File created: 27-Mar-2024 13:53:47 **{WMEDLEY}<library>TEDIT>TEDIT-LOOKS.;242**

edit by: rmk

changes to: (FNS TEDIT.GET.PARALOOKS)

previous date: 20-Mar-2024 11:06:29 {WMEDLEY}<library>TEDIT>TEDIT-LOOKS.;241

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ TEDIT-LOOKSCOMS
       (;; Support for Character looks (font, italic/bold, sub/superscripting, etc) and paragraph looks (margins, centered/justified, tabs, etc.). Uses
        ;; compiled create functions in case DWIM is not available at loadup time.
        (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (RECORDS CHARLOOKS FMTSPEC)
                                                 (MACROS \WORDSETA)
                                                 (MACROS ONOFF)))
        (INITRECORDS CHARLOOKS FMTSPEC PENDINGTAB)
        (FNS \TEDIT.CHARLOOKS.DEFPRINT \TEDIT.FMTSPEC.DEFPRINT)
        [DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (TEDIT.TERMSA.FONTS NIL)
                                              (TEDIT.DEFAULT.FMTSPEC (\CREATE.TEDIT.DEFAULT.FMTSPEC))
                                              (TEDIT.TERMSA.FONTS NIL)
                                              (TEDIT.KNOWN.FONTS '((Classic 'CLASSIC)
                                                                   (Modern 'MODERN)
                                                                   (Terminal 'TERMINAL)
                                                                   (Titan 'TITAN)
                                                                   (Gacha 'GACHA)
                                                                   (Helvetica 'HELVETICA)
                                                                   (Times% Roman 'TIMESROMAN]
        (INITVARS (TEDIT.DEFAULT.FOLIO))
        (VARS (TEDIT.CHARLOOKS.FEATURES '(SUPERSCRIPT INVISIBLE SELECTPOINT PROTECTED SIZE FAMILY OVERLINE
                                                      STRIKEOUT UNDERLINE EXPANSION SLOPE WEIGHT))
              (TEDIT.FACE.MENU (\CREATE.TEDIT.FACE.MENU))
              (TEDIT.SIZE.MENU (\CREATE.TEDIT.SIZE.MENU)))
        (GLOBALVARS TEDIT.CURRENT.FONT TEDIT.CURRENT.CHARLOOKS TEDIT.CURRENT.PARALOOKS TEDIT.KNOWN.FONTS
               TEDIT.FACE.MENU TEDIT.SIZE.MENU TEDIT.DEFAULT.FONT TEDIT.DEFAULT.FMTSPEC TEDIT.TERMSA.FONTS)
        (ADDVARS (FONTVARS (TEDIT.PROMPT.FONT DEFAULTFONT)
                           (TEDIT.ICON.FONT MENUFONT)))
        (COMS                                                   ; Character looks functions
              (FNS CHARLOOKS.FROM.FONT \TEDIT.EQCLOOKS \TEDIT.SAMECLOOKS TEDIT.CARETLOOKS TEDIT.COPY.LOOKS
                   \TEDIT.UNPARSE.CHARLOOKS.LIST TEDIT.MODIFYLOOKS TEDIT.NEW.FONT \TEDIT.CARETLOOKS.VERIFY
                   \TEDIT.CARETPIECE \TEDIT.GET.INSERT.CHARLOOKS \TEDIT.GET.TERMSA.WIDTHS
                   \TEDIT.PARSE.CHARLOOKS.LIST)
              (COMS (FNS \TEDIT.TRANSLATE.ASCIICHARS \TEDIT.CONVERT.TO.FORMATTED)
                    (MACROS \TEDIT.TRANSLATE.ASCII.CHARLOOKS))
              (FNS \TEDIT.UNIQUIFY.CHARLOOKS \TEDIT.UNIQUIFY.PARALOOKS \TEDIT.UNIQUIFY.ALL
                   \TEDIT.FLUSH.UNUSED.LOOKS)
              ;; Public entries
              (FNS TEDIT.LOOKS TEDIT.GET.LOOKS TEDIT.SUBLOOKS TEDIT.FINDLOOKS)
              (FNS \TEDIT.CHANGE.LOOKS \TEDIT.LOOKS \TEDIT.FONTCOPY))
        (COMS                                                   ; Paragraph looks functions
              (FNS \TEDIT.EQFMTSPEC TEDIT.GET.PARALOOKS \TEDIT.PARSE.PARALOOKS.LIST TEDIT.PARALOOKS
                   TEDIT.COPY.PARALOOKS \TEDIT.PARABOUNDS)
              ;; For making paragraph-looks substitutions.
              (FNS TEDIT.SUBPARALOOKS SAMEPARALOOKS))
        (COMS                                                   ; UNDO & History List stuff
              (FNS \TEDIT.UNDO.LOOKS \TEDIT.UNDO.PARALOOKS))
        (COMS                                                   ; Revision-mark support
              (FNS \TEDIT.MARK.REVISION))
        (COMS                                                   ; Added by yabu.fx, for SUNLOADUP without DWIM
              (FNS \CREATE.TEDIT.DEFAULT.FMTSPEC \CREATE.TEDIT.FACE.MENU \CREATE.TEDIT.SIZE.MENU))
        (COMS                                                   ; Style-sheet support
              (FNS \TEDIT.APPLY.STYLES \TEDIT.APPLY.PARASTYLES TEDIT.STYLESHEET TEDIT.POP.STYLESHEET
                   TEDIT.PUSH.STYLESHEET TEDIT.ADD.STYLESHEET)
              ;; *TEDIT-PARASTYLE-CACHE* is an ALIST of  original char/para looks to styled char/para looks.  It is used to cache stylings, and is
              ;; reset when the main stylesheet changes, and when we change paragraph looks, given paras that have private char styles.
              ;; *TEDIT-CURRENTPARA-CACHE* is NIL if we're not in a para that has private char styles, or is the FMTSPEC (styled!) for that
              ;; para, if we are.  Used to decide when we have to flush *TEDIT-PARASTYLE-CACHE* at paragraph boundaries.  Mostly, this'll be
              ;; NIL and not interesting.
              ;; *TEDIT-STYLESHEET-SAVE-LIST* is a list of points inside TEDIT.STYLES, so we can "push" new style sheets on the front, and
              ;; "pop" them off sensibly.  This is the push-stack, in effect.  Used by TEDIT.ADD.STYLESHEET, TEDIT.PUSH.STYLESHEET, and
              ;; TEDIT.POP.STYLESHEET
              (INITVARS (TEDIT.STYLES))
              ;; RMK 2023: Maybe this should be one of the later ones? Only partly implemented
              (GLOBALVARS TEDIT.STYLES)
              (INITVARS (*TEDIT-PARASTYLE-CACHE*)
                     (*TEDIT-CURRENTPARA-CACHE*)
                     (*TEDIT-STYLESHEET-SAVE-LIST*))
              (GLOBALVARS *TEDIT-PARASTYLE-CACHE* *TEDIT-CURRENTPARA-CACHE* *TEDIT-STYLESHEET-SAVE-LIST*))))
```

```
;; Support for Character looks (font, italic/bold, sub/superscripting, etc) and paragraph looks (margins, centered/justified, tabs, etc.). Uses compiled
;; create functions in case DWIM is not available at loadup time.

