File created: 6-Mar-2025 11:42:48 **{WMEDLEY}<library>TEDIT>TEDIT-SCREEN.;867**

edit by: rmk

changes to: (FNS \TEDIT.FORMATLINE)

previous date: 25-Feb-2025 10:40:05 {WMEDLEY}<library>TEDIT>TEDIT-SCREEN.;866

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ TEDIT-SCREENCOMS
       ([DECLARE%: EVAL@COMPILE DONTCOPY
               (EXPORT (RECORDS TAB TABSPEC)
                       (RECORDS LINECACHE)
                       (COMS                                    ; LINEDESCRIPTORS
                             (RECORDS LINEDESCRIPTOR)
                             (I.S.OPRS inlines backlines)
                             (MACROS GETLD FGETLD SETLD FSETLD SETYBOT SETYTOP SETYBASE LINKLD LINEDESCRIPTOR!))
                       (MACROS HCSCALE HCUNSCALE SCALEUP SCALEDOWN)
                       (GLOBALVARS TEDIT.DONT.BREAK.CHARS TEDIT.DONT.LAST.CHARS)
                       (ALISTS (CHARACTERNAMES EM-DASH SOFT-HYPHEN NONBREAKING-HYPHEN NONBREAKING-SPACE))
                       (MACROS DIACRITICP)
                       (MACROS \TEDIT.LINE.TALLP)
                       (COMS                                    ; Formatting slots held by THISLINE
                             (RECORDS THISLINE 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]
        (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.PARA \TEDIT.FORMATLINE.HORIZONTAL
             \TEDIT.FORMATLINE.VERTICAL \TEDIT.FORMATLINE.JUSTIFY \TEDIT.FORMATLINE.TABS \TEDIT.SCALE.TABS
             \TEDIT.FORMATLINE.PURGE.SPACES \TEDIT.FORMATLINE.FLUSH.SOFTHYPHEN \TEDIT.FORMATLINE.EMPTY
             \TEDIT.FORMATLINE.UPDATELOOKS \TEDIT.FORMATLINE.LASTLEGAL \TEDIT.LINES.ABOVE)
        (INITVARS (TEDIT.LINELEADING.BELOW NIL))
        (GLOBALVARS TEDIT.LINELEADING.BELOW)
        (FNS \TLVALIDATE)
                                                                ; Consistency checking
        (INITVARS *TEDIT-CACHED-PARALOOKS*)
                                                                ; Heuristic for \FORMATLINE
        (GLOBALVARS *TEDIT-CACHED-PARALOOKS*)
        (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.BACKFORMAT \TEDIT.PREVIOUS.LINEBREAK \TEDIT.UPDATE.LINES \TEDIT.PANE.CREATELINES
             \TEDIT.SUFFIXLINE.CREATE \TEDIT.LINES.BELOW \TEDIT.MEASURED.LINES \TEDIT.VALID.LINES
             \TEDIT.LASTVALIDLINE \TEDIT.NEXTVALIDLINE \TEDIT.CLEARPANE.BELOW.LINE \TEDIT.INSERTLINE
             \TEDIT.LINE.BOTTOM \TEDIT.SHOW.AT.BOTTOMP \TEDIT.SHOW.AT.TOPP)))

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(RECORD TAB (TABX . TABKIND))

(RECORD TABSPEC (DEFAULTTAB . TABS))
)

(DECLARE%: EVAL@COMPILE

(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 '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. Includes
                                                                ; paragraph and line leading
                          LASCENT                               ; Ascent of the line above YBASE, adjusted for line and
                                                                ; paragraph leading
                          LDESCENT                              ; 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.
                          LCHARLAST                             ; 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.
                          LPARALOOKS                            ; The paragraph looks for this line's paragraph (eventually)
                          (NIL FLAG)                            ; Was LDIRTY: T if this line has changed since it was last
                                                                ; formatted.
                          (NIL FLAG)                            ; Was FORCED-END flag
                          (NIL FLAG)                            ; Was DELETED: T if this line has been completely deleted since
                                                                ; it was last formatted or displayed.  (Used by deletion routines to
                                                                ; detect garbage lines)
                          (NIL FLAG)                            ; Was LHASPROT 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 (STANDARD (IPLUS (GETLD DATUM YBASE)
                                           (GETLD DATUM LASCENT))
                                    FAST
                                    (IPLUS (FGETLD DATUM YBASE)
                                           (FGETLD DATUM LASCENT]
                    [LTRUEYTOP (STANDARD (IPLUS (GETLD DATUM YBASE)
                                                (FGETLD DATUM LTRUEASCENT))
                                         FAST
                                         (IPLUS (FGETLD DATUM YBASE)
                                                (FGETLD DATUM LTRUEASCENT]
                    [LTRUEHEIGHT (STANDARD (IPLUS (GETLD DATUM LTRUEASCENT)
                                                  (FGETLD DATUM LTRUEDESCENT))
                                           FAST
                                           (IPLUS (FGETLD DATUM LTRUEASCENT)
                                                  (FGETLD DATUM LTRUEDESCENT]
                    [LTRUEYBOT (STANDARD (IDIFFERENCE (GETLD DATUM YBASE)
                                                      (FGETLD DATUM LTRUEDESCENT))
                                         FAST
                                         (IDIFFERENCE (FGETLD DATUM YBASE)
                                                      (FGETLD DATUM LTRUEDESCENT]
                    [LLEADBEFORE (STANDARD (IDIFFERENCE (GETLD DATUM LASCENT)
                                                        (FGETLD DATUM LTRUEASCENT))
                                           FAST
                                           (IDIFFERENCE (FGETLD DATUM LASCENT)
```

```
                                                (FGETLD DATUM LTRUEASCENT]
                        [LCHARLIM (STANDARD (ADD1 (GETLD DATUM LCHARLAST))
                                            FAST
                                            (ADD1 (FGETLD DATUM LCHARLAST)))
                                  (STANDARD (SETLD DATUM LCHARLAST (SUB1 NEWVALUE))
                                            FAST
                                            (FSETLD DATUM LCHARLAST (SUB1 NEWVALUE]
                        (LNCH (STANDARD (IDIFFERENCE (GETLD DATUM LCHARLIM)
                                                     (GETLD DATUM LCHAR1))
                                        FAST
                                        (IDIFFERENCE (FGETLD DATUM LCHARLIM)
                                                     (FGETLD DATUM LCHAR1]
        LHEIGHT _ 0 LTRUEASCENT _ 0 LTRUEDESCENT _ 0 YBOT _ 0 YBASE _ 0 LEFTMARGIN _ 0)
)

(/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 SETYBOT MACRO [OPENLAMBDA (LINE BOTTOM)
                          (FSETLD LINE YBASE (IPLUS (GETLD LINE LDESCENT)
                                                    (FSETLD LINE YBOT BOTTOM])

(PUTPROPS SETYTOP MACRO [OPENLAMBDA (LINE TOP)
                         (SETYBOT LINE (IDIFFERENCE TOP (GETLD LINE LHEIGHT])

(PUTPROPS SETYBASE MACRO [OPENLAMBDA (LINE BASE)
                           (FSETLD LINE YBOT (IDIFFERENCE (GETLD LINE LDESCENT)
                                                          (FSETLD LINE YBASE BASE])

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

(PUTPROPS LINEDESCRIPTOR! MACRO ((LD)
                                 (\DTEST LD 'LINEDESCRIPTOR)))
)

(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)))])

(PUTPROPS SCALEUP MACRO [OPENLAMBDA (SCALE ITEM)                ; List = region?
                           (CL:IF (LISTP ITEM)
                               (for I in ITEM collect (FIXR (FTIMES SCALE ITEM)))
                               (FIXR (FTIMES SCALE ITEM)))])

(PUTPROPS SCALEDOWN MACRO [OPENLAMBDA (SCALE ITEM)              ; List = region?
                             (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"))

(DECLARE%: EVAL@COMPILE

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

(DECLARE%: EVAL@COMPILE

(PUTPROPS \TEDIT.LINE.TALLP MACRO ((LINE HEIGHT)
                                   (OR (IGREATERP (FGETLD LINE LHEIGHT)
                                                  50)
                                       (IGREATERP (FGETLD LINE LHEIGHT)
                                                  HEIGHT))))
)

;; Formatting slots held by THISLINE

(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))

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

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

(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)
)
)
```

;; END EXPORTED DEFINITIONS

```
(DEFINEQ
```

(\TEDIT.LINEDESCRIPTOR.DEFPRINT

```
  [LAMBDA (LINE STREAM)                                              ; Edited 19-Nov-2024 16:04 by rmk
                                                                     ; Edited 17-Nov-2024 16:00 by rmk
                                                                     ; Edited 10-Nov-2024 18:28 by rmk
                                                                     ; Edited  4-Nov-2024 19:54 by rmk
                                                                     ; Edited  4-Jul-2024 10:39 by rmk
                                                                     ; Edited 10-May-2024 00:27 by rmk
                                                                     ; 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 LCHARLAST)
                            (CL:IF (GETLD LINE LSTLN)
                                "*"
                                "")
                            (if (GETLD LINE FORCED-END)
                                then (CONCAT " FE" (CL:IF (GETLD LINE LDUMMY)
                                                          "D"
                                                          ""))
                              else (CL:IF (GETLD LINE LDUMMY)
                                          " D"
                                          "")]
         (SETQ LOC (LOC LINE))
         (CONS (CONCAT "{L" (CAR LOC)
                       "/"
                       (CDR LOC)
                       ": " INFO "}"])

)

