 10:57 by rmk
                                                                 ; Edited 15-Mar-2024 22:04 by rmk
                                                                 ; Edited 24-Dec-2023 22:05 by rmk
                                                                 ; Edited  2-Dec-2023 11:34 by rmk
                                                                 ; Edited 20-Nov-2023 13:57 by rmk
                                                                 ; Edited 28-Oct-2023 23:57 by rmk
                                                                 ; Edited 11-Oct-2023 10:47 by rmk
                                                                 ; Edited  2-Aug-2023 12:50 by rmk
                                                                 ; Edited 22-Jun-2023 17:37 by rmk
                                                                 ; Edited 24-Apr-2023 00:08 by rmk
                                                                 ; Edited 10-Apr-2023 12:41 by rmk
                                                                 ; Edited 16-Mar-2023 23:30 by rmk
                                                                 ; Edited  9-Mar-2023 14:06 by rmk
                                                                 ; Edited  7-Mar-2023 23:11 by rmk

    ;; Display the line of text LINE in the edit window where it belongs.

    ;; Validate the incoming arguments so ffetch can be used consistently for all their field extractions.

    (TEXTOBJ! TEXTOBJ)
    (\DTEST LINE 'LINEDESCRIPTOR)
    (LET ((WINDOWDS (WINDOWPROP (FGETPANE PANE PWINDOW)
                            'DSP))
          (THISLINE (\DTEST (FGETTOBJ TEXTOBJ THISLINE)
                            'THISLINE))
          (OLDCACHE (fetch (LINECACHE LCBITMAP) of (FGETTOBJ TEXTOBJ DISPLAYCACHE)))
          (DS (FGETTOBJ TEXTOBJ DISPLAYCACHEDS))
          CACHE XOFFSET CLIPLEFT CLIPRIGHT DISPLAYDATA DDPILOTBBT CURY LHEIGHT)
         [SETQ LHEIGHT (COND
                          ((FGETLD LINE PREVLINE)                ; So if theres a base-to-base measure, we clear everything right.
                           (IMAX (IDIFFERENCE (FGETLD (FGETLD LINE PREVLINE)
                                                      YBOT)
                                        (FGETLD LINE YBOT))
                                 (FGETLD LINE LHEIGHT)))
                          (T (FGETLD LINE LHEIGHT]
         (SETQ CACHE (\TEDIT.LINECACHE (FGETTOBJ TEXTOBJ DISPLAYCACHE)
                            (FGETLD LINE LXLIM)
                            LHEIGHT))
         (CL:UNLESS (EQ CACHE OLDCACHE)                          ; We changed the bitmaps because this line was bigger--update
                                                                 ; the displaystream, too
             (DSPDESTINATION CACHE DS)
             (DSPCLIPPINGREGION (create REGION
                                       LEFT _ 0
                                       BOTTOM _ 0
                                       WIDTH _ (fetch BITMAPWIDTH of CACHE)
                                       HEIGHT _ (fetch BITMAPHEIGHT of CACHE))
                    DS))
         (BLTSHADE WHITESHADE CACHE 0 0 NIL NIL 'REPLACE)        ; Clear the line cache
         (CL:WHEN [AND (IGEQ (FGETLD LINE LCHAR1)
                             1)
                       (ILEQ (FGETLD LINE LCHAR1)
                             (FGETTOBJ TEXTOBJ TEXTLEN))
                       (OR (IGEQ (FGETLD LINE YBOT)
                                 (FGETTOBJ TEXTOBJ WBOTTOM))
                           (IGREATERP LHEIGHT (fetch HEIGHT of (DSPCLIPPINGREGION NIL PANE]

             ;; Only display the line if it contains text (CHAR1 > 0), appears before the end of the text, and is on-screen. Also display clipped lines if
             ;; they are bigger than the window

             (CL:UNLESS (EQ LINE (fetch (THISLINE DESC) of THISLINE))
                                                                 ; No image cache -- re-format and display
                 (\TEDIT.FORMATLINE TEXTOBJ (FGETLD LINE LCHAR1)
                        LINE))
             (MOVETO (FGETLD LINE LX1)
                    (FGETLD LINE DESCENT)
                    DS)
             (SETQ DISPLAYDATA (ffetch (STREAM IMAGEDATA) of DS))  ; IMAGEDATA of the display stream, not textstream
             (SETQ DDPILOTBBT (ffetch DDPILOTBBT of DISPLAYDATA))
```

```
            (SETQ XOFFSET (ffetch DDXOFFSET of DISPLAYDATA))
            ;; The X position of the left edge of the window, since \TEDIT.BLTCHAR works on the screen bitmap itself.
            (SETQ CLIPLEFT (ffetch DDClippingLeft of DISPLAYDATA))
                                                       ; The left and right edges of the clipping region for the text
                                                       ; display window.
            (SETQ CLIPRIGHT (ffetch DDClippingRight of DISPLAYDATA))
            ;; We know that the line's first CLOOKS comes before the first CHAR
            [for CHARSLOT CLOOKS LOOKSTARTX (TX _ (IPLUS XOFFSET (FGETLD LINE LX1)))
                (TERMSA _ (FGETTOBJ TEXTOBJ TXTTERMSA))
                incharslots THISLINE
               do ;; Display the line character by character.  CHAR and CHARW are bound to CHARSLOT values
                  (CL:WHEN (FMEMB CHAR (CHARCODE (EOL FORM)))  ; \FORMATLINE used space-width for EOL and FORM. Display
                                                       ; them that way.
                      (SETQ CHAR (CHARCODE SPACE)))
                  (SELCHARQ CHAR
                       ((TAB Meta,TAB)
                            (CL:WHEN (OR (EQ CHAR (CHARCODE Meta,TAB))
                                         (ffetch CLLEADER of CLOOKS)
                                         (EQ (ffetch CLUSERINFO of CLOOKS)
                                             'DOTTEDLEADER))

                                ;; Not just white space, have to fill in with dots.

                                (\TEDIT.DISPLAYLINE.TABS CHARW DS TX TERMSA LINE CLOOKS DISPLAYDATA DDPILOTBBT
                                       CLIPRIGHT TEXTOBJ))
                            (add TX CHARW))
                       (NIL                                    ; Must be looks. Line-start looks are guaranteed to come before
                                                               ; any character/object
                            (CL:WHEN (type? CHARLOOKS CHARW)
                                (CL:UNLESS LOOKSTARTX          ; LOOKSTARTX: Starting X position for the current-looks text.
                                    (SETQ LOOKSTARTX (IDIFFERENCE TX XOFFSET)))
                                (freplace DDXPOSITION of DISPLAYDATA with (IDIFFERENCE TX XOFFSET))
                                                               ; Make the displaystream reflect our current X position
                                (CL:WHEN CLOOKS                ; Underline/overline/strike the just-finished looks run
                                    (TEDIT.MODIFYLOOKS LINE LOOKSTARTX DS CLOOKS (FGETLD LINE DESCENT)))
                                (SETQ CLOOKS CHARW)
                                (DSPFONT (ffetch CLFONT of CLOOKS)
                                       DS)
                                (CL:UNLESS (EQ 0 (ffetch CLOFFSET of CLOOKS))
                                                               ; Account for super/subscripting
                                    (RELMOVETO 0 (ffetch CLOFFSET of CLOOKS)
                                           DS))
                                (SETQ LOOKSTARTX (IDIFFERENCE TX XOFFSET))))
                       (PROGN (if (IMAGEOBJP CHAR)
                                  then ;; Go to the base line, left edge of the image region.