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

[DATATYPE CHARLOOKS (;; Describes the appearance ("Looks") of characters in a TEdit document.
                     CLFONT                                    ; The font descriptor for these characters
                     CLNAME
                     ;; Name of the font (e.g., HELVETICA) THIS FIELD IS A HINT, OR FOR USE IN CHARLOOKS-BUILDING CODE.  USE
                     ;; FONTPROP TO GET THE RIGHT VALUE FROM CLFONT.
                     CLSIZE                                    ; Font size, in points
                     (CLITAL FLAG)                             ; T if the characters are italic, else NIL
                     (CLBOLD FLAG)                             ; T if the characters are bold, else NIL
                     (CLULINE FLAG)                            ; T if the characters are to be underscored, else NIL
                     (CLOLINE FLAG)                            ; T if the characters are to be overscored, else NIL
                     (CLSTRIKE FLAG)                           ; T if the characters are to be struck thru, else nil.
                     CLOFFSET                                  ; A superscripting offset in points (?) else NIL (SUBSCRIPTING
                                                               ; IF NEGATIVE.)
                     (CLSMALLCAP FLAG)                         ; T if small caps, else NIL
                     (CLINVERTED FLAG)                         ; T if the characters are to be shown white-on-black
                     (CLPROTECTED FLAG)                        ; T if chars can't be selected, else NIL
                     (CLINVISIBLE FLAG)                        ; T if TEDIT is to ignore these chars;  else NIL
                     (CLSELHERE FLAG)
                     ;; T if TEDIT can put selection after this char (for menu blanks) else NIL;  anything typed after this char will NOT BE
                     ;; PROTECTED.
                     (CLCANCOPY FLAG)
                     ;; T if this text can be selected for copying, even tho protected (it will become unprotected after the copy;  for Dribble/TTY
                     ;; interface)
                     (CLUNBREAKABLE FLAG)                      ; Spaces are treated as nonbreaking spaces
                     CLSTYLE                                   ; The style to be used in marking these characters;  overridden
                                                               ; by the other fields
                     CLUSERINFO                                ; Any information that an outsider wants to include
                     CLLEADER                                  ; For creating dotted and other kinds of leader
                     CLRULES
                     ;; For arbitrarily-places horizontal rules.  List of pairs, of (widthinpts  . offsetfrombaselineinpts).  Should be taken account of
                     ;; in ascent/descent calcs.
                     (CLMARK FLAG)
                     ;; Used for a mark-&-sweep of looks at PUT time -- T means this set of looks really IS in use in the document

                     )
       CLOFFSET _ 0 (INIT (DEFPRINT 'CHARLOOKS (FUNCTION \TEDIT.CHARLOOKS.DEFPRINT]

(DATATYPE FMTSPEC (;; Describe the paragraph formatting for a paragraph in a TEdit document.
                   1STLEFTMAR                                  ; Left margin of the first line of the paragraph
                   LEFTMAR                                     ; Left margin of the rest of the lines in the paragraph
                   RIGHTMAR                                    ; Right margin for the paragraph
                   LEADBEFORE                                  ; Leading above the paragraph's first line, in points
                   LEADAFTER                                   ; Leading below the paragraph's bottom line, in points.  NOT
                                                               ; IMPLEMENTED.
                   LINELEAD                                    ; Leading between lines, in points.  This space is added BELOW
                                                               ; each line in the para when TEDIT.LINELEADING.BELOW,
                                                               ; otherwise above, which is how it is documented.
                   FMTBASETOBASE                               ; The baseline-to-baseline spacing between lines in this
                                                               ; paragraph.  THIS OVERRIDES THE LINE LEADING
                   TABSPEC                                     ; The list of tabs for this paragraph, including CAR for a default
                                                               ; tab width
                   QUAD                                        ; How the para is formatted: one of LEFT, RIGHT, CENTERED,
                                                               ; JUSTIFIED
                   FMTSTYLE                                    ; The STYLE that controls this paragraph's appearance
                   FMTCHARSTYLES                               ; The characterstyles that control the appearance of characters
                                                               ; in this para (maybe?  may be part of the fmtstyle.)
                   FMTUSERINFO                                 ; Space for a PLIST of user info
                   FMTSPECIALX                                 ; A special horizontal location on the printed page for this para.
                   FMTSPECIALY                                 ; A special vertical location on the page for this para
                   (FMTHEADINGKEEP FLAG)                       ; This para should be kept with the top line or so of the next
                                                               ; para..
                   FMTPARATYPE                                 ; What kind of para this is: TEXT, PAGEHEADING, whatever
                   FMTPARASUBTYPE                              ; Sub type of the type, e.g., what KIND of page heading this is.
                   FMTNEWPAGEBEFORE                            ; Start a new box (if T) or back up the page formatting tree to
                                                               ; make a new box of the type named in the value -- by going the
                                                               ; least distance back up the tree, then back down until you find
                                                               ; that kind of box.
                   FMTNEWPAGEAFTER                             ; Similarly
                   FMTKEEP                                     ; For information about how this paragraph is to be kept with
                                                               ; other paragraphs.
```

```
                    FMTCOLUMN                                   ; For setting up side-by-side paragraphs easily ala BravoX
                    FMTVERTRULES                                ; For Keeping track of vertical rules in force
                    (FMTMARK FLAG)                              ; Used to keep track of which PARALOOKSs are really being
                                                                ; used -- a mark & collect is done just before a PUT, so that only
                                                                ; 'real' PARALOOKSs make it into the file
                                                                ; Used for a mark&sweep of para looks at PUT time -- T means
                                                                ; this looks really IS in use in the document, so it makes sense to
                                                                ; save it on the file.
                    (FMTHARDCOPY FLAG)                          ; T if this paragraph is to be displayed in hardcopy-format.
                    FMTREVISED                                  ; T (or perhaps a revision level or revision-mark spec??) if this
                                                                ; paragraph is to be marked as changed on output.
                    FMTHARDCOPYSCALE)                           ; The units-per-point (DSPSCALE) of the hardcopy stream that is
                                                                ; simulated in hardcopy-display mode (FMTHARDCOPY=T)
       (INIT (DEFPRINT 'FMTSPEC (FUNCTION \TEDIT.FMTSPEC.DEFPRINT)))
       LEADBEFORE _ 0 LEADAFTER _ 0 LINELEAD _ 0 TABSPEC _ (CONS DEFAULTTAB NIL))
)

(/DECLAREDATATYPE 'CHARLOOKS
       '(POINTER POINTER POINTER FLAG FLAG FLAG FLAG FLAG POINTER FLAG FLAG FLAG FLAG FLAG FLAG FLAG POINTER
              POINTER POINTER POINTER FLAG)
       ;; ---field descriptor list elided by lister---
       '16)

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

(/DECLAREDATATYPE 'FMTSPEC
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER FLAG POINTER POINTER POINTER POINTER POINTER POINTER POINTER FLAG FLAG POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '46)