(/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 is removed
                                           (add TX (IMINUS (CHARW PREVDHYPH)))
                                           (SETQ CHARSLOT (\TEDIT.FORMATLINE.FLUSH.SOFTHYPHEN THISLINE
                                                              PREVDHYPH CHARSLOT)))
                                       (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 (TSTREAM CH#1 LINE REGION IMAGESTREAM FORMATTINGSTATE) ; Edited  6-Mar-2025 11:42 by rmk
                                                                 ; Edited 25-Feb-2025 10:39 by rmk
                                                                 ; Edited 19-Feb-2025 13:36 by rmk
                                                                 ; Edited 10-Feb-2025 09:59 by rmk
                                                                 ; Edited  8-Feb-2025 23:36 by rmk
                                                                 ; Edited 24-Dec-2024 22:15 by rmk
                                                                 ; Edited 23-Dec-2024 19:47 by rmk
                                                                 ; Edited 13-Dec-2024 23:46 by rmk
                                                                 ; Edited 12-Dec-2024 15:20 by rmk
                                                                 ; Edited  9-Dec-2024 21:05 by rmk
                                                                 ; Edited 23-Nov-2024 00:03 by rmk
                                                                 ; Edited 17-Nov-2024 19:56 by rmk
                                                                 ; Edited 31-Oct-2024 15:32 by rmk
                                                                 ; Edited 26-Oct-2024 10:51 by rmk
                                                                 ; Edited  2-Sep-2024 16:06 by rmk
                                                                 ; Edited 27-Aug-2024 18:29 by rmk
                                                                 ; Edited  4-Aug-2024 18:07 by rmk
                                                                 ; Edited 21-May-2024 14:45 by rmk
                                                                 ; Edited 17-Mar-2024 00:27 by rmk
                                                                 ; Edited  5-Feb-2024 09:35 by rmk
                                                                 ; Edited  3-Dec-2023 16:48 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 TSTREAM))
    ;; Note that lines lie within paragraphs, and all pieces within a paragraph have the same PARALOOKS.
    ;; The SPECVARS are accessed and reset under the subfunction\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.
    ;;   LCHARLAST: 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 LCHARLAST
    ;;   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.
    ;;
    (CL:UNLESS IMAGESTREAM
        (SETQ IMAGESTREAM (WINDOWPROP (\TEDIT.PRIMARYPANE TSTREAM)
                                  'DSP)))                        ; For lower image objects?
    (CL:WHEN (type? TEXTOBJ TSTREAM)                             ; Still confused about textobj/stream. Not sure who uses
                                                                 ; TSTREAM freely
        (SETQ TSTREAM (FGETTOBJ TSTREAM STREAMHINT)))
    (PROG ((TEXTOBJ (TEXTOBJ! (GETTSTR TSTREAM TEXTOBJ)))
           (OFFSET 0)
           (TRUEASCENT -1)
           (TRUEDESCENT -1)
           (ASCENTB 0)
           (DESCENTB 0)
           (ASCENTC 0)
           (DESCENTC 0)
           (OVERHANG 0)
           (SPACELEFT 0)
           (TX 0)
           (BOXSTREAM IMAGESTREAM)
           THISLINE LINETYPE WIDTH WMARGIN SCALE PARALOOKS RIGHTMARGIN HASKERN PC CHARSLOT PREVSP 1STLN CHNOB
           FORCED-END CHNO LX1 TX TXB FONT CHARSLOTB TABPENDING PREVHYPH PREVDHYPH START-OF-PIECE UNBREAKABLE
           OLDPIECE OLDPCCHARSLEFT OLDCARETLOOKS)
          (DECLARE (SPECVARS TEXTOBJ LINETYPE CHARSLOT CHNO OFFSET ASCENTC DESCENTC FONT START-OF-PIECE HASKERN
                             UNBREAKABLE))
          (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)))
          (SETQ THISLINE (FGETTOBJ TEXTOBJ THISLINE))
    ;; 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.
;;
    (SETQ OLDCARETLOOKS (FGETTOBJ TEXTOBJ CARETLOOKS))      ; Restore at end--BIN changes things
    (SETQ OLDPIECE (ffetch (TEXTSTREAM PIECE) of TSTREAM))
    (SETQ OLDPCCHARSLEFT (ffetch (TEXTSTREAM PCCHARSLEFT) of TSTREAM))
;;
;; Make sure we have a visible starting piece.
    (SETQ PC (\TEDIT.CHTOPC CH#1 TEXTOBJ T))                ; Get the true staring piece and CH#
    (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)
;;
;; We have the true starting piece and CH#1
;;
    (SETQ LINETYPE (if (NOT (DISPLAYSTREAMP IMAGESTREAM))
                       then 'TRUEHARDCOPY
                     elseif (FGETPLOOKS (PPARALOOKS PC)
                                        FMTHARDCOPY)
                       then 'HARDCOPYDISPLAY
                     else 'TRUEDISPLAY))
    (SETQ IMAGESTREAM (\TEDIT.FORMATLINE.SETUP.PARA TEXTOBJ PC LINE IMAGESTREAM LINETYPE))
;; The unchanging paragraph look has now been established and scaled appropriately. It is returned in the LPARALOOKS, the IMAGESTREAM is
;; unmodified.
    (SETQ PARALOOKS (FGETLD LINE LPARALOOKS))
    (SETQ SCALE (FGETPLOOKS PARALOOKS FMTHARDCOPYSCALE))
    [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 RIGHTMARGIN (if (ZEROP (FGETPLOOKS PARALOOKS RIGHTMAR))
                          then ;; RIGHTMAR = 0 => follow the window/region's width
                               (CL:IF (EQ LINETYPE 'HARDCOPYDISPLAY)
                                   (ITIMES SCALE WIDTH)
                                   WIDTH)
                        else (FGETPLOOKS PARALOOKS RIGHTMAR)))
;; Account for first-line indentation from the true left margin (LEFTMAR)
;; 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 1STLN (OR (IEQP CH#1 1)
                    (AND (IEQP CH#1 START-OF-PIECE)
                         (OR (NOT (\PREV.VISIBLE.PIECE PC))
                             (PPARALAST (\PREV.VISIBLE.PIECE PC]
    (SETQ LX1 (CL:IF 1STLN
                 (FGETPLOOKS PARALOOKS 1STLEFTMAR)
                 (FGETPLOOKS PARALOOKS LEFTMAR)))
    (SETQ WIDTH (IDIFFERENCE RIGHTMARGIN LX1))
;;
;; 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 APPLYLOOKSUPDATEFN) of TSTREAM with T)
    (FSETTOBJ TEXTOBJ CARETLOOKS NIL)                       ; Initialize variables to PC looks.
    (SETQ CHARSLOT (FIRSTCHARSLOT THISLINE))
    (\TEDIT.INSTALL.PIECE TSTREAM PC (- CH#1 START-OF-PIECE))
;;
;; 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.
```

```
(bind CH DX BOX INSPACES FIRSTWHITESLOT PREVCH KERN (FIRSTWHITEX _ TX)
      (INWORD _ T)
      (LASTCHARSLOT _ (LASTCHARSLOT THISLINE))
      (JUSTIFIED _ (EQ 'JUSTIFIED (FGETPLOOKS PARALOOKS QUAD)))
      (TEXTLEN _ (TEXTLEN TEXTOBJ)) for old CHNO by 1 while (ILEQ CHNO TEXTLEN)
   while (SETQ CH (BIN TSTREAM))
   do ;; Get CH's  width DX and maintain line ascent and descent.
      (if (SMALLP CH)
          then                                          ; 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 (M?) is so that the terminator can be selected
                         [SETQ DX (IMAX (\FGETCHARWIDTH FONT (CHARCODE M))
                                        (\FGETCHARWIDTH FONT (CHARCODE EOL]
                         (FILLCHARSLOT CHARSLOT (CL:IF (EQ CH (CHARCODE FORM))
                                                       (CHARCODE FORM)
                                                       (CHARCODE EOL))
                               DX)
                         (SETQ FORCED-END (CL:IF (MEMB CH (CHARCODE (LF CR)))
                                                 (CHARCODE EOL)
                                                 CH))
                         (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)))
               (CL:WHEN HASKERN
                    ;; Unlikely for display--probably backs it up.  The idea is to back up the starting point of CH by backing off the end
                    ;; of the previous character.  We stick the kern inline so that various consumers (displayline, scanline...) can make
                    ;; the adjustments.
                    (SETQ KERN (\FGETLEFTKERN FONT PREVCH CH))
                    (PUSHCHAR CHARSLOT 'KERN KERN)
                    (add DX KERN))
               (SETQ DX (\FGETCHARWIDTH FONT CH))
               (SETQ PREVCH CH)
        else                                            ; 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 BOXSTREAM TX (CL:IF (EQ LINETYPE 'HARDCOPYDISPLAY)
                                                      (SCALEDOWN SCALE WIDTH)
                                                      WIDTH)
                               TSTREAM))
             (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)))
             (CL:WHEN (EQ LINETYPE 'HARDCOPYDISPLAY)    ; Upscale DX from its display width.
                 (SETQ DX (SCALEUP SCALE DX)))
             (SETQ PREVCH NIL))
      [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 PARALOOKS 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 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.
           ;; CHTOLINEX under FIXSEL also needs to deal with this.
           (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 THISLINE CH#1 LINETYPE IMAGESTREAM)
                   ;; 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 PARALOOKS 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 be flushed
                                          (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 APPLYLOOKSUPDATEFN) 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 PARALOOKS 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 LCHARLAST 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 (FGETPLOOKS PARALOOKS FMTPARATYPE)
                                                   'PAGEHEADING)
                                               (FGETPLOOKS PARALOOKS FMTNEWPAGEBEFORE)
                                               (FGETPLOOKS PARALOOKS FMTNEWPAGEAFTER)
                                               [AND (FGETPLOOKS PARALOOKS FMTSPECIALX)
                                                    (NOT (ZEROP (FGETPLOOKS PARALOOKS FMTSPECIALX]
                                               (AND (FGETPLOOKS PARALOOKS FMTSPECIALY)
                                                    (NOT (ZEROP (FGETPLOOKS PARALOOKS FMTSPECIALY]
                                    'GREY))
        (FSETLD LINE FORCED-END FORCED-END)
        (FSETLD LINE LEFTMARGIN (CL:IF 1STLN
                                       (FGETPLOOKS PARALOOKS 1STLEFTMAR)
                                       (FGETPLOOKS PARALOOKS LEFTMAR)))
        (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)
                                          (FGETCLOOKS (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS)
                                                      CLFONT))
                                     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 LPARALOOKS PARALOOKS)
        (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 TEXTOBJ)
        (\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)
                      (FGETPLOOKS PARALOOKS FMTSPECIALX)
                      (NOT (ZEROP (FGETPLOOKS PARALOOKS FMTSPECIALX]
            ;; Maybe SETQ instead of add ??
            (add WMARGIN (FGETPLOOKS PARALOOKS FMTSPECIALX)))
        (add (FGETLD LINE LEFTMARGIN)
             WMARGIN)
        (add (FGETLD LINE RIGHTMARGIN)
             WMARGIN)
        (add (FGETLD LINE LX1)
             WMARGIN)
        (add (FGETLD LINE LXLIM)
             WMARGIN)
    ;; Restore TSTREAM to its condition before any BIN's we might have done.
        (FSETTOBJ TEXTOBJ CARETLOOKS OLDCARETLOOKS)
        (CL:WHEN OLDPIECE
            (\TEDIT.INSTALL.PIECE TSTREAM OLDPIECE (IDIFFERENCE (PLEN OLDPIECE)
                                                                OLDPCCHARSLEFT)))
        (RETURN LINE])
```

(**TEDIT.FORMATLINE.SETUP.PARA**