                                       (SETQ CURY (DSPYPOSITION NIL DS))
                                       (MOVETO (IDIFFERENCE TX XOFFSET)
                                              CURY DS)
                                       (APPLY* (IMAGEOBJPROP CHAR 'DISPLAYFN)
                                              CHAR DS 'DISPLAY (FGETTOBJ TEXTOBJ STREAMHINT))
                                       (DSPFONT (ffetch CLFONT of CLOOKS)
                                              DS)              ; Restore the character font, move to just after the object.
                                       (MOVETO (IDIFFERENCE TX XOFFSET)
                                              CURY DS)
                                elseif TERMSA
                                  then                         ; Using special instrns from TERMSA
                                       (\DSPPRINTCHAR DS CHAR)
                                elseif (DIACRITICP CHAR)
                                  then (MI-TEDIT.BLTCHAR CHAR DS (IPLUS TX (\TEDIT.DIACRITIC.SHIFT CHARSLOT
                                                                                    THISLINE DS))
                                              DISPLAYDATA DDPILOTBBT CLIPRIGHT)
                                       (SETQ CHARW 0)
                                else                           ; Native charcodes
                                     (MI-TEDIT.BLTCHAR CHAR DS TX DISPLAYDATA DDPILOTBBT CLIPRIGHT))
                              (add TX CHARW)))
               finally (replace DDXPOSITION of DISPLAYDATA with (IDIFFERENCE TX XOFFSET))
                                                               ; Make any necessary looks mods to the last run of characters
                     (CL:WHEN CLOOKS
                         (TEDIT.MODIFYLOOKS LINE LOOKSTARTX DS CLOOKS (FGETLD LINE DESCENT)))])
        (BITBLT CACHE 0 0 WINDOWDS 0 (FGETLD LINE YBOT)
               (FGETTOBJ TEXTOBJ WRIGHT)
               LHEIGHT
               'INPUT
               'REPLACE)                                       ; Paint the cached image on the screen (this lessens flicker
                                                               ; during update)
        (CL:WHEN (fetch (FMTSPEC FMTREVISED) of (FGETLD LINE LFMTSPEC))
                                                               ; This paragraph has been revised, so mark it.
            (\TEDIT.MARK.REVISION TEXTOBJ (FGETLD LINE LFMTSPEC)
                   WINDOWDS LINE))
        (SELECTQ (FGETLD LINE LMARK)
            (GREY                                              ; This line has some property that isn't visible to the user.  Tell
                                                               ; him to be careful
                  (BLTSHADE 42405 WINDOWDS 0 (FGETLD LINE YBASE)
                         6 6 'PAINT))
```

```
              (SOLID (BLTSHADE BLACKSHADE WINDOWDS 0 (FGETLD LINE YBASE)
                              6 6 'PAINT))
              (BLTSHADE WHITESHADE WINDOWDS 0 (FGETLD LINE YBASE)
                      6 6 'PAINT))
         (FSETLD LINE LDIRTY NIL)
         LINE])
```

(\TEDIT.DISPLAYLINE.TABS

```
  [LAMBDA (CW DS TX TERMSA LINE CLOOKS DISPLAYDATA DDPILOTBBT CLIPRIGHT TEXTOBJ)
                                                                    ; Edited 10-Oct-2023 23:29 by rmk
                                                                    ; Edited  4-Oct-2023 21:16 by rmk
                                                                    ; Edited  3-Jul-2023 22:02 by rmk
                                                                    ; Edited  4-Mar-2023 22:17 by rmk
                                                                    ; Edited  1-Oct-2022 11:35 by rmk
                                                                    ; Edited 24-Sep-2022 21:19 by rmk

    ;; Fills in tab-space CW with dotted leaders.  LINE is only needed to get the FMTSPEC.  TEXTOBJ only needed to get the hardcopy-display
    ;; stream.

    (bind TTX DOTWIDTH (FMTSPEC _ (GETLD LINE LFMTSPEC))
       first
             ;; The dots on successive lines may not align so well, in hardcopy display mode.  But that's not a mode that looks good anyway.  The
             ;; TERMSA probably screws it anyway.

             [SETQ DOTWIDTH (CL:IF (fetch (FMTSPEC FMTHARDCOPY) of FMTSPEC)
                                 [HCUNSCALE (fetch (FMTSPEC FMTHARDCOPYSCALE) of FMTSPEC)
                                        (CHARWIDTH (CHARCODE %.)
                                               (FONTCOPY (fetch CLFONT of CLOOKS)
                                                      'DEVICE
                                                      (FGETTOBJ TEXTOBJ DISPLAYHCPYDS]
                                 (CHARWIDTH (CHARCODE %.)
                                        (fetch CLFONT of CLOOKS)))]
             [SETQ TTX (IPLUS TX DOTWIDTH (IDIFFERENCE DOTWIDTH (IREMAINDER TX DOTWIDTH]
       while (ILEQ TTX (IPLUS TX CW)) do (if TERMSA
                                               then                 ; Using special instrns from TERMSA
                                                    (\DSPPRINTCHAR DS (CHARCODE %.))
                                             else                   ; Native charcodes
                                                  (MI-TEDIT.BLTCHAR (CHARCODE %.)
                                                         DS
                                                         (IDIFFERENCE TTX DOTWIDTH)
                                                         DISPLAYDATA DDPILOTBBT CLIPRIGHT))
                                           (add TTX DOTWIDTH])
```

(\TEDIT.LINECACHE

```
  [LAMBDA (CACHE WIDTH HEIGHT)                                      (* jds "21-Apr-84 00:52")

            (* Given a candidate line cache, return the bitmap, making sure it's at least WIDTH by HEIGHT big.)

    (PROG ((BITMAP (fetch LCBITMAP of CACHE))
           CW CH)
          (SETQ CW (fetch BITMAPWIDTH of BITMAP))
          (SETQ CH (fetch BITMAPHEIGHT of BITMAP))
          (COND
             ((AND (IGEQ CW WIDTH)
                   (IGEQ CH HEIGHT))
              (RETURN BITMAP))
             (T (RETURN (replace LCBITMAP of CACHE with (BITMAPCREATE (IMAX CW WIDTH)
                                                                   (IMAX CH HEIGHT])
```

(\TEDIT.CREATE.LINECACHE

```
  [LAMBDA (%#CACHES)                                                (* jds "21-Apr-84 00:58")
                                                                    (* Create a linked-together set of LINECACHEs, for saving line
                                                                    images.)
    (PROG [(CACHES (for I from 1 to %#CACHES collect (create LINECACHE
                                                         LCBITMAP _ (BITMAPCREATE 100 15]
          [for CACHE on CACHES do                                   (* Link the caches together.)
                                  (replace LCNEXTCACHE of (CAR CACHE) with (OR (CADR CACHE)
                                                                                (CAR CACHES]
          (RETURN CACHES])
```

(\TEDIT.BLTCHAR

```
  [LAMBDA (CHARCODE DISPLAYSTREAM CURX DISPLAYDATA DDPILOTBBT CLIPRIGHT)
                                                                    ; Edited 15-Mar-2024 14:39 by rmk
                                                                    (* jds " 9-Jan-86 17:14")

    ;; Version of BLTCHAR peculiar to TEdit -- relies on \TEDIT.DISPLAYLINE to make sure things keep working right.

    ;; puts a character on a guaranteed display stream.  Much of the information needed by the BitBlt microcode is prestored by the routines that
    ;; change it.  This is kept in the BitBltTable.                  ; knows about the representation of display stream image data
                                                                    ; MUST NOT POINT AT A WINDOW'S DISPLAYSTREAM!!!

    ;; ASSUMES THAT WE NEVER WANT TO PRINT TO THE LEFT OF ORIGIN 0 ON THE LINE CACHE BITMAP, OR THAT IF WE DO, ALL BETS
    ;; ARE OFF

    (DECLARE (LOCALVARS . T))
    (PROG (NEWX LEFT RIGHT IMAGEWIDTH (CHAR8CODE (\CHAR8CODE CHARCODE)))
          [COND
```

```
            ((NEQ (ffetch DDCHARSET of DISPLAYDATA)
                  (\CHARSET CHARCODE))
             (\CHANGECHARSET.DISPLAY DISPLAYDATA (\CHARSET CHARCODE]
         (SETQ IMAGEWIDTH (\GETBASE (fetch DDCHARIMAGEWIDTHS of DISPLAYDATA)
                                 (\CHAR8CODE CHARCODE)))
         (SETQ NEWX (IPLUS CURX IMAGEWIDTH))
         (SETQ LEFT (IMAX 0 CURX))
         (SETQ RIGHT (IMIN CLIPRIGHT NEWX))
         (COND
            ((ILESSP LEFT RIGHT)                          ; Only print anything if there is a place to put it
             (UNINTERRUPTABLY
                 (freplace PBTDESTBIT of DDPILOTBBT with LEFT)    ; Set up the bitblt-table source left
                 (freplace PBTWIDTH of DDPILOTBBT with (IMIN IMAGEWIDTH (IDIFFERENCE RIGHT LEFT)))
                 (freplace PBTSOURCEBIT of DDPILOTBBT with (\GETBASE (fetch DDOFFSETSCACHE of DISPLAYDATA)
                                                                     (\CHAR8CODE CHARCODE)))
                 (\PILOTBITBLT DDPILOTBBT 0))
             T])

(\TEDIT.DIACRITIC.SHIFT
  [LAMBDA (CHARSLOT THISLINE IMAGESTREAM)                 ; Edited  2-Dec-2023 15:58 by rmk
                                                          ; Edited 28-Oct-2023 23:51 by rmk

    ;; Called when CHARSLOT contains a diacritic.  Computes the X position shift that will center the diacritic over the next character.  If negative, the
    ;; caller should move forward by the shift the next character rather than  the diacritic.  In effect, the diacritic should be treated as if its width is
    ;; (IMINUS SHIFT) and the next character should be treated as if its with is incremented by (IMINUS SHIFT).