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

(DECLARE%: EVAL@COMPILE

(PUTPROPS \WORDSETA DMACRO (OPENLAMBDA (A J V)
                              [CHECK (AND (ARRAYP A)
                                          (ZEROP (fetch (ARRAYP ORIG) of A))
                                          (EQ \ST.POS16 (fetch (ARRAYP TYP) of A]
                              (CHECK (IGREATERP (fetch (ARRAYP LENGTH) of A)
                                                J))
                              (\PUTBASE (fetch (ARRAYP BASE) of A)
                                     (IPLUS (fetch (ARRAYP OFFST) of A)
                                            J)
                                     V)))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS ONOFF MACRO [OPENLAMBDA (VAL)
                        (COND
                           (VAL 'ON)
                           (T 'OFF])
)
)
```

;; END EXPORTED DEFINITIONS

```
(/DECLAREDATATYPE 'CHARLOOKS
       '(POINTER POINTER POINTER FLAG FLAG FLAG FLAG FLAG POINTER FLAG FLAG FLAG FLAG FLAG FLAG FLAG POINTER
              POINTER POINTER POINTER FLAG)
       ;; ---field descriptor list elided by lister---
       '16)

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

(/DECLAREDATATYPE 'FMTSPEC
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER FLAG POINTER POINTER POINTER POINTER POINTER POINTER POINTER FLAG FLAG POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '46)

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

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

(DEFINEQ

(\TEDIT.CHARLOOKS.DEFPRINT
  [LAMBDA (LOOKS STREAM CPL NOLOC)                              ; Edited 26-Aug-2023 11:10 by rmk
```

```
    ;; CPL seems to be a hidden argument passed on calls from \PRINT-USING-DEFPRINT, usually with value 0.  So NOLOC is one beyond that
    (LET ((LOC (LOC LOOKS))
          (FACE (CONCAT (CL:IF (fetch (CHARLOOKS CLBOLD) of LOOKS)
                               "B"
                               "M")
                        (CL:IF (fetch (CHARLOOKS CLITAL) of LOOKS)
                               "I"
                               "R")))
          INFO)
         (SETQ INFO (CONCAT (L-CASE (fetch (CHARLOOKS CLNAME) of LOOKS)
                                    T)
                            (fetch (CHARLOOKS CLSIZE) of LOOKS)
                            (CL:IF (STREQUAL FACE "MR")
                                   ""
                                   FACE)))
         (CONS (CL:IF NOLOC
                      INFO
                      (CONCAT "{CL" (CAR LOC)
                              "/"
                              (CDR LOC)
                              ":" INFO "}"))])
```

(\TEDIT.FMTSPEC.DEFPRINT

```
  [LAMBDA (FMTSPEC STREAM)                                    ; Edited 26-Aug-2023 11:11 by rmk
    (LET ((LOC (LOC FMTSPEC)))
         (CONS (CONCAT "{FMT" (CAR LOC)
                       "/"
                       (CDR LOC)
                       ":"
                       (SUBSTRING (fetch (FMTSPEC QUAD) of FMTSPEC)
                                  1 2)
                       "_"
                       (fetch (FMTSPEC LEFTMAR) of FMTSPEC)
                       "_"
                       (fetch (FMTSPEC RIGHTMAR) of FMTSPEC)
                       "}"])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQQ TEDIT.TERMSA.FONTS NIL)

(RPAQ TEDIT.DEFAULT.FMTSPEC (\CREATE.TEDIT.DEFAULT.FMTSPEC))

(RPAQQ TEDIT.TERMSA.FONTS NIL)

(RPAQQ TEDIT.KNOWN.FONTS
       ((Classic 'CLASSIC)
        (Modern 'MODERN)
        (Terminal 'TERMINAL)
        (Titan 'TITAN)
        (Gacha 'GACHA)
        (Helvetica 'HELVETICA)
        (Times% Roman 'TIMESROMAN)))
)

(RPAQ? TEDIT.DEFAULT.FOLIO )

(RPAQQ TEDIT.CHARLOOKS.FEATURES (SUPERSCRIPT INVISIBLE SELECTPOINT PROTECTED SIZE FAMILY OVERLINE STRIKEOUT
                                            UNDERLINE EXPANSION SLOPE WEIGHT))

(RPAQ TEDIT.FACE.MENU (\CREATE.TEDIT.FACE.MENU))

(RPAQ TEDIT.SIZE.MENU (\CREATE.TEDIT.SIZE.MENU))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.CURRENT.FONT TEDIT.CURRENT.CHARLOOKS TEDIT.CURRENT.PARALOOKS TEDIT.KNOWN.FONTS TEDIT.FACE.MENU
       TEDIT.SIZE.MENU TEDIT.DEFAULT.FONT TEDIT.DEFAULT.FMTSPEC TEDIT.TERMSA.FONTS)
)

(ADDTOVAR FONTVARS (TEDIT.PROMPT.FONT DEFAULTFONT)
                   (TEDIT.ICON.FONT MENUFONT))

;; Character looks functions

(DEFINEQ
```

(CHARLOOKS.FROM.FONT

```
  [LAMBDA (FONT)                                              ; Edited 15-Oct-2023 18:56 by rmk
                                                              ; Edited 25-Aug-2023 20:03 by rmk
                                                              ; Edited 30-May-91 21:45 by jds
    ;; Create a CHARLOOKS from a font, filling in such fields as can be inferred from the font descriptor.
```

```
    (LET ((LOOKS (create CHARLOOKS
                         CLFONT _ FONT)))
         (CL:UNLESS (FONTP FONT)
             (\ILLEGAL.ARG FONT))                              ; It HAS to be a font, first off.
         (freplace (CHARLOOKS CLNAME) of LOOKS with (FONTPROP FONT 'FAMILY))
         (CL:WHEN [EQ 'BOLD (CAR (FONTPROP FONT 'FACE]
             (freplace (CHARLOOKS CLBOLD) of LOOKS with T))    ; Set the boldness bit, if it's a bold font.
         (CL:WHEN [EQ 'ITALIC (CADR (FONTPROP FONT 'FACE]
             (freplace (CHARLOOKS CLITAL) of LOOKS with T))    ; Set the italic bit, if it's italic
         (freplace (CHARLOOKS CLSIZE) of LOOKS with (FONTPROP FONT 'SIZE))
         LOOKS])
```

(\TEDIT.EQCLOOKS