```
  [LAMBDA (TEXTOBJ PC LINE IMAGESTREAM LINETYPE)                     ; Edited 19-Feb-2025 13:37 by rmk
                                                                     ; Edited  8-Feb-2025 23:36 by rmk
                                                                     ; Edited  7-Feb-2025 08:09 by rmk
                                                                     ; Edited 22-Nov-2024 11:14 by rmk
                                                                     ; Edited 21-Oct-2024 00:33 by rmk
                                                                     ; Edited  4-Aug-2024 15:08 by rmk
                                                                     ; Edited 28-Jul-2024 21:01 by rmk
                                                                     ; 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 possibly-modified PARALOOKS of PC is stored in LINE.

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

    ;; 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 PARALOOKS.

    ;; See comments in TEDIT-LOOKSCOMS about the style-cache variables.  Probably not completely or correctly coordinated with this code.
```

```
    (TEXTOBJ! TEXTOBJ)
    (LET ([PLOOKS (PARALOOKS! (PPARALOOKS (OR PC (\PREV.VISIBLE.PIECE (FGETTOBJ TEXTOBJ SUFFIXPIECE))
                                              (FGETTOBJ TEXTOBJ SUFFIXPIECE]
          SCALE)
         (SETQ PLOOKS (\TEDIT.APPLY.PARASTYLES PLOOKS PC TEXTOBJ))
         (SELECTQ LINETYPE
              (TRUEHARDCOPY (SETQ PLOOKS (\TEDIT.HCPYFMTSPEC PLOOKS IMAGESTREAM)))
              (TRUEDISPLAY (CL:UNLESS (FGETPLOOKS PLOOKS FMTHARDCOPYSCALE)
                                  (FSETPLOOKS PLOOKS FMTHARDCOPYSCALE 1)))
              (HARDCOPYDISPLAY ;; Coerce the image stream and PARALOOKS for HARDCOPYDISPLAY.
                   [SETQ IMAGESTREAM (OR (FGETTOBJ TEXTOBJ DISPLAYHCPYDS)
                                         (FSETTOBJ TEXTOBJ DISPLAYHCPYDS (OPENIMAGESTREAM '{NODIRCORE}
                                                                                          'POSTSCRIPT]
                   (SETQ SCALE (DSPSCALE NIL IMAGESTREAM))
                   [SETQ PLOOKS (create PARALOOKS using PLOOKS FMTHARDCOPYSCALE _ SCALE RIGHTMAR _
                                                         (SCALEUP SCALE (FGETPLOOKS PLOOKS RIGHTMAR))
                                                         1STLEFTMAR _ (SCALEUP SCALE (FGETPLOOKS PLOOKS 1STLEFTMAR))
                                                         LEFTMAR _ (SCALEUP SCALE (FGETPLOOKS PLOOKS LEFTMAR))
                                                         FMTTABS _ (\TEDIT.SCALE.TABS (FGETPLOOKS PLOOKS FMTTABS)
                                                                                      SCALE)
                                                         FMTDEFAULTTAB _ (SCALEUP SCALE (FGETPLOOKS PLOOKS
                                                                                                    FMTDEFAULTTAB])
              (\TEDIT.THELP "BAD LINE TYPE" LINETYPE))
         (CL:UNLESS (OR (EQ PLOOKS *TEDIT-CACHED-PARALOOKS*)
                        (NOT (FGETPLOOKS PLOOKS FMTCHARSTYLES)))
             ;; 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-PARALOOKS* PLOOKS))
         (SETLD LINE LPARALOOKS PLOOKS)
         IMAGESTREAM])
```

(**TEDIT.FORMATLINE.HORIZONTAL**

```
  [LAMBDA (LINE THISLINE PREVSP SPACELEFT OVERHANG LINETYPE)     ; Edited 19-Feb-2025 13:35 by rmk
                                                                 ; Edited  8-Feb-2025 23:37 by rmk
                                                                 ; 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* ((PARALOOKS (FGETLD LINE LPARALOOKS))
           (SCALE (FGETPLOOKS PARALOOKS FMTHARDCOPYSCALE)))
          ;; Distribute SPACELEFT according to QUAD.
          (freplace (THISLINE TLSPACEFACTOR) of THISLINE with 1)
          (CL:WHEN (EQ 'JUSTIFIED (GETPLOOKS PARALOOKS QUAD))
                 (\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 (FGETPLOOKS PARALOOKS QUAD)
               (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 19-Feb-2025 13:37 by rmk
                                                               ; Edited  8-Feb-2025 23:37 by rmk
                                                               ; Edited 29-Oct-2024 11:07 by rmk
                                                               ; Edited 26-Oct-2024 10:26 by rmk
                                                               ; 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

    ;; 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 ((PARALOOKS (FGETLD LINE LPARALOOKS))
          (ASCENT (FGETLD LINE LTRUEASCENT))
          (DESCENT (FGETLD LINE LTRUEDESCENT)))
         (CL:WHEN (FGETLD LINE 1STLN LINE)                     ; Set pre-paragraph leading
             (add ASCENT (FGETPLOOKS PARALOOKS LEADBEFORE)))
         (CL:WHEN (FGETLD LINE LSTLN)                          ; Set post-paragraph leading
             (add DESCENT (FGETPLOOKS PARALOOKS LEADAFTER)))
        ;; Documentation says that lineleading goes above the line, which automatically makes for reasonable selection marking.  It went below in
        ;; the original implementation, selections were very odd for large line leadings.  Documentation also says that the lineleading is added to the
        ;; paragraph leading, so we add it to the ascent even of the 1STLN.  I.e. it is not just between-the-lines spacing.

        (add ASCENT (FGETPLOOKS PARALOOKS LINELEAD))
        (FSETLD LINE LASCENT ASCENT)
        (FSETLD LINE LDESCENT DESCENT)
        (FSETLD LINE LHEIGHT (IPLUS ASCENT DESCENT])
```

(\TEDIT.FORMATLINE.JUSTIFY

```
  [LAMBDA (LINE THISLINE PREVSP SPACELEFT LINETYPE)            ; Edited  4-Aug-2024 11:43 by rmk
                                                               ; 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 (FGETLD LINE LXLIM)
                     SPACELEFT)
                (freplace (THISLINE TLSPACEFACTOR) of THISLINE with (FQUOTIENT (IPLUS NATURALWIDTHS SPACELEFT)
                                                                               NATURALWIDTHS))))])
```

(**TEDIT.FORMATLINE.TABS**

```
  [LAMBDA (TEXTOBJ PARALOOKS SCALE CHARSLOT LX1 TX PRIORTAB CLEANINGUP)
                                                                                    ; Edited 19-Feb-2025 13:37 by rmk
                                                                                    ; Edited  8-Feb-2025 20:18 by rmk
                                                                                    ; Edited 21-Oct-2024 00:33 by rmk
                                                                                    ; Edited 27-Aug-2024 18:29 by rmk
                                                                                    ; Edited 28-Jul-2024 20:49 by rmk
                                                                                    ; 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.  DEFTAB and TABS are 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)
                (TABS (FGETPLOOKS PARALOOKS FMTTABS))
                (DEFTAB (FGETPLOOKS PARALOOKS FMTDEFAULTTAB)))
          (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)))
                                                                (\TEDIT.THELP)))
                                                        (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 TABS 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 (if NEXTTAB
                       then                                         ; There is a real tab to go to; use its location.
                            (fetch TABX of NEXTTAB)
                     else (SETQ GRAIN (FOLDLO SCALE 2))

                          ;; No real tab; use the next multiple of the default spacing.