    (for CS (DWIDTH _ (CHARW CHARSLOT))
        incharslots
        (NEXTCHARSLOT CHARSLOT) when CHAR do (RETURN (FIXR (FQUOTIENT (- CHARW DWIDTH)
                                                                      2)))
       finally (RETURN 0])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS MI-TEDIT.BLTCHAR MACRO [(CHARCODE DISPLAYSTREAM CURX DISPLAYDATA DDPILOTBBT CLIPRIGHT)
                                  (COND
                                     ((EQ 'MAIKO (MACHINETYPE))
                                      (SUBRCALL TEDIT.BLTCHAR CHARCODE DISPLAYSTREAM CURX DISPLAYDATA
                                             DDPILOTBBT CLIPRIGHT))
                                     (T (\TEDIT.BLTCHAR CHARCODE DISPLAYSTREAM CURX DISPLAYDATA DDPILOTBBT
                                               CLIPRIGHT])
)
)

(DEFINEQ

(\TEDIT.UPDATE.SCREEN
  [LAMBDA (TEXTOBJ)                                       ; Edited 15-Mar-2024 22:00 by rmk
                                                          ; Edited 16-Dec-2023 23:52 by rmk
                                                          ; Edited 12-Oct-2023 15:27 by rmk
                                                          ; Edited 17-Sep-2023 11:50 by rmk
                                                          ; Edited 22-May-2023 22:19 by rmk
                                                          ; Edited 17-May-2023 08:58 by rmk
                                                          ; Edited  5-May-2023 13:16 by rmk
                                                          ; Edited  5-Oct-2022 21:30 by rmk
    (CL:UNLESS (GETTOBJ TEXTOBJ TXTDON'TUPDATE)
        [LET ((DIRTYCHARS (\TEDIT.FIND.DIRTYCHARS TEXTOBJ)))
             (if DIRTYCHARS
                 then
                      ;; As long as we have this path, we don't want the line updater to update the selection. Updating the looks affects the line
                      ;; (so we need to know what characters changed), but the characters don't move around.  We want the rest of the insertion
                      ;; callers to avoid this entry.

                      (\TEDIT.UPDATE.LINES TEXTOBJ 'APPEARANCE (CAR DIRTYCHARS)
                             (CDR DIRTYCHARS))
               else (for PANE inpanes TEXTOBJ do (\TEDIT.FILLPANE (fetch (TEXTWINDOW PLINES) of PANE)
                                                        TEXTOBJ PANE]
        (FSETTOBJ TEXTOBJ TXTNEEDSUPDATE NIL))])

(\TEDIT.BACKFORMAT
  [LAMBDA (TEXTOBJ DY CH1 HEIGHT)                         ; Edited 20-Mar-2024 06:46 by rmk
                                                          ; Edited 15-Mar-2024 19:44 by rmk
                                                          ; Edited 30-Nov-2023 21:16 by rmk
                                                          ; Edited  3-Nov-2023 12:02 by rmk
                                                          ; Edited  6-Apr-2023 16:46 by rmk
                                                          ; Edited  5-Apr-2023 09:13 by rmk
                                                          ; Edited 30-May-91 15:58 by jds

    ;; This computes the shortest sequence of globally correct lines above and including the line with CH1 whose total height is GEQ DY

    ;; Returns the head line of the chain whose YBOT is the actual height (possibly a little greater than DY).

    ;; This computes block by block, where the first line of a block either starts a paragraph or comes immediately after a forced break.

    (bind L1 PAIR (CHNO _ CH1) until (IGREATERP HEIGHT DY) while (IGEQ CHNO 1)
        do (SETQ PAIR (\TEDIT.LINES.ABOVE TEXTOBJ CHNO HEIGHT))     ; The block may go beyond DY
           (LINKLD (CADR PAIR)
                  L1)                                               ; This block's LN links to previous L1
           (SETQ L1 (CAR PAIR))
           (SETQ HEIGHT (GETLD L1 YTOP))
           (SETQ CHNO (SUB1 (GETLD L1 LCHAR1)))
        finally                                                     ; Perhaps the break was beyond DY
              (RETURN (find L inlines L1 suchthat (ILEQ (FGETLD L YBOT)
                                                        DY])
```

(\TEDIT.PREVIOUS.LINEBREAK

```
  [LAMBDA (TEXTOBJ CHNO)                                            ; Edited 17-Mar-2024 12:05 by rmk
                                                                    ; Edited 11-Dec-2023 21:59 by rmk
                                                                    ; Edited 16-Oct-2023 23:19 by rmk
                                                                    ; Edited 31-Mar-2023 17:44 by rmk
                                                                    ; Edited 28-Mar-2023 09:03 by rmk
                                                                    ; Edited 26-Mar-2023 12:55 by rmk

    ;; Returns the character number of the first character at or before CHNO that would follow a forced line-end or a paragraph end.  Line-formatting
    ;; from that character onward would be consistent with any earlier line-breaks (and wouldn't change if earlier breaks changed).

    (if (ILEQ CHNO 1)
        then 1
      elseif (AND NIL (FGETTOBJ TEXTOBJ FORMATTEDP))
        then ;; [Disabled] For a para-formatted object, back up to the prior linebreak (PPARALAST).  But if EOL's are not always paragraph
             ;; boundaries, this might back up way too far.

             (CAR (\TEDIT.PARA.FIRST TEXTOBJ CHNO))
      else ;; Otherwise, move back thru the text until we find a for-sure line break.

           (CL:WHEN (IGREATERP CHNO (FGETTOBJ TEXTOBJ TEXTLEN))
               (SETQ CHNO (FGETTOBJ TEXTOBJ TEXTLEN)))
           (LET ((TSTREAM (FGETTOBJ TEXTOBJ STREAMHINT))
                 NCHARS)
                (\TEDIT.TEXTSETFILEPTR TSTREAM (SUB1 CHNO))         ; Start at (SUB1 CHNO) because fileptrs are one back from
                                                                    ; characters
                [SETQ NCHARS (find I from 1 suchthat (MEMB (\TEDIT.TEXTBACKFILEPTR TSTREAM)
                                                           (CHARCODE (EOL FORM %#EOL Meta,EOL CR LF NIL]

                ;; If we didn't find a preceding EOL, we must have backed to the beginning of the file (NIL).

                (CL:IF NCHARS
                    (ADD1 (IDIFFERENCE CHNO NCHARS))
                    1)])
```

(\TEDIT.FILLPANE

```
  [LAMBDA (PREVLINE TEXTOBJ PANE)                                   ; Edited 20-Mar-2024 06:43 by rmk
                                                                    ; Edited 15-Mar-2024 14:39 by rmk
                                                                    ; Edited 11-Jan-2024 19:32 by rmk
                                                                    ; Edited  2-Jan-2024 12:45 by rmk
                                                                    ; Edited 24-Dec-2023 22:00 by rmk
                                                                    ; Edited  2-Dec-2023 23:05 by rmk
                                                                    ; Edited  3-Nov-2023 12:03 by rmk
                                                                    ; Edited 17-Sep-2023 14:51 by rmk
                                                                    ; Edited  8-May-2023 21:59 by rmk
                                                                    ; Edited  5-May-2023 10:54 by rmk
                                                                    ; Edited 26-Apr-2023 21:02 by rmk

    ;; This executes whether or not TXTNEEDSUPDATE, callers decide that.

    (LET (LINE)
         ;;
         ;; Find the first on-screen line after PREVLINE, if any.  If the scrolling and other algorithms are tidy, we shouldn't expect to find any lines
         ;; hanging around above the pane.  If none, start with PREVLINE, maybe the dummy.

         (SETQ LINE (find L (PHEIGHT _ (fetch HEIGHT of (DSPCLIPPINGREGION NIL PANE)))
                         inlines
                         (GETLD PREVLINE NEXTLINE) suchthat (ILESSP (FGETLD L YBOT)
                                                                    PHEIGHT)))

         ;;

         (CL:WHEN LINE
             (SETQ PREVLINE (GETLD LINE PREVLINE)))

         ;;

         ;; Format and display any lines that are still needed to fill out the pane.