```
  [LAMBDA (CLOOK1 CLOOK2)                                      ; Edited  1-Dec-2023 19:27 by rmk
                                                               ; Edited  9-Nov-2023 00:46 by rmk
                                                               ; Edited 24-Jul-2023 17:18 by rmk
                                                               ; Edited  1-Jun-93 11:49 by sybalsky:mv:envos
    ;; Given two sets of CHARLOOKS, are they effectively the same?
    (LET ((FONT1 (fetch (CHARLOOKS CLFONT) of CLOOK1))
          (FONT2 (fetch (CHARLOOKS CLFONT) of CLOOK2)))
         (OR (EQ CLOOK1 CLOOK2)
             (AND [OR (EQ FONT1 FONT2)
                      (AND (type? FONTCLASS FONT1)
                           (type? FONTCLASS FONT2)
                           (EQUAL (fetch (FONTCLASS DISPLAYFD) of FONT1)
                                  (fetch (FONTCLASS DISPLAYFD) of FONT1))
                           (EQUAL (FONTCLASSUNPARSE FONT1)
                                  (FONTCLASSUNPARSE FONT2]
                  (EQ (ffetch (CHARLOOKS CLPROTECTED) of CLOOK1)
                      (ffetch (CHARLOOKS CLPROTECTED) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLINVISIBLE) of CLOOK1)
                      (ffetch (CHARLOOKS CLINVISIBLE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLSELHERE) of CLOOK1)
                      (ffetch (CHARLOOKS CLSELHERE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLCANCOPY) of CLOOK1)
                      (ffetch (CHARLOOKS CLCANCOPY) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLULINE) of CLOOK1)
                      (ffetch (CHARLOOKS CLULINE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLOLINE) of CLOOK1)
                      (ffetch (CHARLOOKS CLOLINE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLINVERTED) of CLOOK1)
                      (ffetch (CHARLOOKS CLINVERTED) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLSTRIKE) of CLOOK1)
                      (ffetch (CHARLOOKS CLSTRIKE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLOFFSET) of CLOOK1)
                      (ffetch (CHARLOOKS CLOFFSET) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLSMALLCAP) of CLOOK1)
                      (ffetch (CHARLOOKS CLSMALLCAP) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLSTYLE) of CLOOK1)
                      (ffetch (CHARLOOKS CLSTYLE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLUNBREAKABLE) of CLOOK1)
                      (ffetch (CHARLOOKS CLUNBREAKABLE) of CLOOK2))
                  (EQ (ffetch (CHARLOOKS CLUSERINFO) of CLOOK1)
                      (ffetch (CHARLOOKS CLUSERINFO) of CLOOK2])
```

(\TEDIT.SAMECLOOKS

```
  [LAMBDA (CLOOK1 CLOOK2 FEATURES)                             ; Edited 24-Jul-2023 17:17 by rmk
                                                               ; Edited 30-May-91 21:45 by jds
    ;; Predicate to determine if CLOOK1 and CLOOK2 are the same in all the characteristics listed in FEATURES
    (for F in FEATURES always (SELECTQ F
                                  (FAMILY (EQ (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                        'FAMILY)
                                              (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                        'FAMILY)))
                                  (SIZE (EQ (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                      'SIZE)
                                            (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                      'SIZE)))
                                  (EXPANSION (EQ (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                           'EXPANSION)
                                                 (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                           'EXPANSION)))
                                  (SLOPE (EQ (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                       'SLOPE)
                                             (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                       'SLOPE)))
                                  (WEIGHT (EQ (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                        'WEIGHT)
                                              (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                        'WEIGHT)))
                                  (SUPERSCRIPT (EQ (fetch (CHARLOOKS CLOFFSET) of CLOOK1)
                                                   (fetch (CHARLOOKS CLOFFSET) of CLOOK2)))
                                  (INVISIBLE (EQ (fetch (CHARLOOKS CLINVISIBLE) of CLOOK1)
```

```
                                    (fetch (CHARLOOKS CLINVISIBLE) of CLOOK2)))
                          (SELECTPOINT (EQ (fetch (CHARLOOKS CLSELHERE) of CLOOK1)
                                           (fetch (CHARLOOKS CLSELHERE) of CLOOK2)))
                          (PROTECTED (EQ (fetch (CHARLOOKS CLPROTECTED) of CLOOK1)
                                         (fetch (CHARLOOKS CLPROTECTED) of CLOOK2)))
                          (OVERLINE (EQ (fetch (CHARLOOKS CLOLINE) of CLOOK1)
                                        (fetch (CHARLOOKS CLOLINE) of CLOOK2)))
                          (STRIKEOUT (EQ (fetch (CHARLOOKS CLSTRIKE) of CLOOK1)
                                         (fetch (CHARLOOKS CLSTRIKE) of CLOOK2)))
                          (UNDERLINE (EQ (fetch (CHARLOOKS CLULINE) of CLOOK1)
                                         (fetch (CHARLOOKS CLULINE) of CLOOK2)))
                          (UNBREAKABLE (fetch (CHARLOOKS CLUNBREAKABLE) of CLOOK1)
                                       (fetch (CHARLOOKS CLUNBREAKABLE) of CLOOK2))
                          (FACE (EQUAL (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK1)
                                                 'FACE)
                                       (FONTPROP (fetch (CHARLOOKS CLFONT) of CLOOK2)
                                                 'FACE)))
                          (ERROR (CONCAT F " is an unknown feature of character looks.  Detected in
                                          SAMECLOOKS"])
```

(TEDIT.CARETLOOKS

```
[LAMBDA (STREAM LOOKS)                                  ; Edited 15-Oct-2023 17:12 by rmk
                                                        ; Edited 28-May-2023 14:15 by rmk
                                                        ; Edited  6-Apr-2023 21:42 by rmk
                                                        ; Edited  8-Sep-2022 11:25 by rmk
                                                        ; Edited 30-May-91 21:40 by jds
```

;; Set the 'Caret looks' for a TEdit document, i.e., the looks that will be applied to newly-typed characters from here on.

```
(LET ((TEXTOBJ (TEXTOBJ STREAM)))                       ; Parse up the looks he gave us, to make sure they're a valid
                                                        ; CHARLOOKS
     (change (FGETTOBJ TEXTOBJ CARETLOOKS)
             (\TEDIT.CARETLOOKS.VERIFY TEXTOBJ (\TEDIT.PARSE.CHARLOOKS.LIST LOOKS DATUM TEXTOBJ])
```

(TEDIT.COPY.LOOKS

```
[LAMBDA (STREAM SOURCE DEST)                            ; Edited 17-Mar-2024 00:27 by rmk
                                                        ; Edited  9-Feb-2024 11:42 by rmk
                                                        ; Edited 18-Apr-2023 23:53 by rmk
                                                        ; Edited 22-Oct-2022 15:27 by rmk
                                                        ; Edited 22-Aug-2022 13:14 by rmk
                                                        ; Edited 30-May-91 21:43 by jds
```

;; Copy the CHARACTER LOOKS of one piece of text (actually, the first selected character) to another piece of text

```
(PROG ((TEXTOBJ (TEXTOBJ STREAM))
       LOOKS LEN)                                       ; get the character looks of the first character of SOURCE
      [SETQ LOOKS (PLOOKS (if (FIXP SOURCE)
                              then (\TEDIT.CHTOPC SOURCE TEXTOBJ)
                            elseif (type? SELECTION SOURCE)
                              then (\TEDIT.CHTOPC (fetch (SELECTION CH#) of SOURCE)
                                                  (fetch (SELECTION SELTEXTOBJ) of SOURCE))
                            else (\ILLEGAL.ARG SOURCE]
      (COND
         ((type? SELECTION DEST)                        ; make sure that the destination selection is in this document
          (CL:UNLESS (EQ TEXTOBJ (fetch (SELECTION SELTEXTOBJ) of DEST))
                 (\LISPERROR "Destination selection is not in stream " STREAM)))
         (T                                             ; set the LEN arg for TEDIT.LOOKS to be 1 since we just have a
                                                        ; char pos.
            (SETQ LEN 1)))
      (TEDIT.LOOKS TEXTOBJ LOOKS DEST LEN])
```

(\TEDIT.UNPARSE.CHARLOOKS.LIST