                          (ITIMES DEFTAB (ADD1 (IQUOTIENT (IPLUS GRAIN TX)
                                                          DEFTAB]
                                                                    ; 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.SCALE.TABS**

```
[LAMBDA (TABS SCALE)                                                ; Edited 29-Jul-2024 14:36 by rmk
                                                                    ; Edited 28-Jul-2024 10:11 by rmk
                                                                    ; 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:IF (EQ SCALE 1)
    TABS
    [for TAB in TABS collect (create TAB using TAB TABX _ (HCSCALE SCALE (fetch (TAB TABX) of TAB])])
```

(**TEDIT.FORMATLINE.PURGE.SPACES**

```
[LAMBDA (PREVSP UNTILSP)                                            ; Edited 21-Oct-2024 00:26 by rmk
                                                                    ; 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
                                                   (\TEDIT.THELP 'PURGE PREVSP))
                                               (replace (CHARSLOT CHAR) of OPREVSP with (CHARCODE SPACE))))
PREVSP])
```

(**TEDIT.FORMATLINE.FLUSH.SOFTHYPHEN**

```
[LAMBDA (THISLINE PREVDHYPH CHARSLOT)                               ; Edited  2-Sep-2024 16:09 by rmk
```

;; PREVDHYPH is the THISLINE character slot of a preceding soft hyphen that is now being discarded in favor of a later potential linebreak. This
;; function purges it from THISLINE by moving the contents of all of the slots from the one after PREDVDHYPH backwards by one slot. The value
;; is the new (one-back) last slot

;; THISLINE needed only to suppress unused reference in incharslots I.S.OPR.

```
(CL:WHEN PREVDHYPH
    (for CS NEXT incharslots PREVDHYPH do (SETQ NEXT (NEXTCHARSLOT CS))
                                          (FILLCHARSLOT CS (CHAR NEXT)
                                                           (CHARW NEXT))
```

```
            repeatuntil (EQ NEXT CHARSLOT) finally (RETURN CS)))])
```

(\TEDIT.FORMATLINE.EMPTY

```
  [LAMBDA (TEXTOBJ CH#1 LINE)                                          ; Edited 19-Feb-2025 13:37 by rmk
                                                                       ; Edited  8-Feb-2025 23:37 by rmk
                                                                       ; Edited  7-Feb-2025 08:09 by rmk
                                                                       ; Edited 22-Nov-2024 22:29 by rmk
                                                                       ; Edited 17-Nov-2024 16:00 by rmk
                                                                       ; Edited  4-Aug-2024 14:51 by rmk
                                                                       ; Edited 25-Jun-2024 14:51 by rmk
                                                                       ; Edited 10-May-2024 00:24 by rmk
                                                                       ; 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
                                                                       ;
    ;; 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. But it
    ;; really wants to the looks of the preceding piece.

    (LINEDESCRIPTOR! LINE)
    (LET (CHARSLOT FONT CLOOKS TRUEASCENT TRUEDESCENT LM PLOOKS (THISLINE (FGETTOBJ TEXTOBJ THISLINE)))
         (\TEDIT.FORMATLINE.SETUP.PARA TEXTOBJ NIL LINE (WINDOWPROP (\TEDIT.PRIMARYPANE TEXTOBJ)
                                                                    'DSP)
               'TRUEDISPLAY)
         (SETQ PLOOKS (FGETLD LINE LPARALOOKS))

         ;; Get the current caret looks, so that LHEIGHT and \DISPLAYLINE work.  Font preferences:  the font of the previous piece, else the default
         ;; (from the last piece).  Previous code preferred the current caret looks, but that might have nothing to do with the end-of-document.

         [SETQ CLOOKS (PCHARLOOKS (OR (\PREV.VISIBLE.PIECE (FGETTOBJ TEXTOBJ SUFFIXPIECE))
                                      (FGETTOBJ TEXTOBJ SUFFIXPIECE]
         (SETQ FONT (GETCLOOKS CLOOKS CLFONT))
         (SETQ TRUEASCENT (FONTPROP FONT 'ASCENT))
         (SETQ TRUEDESCENT (FONTPROP FONT 'DESCENT))
         (SETQ LM (IPLUS \TEDIT.LINEREGION.WIDTH (FGETTOBJ TEXTOBJ WLEFT)
                         (FGETPLOOKS PLOOKS 1STLEFTMAR)))
         (with LINEDESCRIPTOR LINE (SETQ LDUMMY T)
               (SETQ LCHAR1 CH#1)
               (SETQ LCHARLAST CH#1)
               (SETQ 1STLN T)
               (SETQ LSTLN T)
               (SETQ LMARK NIL)
               (SETQ LX1 LM)
               (SETQ LXLIM LM)
               (SETQ FORCED-END (CHARCODE EOL))
               (SETQ LHASPROT NIL)
               (SETQ LPARALOOKS PLOOKS)
               (SETQ LEFTMARGIN LM)
               (SETQ RIGHTMARGIN (CL:IF (ZEROP (FGETPLOOKS PLOOKS RIGHTMAR))
                                        (IDIFFERENCE (FGETTOBJ TEXTOBJ WRIGHT)
                                               \TEDIT.LINEREGION.WIDTH)
                                        (FGETPLOOKS PLOOKS RIGHTMAR)))
               (SETQ LTRUEASCENT TRUEASCENT)
               (SETQ LTRUEDESCENT TRUEDESCENT)
               (SETQ LHEIGHT (IPLUS TRUEASCENT TRUEDESCENT)))
         (SETQ CHARSLOT (FIRSTCHARSLOT THISLINE))
         (FILLCHARSLOT CHARSLOT NIL CLOOKS)
         (replace (THISLINE NEXTAVAILABLECHARSLOT) of THISLINE with (NEXTCHARSLOT CHARSLOT))
         (freplace (THISLINE DESC) of THISLINE with LINE)

         ;; 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 19-Dec-2024 11:50 by rmk
                                                                       ; Edited 13-Dec-2024 17:09 by rmk
                                                                       ; Edited  4-Aug-2024 15:09 by rmk
                                                                       ; Edited 28-Jul-2024 20:52 by rmk
                                                                       ; Edited  9-May-2024 10:28 by rmk
                                                                       ; 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 OFFSET ASCENTC DESCENTC FONT IMAGESTREAM HASKERN UNBREAKABLE))
    (CL:UNLESS PC                                                   ; Ran off the end ? Skips the ENDOFSTREAMOP
        (RETFROM (FUNCTION \TEDIT.TEXTBIN)
               NIL))
    (LET (PLOOKS INVISIBLERUN CLOFFSET)
         (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 (FGETTOBJ (ffetch (TEXTSTREAM TEXTOBJ) of TSTREAM)
                                                   CARETLOOKS))
                      (FSETTOBJ (ffetch (TEXTSTREAM TEXTOBJ) of TSTREAM)
                             CARETLOOKS PLOOKS)

                      ;;

                      (SETQ OFFSET (OR (FGETCLOOKS PLOOKS CLOFFSET)
                                       0))
                      (SETQ FONT (FGETCLOOKS PLOOKS CLFONT))        ; 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 (SCALEUP (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))
                                                                    ; ASCENT before switching fonts
                                 (SETQ DESCENTC (ffetch \SFDescent of FONT))
                                 (CL:WHEN (EQ LINETYPE 'HARDCOPYDISPLAY)
                                                                    ; Switch widths to hardcopy
                                     (SETQ FONT (FONTCOPY FONT 'DEVICE IMAGESTREAM)))]
                      (SETQ HASKERN (FFETCH (FONTDESCRIPTOR FONTHASLEFTKERNS) of FONT))
                                                                    ; T if FONT contains left-kern information
                      (SETQ UNBREAKABLE (FGETCLOOKS PLOOKS CLUNBREAKABLE))
                      (PUSHCHAR CHARSLOT NIL PLOOKS))
               else ;; Adjust the CHNO to pass over invisible pieces--they don't show up in the THISLINE vector or on the screen.  Then recurse to
                    ;; here for the next visible piece.
                    (add CHNO INVISIBLERUN)
                    (\TEDIT.INSTALL.PIECE TSTREAM PC 0))
        PC])
```