         (SETQ PREVLINE (\TEDIT.LINES.BELOW PREVLINE NIL PANE TEXTOBJ))

         ;;

         (CL:WHEN (\TEDIT.INSURE.TRAILING.LINE TEXTOBJ PREVLINE)
             (\TEDIT.DISPLAYLINE TEXTOBJ (GETLD PREVLINE NEXTLINE)
                    PANE))

         ;;

         (\TEDIT.CLEARPANE.BELOW.LINE PREVLINE PANE TEXTOBJ)
         (\TEDIT.SET.WINDOW.EXTENT TEXTOBJ PANE)
         (FSETTOBJ TEXTOBJ TXTNEEDSUPDATE NIL])
```

```
(\TEDIT.UPDATE.LINES
  [LAMBDA (TEXTOBJ REASON FIRSTCHANGEDCHNO NCHARSCHANGED DONTDISPLAY)
                                                              ; Edited 20-Mar-2024 06:43 by rmk
                                                              ; Edited 24-Dec-2023 22:00 by rmk
                                                              ; Edited 18-Dec-2023 00:12 by rmk
                                                              ; Edited 16-Dec-2023 13:43 by rmk
                                                              ; Edited  4-Dec-2023 20:37 by rmk
                                                              ; Edited 22-Jun-2023 15:50 by rmk
                                                              ; Edited 11-Jun-2023 18:34 by rmk
                                                              ; Edited  4-May-2023 10:29 by rmk

    ;; This updates the lines in each pane given that NCHARSCHANGED characters with respect to FIRSTCHANGEDCHNO have been modified.  It
    ;; tries to reuse formatting information and screen bitmap images that are valid after the change.

    ;; See line-segmentation comments in \TEDIT.VALID.LINES.

    (CL:UNLESS (GETTOBJ TEXTOBJ TXTDON'TUPDATE)
        (CL:WHEN (type? SELECTION FIRSTCHANGEDCHNO)
            (SETQ NCHARSCHANGED (FGETSEL FIRSTCHANGEDCHNO DCH))
            (SETQ FIRSTCHANGEDCHNO (FGETSEL FIRSTCHANGEDCHNO CH#)))

        ;; If DONTDISPLAY, we ensure lines that are properly formatted and positioned but not displayed.

        (for PANE VALIDS NEXTVALID LASTGAPLINE DELTA inpanes TEXTOBJ as VALIDS
           in (\TEDIT.VALID.LINES TEXTOBJ FIRSTCHANGEDCHNO NCHARSCHANGED REASON) when VALIDS
           do ;; Create/format/display new lines between LASTVALID=(CAR VALIDS) and NEXTVALID

              (SETQ NEXTVALID (CDR VALIDS))
              (SETQ LASTGAPLINE (\TEDIT.LINES.BELOW (CAR VALIDS)
                                       (AND NEXTVALID (SUB1 (FGETLD NEXTVALID LCHAR1)))
                                       PANE TEXTOBJ DONTDISPLAY))
              (LINKLD LASTGAPLINE NEXTVALID)

              ;; The chain that ended at LASTVALID now continues thru LASTGAPLINE to NEXVALID and below.  But the Ypositions of
              ;; NEXTVALID lines have not yet been adjusted, and their images have not been displayed. The top of NEXTVALID should align with
              ;; the bottom of LASTGAPLINE: their Y positions are changed by DELTA.  DELTA is positive if NEXTVALID is moving up (deletion ),
              ;; otherwise insertion. Appearance can go either way.

              (if NEXTVALID
                  then (SETQ DELTA (IDIFFERENCE (FGETLD LASTGAPLINE YBOT)
                                          (FGETLD NEXTVALID YTOP)))

                       ;; Unless DONTDISPLAY, the bitmap for lines from NEXTVALID down has been preserved, even in the insertion case.

                       ;; The gap is filled in with formatted and displayed lines, the last of which now links to NEXTVALID.  NEXTVALID and later
                       ;; lines have good character positions and good bitmaps, but their YPOS are not correct and their bitmaps are not in the
                       ;; right place.

                       ;;

                       ;; In the deletion case, NEXTVALID's current YPOS  will be at or below its target value as determined by the gap-filler, but
                       ;; the gap-filling hasn't disturbed the image.  The bitmap can be raised and the pane filled in below.

                       ;;

                       ;; In the insertion case, the YPOS maybe above the target, but we don't know what it should be until we fill in and display
                       ;; the gap lines.  The gap line-display may have smashed some of the display bits that we otherwise would be available to
                       ;; move down..

                       ;;

                       (if DONTDISPLAY
                           then (for L inlines NEXTVALID do (\TEDIT.LINE.BOTTOM L))
                         elseif (IGREATERP DELTA 0)
                           then ;; Deletion/appearance

                                (\TEDIT.RAISE.LINES NEXTVALID (FGETLD LASTGAPLINE YBOT)
                                       PANE TEXTOBJ)
                         elseif (ILESSP DELTA 0)
                           then ;; Insertion/appearance: bitmaps of NEXTVALID can be shifted down

                                (\TEDIT.LOWER.LINES NEXTVALID LASTGAPLINE PANE TEXTOBJ))
                else (\TEDIT.CLEARPANE.BELOW.LINE LASTGAPLINE PANE TEXTOBJ)
                     (\TEDIT.INSURE.TRAILING.LINE TEXTOBJ LASTGAPLINE)))
        (FSETTOBJ TEXTOBJ TXTNEEDSUPDATE NIL))])

(\TEDIT.CREATEPLINE
  [LAMBDA (TEXTOBJ PANE FIRSTLINE)                              ; Edited 13-Mar-2024 17:02 by rmk
                                                              ; Edited 21-Feb-2024 23:36 by rmk
                                                              ; Edited  2-Jan-2024 13:04 by rmk
                                                              ; Edited 29-Dec-2023 15:48 by rmk

    ;; Creates the initial dummy line PLINES for PANE.  Connects it to FIRSTLINE if provided.

    (LET (DUMMYLINE)

         ;; Initialize with a dummy empty first line with LCHAR1 and LCHARLIM=0 above the pane top. 0 means in particular that the LCHARLIM is
         ;; just before the first character of the file (if there is one).

         ;; 1STLN and LSTLN are NIL, since we don't want to make end paragraph-boundary inferences

         (SETQ DUMMYLINE
          (create LINEDESCRIPTOR
                 LDUMMY _ T
                 YBOT _ (fetch HEIGHT of (DSPCLIPPINGREGION NIL PANE))
                 LCHAR1 _ 0
                 LCHARLIM _ 0
                 RIGHTMARGIN _ (SUB1 (FGETTOBJ TEXTOBJ WRIGHT))
                 LHEIGHT _ 0
                 LX1 _ 0
                 LXLIM _ (FGETTOBJ TEXTOBJ WRIGHT)
                 FORCED-END _ (CHARCODE EOL)
                 ASCENT _ 0
                 DESCENT _ 0
                 LTRUEASCENT _ 0
                 LTRUEDESCENT _ 0
                 LFMTSPEC _ TEDIT.DEFAULT.FMTSPEC
                 1STLN _ NIL
                 LSTLN _ NIL))
         (replace (TEXTWINDOW PLINES) of PANE with DUMMYLINE)       ; Install PANE's new dummy line
         (LINKLD DUMMYLINE FIRSTLINE)                               ; Link the possible first line
         DUMMYLINE])
```

(\TEDIT.FIND.DIRTYCHARS

```
  [LAMBDA (TEXTOBJ)                                                 ; Edited  4-Jan-2024 23:34 by rmk
                                                                    ; Edited  2-Jan-2024 12:15 by rmk
                                                                    ; Edited  2-Dec-2023 23:06 by rmk
                                                                    ; Edited  3-Nov-2023 12:04 by rmk
                                                                    ; Edited  8-May-2023 13:18 by rmk
                                                                    ; Edited 28-Apr-2023 15:30 by rmk

    ;; Returns a pair (firstdirty . ndirties)  figuring the first and maxium range of dirty characters. For programs that mark DIRTY when they modify lines.
    ;; The dirty LCHAR*'s are the same in all panes where they exist.

    (for PANE PLINES FIRSTDIRTYLINE (LASTDIRTYCHAR _ 1)
        inpanes
        (PROGN TEXTOBJ) eachtime (SETQ PLINES (fetch (TEXTWINDOW PLINES) of PANE))
       when (SETQ FIRSTDIRTYLINE (find L inlines (GETLD PLINES NEXTLINE) suchthat (FGETLD L LDIRTY)))
       do ;; Some panes may have more lines than others--we want to get the largest dirty range.