```
[LAMBDA (LOOKS)                                         ; Edited 24-Jul-2023 17:28 by rmk
                                                        ; Edited 11-Feb-2023 14:51 by rmk
                                                        ; Edited 30-May-91 21:45 by jds
```

;; Convert a CHARLOOKS into an equivalent PList-form for external consumption

```
(LET (NEWLOOKS OFFSET)
     (SETQ NEWLOOKS
      (for PROP in (LIST (fetch (CHARLOOKS CLSTYLE) of LOOKS)
                         (fetch (CHARLOOKS CLUSERINFO) of LOOKS)
                         (ONOFF (fetch (CHARLOOKS CLINVERTED) of LOOKS))
                         (FONTPROP (fetch (CHARLOOKS CLFONT) of LOOKS)
                                'WEIGHT)
                         (FONTPROP (fetch (CHARLOOKS CLFONT) of LOOKS)
                                'SLOPE)
                         (FONTPROP (fetch (CHARLOOKS CLFONT) of LOOKS)
                                'EXPANSION)
                         (ONOFF (fetch (CHARLOOKS CLULINE) of LOOKS))
                         (ONOFF (fetch (CHARLOOKS CLSTRIKE) of LOOKS))
                         (ONOFF (fetch (CHARLOOKS CLOLINE) of LOOKS))
                         (ONOFF (fetch (CHARLOOKS CLUNBREAKABLE) of LOOKS))
                         (FONTPROP (fetch (CHARLOOKS CLFONT) of LOOKS)
                                'FAMILY)
                         (FONTPROP (fetch (CHARLOOKS CLFONT) of LOOKS)
                                'SIZE)
```

```
                          (ONOFF (fetch (CHARLOOKS CLPROTECTED) of LOOKS))
                          (ONOFF (fetch (CHARLOOKS CLSELHERE) of LOOKS))
                          (ONOFF (fetch (CHARLOOKS CLINVISIBLE) of LOOKS)))
            as PROPNAME
            in '(STYLE USERINFO INVERTED WEIGHT SLOPE EXPANSION UNDERLINE STRIKEOUT OVERLINE UNBREAKABLE FAMILY
                       SIZE PROTECTED SELECTPOINT INVISIBLE)
            join (LIST PROPNAME PROP)))
        (SETQ OFFSET (fetch (CHARLOOKS CLOFFSET) of LOOKS))
        (push NEWLOOKS (COND
                           ((IGREATERP OFFSET 0)
                            'SUPERSCRIPT)
                           ((ILESSP OFFSET 0)
                            'SUBSCRIPT)
                           (T 'SUPERSCRIPT))
              (IABS (OR OFFSET 0)))
       NEWLOOKS])
```

(**TEDIT.MODIFYLOOKS**

```
[LAMBDA (LINE STARTX DS LOOKS LINEBASEY)                    ; Edited 20-Nov-2023 14:18 by rmk
                                                            ; Edited 27-May-2023 12:11 by rmk
                                                            ; Edited 24-Sep-2022 11:12 by rmk
                                                            ; Edited 30-May-91 21:45 by jds
```

;; Modify the screen to allow for underlining, etc. Also, restore the vertical offset to the baseline.

```
(LET ((CURX (DSPXPOSITION NIL DS))
      (CURY (DSPYPOSITION NIL DS))
      (FONT (fetch (CHARLOOKS CLFONT) of LOOKS)))
     (CL:WHEN (fetch (CHARLOOKS CLULINE) of LOOKS)          ; It's underlined.
         (MOVETO STARTX (ADD1 (IDIFFERENCE (IPLUS CURY)
                                           (GETLD LINE LTRUEDESCENT)))
                DS)
         (RELDRAWTO (IDIFFERENCE CURX STARTX)
                0 1 'PAINT DS))
     (CL:WHEN (fetch (CHARLOOKS CLOLINE) of LOOKS)          ; Over-line
         (MOVETO STARTX [IPLUS CURY (SUB1 (FONTPROP FONT 'ASCENT]
                DS)
         (RELDRAWTO (IDIFFERENCE CURX STARTX)
                0 1 'PAINT DS))
     (CL:WHEN (fetch (CHARLOOKS CLSTRIKE) of LOOKS)         ; Struck-thru
         (MOVETO STARTX (IPLUS CURY (IQUOTIENT (FONTPROP FONT 'ASCENT)
                                               3))
                DS)
         (RELDRAWTO (IDIFFERENCE CURX STARTX)
                0 1 'PAINT DS))
     (CL:WHEN (fetch (CHARLOOKS CLINVERTED) of LOOKS)       ; Inverse video
         (BLTSHADE BLACKSHADE DS STARTX (IDIFFERENCE CURY (FONTPROP FONT 'DESCENT))
                (IDIFFERENCE CURX STARTX)
                (FONTPROP FONT 'HEIGHT)
                'INVERT))
     (MOVETO CURX LINEBASEY DS])
```

(**TEDIT.NEW.FONT**

```
[LAMBDA (TEXTOBJ)                                           (* jds " 8-Feb-85 11:27")
  (PROG [(NAME (\TEDIT.MAKEFILENAME (TEDIT.GETINPUT TEXTOBJ "Name of font:  "]
        (AND NAME [SETQ TEDIT.KNOWN.FONTS (NCONC1 TEDIT.KNOWN.FONTS (LIST NAME (KWOTE (U-CASE NAME]
             (RETURN (U-CASE NAME])
```

(**\TEDIT.CARETLOOKS.VERIFY**

```
[LAMBDA (TEXTOBJ NEWLOOKS)                                  ; Edited 15-Oct-2023 20:13 by rmk
                                                            ; Edited 30-May-91 21:41 by jds
```

;; Check with the user's CARETLOOKSFN to see if he wants to make changes

```
(LET ((CARETFN (GETTEXTPROP TEXTOBJ 'CARETLOOKSFN))
      LOOKS)
     (SETQ LOOKS (AND CARETFN (APPLY* CARETFN NEWLOOKS TEXTOBJ)))
     (if (EQ LOOKS 'DON'T)
         then                                               ; He said not to change the looks.
              (OR (FGETTOBJ TEXTOBJ CARETLOOKS)
                  (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS))
       else (\TEDIT.UNIQUIFY.CHARLOOKS (OR LOOKS NEWLOOKS)
                   TEXTOBJ])
```

(**\TEDIT.CARETPIECE**

```
[LAMBDA (TEXTOBJ)                                           ; Edited 17-Mar-2024 00:27 by rmk
                                                            ; Edited  6-Apr-2023 21:32 by rmk
  (\TEDIT.CHTOPC (TEDIT.GETPOINT TEXTOBJ)
         TEXTOBJ])
```

(**\TEDIT.GET.INSERT.CHARLOOKS**

```
[LAMBDA (TEXTOBJ SEL)                                       ; Edited 17-Mar-2024 00:27 by rmk
                                                            ; Edited 16-Feb-2024 22:48 by rmk
                                                            ; Edited 15-Dec-2023 08:40 by rmk
```

```
                                                                    ; Edited  3-Aug-2023 22:39 by rmk
                                                                    ; Edited  9-Oct-2022 13:57 by rmk
                                                                    ; Edited 22-Aug-2022 13:21 by rmk
                                                                    ; Edited 30-May-91 21:45 by jds
```

;; Return the looks at SEL, or defaults. Reset CLPROTECTED if need be.