(**TEDIT.FORMATLINE.LASTLEGAL**

```
  [LAMBDA (THISLINE CH#1 LINETYPE IMAGESTREAM)                      ; Edited 25-Jun-2024 15:44 by rmk
                                                                    ; 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 TX CHNO CHARSLOT TRUEASCENT TRUEDESCENT 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 (TSTREAM CHN BOTTOMY)                                   ; Edited 24-Nov-2024 11:47 by rmk
                                                                  ; Edited 20-Nov-2024 12:37 by rmk
                                                                  ; Edited 17-Nov-2024 16:02 by rmk
                                                                  ; Edited  2-Nov-2024 23:21 by rmk
                                                                  ; Edited 28-Oct-2024 16:14 by rmk
                                                                  ; Edited 27-Oct-2024 00:05 by rmk
                                                                  ; Edited 13-Jun-2024 17:03 by rmk
                                                                  ; Edited 18-May-2024 10:10 by rmk
                                                                  ; Edited  3-May-2024 23:33 by rmk
                                                                  ; 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 BOTTOMY, if provided.  This
    ;; is 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.

    (bind LTOP LBOT LINE HEIGHT CHNO (TEXTOBJ _ (GETTSTR TSTREAM TEXTOBJ))
       first (CL:WHEN (IGREATERP CHN (TEXTLEN TEXTOBJ))
                  (SETQ CHN (TEXTLEN TEXTOBJ)))
             (SETQ CHNO (\TEDIT.PREVIOUS.LINEBREAK TSTREAM CHN))   ; The end-of-line character
             (SETQ LTOP (\TEDIT.FORMATLINE TSTREAM CHNO))          ; A line containiing only the EOL
             (SETQ LBOT LTOP)
             (SETQ CHNO (FGETLD LTOP LCHARLIM))
       until (IGREATERP CHNO CHN) do (SETQ LINE (\TEDIT.FORMATLINE TSTREAM CHNO))
                                     (LINKLD LBOT LINE)
                                     (SETQ LBOT LINE)
                                     (SETQ CHNO (FGETLD LINE LCHARLIM))
       finally ;; We now have the line chain, but they aren't positioned.  Set the YBOT of LBOT to BOTTOMY
               (CL:WHEN BOTTOMY
                   (for L (YB _ BOTTOMY)
                      backlines LBOT do (SETYBOT L YB)
                                        (add YB (FGETLD L LHEIGHT))))
               (RETURN (CONS LTOP LBOT])

)

(RPAQ? TEDIT.LINELEADING.BELOW NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.LINELEADING.BELOW)
)

(DEFINEQ
```

(\TLVALIDATE

```
  [LAMBDA (THISLINE)                                              ; Edited 21-Oct-2024 00:26 by rmk
                                                                  ; 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)
                                 (\TEDIT.THELP "INVALID THISLINE" DESC))
          (CL:WHEN (EQ TLSPACEFACTOR 'NOSPACEFACTOR)
                 (\TEDIT.THELP "INVALID THISLINE" TLSPACEFACTOR))
          (CL:WHEN (EQ TLFIRSTSPACE 'NOTLFIRSTSPACE)
                 (\TEDIT.THELP "INVALID THISLINE" TLFIRSTSPACE))
          (CL:UNLESS (CHARSLOTP NEXTAVAILABLECHARSLOT THISLINE)
              (\TEDIT.THELP "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
                                                     (\TEDIT.THELP "INVALID THISLINE" 'BADCHAR))
                                                (CL:UNLESS (SMALLP CHARW)
                                                     (\TEDIT.THELP "INVALID THISLINE" 'BADCHARW))
                                           elseif (OR (SMALLP CHARW)
                                                      (type? CHARLOOKS CHARW))
                                           else (\TEDIT.THELP "INVALID THISLINE" 'BADCHARW])

)

;; Consistency checking

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

;; Heuristic for \FORMATLINE

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *TEDIT-CACHED-PARALOOKS*)
)

(DEFINEQ
```

(\TEDIT.DISPLAYLINE

```
  [LAMBDA (TEXTOBJ LINE PANE)                                   ; Edited 19-Feb-2025 13:35 by rmk
                                                                ; Edited  8-Feb-2025 23:37 by rmk
                                                                ; Edited 13-Dec-2024 23:51 by rmk
                                                                ; Edited 11-Dec-2024 23:14 by rmk
                                                                ; Edited 31-Oct-2024 09:56 by rmk
                                                                ; Edited 26-Oct-2024 10:43 by rmk
                                                                ; Edited 25-Aug-2024 23:18 by rmk
                                                                ; Edited 23-Aug-2024 22:52 by rmk
                                                                ; Edited 19-Jul-2024 23:17 by rmk
                                                                ; Edited 28-Jun-2024 11:43 by rmk
                                                                ; Edited 13-Jun-2024 17:08 by rmk
                                                                ; Edited 10-May-2024 00:24 by rmk
                                                                ; 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. This constructs the line image in a scratch bitmap then copies it to PANE.
    ;; Presumably this is to avoid the flicker of incremental updates.

    ;; Validate the incoming arguments so ffetch can be used consistently for all their field extractions.
    (TEXTOBJ! TEXTOBJ)
    (\DTEST LINE 'LINEDESCRIPTOR)
    (LET ((WINDOWDS (WINDOWPROP PANE '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)))
;; Only display the line if it contains text (CHAR1 > 0), appears before the end of the text.  Original code also suppressed lines that
;; were partially off-screen, which meant that large bitmaps wouldn't show.
(CL:UNLESS (EQ LINE (fetch (THISLINE DESC) of THISLINE))
                                                        ; No image cache -- re-format and display
    (\TEDIT.FORMATLINE (FGETTOBJ TEXTOBJ STREAMHINT)
           (FGETLD LINE LCHAR1)
           LINE))
(MOVETO (FGETLD LINE LX1)
       (FGETLD LINE LDESCENT)
       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))
                            (FGETCLOOKS CLOOKS CLLEADER)
                            (EQ (FGETCLOOKS CLOOKS CLUSERINFO)
                                '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 LDESCENT)))
                   (SETQ CLOOKS CHARW)
                   (DSPFONT (FGETCLOOKS CLOOKS CLFONT)
                          DS)
                   (CL:UNLESS (EQ 0 (FGETCLOOKS CLOOKS CLOFFSET))
                                                        ; Account for super/subscripting
                       (RELMOVETO 0 (FGETCLOOKS CLOOKS CLOFFSET)
                              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 (FGETCLOOKS CLOOKS CLFONT)
                                  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)
                 elseif (EQ 'KERN CHAR)
                   then (RELMOVETO 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 LDESCENT)))])
      (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 (GETPLOOKS (FGETLD LINE LPARALOOKS)
                      FMTREVISED)                               ; This paragraph has been revised, so mark it.
          (\TEDIT.MARK.REVISION TEXTOBJ (FGETLD LINE LPARALOOKS)
                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))
      LINE])
```

(\TEDIT.DISPLAYLINE.TABS

```
[LAMBDA (CW DS TX TERMSA LINE CLOOKS DISPLAYDATA DDPILOTBBT CLIPRIGHT TEXTOBJ)
                                                                ; Edited 19-Feb-2025 13:36 by rmk
                                                                ; Edited  8-Feb-2025 23:37 by rmk
                                                                ; 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 PARALOOKS. TEXTOBJ only needed to get the hardcopy-display
;; stream.

```
(bind TTX DOTWIDTH (PARALOOKS _ (GETLD LINE LPARALOOKS))
   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 (GETPLOOKS PARALOOKS FMTHARDCOPY)
                               [HCUNSCALE (FGETPLOOKS PARALOOKS FMTHARDCOPYSCALE)
                                      (CHARWIDTH (CHARCODE %.)
                                             (FONTCOPY (GETCLOOKS CLOOKS CLFONT)
                                                    'DEVICE
                                                    (FGETTOBJ TEXTOBJ DISPLAYHCPYDS]
                               (CHARWIDTH (CHARCODE %.)
                                      (GETCLOOKS CLOOKS CLFONT)))]
      [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.BACKFORMAT

```
  [LAMBDA (TSTREAM DY CH1 HEIGHT TRUETOP)                       ; Edited 26-Oct-2024 23:10 by rmk
                                                                ; Edited  3-May-2024 23:33 by rmk
                                                                ; 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 BOTTOMNEWLINE (TEXTOBJ _ (GETTSTR TSTREAM TEXTOBJ))
          (CHNO _ CH1) until (IGREATERP HEIGHT DY) while (IGEQ CHNO 1) do (SETQ PAIR (\TEDIT.LINES.ABOVE TSTREAM
                                                                                          CHNO HEIGHT))
                                                                          (CL:UNLESS BOTTOMNEWLINE
                                                                              (SETQ BOTTOMNEWLINE (CDR PAIR)))
```

```
                                                                    ; The block may go beyond DY
                                                                           (LINKLD (CDR PAIR)
                                                                                   L1)
                                                                    ; This block's LN links to previous L1
                                                                           (SETQ L1 (CAR PAIR))
                                                                           (SETQ HEIGHT (CL:IF TRUETOP
                                                                                               (GETLD L1 LTRUEYTOP)
                                                                                               (GETLD L1 YTOP)))
                                                                           (SETQ CHNO (SUB1 (GETLD L1 LCHAR1)))
      finally                                                       ; Perhaps the break was beyond DY
             (RETURN (CONS (OR (find L inlines L1 suchthat (ILEQ (FGETLD L YBOT)
                                                                 DY))
                               (RETURN NIL))
                           BOTTOMNEWLINE])
```

(\TEDIT.PREVIOUS.LINEBREAK

```
[LAMBDA (TSTREAM CHNO)                                              ; Edited 18-May-2024 18:53 by rmk
                                                                    ; Edited  3-May-2024 23:33 by rmk
                                                                    ; 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 FOLLOWS 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
      else ;; Otherwise, move back thru the text until we find a for-sure line break.
           (LET ((TEXTOBJ (fetch (TEXTSTREAM TEXTOBJ) of TSTREAM))
                 NCHARS)
                (if (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 (CL:WHEN (IGREATERP CHNO (FGETTOBJ TEXTOBJ TEXTLEN))
                                (SETQ CHNO (FGETTOBJ TEXTOBJ TEXTLEN)))
                       (\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.UPDATE.LINES