          [SETQ LASTDIRTYCHAR (IMAX LASTDIRTYCHAR (for L (PREV _ FIRSTDIRTYLINE)
                                                      inlines FIRSTDIRTYLINE while (FGETLD L LDIRTY)
                                                     do (SETQ PREV L) finally (RETURN (FGETLD PREV LCHARLIM]
       finally (RETURN (CL:WHEN FIRSTDIRTYLINE
                          (CONS (GETLD FIRSTDIRTYLINE LCHAR1)
                                (IDIFFERENCE (ADD1 LASTDIRTYCHAR)
                                       (FGETLD FIRSTDIRTYLINE LCHAR1))))])
```

(\TEDIT.LINES.BELOW

```
  [LAMBDA (PREVLINE LASTCHAR PANE TEXTOBJ DONTDISPLAY)              ; Edited 15-Mar-2024 19:22 by rmk
                                                                    ; Edited 23-Dec-2023 23:38 by rmk
                                                                    ; Edited 17-Dec-2023 15:56 by rmk
                                                                    ; Edited 14-Dec-2023 12:46 by rmk

    ;; Formats lines after PREVLINE down to the one that contains LASTCHAR and/or does not run off the bottom of PANE.

    ;; Assumes that PREVLINE is correctly formatted and Y-positioned, and already displayed in PANE (if desired).

    ;; Sets the Y positions of all lines relative to PREVLINE, and returns the last properly formatted, positioned, and displayed line, perhaps PREVLINE
    ;; itself if there was nothing below it.

    ;; Also displays the lines, unless DONTDISPLAY. This is an optimization: THISLINE caches the just formatted line, doesn't have to be formatted
    ;; again if it is immediately displayed.  Calling it with DONTDISPLAY NIL followed by DONTDISPLAY T gives exactly the same result as calling it
    ;; once with DONTDISPLAY T.

    (CL:WHEN PREVLINE
        (SETQ LASTCHAR (CL:IF LASTCHAR
                           (IMIN LASTCHAR (FGETTOBJ TEXTOBJ TEXTLEN))
                           (FGETTOBJ TEXTOBJ TEXTLEN)))

        ;; If PREVLINE is LDUMMY (= PLINES of PANE), we pretend it has an LCHARLIM one before the LCHAR1 of its nextline, otherwise 0.

        (for L NEXT (LCHARLIM _ (CL:IF (AND (FGETLD PREVLINE LDUMMY)
                                            (FGETLD PREVLINE NEXTLINE))
                                    (SUB1 (FGETLD (FGETLD PREVLINE NEXTLINE)
                                                  LCHAR1))
                                    (FGETLD PREVLINE LCHARLIM)))
             (YBOT _ (FGETLD PREVLINE YBOT))
             (PBOTTOM _ (fetch (REGION BOTTOM) of (DSPCLIPPINGREGION NIL PANE)))
             inlines PREVLINE first (CL:WHEN (OR (IGREATERP LCHARLIM LASTCHAR)
                                                 (ILEQ YBOT PBOTTOM))
                                      (FSETLD PREVLINE NEXTLINE NIL)
                                                                    ; Eliminate dangling garbage
                                      (RETURN PREVLINE))
           while (SETQ NEXT (\TEDIT.FORMATLINE TEXTOBJ (ADD1 LCHARLIM)))
           do ;; L is formatted, positioned, linked, displayed. Next is the following line unless at the end.

              (LINKLD L NEXT)                                       ; Put NEXT into the iteration
              (SETQ YBOT (\TEDIT.LINE.BOTTOM NEXT))                 ; Link needed for Y position
              (SETQ LCHARLIM (FGETLD NEXT LCHARLIM))
              (CL:WHEN (OR (IGREATERP LCHARLIM LASTCHAR)
                           (ILEQ YBOT PBOTTOM))
                  (FSETLD L NEXTLINE NIL)                           ; Overshot, flush link
                  (RETURN L))
              (CL:UNLESS DONTDISPLAY                                  ; Cached formatting is good for display
                  (\TEDIT.DISPLAYLINE TEXTOBJ NEXT PANE))
           finally
                ;; Ran out of lines

                (RETURN (OR L PREVLINE))))])
```

(\FORMAT.GAP.LINES

```
  [LAMBDA (VALIDS PANE TEXTOBJ DONTDISPLAY)                           ; Edited 15-Mar-2024 19:23 by rmk
                                                                      ; Edited  4-Dec-2023 20:42 by rmk
                                                                      ; Edited 20-Nov-2023 10:47 by rmk
                                                                      ; Edited  3-Nov-2023 12:05 by rmk
                                                                      ; Edited 15-May-2023 17:31 by rmk
                                                                      ; Edited 28-Apr-2023 17:35 by rmk
                                                                      ; Edited 26-Apr-2023 18:39 by rmk

    ;; VALIDS is a pair (LASTVALID . NEXTVALID) as described in \TEDIT.VALID.LINES. Our job is to format and display the lines between
    ;; LASTVALID and NEXTVALID, laying them out in the region starting below the given LASTVALID.

    ;; The screen has valid images for  lines from the top down to LASTVALID (segment 1 as described in \TEDIT.VALID.LINES). We don't touch those
    ;; lines or their bitmaps.

    ;; We also don't smash the bitmaps for NEXTVALID lines whose initial YTOP is below the YBOT of the last formatted valid line.  This is guaranteed
    ;; for deletions, maybe not for insertions where the gap displaying can encroach on the valid bitmaps.  The caller has to sort the bitmap overlaps.

    ;; Returns the new LASTVALID whose NEXTLINE is the given NEXTVALID.

    (CL:UNLESS PANE (SETQ DONTDISPLAY T))
    (for L LASTINVALIDCHNO PBOTTOM LCHARLIM YBOT (LASTVALID _ (CAR VALIDS))
         (NEXTVALID _ (CDR VALIDS))
         [PBOTTOM _ (CL:UNLESS DONTDISPLAY
                        (fetch BOTTOM of (DSPCLIPPINGREGION NIL PANE)))]
         inlines LASTVALID first (SETQ YBOT (GETLD LASTVALID YBOT))
                                 (SETQ LCHARLIM (FGETLD LASTVALID LCHARLIM))
                                                                      ; LCHARLIM=0 if change in document's first line
                                 (SETQ LASTINVALIDCHNO (CL:IF NEXTVALID
                                                           (SUB1 (FGETLD NEXTVALID LCHAR1))
                                                           (FGETTOBJ TEXTOBJ TEXTLEN)))
       eachtime (SETQ LCHARLIM (FGETLD L LCHARLIM)) until (OR (AND PBOTTOM (ILEQ YBOT PBOTTOM))
                                                              (IEQP LCHARLIM LASTINVALIDCHNO))
       do (if (AND PBOTTOM (ILEQ YBOT PBOTTOM))
              then (LINKLD LASTVALID NIL)                             ; Insertion ran off the bottom, flush now-invisible lines
                   (RETURN LASTVALID)
            elseif (IEQP LCHARLIM LASTINVALIDCHNO)
              then
                   ;; We reached the end of the gap. But we may have smashed the bitmaps of the initial NEXTVALID lines, so we have to
                   ;; format/display a little bit more until we clear the overlap.  If we are displaying, we first move the non-overlapping bitmap
                   ;; downwards on the screen, out of danger, then reformat and provide fresh images for the overlapping lines.

                   (CL:WHEN (IGREATERP (FGETLD NEXTVALID YTOP)
                                       (FGETLD LASTVALID YBOT))
                            (BITBLT)))
          (SETQ LASTVALID (\TEDIT.FORMATLINE TEXTOBJ (ADD1 LCHARLIM)))
          (LINKLD L LASTVALID)
          (SETQ YBOT (\TEDIT.LINE.BOTTOM LASTVALID))
          (SETQ LCHARLIM (FGETLD LASTVALID LCHARLIM))
          (CL:UNLESS DONTDISPLAY

              ;; The THISLINE cache for  NEXT is good if we display immediately after formatting

              (\TEDIT.DISPLAYLINE TEXTOBJ LASTVALID PANE))
       finally (LINKLD LASTVALID NEXTVALID)
               (RETURN LASTVALID])
```

(\TEDIT.LOWER.LINES

```
  [LAMBDA (NEXTVALID LASTVALID PANE TEXTOBJ)                          ; Edited 15-Mar-2024 14:40 by rmk
                                                                      ; Edited 20-Jan-2024 23:15 by rmk
                                                                      ; Edited  2-Jan-2024 00:26 by rmk
                                                                      ; Edited  4-Dec-2023 11:25 by rmk
                                                                      ; Edited 24-Nov-2023 13:01 by rmk
                                                                      ; Edited 11-May-2023 11:34 by rmk
                                                                      ; Edited 28-Apr-2023 08:51 by rmk

    ;; NEXTVALID is the top line of a region in PANE that extends to the pane-bottom or text end--that is, the pane bitmap in that region correctly
    ;; reflects the lines (and possibly empty space at text-end).