```
(LET ((PC (\TEDIT.CHTOPC (IMAX 1 (IMIN (FGETTOBJ TEXTOBJ TEXTLEN)
                                       (TEDIT.GETPOINT TEXTOBJ SEL)))
                  TEXTOBJ))
      LOOKS)
     (SETQ LOOKS (if PC
                     then (PLOOKS PC)
                   elseif (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS)
                   else (CHARLOOKS.FROM.FONT DEFAULTFONT)))
     (CL:WHEN (fetch (CHARLOOKS CLPROTECTED) of LOOKS)        ; Unprotect by copying to a new CHARLOOKS.
         (SETQ LOOKS (create CHARLOOKS using LOOKS CLPROTECTED _ NIL CLSELHERE _ NIL)))
     (\TEDIT.CARETLOOKS.VERIFY TEXTOBJ LOOKS])
```

(\TEDIT.GET.TERMSA.WIDTHS

```
[LAMBDA (TERMSA FONT)                                           (* jds "22-OCT-83 21:36")
```

(* If the guy is using a terminal table, get an updated set of widths to reflect that.)

```
(PROG ((NWIDTHS (ARRAY 256 'SMALLP 0 0)))
      (for I from 0 to 255 do (\WORDSETA NWIDTHS I (TEDIT.CHARWIDTH I FONT TERMSA)))
      (RETURN NWIDTHS])
```

(\TEDIT.PARSE.CHARLOOKS.LIST

```
[LAMBDA (NLOOKS DEFAULTCLOOKS TEXTOBJ)                          ; Edited 13-Nov-2023 01:08 by rmk
                                                                ; Edited 11-Nov-2023 16:09 by rmk
                                                                ; Edited 16-Oct-2023 09:02 by rmk
                                                                ; Edited 24-Jul-2023 17:24 by rmk
                                                                ; Edited 30-May-91 21:46 by jds
```

;; NLOOKS is either a CHARLOOKS, a FONTDESCRIPTOR, or an ALIST-format looks spec. If NLOOKS is not already a CHARLOOKS, it is ;; coerced into one.

;; ALIST is the complicated case. The various properties are extracted from the list, wutg values for unspecified properties taken from ;; DEFAULTCLOOKS. If DEFAULTCLOOKS is not provided, default values are taken from the DEFAULTCHARLOOKS of TEXTOBJ.

```
(if (type? CHARLOOKS NLOOKS)
    then NLOOKS
  elseif (FONTP NLOOKS)
    then (CHARLOOKS.FROM.FONT NLOOKS)
  else (LET (FAMILY FONT FACE SIZEINC SIZE PROT SELHERE ULINE OLINE STRIKE SUPER OFFSETINC WEIGHT SLOPE
             EXPANSION SUB INVISIBLE UNBREAKABLE STYLE STYLESET UISET USERINFO NEWFONTSPEC NEWFONT
             INVERSEVIDEO)                                      ; Construct the set of new looks to apply:
                                                                ; We got an AList -- prepare looks changes in that form
       ;; First get the new font
       [SETQ FONT (FONTP (LISTGET NLOOKS 'FONT]
       (CL:WHEN (SETQ FAMILY (LISTGET NLOOKS 'FAMILY))
           (PUSH NEWFONTSPEC 'FAMILY FAMILY))
       (CL:WHEN (SETQ SIZE (LISTGET NLOOKS 'SIZE))
           (PUSH NEWFONTSPEC 'SIZE SIZE))
       (SETQ SIZEINC (OR (LISTGET NLOOKS 'SIZEINCREMENT)
                         0))
       (SETQ FACE (LISTGET NLOOKS 'FACE))
       (SETQ WEIGHT (LISTGET NLOOKS 'WEIGHT))
       (SETQ SLOPE (LISTGET NLOOKS 'SLOPE))
       (SETQ EXPANSION (LISTGET NLOOKS 'EXPANSION))
       (COND
          ((OR WEIGHT SLOPE EXPANSION)                          ; Setting one of these inhibits the FACE parameter
           (CL:WHEN WEIGHT
               (PUSH NEWFONTSPEC 'WEIGHT WEIGHT))
           (CL:WHEN SLOPE
               (PUSH NEWFONTSPEC 'SLOPE SLOPE))
           (CL:WHEN EXPANSION
               (PUSH NEWFONTSPEC 'EXPANSION EXPANSION)))
          (FACE (PUSH NEWFONTSPEC 'FACE FACE)))
       (CL:UNLESS DEFAULTCLOOKS
           [SETQ DEFAULTCLOOKS (OR (AND TEXTOBJ (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS))
                                   (CHARLOOKS.FROM.FONT (FONTCOPY NIL NEWFONTSPEC])
       [PUSH NEWFONTSPEC 'SIZE (IPLUS SIZEINC (FONTPROP (fetch (CHARLOOKS CLFONT) of DEFAULTCLOOKS)
                                                        'SIZE]
       (SETQ NEWFONT (OR FONT (\TEDIT.FONTCOPY (fetch (CHARLOOKS CLFONT) of DEFAULTCLOOKS)
                                  NEWFONTSPEC TEXTOBJ)))
       ;;
       ;; Now for other CHARLOOKS properties
       (SETQ PROT (LISTGET NLOOKS 'PROTECTED))
       (SETQ SELHERE (LISTGET NLOOKS 'SELECTPOINT))
       (SETQ ULINE (LISTGET NLOOKS 'UNDERLINE))
       (SETQ OLINE (LISTGET NLOOKS 'OVERLINE))
       (SETQ INVERSEVIDEO (LISTGET NLOOKS 'INVERTED))
       (SETQ STRIKE (LISTGET NLOOKS 'STRIKEOUT))
```

```
                (SETQ INVISIBLE (LISTGET NLOOKS 'INVISIBLE))
                (SETQ SUPER (LISTGET NLOOKS 'SUPERSCRIPT))
                (SETQ SUB (LISTGET NLOOKS 'SUBSCRIPT))
                (SETQ OFFSETINC (LISTGET NLOOKS 'OFFSETINCREMENT))
                (SETQ UNBREAKABLE (LISTGET NLOOKS 'UNBREAKABLE))
                (SETQ STYLE (LISTGET NLOOKS 'STYLE))
                (SETQ STYLESET (FMEMB 'STYLE NLOOKS))
                (SETQ USERINFO (LISTGET NLOOKS 'USERINFO))
                (SETQ UISET (FMEMB 'USERINFO NLOOKS))
                [SETQ NLOOKS (create CHARLOOKS using DEFAULTCLOOKS CLFONT _ NEWFONT CLSIZE _ SIZE CLBOLD _
                                                     (EQ 'BOLD (FONTPROP NEWFONT 'WEIGHT))
                                                     CLITAL _ (EQ 'ITALIC (FONTPROP NEWFONT 'SLOPE]
```

;; NLOOKS has the new font but all other properties come from the default. Override if specified.