```
[LAMBDA (TEXTOBJ REASON FIRSTCHANGEDCHNO NCHARSCHANGED)             ; Edited  1-Feb-2025 10:34 by rmk
                                                                    ; Edited 21-Jan-2025 13:25 by rmk
                                                                    ; Edited  7-Jan-2025 11:55 by rmk
                                                                    ; Edited  7-Dec-2024 21:52 by rmk
                                                                    ; Edited 29-Nov-2024 22:56 by rmk
                                                                    ; Edited 26-Nov-2024 03:35 by rmk
                                                                    ; Edited 22-Nov-2024 17:57 by rmk
                                                                    ; Edited 20-Nov-2024 14:52 by rmk
                                                                    ; Edited 17-Nov-2024 19:52 by rmk
                                                                    ; Edited 11-Nov-2024 23:51 by rmk
                                                                    ; Edited  1-Nov-2024 22:05 by rmk
                                                                    ; Edited 13-Sep-2024 22:27 by rmk
                                                                    ; Edited  3-Jul-2024 15:42 by rmk
                                                                    ; Edited  7-May-2024 10:41 by rmk
                                                                    ; Edited 20-Mar-2024 06:43 by rmk
                                                                    ; Edited  4-Dec-2023 20:37 by rmk
                                                                    ; Edited 22-Jun-2023 15:50 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)
        [if (type? SELECTION FIRSTCHANGEDCHNO)
            then (SETQ NCHARSCHANGED (FGETSEL FIRSTCHANGEDCHNO DCH))
                 (SETQ FIRSTCHANGEDCHNO (FGETSEL FIRSTCHANGEDCHNO CH#))
          elseif (type? SELPIECES FIRSTCHANGEDCHNO)
            then [SETQ NCHARSCHANGED (ADD1 (IDIFFERENCE (FGETSPC FIRSTCHANGEDCHNO SPLASTCHAR)
                                                        (FGETSPC FIRSTCHANGEDCHNO SPFIRSTCHAR]
                 (SETQ FIRSTCHANGEDCHNO (FGETSPC FIRSTCHANGEDCHNO SPFIRSTCHAR))
          else (CL:UNLESS FIRSTCHANGEDCHNO (SETQ FIRSTCHANGEDCHNO 1))
               (CL:UNLESS NCHARSCHANGED
                   (SETQ NCHARSCHANGED (FGETTOBJ TEXTOBJ TEXTLEN)))]
```

```
        (\TEDIT.SHOWSEL NIL NIL TEXTOBJ)
        (for PANE VALIDS LASTVALID NEXTVALID LASTGAPLINE UPPERBITMAPLINES BITMAPLINES inpanes TEXTOBJ
           when (SETQ VALIDS (\TEDIT.VALID.LINES PANE FIRSTCHANGEDCHNO NCHARSCHANGED REASON (FGETTOBJ TEXTOBJ
                                                                                              STREAMHINT)))
           do ;; Create/format/position/display new lines between LASTVALID and NEXTVALID exclusive
              (SETQ LASTVALID (CAR VALIDS))
              (SETQ NEXTVALID (CDR VALIDS))                    ; MEASURED.LINES creates, measures, and links the lines from
                                                               ; LASTVALID to the last pre-NEXTVALID character, without
                                                               ; displaying.  They may be in the bitmap.
              [SETQ LASTGAPLINE (\TEDIT.MEASURED.LINES LASTVALID PANE TEXTOBJ (CL:IF NEXTVALID
                                                                                  (SUB1 (FGETLD NEXTVALID LCHAR1
                                                                                                ))
                                                                                  (TEXTLEN TEXTOBJ))]
              ;; The chain that ended at LASTVALID now continues thru LASTGAPLINE to NEXVALID and below.
              (LINKLD LASTGAPLINE NEXTVALID)
              (if NEXTVALID
                  then (SETQ BITMAPLINES (\TEDIT.BITMAPLINES PANE NEXTVALID))
                else (\TEDIT.SUFFIXLINE.CREATE PANE TEXTOBJ LASTGAPLINE))
              ;; If LASTVALID is not visible (above the pane), make sure that its NEXT is linked to the PANE's prefix
              (CL:WHEN (IGEQ (FGETLD LASTVALID YBOT)
                             (PANETOP PANE))
                  (LINKLD (PANEPREFIX PANE)
                          (FGETLD LASTVALID NEXTLINE)))
              (\TEDIT.SHIFTLINES LASTVALID PANE TEXTOBJ BITMAPLINES UPPERBITMAPLINES)))])
```

(\TEDIT.PANE.CREATELINES

```
  [LAMBDA (TEXTOBJ PANE LCHARLAST YBOT)                    ; Edited  8-Feb-2025 23:52 by rmk
                                                           ; Edited 29-Nov-2024 09:14 by rmk
                                                           ; Edited 20-Nov-2024 14:26 by rmk
                                                           ; Edited 17-Nov-2024 19:53 by rmk
                                                           ; Edited 10-Nov-2024 18:45 by rmk
                                                           ; Edited  4-Nov-2024 17:02 by rmk
                                                           ; Edited 26-Oct-2024 10:25 by rmk
                                                           ; Edited 29-Jun-2024 23:29 by rmk
                                                           ; Edited 28-Jun-2024 13:34 by rmk
                                                           ; Edited 21-Jun-2024 22:25 by rmk
                                                           ; Edited 19-Jun-2024 08:26 by rmk
                                                           ; Edited 17-Jun-2024 08:52 by rmk
                                                           ; 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.  This covers all of the characters before the first character visible in PANE, which is
    ;; LCHARLAST+1.  LCHARLAST defaults to 0. The bottom of the dummy line is the top of PANE.
    (LET (PREFIX)
         ;; Initialize with a dummy empty first line with LCHAR1 and LASTCHAR above the pane top.
         ;; 1STLN and LSTLN are NIL, since we don't want to make end paragraph-boundary inferences
         (SETQ PREFIX
          (create LINEDESCRIPTOR
                  LDUMMY _ T
                  LCHAR1 _ 0
                  LCHARLAST _ (OR LCHARLAST 0)
                  RIGHTMARGIN _ (SUB1 (FGETTOBJ TEXTOBJ WRIGHT))
                  LHEIGHT _ 0
                  LX1 _ 0
                  LXLIM _ (FGETTOBJ TEXTOBJ WRIGHT)
                  FORCED-END _ (CHARCODE EOL)
                  LASCENT _ 0
                  LDESCENT _ 0
                  LTRUEASCENT _ 0
                  LTRUEDESCENT _ 0
                  LPARALOOKS _ TEDIT.DEFAULT.FMTSPEC
                  1STLN _ NIL
                  LSTLN _ NIL))
         (SETYBOT PREFIX (OR YBOT (PANEHEIGHT PANE)))
         (FSETPANEPROP (PANEPROPS PANE)
                PREFIXLINE PREFIX)
         (\TEDIT.SUFFIXLINE.CREATE PANE TEXTOBJ PREFIX)
         PREFIX])
```

(\TEDIT.SUFFIXLINE.CREATE

```
  [LAMBDA (PANE TEXTOBJ PREVLINE)                          ; Edited 29-Nov-2024 10:54 by rmk
                                                           ; Edited 22-Nov-2024 10:22 by rmk
                                                           ; Edited 20-Nov-2024 14:25 by rmk
    ;; A new suffix line is created, if needed, and linked whenever the bottom is reached.  This gets the paragraph leading and height parameters from
    ;; the previous line. This may already be formatted as a dummy, if LCHARLIM is past the end.
    (LET ([SUFFIX (LINEDESCRIPTOR! (\TEDIT.FORMATLINE TEXTOBJ (GETLD PREVLINE LCHARLIM)
                                                      (PANESUFFIX PANE]
          EMPTYLINE)
```

```
        (FSETLD SUFFIX LDUMMY T)
        (SETYTOP SUFFIX (FGETLD PREVLINE YBOT))
        (FSETPANEPROP (PANEPROPS PANE)
              SUFFIXLINE SUFFIX)
        (LINKLD PREVLINE SUFFIX)
        (CL:WHEN (FGETLD PREVLINE FORCED-END)
            (SETQ EMPTYLINE (create LINEDESCRIPTOR using SUFFIX LDUMMY _ NIL LCHARLIM _ (FGETLD SUFFIX LCHAR1)
                                                         FORCED-END _ NIL))
            (LINKLD PREVLINE EMPTYLINE)
            (LINKLD EMPTYLINE SUFFIX))
        SUFFIX])
```

(\TEDIT.LINES.BELOW

```
  [LAMBDA (PREVLINE PANE TEXTOBJ)                          ; Edited 21-Jan-2025 13:31 by rmk
                                                           ; Edited 24-Nov-2024 14:57 by rmk
                                                           ; Edited 22-Nov-2024 00:53 by rmk
                                                           ; Edited 20-Nov-2024 12:37 by rmk
                                                           ; Edited 18-Nov-2024 21:12 by rmk
                                                           ; Edited 17-Nov-2024 16:03 by rmk
                                                           ; Edited 13-Nov-2024 12:20 by rmk
                                                           ; Edited 11-Nov-2024 23:01 by rmk
                                                           ; Edited  9-Nov-2024 11:22 by rmk
                                                           ; Edited  7-Nov-2024 22:21 by rmk
                                                           ; Edited  4-Nov-2024 16:49 by rmk
                                                           ; Edited  1-Nov-2024 22:11 by rmk
                                                           ; Edited 28-Oct-2024 21:28 by rmk
                                                           ; Edited 26-Oct-2024 15:49 by rmk
                                                           ; Edited 13-Sep-2024 22:24 by rmk
                                                           ; Edited 28-Jun-2024 15:21 by rmk
                                                           ; Edited 10-May-2024 00:20 by rmk
                                                           ; Edited  9-Apr-2024 10:13 by rmk
                                                           ; Edited 15-Mar-2024 19:22 by rmk
                                                           ; Edited 23-Dec-2023 23:38 by rmk
                                                           ; Edited 14-Dec-2023 12:46 by rmk
    (CL:UNLESS PREVLINE
        (SETQ PREVLINE (PANEPREFIX PANE)))
```

;; Formats and displays lines after PREVLINE down to the one is at least partially visible at the bottom of PANE. Each line is positioned with
;; respect to its predecessor and linked to it. The last visible line is set as the BOTTOMLINE of PANE, PANE's suffix is adjusted to cover the
;; bitmap and all the unseen later characters. Returns the last displayed line.