    ;; Insertion case.  The inserted gap lines may cover some of the bitmap of the nextvalid lines.  In that case NEXTVALID:YBOT is greater than the
    ;; new LASTVALID:YBOT

    ;; If PANE has been moved so that it is not entirely within the screen, then don't try to find the relevant bits, just repaint the whole window.

    (CL:UNLESS (\TEDIT.OFFSCREEN.SCROLL TEXTOBJ PANE 'VERTICAL)

        ;; Completely on screen, we can take advantage of screen bitmap.

        [PROG ((NEWTOP (GETLD LASTVALID YBOT))
               (PREG (DSPCLIPPINGREGION NIL PANE))
               (LTOP (GETLD NEXTVALID YTOP))
               (LVBOT 0)
               LOWER PWIDTH LASTVISIBLE)
              (SETQ LOWER (IDIFFERENCE LTOP NEWTOP))                  ; How far down to go
              (CL:UNLESS (IGREATERP LOWER 0)                          ; Maybe it's not moving
                  (RETURN))
```

```
;; Make the YPOS of the lowered lines consistent with the intended positions of their images.  LASTVISIBLE is the last line that was
;; previously visible (and whose image will be lowered).

         (for L (PBOTTOM _ (fetch BOTTOM of PREG))
             inlines NEXTVALID while (IGEQ (IDIFFERENCE (FGETLD L YBOT)
                                                        LOWER)
                                           PBOTTOM)
            do (SETQ LASTVISIBLE L))

;;

         (SETQ PWIDTH (fetch WIDTH of PREG))                          ; Width of the pane

;; Lower what we think is the image of NEXTVALID and all visible lines below it. This may lower some garbage, if  the LASTVALID printer
;; encroached on NEXTVALID's image.

         (CL:WHEN LASTVISIBLE
             (SETQ LVBOT (IDIFFERENCE (GETLD LASTVISIBLE YBOT)
                                LOWER)))
         (BITBLT PANE 0 (IPLUS LVBOT LOWER)
                PANE 0 LVBOT PWIDTH (IDIFFERENCE (IDIFFERENCE LTOP LVBOT)
                                           LOWER)
                'INPUT
                'REPLACE)

;;

;; The bottom of the pane is good.  But if LASTVALID encroached into the bitmap of some of NEXTVALID and some of its descendants,
;; those need to be redisplayed.   And the ypositions of NEXTVALID and all lines down to LASTVISIBLE have to be lowered.

;;

         (CL:WHEN LASTVISIBLE                                         ; Smash the invisible tail
             (SETLD LASTVISIBLE NEXTLINE NIL))
         (for L YBOT inlines NEXTVALID do (SETQ YBOT (IDIFFERENCE (FGETLD L YBOT)
                                                           LOWER))
                                          (if (IGEQ (FGETLD L YTOP)
                                                    NEWTOP)
                                              then (SETYPOS L YBOT)
                                                   (\TEDIT.DISPLAYLINE TEXTOBJ L PANE)
                                            else (SETYPOS L YBOT)))

;; Clear whatever might be left over below the last visible line

         (CL:WHEN LASTVISIBLE
             (BLTSHADE WHITESHADE PANE 0 0 PWIDTH (GETLD LASTVISIBLE YBOT)
                    'REPLACE))])])
```

(\TEDIT.RAISE.LINES

```
  [LAMBDA (LINE NEWTOP PANE TEXTOBJ)                          ; Edited 20-Mar-2024 10:57 by rmk
                                                              ; Edited 20-Jan-2024 23:14 by rmk
                                                              ; Edited  2-Jan-2024 00:31 by rmk
                                                              ; Edited 14-Dec-2023 17:20 by rmk
                                                              ; Edited  4-Dec-2023 20:57 by rmk
                                                              ; Edited 24-Nov-2023 13:01 by rmk
                                                              ; Edited 14-May-2023 21:55 by rmk
                                                              ; Edited 11-May-2023 11:34 by rmk
                                                              ; Edited 28-Apr-2023 08:51 by rmk

    ;; LINE is the top line of a region in PANE that extends to the pane-bottom or text end--that is, the pane bitmap in that region correctly reflects the
    ;; lines (and possibly empty space at text-end).

    ;; This raises the image of that region so that its new top is at NEWTOP. It then fills in and displays lines below the region's new location that may
    ;; be neeeded to fill in the pane.

    ;; If PANE has been moved so that it is not entirely within the screen, then don't try to find the relevant bits, just repaint the whole window.

    (TEXTOBJ! TEXTOBJ)
    (PROG ((PREG (DSPCLIPPINGREGION NIL PANE))
           (LTOP (GETLD LINE YTOP))
           RAISE PWIDTH PBOTTOM LASTVISIBLE)
          (SETQ RAISE (IDIFFERENCE NEWTOP (FGETLD LINE YTOP)))
          (CL:UNLESS (IGREATERP RAISE 0)                      ; Maybe it's not moving
              (RETURN))

     ;; Make the YPOS of the raised lines consistent with the new positions of their images.  LASTVISIBLE is the last line that was previously visible
     ;; (and whose image has been raised).

          (SETQ PBOTTOM (fetch BOTTOM of PREG))
          (for L inlines LINE while (IGEQ (FGETLD L YBOT)
                                          PBOTTOM)
             do (SETYPOS L (IPLUS RAISE (FGETLD L YBOT)))
                (SETQ LASTVISIBLE L))

     ;;

          (CL:UNLESS (\TEDIT.OFFSCREEN.SCROLL TEXTOBJ PANE 'VERTICAL)

              ;; Completely on screen, we can work with  screen bitmap.  But first, are we at the end of the text?  Just clear.

              ;; Lines are positioned, but images may not exist. Raise the image of LINE and all visible lines below it.

              (SETQ PWIDTH (fetch WIDTH of PREG))
              (BITBLT PANE 0 0 PANE 0 RAISE PWIDTH LTOP 'INPUT 'REPLACE)

              ;;

              ;; Now for the bottom of the pane.  First clear it.
              (BLTSHADE WHITESHADE PANE 0 0 PWIDTH (FGETLD LASTVISIBLE YBOT)
                     'REPLACE)
              ;;
              ;; If the last visible line in the pane (whose image is now elevated) is not the last line of the text, we build and display new lines to fill
              ;; out the pane.
              (\TEDIT.LINES.BELOW LASTVISIBLE NIL PANE TEXTOBJ)
              (RETURN))])
```

(\TEDIT.VALID.LINES

```
  [LAMBDA (TEXTOBJ FIRSTCHANGEDCHNO NCHARSCHANGED REASON)         ; Edited 20-Mar-2024 06:46 by rmk
                                                                  ; Edited 15-Mar-2024 19:44 by rmk
                                                                  ; Edited 22-Feb-2024 01:05 by rmk
                                                                  ; Edited  3-Nov-2023 12:07 by rmk
                                                                  ; Edited 14-Jun-2023 15:55 by rmk
                                                                  ; Edited 17-May-2023 09:32 by rmk
                                                                  ; Edited 15-May-2023 17:51 by rmk