```
                [AND PROT (replace (CHARLOOKS CLPROTECTED) of NLOOKS with (EQ PROT 'ON]
                [AND SELHERE (replace (CHARLOOKS CLSELHERE) of NLOOKS with (EQ SELHERE 'ON]
                [AND ULINE (replace (CHARLOOKS CLULINE) of NLOOKS with (EQ ULINE 'ON]
                [AND OLINE (replace (CHARLOOKS CLOLINE) of NLOOKS with (EQ OLINE 'ON]
                [AND STRIKE (replace (CHARLOOKS CLSTRIKE) of NLOOKS with (EQ STRIKE 'ON]
                [AND INVISIBLE (replace (CHARLOOKS CLINVISIBLE) of NLOOKS with (EQ INVISIBLE 'ON]
                [AND UNBREAKABLE (replace (CHARLOOKS CLUNBREAKABLE) of NLOOKS with (EQ UNBREAKABLE 'ON]
                [AND INVERSEVIDEO (replace (CHARLOOKS CLINVERTED) of NLOOKS with (EQ INVERSEVIDEO 'ON]
                (AND SUPER (replace (CHARLOOKS CLOFFSET) of NLOOKS with SUPER))
                (AND SUB (replace (CHARLOOKS CLOFFSET) of NLOOKS with (IMINUS SUB)))
                (AND STYLESET (replace (CHARLOOKS CLSTYLE) of NLOOKS with STYLE))
                (AND UISET (replace (CHARLOOKS CLUSERINFO) of NLOOKS with USERINFO))
                (AND OFFSETINC (add (fetch (CHARLOOKS CLOFFSET) of NLOOKS)
                                    OFFSETINC))
                NLOOKS])

)

(DEFINEQ

(\**TEDIT.TRANSLATE.ASCIICHARS**
  [LAMBDA (TEXTOBJ NOASCIIFONTS)                                    ; Edited 17-Mar-2024 00:25 by rmk
                                                                    ; Edited  1-Dec-2023 22:28 by rmk
                                                                    ; Edited 27-Nov-2023 16:13 by rmk
                                                                    ; Edited 26-Nov-2023 11:19 by rmk
                                                                    ; Edited 14-Nov-2023 19:21 by rmk
                                                                    ; Edited  9-Nov-2023 23:56 by rmk
    ;; Converts characters in Alto/Ascii font pieces to their XCCS character and font (more or less) equivalents. The affected characters are put in their
    ;; own string pieces with their new CHARLOOKS. Asciifont pieces are completely replaced if NOASCIIFONTS, otherwise untranslated characters
    ;; remain in their Asciifonts.

    ;; It is tricky to mix the pieces iteration with the TEDIT.RPLCHARCODE, the within-piece indexing has to be adjusted to continue the
    ;; iteration,because the replacement may split the piece.

    ;; ASCIITONSTRANSLATIONS and the mapping arrays are from INTERPRESS.

    ;; \ASCIITOSTAR is the default translation array, for Gacha, Timesromand. HIPPO, MATH ... have their own.

    (DECLARE (GLOBALVARS ASCIITONSTRANSLATIONS \ASCIITOSTAR))
    (SETQ TEXTOBJ (TEXTOBJ TEXTOBJ))
    (CL:WHEN (thereis CL in (FGETTOBJ TEXTOBJ TXTCHARLOOKSLIST) unless (EQ 'CLASSIC (fetch (CHARLOOKS CLNAME)
                                                                                        of CL))
                suchthat
                    ;; CLASSIC is in the list presumably to provide a coercion to MODERN for Interpress. We don't want to translate it.
                    (ASSOC (fetch (CHARLOOKS CLNAME) of CL)
                           ASCIITONSTRANSLATIONS))
        (for PC CLOOKS TRANS MAPARRAY NEWFONTNAME STRING FAT CLOOKSLIST CLNAME TARRAYLAST inpieces (
                                                                                        \TEDIT.FIRSTPIECE
                                                                                            TEXTOBJ)
            eachtime (SETQ CLOOKS (PLOOKS PC))
                     (SETQ CLNAME (fetch (CHARLOOKS CLNAME) of CLOOKS))
            unless (OR (EQ OBJECT.PTYPE (PTYPE PC))
                       (EQ CLNAME 'CLASSIC))
            when (SETQ TRANS (ASSOC CLNAME ASCIITONSTRANSLATIONS))
            do ;; PC needs some work.
                (SETQ MAPARRAY (CADR TRANS))
                (SETQ NEWFONTNAME (CADDR TRANS))
                (CL:WHEN MAPARRAY                                   ; Idiosyncratic fonts (MATH, CYRILLIC).
                    (SETQ MAPARRAY (GETATOMVAL MAPARRAY))           ; Global value
                    (CL:WHEN (AND NOASCIIFONTS (PREVPIECE PC))
                        ;; Look backward for NEWFONTNAME, since that piece has already been coerced. The idea is to get Cyrillic to continue
                        ;; the previous looks (serif, san-serif)
                        [SETQ NEWFONTNAME (fetch (CHARLOOKS CLNAME) of (PLOOKS (PREVPIECE PC]))
                (if (OR MAPARRAY NOASCIIFONTS)
                    then ;; Translate all characters in idiosyncratic fonts, flush everything and change the looks even for Helvetica etc. if NO
                         ;; ALTOFONTS
                        (CL:UNLESS MAPARRAY (SETQ MAPARRAY \ASCIITOSTAR))
                        (SETQ TARRAYLAST (SUB1 (ARRAYSIZE MAPARRAY)))
                        ;; Create a string with the translated codes, then convert the existing piece to a string piece holding that string.
                        (SETQ STRING (ALLOCSTRING (PLEN PC)))
```