```
    (for L NEXT YBOT (BOTTOM _ (\TEDIT.ONSCREEN? PANE 'BOTTOM))
        inlines PREVLINE eachtime (SETQ NEXT (\TEDIT.FORMATLINE TEXTOBJ (FGETLD L LCHARLIM)))
       until (FGETLD NEXT LDUMMY) do (SETQ YBOT (\TEDIT.LINE.BOTTOM L NEXT))
                                     (SETYBOT NEXT YBOT)
                                     (CL:WHEN (ILESSP YBOT BOTTOM)
                                                           ; Ran off the bottom
                                         (RETURN (if (\TEDIT.SHOW.AT.BOTTOMP NEXT PANE)
                                                     then (\TEDIT.DISPLAYLINE TEXTOBJ NEXT PANE)
                                                          (LINKLD L NEXT)
                                                                ; Keep NEXT with partial display
                                                          NEXT
                                                 else           ; Overshot, throw NEXT away
                                                    L)))
                                     (LINKLD L NEXT)
                                     (CL:WHEN (FGETLD NEXT LDUMMY)
                                                           ; Suffix line: end of pane
                                         (RETURN L))
                                     (\TEDIT.DISPLAYLINE TEXTOBJ NEXT PANE)
                                                           ; Cached formatting is good for display
        finally ;; Ran off the end
                (RETURN L])
```

(\TEDIT.MEASURED.LINES

```
  [LAMBDA (PREVLINE PANE TEXTOBJ LASTCHAR)                 ; Edited 21-Jan-2025 13:30 by rmk
                                                           ; Edited  7-Dec-2024 16:55 by rmk
                                                           ; Edited  1-Dec-2024 11:26 by rmk
                                                           ; Edited 20-Nov-2024 12:37 by rmk
                                                           ; Edited 18-Nov-2024 20:01 by rmk
```

;; Formats and positions lines following PREVLINE up to and including the line that contains LASTCHAR or the last line that would be visible in
;; PANE. Returns the last formatted (and visible) line. Lines are not displayed.

```
    (for L NEXT NEXTCHAR1 YBOT (PBOTTOM _ (PANEBOTTOM PANE))
        inlines PREVLINE eachtime (SETQ NEXTCHAR1 (FGETLD L LCHARLIM)) while (ILEQ NEXTCHAR1 LASTCHAR)
       do (SETQ NEXT (\TEDIT.FORMATLINE TEXTOBJ NEXTCHAR1))      ; Always a next if the while succeeds
          (SETQ YBOT (\TEDIT.LINE.BOTTOM L NEXT))
          (SETYBOT NEXT YBOT)
          (CL:WHEN (ILESSP YBOT PBOTTOM)                        ; NEXT runs off the bottom
              (RETURN (if (\TEDIT.SHOW.AT.BOTTOMP NEXT PANE)
                          then (LINKLD L NEXT)                  ; Keep NEXT with partial display
                               NEXT
                        else                                    ; Overshot, throw NEXT away
                             L)))
          (LINKLD L NEXT)                                       ; Keeps the iteration going
       finally ;; Ran out of characters.
```

```
            (RETURN L])
```

(\TEDIT.VALID.LINES

```
  [LAMBDA (PANE FIRSTCHANGEDCHNO NCHARSCHANGED REASON TSTREAM)    ; Edited 21-Jan-2025 15:22 by rmk
                                                                 ; Edited  6-Jan-2025 15:19 by rmk
                                                                 ; Edited 22-Nov-2024 16:54 by rmk
                                                                 ; Edited 20-Nov-2024 12:37 by rmk
                                                                 ; Edited 21-Oct-2024 00:33 by rmk
                                                                 ; Edited  5-Jul-2024 22:58 by rmk
                                                                 ; Edited  4-Jul-2024 10:48 by rmk
                                                                 ; Edited 23-May-2024 12:48 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 PANE. Return NIL if the change is not visible in PANE.
    ;; Otherwise, this divides the lines in PANE into 3 segments:

    ;;    1. a prefix of lines from the top visible line (next of PANEPREFIX) 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-change 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 the prefix line 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 and PREVLINE of
    ;; NEXTVALID).

    ;;

    ;; This assumes that the change has already been installed in the piece table after character FIRSTCHANGEDCHNO.  The LCHAR1/LAST 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 line with LCHARLAST and YBOT just above the
    ;; changed top line and returned it as LASTVALID.  Either way, the next of PREFIXLINE 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 LCHARLAST 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 bitblting 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.
    (PROG ((TEXTOBJ (fetch (TEXTSTREAM TEXTOBJ) of TSTREAM))
           (LASTCHANGEDCHNO (SUB1 (IPLUS FIRSTCHANGEDCHNO NCHARSCHANGED)))
           (PREFIXLINE (PANEPREFIX PANE))
           (SUFFIXLINE (PANESUFFIX PANE))
           (DELTA (SELECTQ REASON
                       (INSERTION NCHARSCHANGED)
                       (DELETION (IMINUS NCHARSCHANGED))
                       ((CHANGED LOOKS)
                            NIL)
                       (\TEDIT.THELP "BAD REASONS FOR VALID LINES")))
           FIRSTVISIBLECHNO LASTVISIBLECHNO FIRSTCHANGEDLINE LASTCHANGEDLINE LASTVALIDLINE NEXTVALIDLINE)
          (CL:WHEN (EQ 0 (TEXTLEN TEXTOBJ))                      ; Empty document
              (RETURN (CONS PREFIXLINE)))
          (CL:UNLESS SUFFIXLINE
              (\TEDIT.THELP "NO SUFFIXLINE")
              (RETURN NIL))
          (SETQ LASTVISIBLECHNO (SUB1 (FGETLD SUFFIXLINE LCHAR1)))
          (CL:WHEN (IGREATERP FIRSTCHANGEDCHNO LASTVISIBLECHNO)  ; Change after previously visible lines
              (CL:UNLESS (ILEQ LASTCHANGEDCHNO (TEXTLEN TEXTOBJ))
                                                                 ; Change is after PANE, nothing to do
                  (RETURN NIL))
              ;; Adding at the end of the document:  insert a new line
              (\TEDIT.INSERTLINE (\TEDIT.FORMATLINE TEXTOBJ FIRSTCHANGEDCHNO)
                     SUFFIXLINE))
          (SETQ FIRSTVISIBLECHNO (FGETLD PREFIXLINE LCHARLIM))
          (SETQ FIRSTCHANGEDLINE (find L inlines (FGETLD PREFIXLINE NEXTLINE) suchthat (FWITHINLINEP
                                                                                            FIRSTCHANGEDCHNO L))
           )
```

```
        (CL:UNLESS FIRSTCHANGEDLINE                          ; Changes are not visible
            (RETURN NIL))
```

;; Change is visible in PANE, look for the last valid line (in PANE).

```
        (SETQ LASTVALIDLINE (\TEDIT.LASTVALIDLINE FIRSTCHANGEDLINE FIRSTVISIBLECHNO PANE TSTREAM))
```

;; Now for the after-change lines

```
        (SETQ LASTCHANGEDLINE (find L inlines FIRSTCHANGEDLINE suchthat (FWITHINLINEP LASTCHANGEDCHNO L)))
        (CL:WHEN LASTCHANGEDLINE
```

;; Last changed line is visible, its changes may cause character to shift to or from lower lines.

```
            (SETQ NEXTVALIDLINE (\TEDIT.NEXTVALIDLINE LASTCHANGEDLINE TSTREAM)))
        (CL:WHEN NEXTVALIDLINE
            (FSETLD NEXTVALIDLINE PREVLINE NIL)
            (CL:WHEN DELTA
```

;; If the modification added or substracted to the number of characters, translate the character positions of the still-valid lines
;; that are visible later than the change.

```
                (for L inlines NEXTVALIDLINE do (add (FGETLD L LCHAR1)
                                                     DELTA)
                                                (add (FGETLD L LCHARLAST)
                                                     DELTA))))
```

;;

```
        (CL:WHEN LASTVALIDLINE
            (FSETLD LASTVALIDLINE NEXTLINE NIL)              ; Chop out the now useless lines
            (RETURN (CONS LASTVALIDLINE NEXTVALIDLINE)))])
```

(\TEDIT.LASTVALIDLINE

```
  [LAMBDA (FIRSTCHANGEDLINE FIRSTCHANGEDCHNO PANE TSTREAM)       ; Edited 18-Feb-2025 12:45 by rmk
                                                                 ; Edited 29-Nov-2024 09:14 by rmk
                                                                 ; Edited 20-Nov-2024 12:37 by rmk
                                                                 ; Edited 18-Nov-2024 23:16 by rmk
                                                                 ; Edited 17-Nov-2024 19:08 by rmk
                                                                 ; Edited 16-Nov-2024 13:25 by rmk
                                                                 ; Edited 28-Oct-2024 16:05 by rmk
                                                                 ; Edited 28-Jun-2024 15:22 by rmk
                                                                 ; Edited 16-Jun-2024 08:27 by rmk
                                                                 ; Edited 13-Jun-2024 22:09 by rmk
                                                                 ; Edited 25-May-2024 00:28 by rmk
                                                                 ; Edited 23-May-2024 12:47 by rmk
                                                                 ; Edited 18-May-2024 10:13 by rmk
```

;; We hope to return an existing line in PANE that is impervious to the change at FIRSTCHARCHANGECHNO. This would be the impervious line
;; closest to FIRSTCHANGEDLINE, usually the immediately preceding line. That line is valid: it and lines above it do not need reformatting or
;; redisplay. But if PANE does not contain an impervious line, and we are not at the beginning of the document, we have to construct lines above
;; PANE until we get to an impervious line, so that we can format forwards.

;; A line L is impervious to a change in L+1 if it has a forced end, or if L has at least one separator (space, tab) prior to its change point. The
;; change point is FIRSTCHANGEDCHNO for the first line. If we have to go to earlier lines, then any separator anywhere on the line (at or before
;; LCHARLAST) will stop the back-propagation.