    ;; Called when changes have been made to the document that affect the lines displayed in each pane. If a change is not visible in a given pane,
    ;; then NIL is returned for that pane.  Otherwise, this divides the lines in the pane into 3 segments:
    ;;     1. a prefix of lines from the top visible line (next of PLINES) to the LASTVALID line, the line just before the first changed line.
    ;;     2. an intermediate sequence of lines that are (or may be) no longer valid because of the change.
    ;;     3. a suffix of post-chamge lines, starting with NEXTVALID, that are known still to be valid.
    ;; A line is "valid" if its line breaking is unaffected by the change and the bits in the screen bitmap that represented it before the change are still
    ;; correct.
    ;;
    ;; The segmentation information is returned to the caller as a pair of lines (LASTVALID . NEXTVALID).   Segment 1 is then the sequence of lines
    ;; chained from PLINES to LASTVALID, segment 3 is the sequence beginning at NEXTVALID.  The segment 2 lines originally between LASTVALID
    ;; and NEXTVALID are useless, so here we just nuke them out (by smashing the NEXTLINE of LASTVALID).
    ;;
    ;; This assumes that the change has already been installed in the piece table after character FIRSTCHANGEDCHNO.  The LCHAR1/LIM valus for
    ;; lines through LASTVALID are unaffected by the change, the values for all later lines are off by NCHARSCHANGED (negative for deletions,
    ;; positive for insertions).  The positions for NEXTVALID and beyond are adjusted so that they are correct with respect to the revised piece table.
    ;; Note that this only deals with the character numbers of lines that will persist.  Although the Y positions for segment 1 lines are good,segment 3
    ;; positions cannot be adjusted until the replacements for segment 2 lines have been calculated.
    ;;
    ;; Edge conditions:
    ;; If the first visible line is changed, then there are no existing segment 1 lines and no existing  LASTVALID line to return.  If the first changed line is
    ;; also the first line of the document, then LASTVALID is NIL.  Otherwise, we fabricate a new a new line with LCHARLIM and YBOT just above the
    ;; changed top line and returned it as LASTVALID.  Either way, the next of PLINES is set to NIL to indicate that there is no chain of real segment 1
    ;; lines with valid formatting and reusable bitmaps.
    ;;
    ;; If the last visible line is changed, then there is no NEXTVALID line, indicated by NEXTVALID=NIL. The next valid could be a currently
    ;; non-existent line just below the pane if we are not at the end of the document.  If LCHARLIM of the last visible line is TEXTLEN, there is at best a
    ;; trailing line.
    ;;
    ;; Note that this is mostly an optimization to avoid unnecessary reformatting and redisplaying of still-valid lines in favor of bitbltting a block of their
    ;; currently visible images.  Smashing all lines to NIL and refilling each pane would also give the correct behavior, but slower.  Intermediate would
    ;; be smashing all lines below the last valid.
    ;;
    (for PANE PLINES FIRSTCHANGEDLINE LASTCHANGEDLINE LASTVALIDLINE NEXTVALIDLINE (LASTCHANGEDCHNO
                                                                                      _
                                                                                   (SUB1 (IPLUS FIRSTCHANGEDCHNO
                                                                                                NCHARSCHANGED)))
        inpanes TEXTOBJ eachtime (SETQ PLINES (\DTEST (fetch (TEXTWINDOW PLINES) of PANE)
                                                     'LINEDESCRIPTOR))
                                 (SETQ LASTVALIDLINE PLINES)
                                 (SETQ NEXTVALIDLINE NIL)
       collect [SETQ FIRSTCHANGEDLINE (find L inlines (FGETLD PLINES NEXTLINE)
                                         suchthat

                                                 ;; Either within a line or immediately after a line that did not end with an EOL

                                                 (OR (LINESELECTEDP L FIRSTCHANGEDCHNO LASTCHANGEDCHNO)
                                                     (AND (NOT (FGETLD L FORCED-END))
                                                          (IEQP FIRSTCHANGEDCHNO (ADD1 (FGETLD L LCHARLIM]
               [SETQ LASTCHANGEDLINE (find L inlines (OR FIRSTCHANGEDLINE (FGETLD PLINES NEXTLINE))
                                        suchthat (OR (WITHINLINEP LASTCHANGEDCHNO L)
                                                     (AND (NOT (FGETLD L FORCED-END))
                                                          (IEQP LASTCHANGEDCHNO (ADD1 (FGETLD L LCHARLIM]
               (CL:WHEN (OR FIRSTCHANGEDLINE LASTCHANGEDLINE)       ; The change is visible in this pane.

                   ;; Figure out the LASTVALIDLINE--somewhere before the FIRSTCHANGEDLINE.  Could be PLINES as initialized above

                   (CL:WHEN FIRSTCHANGEDLINE

                       ;; First changed line is visible.  Rejustification could propagate changes backwards until a forced-end, so that's the clear last
                       ;; valid. That may overshoot and cause to much action on redisplaying.  The only way we can tighten up is to then format
                       ;; lines forward from the break, stopping before the first line whose LCHARLIM would change.

                       (SETQ LASTVALIDLINE (find L backlines (FGETLD FIRSTCHANGEDLINE PREVLINE))
                                        suchthat (FGETLD L FORCED-END)))
                       (CL:WHEN (AND (EQ LASTVALIDLINE PLINES)
                                     (IGREATERP (FGETLD FIRSTCHANGEDLINE LCHAR1)
                                            1))

                           ;; We ran back to the top of the pane without finding a forced-end. If it's not the beginning of the document, we need to
                           ;; insert a new line with the proper LCHARLIM and YBOT just above the pane.

                           [SETQ LASTVALIDLINE (CADR (\TEDIT.LINES.ABOVE TEXTOBJ (SUB1 (FGETLD FIRSTCHANGEDLINE
                                                                                               LCHAR1))
                                                               (FGETLD FIRSTCHANGEDLINE YTOP]
                           (\TEDIT.INSERTLINE LASTVALIDLINE PLINES T)))
                   ;;

                   ;; The next valid must be somewhere after the last changed line, and after a stable line break. But we will run out of lines if there
                   ;; is no visible paragraph break: the change reached the bottom, or the paragraph break after the change is below the pane, or
                   ;; the document ended.  In that case the last line in the chain is not valid and presumably the gap filler will fill to the end of the
                   ;; window.

                   [SETQ NEXTVALIDLINE (for L inlines LASTCHANGEDLINE when (OR (FGETLD L FORCED-END)
                                                                             (GETLD L LSTLN))
                                          do (RETURN (FGETLD L NEXTLINE]

                   ;; Translate the character positions of the still-valid lines that are visible later than the change.

                   (for L DELTA inlines NEXTVALIDLINE first (SETQ DELTA (SELECTQ REASON
                                                                            (INSERTION NCHARSCHANGED)
                                                                            (DELETION (IMINUS NCHARSCHANGED))
                                                                            (APPEARANCE (RETURN))
                                                                            (SHOULDNT "BAD REASONS FOR VALID LINES")
                                                                         ))
                      do (add (FGETLD L LCHAR1)
                              DELTA)
                         (add (FGETLD L LCHARLIM)
                              DELTA))
                   ;;

                   (CL:WHEN LASTVALIDLINE
                       (SETLD LASTVALIDLINE NEXTLINE NIL)           ; Chop off the useless lines
                       (CONS LASTVALIDLINE NEXTVALIDLINE)))])
```

(\TEDIT.CLEARPANE.BELOW.LINE

```
  [LAMBDA (LINE PANE TEXTOBJ)                                  ; Edited 20-Nov-2023 14:02 by rmk
                                                               ; Edited 22-Sep-2023 20:33 by rmk
                                                               ; Edited 25-Apr-2023 23:06 by rmk
                                                               ; Edited 30-May-91 15:59 by jds

    ;; According to the manual, the user overflow function is called whenever a line falls out of the window (pane?), but it isn't told anything else. The
    ;; use-case mentioned is coordination with the REGION property wherein TEDIT is running in part of a window.  But how does the userfn know
    ;; where it is?

    (CL:UNLESS (AND (GETTEXTPROP TEXTOBJ 'OVERFLOWFN)
                    (APPLY* (GETTEXTPROP TEXTOBJ 'OVERFLOWFN)
                           PANE TEXTOBJ))

        ;; Clears the  pane below LINE to white.

        (LET ((PREG (DSPCLIPPINGREGION NIL PANE)))
             (BLTSHADE WHITESHADE PANE 0 (fetch BOTTOM of PREG)
                    (fetch WIDTH of PREG)
                    (IDIFFERENCE (GETLD LINE YBOT)
                           (fetch BOTTOM of PREG))
                    'REPLACE)))])
```

(\TEDIT.INSERTLINE

```
  [LAMBDA (NEWLINE OLDLINE AFTER)                              ; Edited 31-May-2023 00:18 by rmk
                                                               ; Edited 26-Feb-2023 22:36 by rmk
                                                               ; Edited 24-Feb-2023 23:12 by rmk
                                                               ; Edited 23-Feb-2023 22:41 by rmk
                                                               ; Edited 30-May-91 16:05 by jds

    ;; Inserts NEWLINE in the line-descriptor chain either AFTER OLDLINE or before it (AFTER=NIL)

    (LET (LINE)
         (if AFTER
             then (SETQ LINE (GETLD OLDLINE NEXTLINE))
                  (CL:WHEN LINE (SETLD LINE PREVLINE NEWLINE))
                  (SETLD NEWLINE NEXTLINE LINE)
                  (SETLD NEWLINE PREVLINE OLDLINE)
                  (SETLD OLDLINE NEXTLINE NEWLINE)
           else (SETQ LINE (GETLD OLDLINE PREVLINE))
                (CL:WHEN LINE (SETLD LINE NEXTLINE NEWLINE))
                (SETLD NEWLINE PREVLINE LINE)
                (SETLD NEWLINE NEXTLINE OLDLINE)
                (SETLD OLDLINE PREVLINE NEWLINE])
```

(\TEDIT.INSURE.TRAILING.LINE

```
  [LAMBDA (TEXTOBJ LASTLINE)                                   ; Edited 15-Mar-2024 19:31 by rmk
                                                               ; Edited 16-Dec-2023 00:12 by rmk
                                                               ; Edited 15-Jul-2023 13:53 by rmk
```

```
                                                                        ; Edited  8-May-2023 22:00 by rmk
                                                                        ; Edited  5-May-2023 10:54 by rmk
    ;; Fabricates a final line to insure that there is a place for the caret to blink after the last EOL of the text.  Something for \FIXSEL to move to.