```
  (LET* ((PREFIXLINE (PANEPREFIX PANE))
         (FIRSTPANECHAR (AND (FGETLD PREFIXLINE NEXTLINE)
                             (FGETLD (FGETLD PREFIXLINE NEXTLINE)
                                     LCHAR1)))
         PREV)
        (if (bind (L _ FIRSTCHANGEDLINE)
                  (LIMCHAR _ (SUB1 FIRSTCHANGEDCHNO)) while (SETQ PREV (FGETLD L PREVLINE))
            do ;; The previous line is valid if its ending was forced, or if L has at least one space/tab earlier then the limit.  Note that
               ;; PREFIXLINE  is always forced-end, it stops the iteration..
               (CL:WHEN (FGETLD PREV FORCED-END)
                   (RETURN (if (NEQ PREFIXLINE PREV)
                               then PREV
                             elseif (EQ 1 FIRSTPANECHAR)
                               then                          ; PANE is at the top
                                    PREFIXLINE)))
               (\TEDIT.TEXTSETFILEPTR TSTREAM (SUB1 (FGETLD L LCHAR1)))
               (CL:WHEN [find I from 1 to (IDIFFERENCE LIMCHAR (FGETLD L LCHAR1))
                           suchthat (MEMB (BIN TSTREAM)
                                          (CHARCODE (SPACE TAB]
                      (RETURN PREV))
               (SETQ L PREV)
               (SETQ LIMCHAR (FGETLD L LCHARLIM))
            repeatwhile L)
          else ;; None of the existing lines above FIRSTCHANGEDLINE are valid.  We return a valid line that is positioned just above PANE such
               ;; that everything past its LCHARLAST is valid.  That line has no current bitmap and will not be displayed, but it signals where the
               ;; gap begins.

               ;; Note that that line is not linked into the chain, PANEPREFIX doesn't know about it.

               ;; We could go forward from the CAR or backwards from the CADR to find the valid line just above the pane. Maybe fewer lines
               ;; backwards, if we're working at the bottom of a paragraph?

               (find L (PTOP _ (PANEHEIGHT PANE))
                     backlines
                     (CDR (\TEDIT.LINES.ABOVE TSTREAM (SUB1 FIRSTPANECHAR)
                                              (FGETLD FIRSTCHANGEDLINE YTOP)))
```

```
                    suchthat (IGEQ (FGETLD L YBOT)
                                   PTOP])
```

(\TEDIT.NEXTVALIDLINE

```
[LAMBDA (LASTCHANGEDLINE TSTREAM)                          ; Edited 21-Jan-2025 15:27 by rmk
                                                           ; Edited 29-Nov-2024 23:31 by rmk
                                                           ; Edited 16-Nov-2024 11:00 by rmk
```

;; We know we can stop when we see a forced end-- characters won't move around after that. In the usual case, the forced-end is a also the last ;; line of a paragraph, but we can't just take the first line of the next paragraph because we can't deal here with whatever paragraph leading it might ;; have (and the venue sysout also screwed up in that case).

;; So we go for the second line of the next paragraph, if there is one

;; The line after a forced end is valid. But maybe we can figure out how to stop sooner?

```
(for L inlines LASTCHANGEDLINE when (FGETLD L FORCED-END) do
```

;; A forced end is usually the last line of a paragraph, and its next line is probably valid. But we skip ;; that one, because we don't know how to deal here with its paragraph leading. If forced but not last, ;; presumably it was a meta-EOL linebreak, no special leading to worry about.

```
                                                           (CL:WHEN (FGETLD L LSTLN)
                                                               (SETQ L (FGETLD L NEXTLINE)))
                                                           (RETURN (AND L (FGETLD L NEXTLINE])
```

(\TEDIT.CLEARPANE.BELOW.LINE

```
[LAMBDA (LINE PANE TEXTOBJ)                                ; Edited  1-Dec-2024 11:27 by rmk
                                                           ; Edited 20-Nov-2024 10:03 by rmk
                                                           ; Edited 11-Nov-2024 00:22 by rmk
                                                           ; Edited  4-Oct-2024 08:10 by rmk
                                                           ; Edited 26-Jun-2024 23:50 by rmk
                                                           ; Edited 13-Jun-2024 21:51 by rmk
                                                           ; 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:WHEN LINE
    (CL:UNLESS (AND (GETTEXTPROP TEXTOBJ 'OVERFLOWFN)
                    (APPLY* (GETTEXTPROP TEXTOBJ 'OVERFLOWFN)
                           PANE TEXTOBJ))
```

;; Clears the pane below LINE to white.

```
        (LET ((DISTBELOW 0))                               ; See note in SHOWSEL.HILIGHT
             (BLTSHADE WHITESHADE PANE 0 (PANEBOTTOM PANE)
                    (PANEWIDTH PANE)
                    (IDIFFERENCE (IDIFFERENCE (GETLD LINE YBOT)
                                              (PANEBOTTOM PANE))
                           DISTBELOW)
                    'REPLACE))))])
```

(\TEDIT.INSERTLINE

```
[LAMBDA (NEWLINE OLDLINE AFTER)                            ; Edited 17-May-2024 22:49 by rmk
                                                           ; 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))
     NEWLINE])
```

(\TEDIT.LINE.BOTTOM

```
[LAMBDA (PREVLINE LINE)                                    ; Edited 19-Feb-2025 13:36 by rmk
                                                           ; Edited  8-Feb-2025 23:38 by rmk
                                                           ; Edited 17-Nov-2024 00:38 by rmk
                                                           ; Edited  7-Nov-2024 16:57 by rmk
                                                           ; Edited 26-Oct-2024 15:45 by rmk
                                                           ; Edited 16-Jun-2024 23:43 by rmk
                                                           ; 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 PREVLINE. 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.

```
(LINEDESCRIPTOR! PREVLINE)
(LINEDESCRIPTOR! LINE)
(LET* ((PREVYBOT (FGETLD PREVLINE YBOT))
       (PARALOOKS (FGETLD LINE LPARALOOKS))
       (BASETOBASE (GETPLOOKS PARALOOKS FMTBASETOBASE))
       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 (GETPLOOKS (FGETLD PREVLINE LPARALOOKS)
                                                                     LEADAFTER)
                                                          (GETPLOOKS PARALOOKS LEADBEFORE)
                                                          (FGETLD LINE LTRUEHEIGHT)))
                      else ;; Between lines inside a paragraph, make the baselines BASETOBASE apart.  Oldcode subtracted paragraph
                           ;; leading
                           (IDIFFERENCE (IDIFFERENCE (FGETLD PREVLINE YBASE)
                                                     BASETOBASE)
                                        (FGETLD LINE LDESCENT]
       NEWYBOT])
```

(\TEDIT.SHOW.AT.BOTTOMP

```
[LAMBDA (LINE PANE)                                                 ; Edited  1-Dec-2024 11:27 by rmk
                                                                    ; Edited 29-Nov-2024 09:14 by rmk
                                                                    ; Edited 20-Nov-2024 10:17 by rmk
                                                                    ; Edited  2-Nov-2024 18:46 by rmk
                                                                    ; Edited 30-Oct-2024 18:03 by rmk
```

;; True if LINE is displayable at the bottom of PANE, with the heuristic goal of showing whole lines but not if the line itself is too big to be shown at ;; all.

```
(CL:WHEN (IGREATERP (FGETLD LINE YTOP)
                    (PANEBOTTOM PANE))                              ; For sure, the line's top has to be above PANE's bottom
    (OR (IGREATERP (FGETLD LINE YBOT)
                   (PANEBOTTOM PANE))
        (\TEDIT.LINE.TALLP LINE (PANEHEIGHT PANE)))])
```

(\TEDIT.SHOW.AT.TOPP

```
[LAMBDA (LINE TARGETY PHEIGHT)                                      ; Edited 21-Nov-2024 15:02 by rmk
                                                                    ; Edited  2-Nov-2024 18:44 by rmk
                                                                    ; Edited 30-Oct-2024 20:18 by rmk
```

;; True if LINE's true top is below TARGETY, or if the botom is below TARGETY and the line is very tall.

```
(CL:UNLESS (FGETLD LINE LDUMMY)
    (OR (ILEQ (FGETLD LINE LTRUEYTOP)
              TARGETY)
        (AND (ILEQ (FGETLD LINE YBOT)
                   TARGETY)
             (\TEDIT.LINE.TALLP LINE PHEIGHT)))])

)
```

FUNCTION INDEX

MACRO INDEX

RECORD INDEX

I.S.OPR INDEX

VARIABLE INDEX

CONSTANT INDEX