    ;; \TEDIT.FORMATLINE may be overkill--maybe we really want to construct exactly what we want.  But \TEDIT.FORMATLINE does get the
    ;; LHEIGHT.

    (CL:WHEN (AND (GETLD LASTLINE FORCED-END)
                  (IEQP (FGETLD LASTLINE LCHARLIM)
                        (FGETTOBJ TEXTOBJ TEXTLEN)))
        (LET [(LINE (\TEDIT.FORMATLINE.EMPTY TEXTOBJ (ADD1 (FGETTOBJ TEXTOBJ TEXTLEN]
             (SETYPOS LINE (IDIFFERENCE (FGETLD LASTLINE YBOT)
                                  (FGETLD LINE LHEIGHT)))
             (LINKLD LASTLINE LINE)
             LINE))])
```

(\TEDIT.MARK.LINES.DIRTY

```
  [LAMBDA (TEXTOBJ FIRSTCHAR LASTCHAR)                                  ; Edited 11-Dec-2023 10:43 by rmk
                                                                        ; Edited  2-Dec-2023 23:07 by rmk
                                                                        ; Edited  3-Nov-2023 12:07 by rmk
                                                                        ; Edited 28-May-2023 14:05 by rmk
                                                                        ; Edited 20-May-2023 16:44 by rmk
                                                                        ; Edited  7-Apr-2023 19:25 by rmk
                                                                        ; Edited 30-May-91 16:05 by jds
    ;; Mark as dirty the lines that intersect the range FIRSTCHAR to LASTCHAR  inclusive, and assert that all panes need to be updated.

    [if (type? SELECTION FIRSTCHAR)
        then (SETQ LASTCHAR (SUB1 (GETSEL FIRSTCHAR CHLIM)))
             (SETQ FIRSTCHAR (GETSEL FIRSTCHAR CH#))
      elseif (type? SELPIECES FIRSTCHAR)
        then (SETQ LASTCHAR (ffetch (SELPIECES SPLASTCHAR) of FIRSTCHAR))
             (SETQ FIRSTCHAR (ffetch (SELPIECES SPFIRSTCHAR) of FIRSTCHAR))
      else (SETQ FIRSTCHAR (IMIN FIRSTCHAR (TEXTLEN TEXTOBJ)))
           (SETQ LASTCHAR (CL:IF (EQ LASTCHAR -1)
                                 (TEXTLEN TEXTOBJ TEXTOBJ)
                                 (IMIN LASTCHAR (TEXTLEN TEXTOBJ))))]
    (for PANE inpanes TEXTOBJ do (for LINES inlines (find L inlines (fetch (TEXTWINDOW PLINES) of PANE)
                                                        suchthat

                                                              ;; The first line ending after FIRSTCHAR

                                                              (IGEQ (FGETLD L LCHARLIM)
                                                                    FIRSTCHAR))
                                     do (FSETTOBJ TEXTOBJ TXTNEEDSUPDATE T)
                                        (for L inlines LINES while (ILEQ (FGETLD L LCHAR1)
                                                                         LASTCHAR)
                                           do ;; All the lines that begin before LASTCHAR

                                              (FSETLD L LDIRTY T))
                                        (RETURN])
```

(\TEDIT.LINE.BOTTOM

```
  [LAMBDA (LINE)                                                        ; Edited  4-Dec-2023 13:59 by rmk
                                                                        ; Edited 25-Apr-2023 23:00 by rmk
                                                                        ; Edited 23-Apr-2023 00:05 by rmk
                                                                        ; Edited 24-Sep-87 10:00 by jds
    ;; Computes LINE's YBOT value relative to the Y position of the line before.  Takes into account the (undocumented) BASETOBASE leading, as
    ;; well as paragraph leadings.

    ;; BASETOBASE leading differs from normal LINELEADING in that the distance between the baselines of adjacent within-paragraph lines should
    ;; be the given constant, whether or not the previous line has a non standard descent (a subscript) or the next line has a nonstandard ascent.

    ;; We can't fetch the YBASE of PREV directly, since we

    (\DTEST LINE 'LINEDESCRIPTOR)
    (LET* ((PREV (\DTEST (FGETLD LINE PREVLINE)
                         'LINEDESCRIPTOR))
           (PREVYBOT (FGETLD PREV YBOT))
           (FMTSPEC (GETLD LINE LFMTSPEC))
           (BASETOBASE (fetch (FMTSPEC FMTBASETOBASE) of FMTSPEC))
           NEWYBOT)
          [SETQ NEWYBOT (if (NOT BASETOBASE)
                            then ;; \TEDIT.FORMATLINE.VERTICAL already compensated for paragraph leading.

                                 (IDIFFERENCE PREVYBOT (FGETLD LINE LHEIGHT))
                          elseif (FGETLD LINE 1STLN)
                            then ;; This is the first line of a new paragraph, and the previous line must therefore have been a last.  Both
                                 ;; paragraph leadings apply in the gap, but the line leading is irrelevant.

                                 (IDIFFERENCE PREVYBOT (IPLUS (fetch (FMTSPEC LEADAFTER) of (FGETLD PREV LFMTSPEC)
                                                                     )
                                                              (fetch (FMTSPEC LEADBEFORE) of FMTSPEC)
                                                              (FGETLD LINE LTRUEHEIGHT)))
                          else ;; Between lines inside a paragraph, make the baselines BASETOBASE apart.  Oldcode subtracted paragraph
                               ;; leading

                               (IDIFFERENCE (IDIFFERENCE (FGETLD PREV YBASE)
                                                   BASETOBASE)
```

```
                                     (FGETLD LINE DESCENT]
            (SETYPOS LINE NEWYBOT)
            NEWYBOT])
```

(\TEDIT.NCONC.LINES

```
  [LAMBDA (HEADLINE TAILLINE HEADYTOP LASTBOTTOM)                    ; Edited  1-Dec-2023 11:45 by rmk

    ;; The lines headed by HEADLINE and TAILLINE are linked in a single chain, and their Y positions are adjusted so that the top of HEADLINE is at
    ;; HEADYTOP (if given) and it and all other lines are positioned relative to that, based on their LHEIGHTs.  If LASTBOTTOM is provided, then lines
    ;; below it will be chopped off.  Returns the last line in the chain.

    (CL:WHEN HEADLINE
        (CL:UNLESS HEADYTOP
            (SETQ HEADYTOP (FGETLD HEADLINE YTOP)))
        (CL:UNLESS LASTBOTTOM (SETQ LASTBOTTOM MIN.SMALLP))
        (for L (YBOT _ HEADYTOP)
            inlines HEADLINE do ;; YBOT is the bottom of the previous line, move it down

                                   (SETQ YBOT (IDIFFERENCE YBOT (FGETLD L LHEIGHT)))
                                   (SETYPOS L YBOT)
                                   (CL:UNLESS (IGREATERP YBOT LASTBOTTOM)
                                                                     ; 1 above the bottom or below, back up and chop off
                                       (FSETLD $$PREVLINE NEXTLINE NIL)
                                       (RETURN $$PREVLINE))
                                   (CL:UNLESS (FGETLD L NEXTLINE)    ; Concatenate, keep going
                                       (CL:UNLESS TAILLINE (RETURN L))
                                       (LINKLD L TAILLINE)
                                       (SETQ TAILLINE NIL))))])

)
```

FUNCTION INDEX

MACRO INDEX

RECORD INDEX

I.S.OPR INDEX

VARIABLE INDEX

CONSTANT INDEX