```
        (for OFFSET CODE NEWCODE from 1 to (PLEN PC)
           do ;; Out-of-range alone and zero newcodes alone (some arrays are not filled in).
              (SETQ CODE (\TEDIT.PIECE.NTHCHARCODE TEXTOBJ PC OFFSET))
              (RPLCHARCODE STRING OFFSET (if [OR (IGREATERP CODE TARRAYLAST)
                                                 (ZEROP (SETQ NEWCODE (ELT MAPARRAY CODE]
                                             then CODE
                                            else NEWCODE)))
        (SETQ FAT (ffetch (STRINGP FATSTRINGP) of STRING))
        (FSETPC PC PTYPE (CL:IF FAT
                                FATSTRING.PTYPE
                                THINSTRING.PTYPE))
        (FSETPC PC PCONTENTS STRING)
        (FSETPC PC PFPOS NIL)
        (FSETPC PC PBINABLE (NOT FAT))
        (FSETPC PC PBYTESPERCHAR (CL:IF FAT
                                        2
                                        1))
        (FSETPC PC PBYTELEN (CL:IF FAT
                                   (UNFOLD (PLEN PC)
                                           2)
                                   (PLEN PC)))
        (FSETPC PC PLOOKS (\TEDIT.TRANSLATE.ASCII.CHARLOOKS TEXTOBJ CLOOKS NEWFONTNAME))
     else ;; Must be a text font (GACHA, TIMESROMAN, HELVETICA) \ASCIITOSTAR is the translation array, mostly identities.
        ;; The way TEDIT.RPLCHARCODE works, the PC piece is always the suffix after the last change. So offset 1 is always the
        ;; next character to be examined after a change, and PLEN is always shrinking. START has to be adjusted after each hit to
        ;; reflect the new starting CHNO of the shortened PC.
        (bind (OFFSET _ 0)
              OLDCODE NEWCODE START eachtime (add OFFSET 1)
                                             (SETQ OLDCODE (\TEDIT.PIECE.NTHCHARCODE TEXTOBJ PC OFFSET)
                                              )
           while OLDCODE when (ILEQ (SETQ OLDCODE (\TEDIT.PIECE.NTHCHARCODE TEXTOBJ PC OFFSET))
                                    255)
           unless (EQ OLDCODE (SETQ NEWCODE (ELT \ASCIITOSTAR OLDCODE)))
           do (CL:UNLESS START
                  (SETQ START (\TEDIT.PCTOCH PC TEXTOBJ)))
              (TEDIT.RPLCHARCODE TEXTOBJ (IPLUS START OFFSET -1)
                     NEWCODE
                     (FSETPC PC PLOOKS (\TEDIT.TRANSLATE.ASCII.CHARLOOKS TEXTOBJ CLOOKS NEWFONTNAME)
                           ))
           ;; Move START up to the new START of PC, set OFFSET back to its beginning.
           (add START OFFSET)
           (SETQ OFFSET 0)))
     finally ;; Here we change the default and caret looks.  Perhaps this should be done only if NOASCIIFONTS.  But there is a risk that Ascii
             ;; fonts and characters would slip in by future editing.
        (CL:WHEN NOASCIIFONTS
            (SETQ CLOOKS (FGETTOBJ TEXTOBJ DEFAULTCHARLOOKS))
            (SETQ CLNAME (fetch (CHARLOOKS CLNAME) of CLOOKS))
            (CL:WHEN (AND (NEQ CLNAME 'CLASSIC)
                          (SETQ TRANS (ASSOC CLNAME ASCIITONSTRANSLATIONS)))
                (FSETTOBJ TEXTOBJ DEFAULTCHARLOOKS (\TEDIT.TRANSLATE.ASCII.CHARLOOKS TEXTOBJ CLOOKS
                                                                                     (CADDR TRANS))))
            (SETQ CLOOKS (FGETTOBJ TEXTOBJ CARETLOOKS))
            (SETQ CLNAME (fetch (CHARLOOKS CLNAME) of CLOOKS))
            (CL:WHEN (AND (NEQ CLNAME 'CLASSIC)
                          (SETQ TRANS (ASSOC CLNAME ASCIITONSTRANSLATIONS)))
                (FSETTOBJ TEXTOBJ CARETLOOKS (\TEDIT.TRANSLATE.ASCII.CHARLOOKS TEXTOBJ CLOOKS
                                                                               (CADDR TRANS)))))
        (CL:WHEN CLOOKSLIST
            ;; Something happened, get rid of any lingering old looks
            (\TEDIT.UNIQUIFY.ALL TEXTOBJ))))])
```

(**TEDIT.CONVERT.TO.FORMATTED**

```
  [LAMBDA (TEXTOBJ START END)                                    ; Edited 20-Mar-2024 11:00 by rmk
                                                                 ; Edited 17-Mar-2024 12:06 by rmk
                                                                 ; Edited 15-Mar-2024 13:53 by rmk
                                                                 ; Edited  6-Jan-2024 15:10 by rmk
                                                                 ; Edited 11-Dec-2023 10:02 by rmk
                                                                 ; Edited  9-Nov-2023 15:37 by rmk
                                                                 ; Edited  5-Nov-2023 11:22 by rmk
                                                                 ; Edited 24-Sep-2023 23:30 by rmk
                                                                 ; Edited 22-May-2023 22:50 by rmk
                                                                 ; Edited 20-May-2023 16:44 by rmk
                                                                 ; Edited  8-May-2023 08:44 by rmk
                                                                 ; Edited 29-Apr-93 19:47 by jds
    ;; Turn an unformatted TEdit file into a formatted TEdit file, essentially simulating ANY.EOLC by ensuring that end-of-line indicators (CR, CRLF, LF)
    ;; are canonicalized as EOL's and then interpreted as paragraph ends. Pieces are split so that EOLs are always at piece-end.  If it wasn't formatted
    ;; before, it presumably didn't have any looks to worry about, just the defaults.

    ;; Using BIN for the main iteration is a little tricky when TEDIT.RPLCHARCODE is used to make the single-character change.  RPLCHARCODE
    ;; can split the pieces and parameters in the TSTREAM that are used to drive the high-speed (BINABLE) operation.  It should perhaps figure out
    ;; how to fix the stream internally, but for now the \TEXTSETFILEPTR gets things consistent again.
```

```
    (TEXTOBJ! TEXTOBJ)
    (CL:UNLESS (OR (FGETTOBJ TEXTOBJ FORMATTEDP)
                   (ZEROP (FGETTOBJ TEXTOBJ TEXTLEN)))
        (CL:UNLESS START (SETQ START 1))
        (CL:UNLESS END
            (SETQ END (FGETTOBJ TEXTOBJ TEXTLEN)))
        (FSETTOBJ TEXTOBJ \DIRTY T)
        (CL:WHEN (IGEQ END START)
            (for CHNO (TSTREAM _ (FGETTOBJ TEXTOBJ STREAMHINT)) from START first
                                                          ;; CHNO is in characters, one more than
                                                          ;; stream positions
                                                          (\TEDIT.TEXTSETFILEPTR
                                                           TSTREAM
                                                           (SUB1 START))
              do (SELCHARQ (BIN TSTREAM)
                       (LF ;; Linefeed not preceded by CR, replace by EOL and mark it paragraph-last. \TEXTSETFILEPTR to make sure that
                           ;; the next BIN does what we want
                           (TEDIT.RPLCHARCODE TEXTOBJ CHNO (CHARCODE EOL))
                           (FSETPC (\TEDIT.CHTOPC CHNO TEXTOBJ)
                                  PPARALAST T)
                           (\TEDIT.TEXTSETFILEPTR TSTREAM CHNO))
                       (CR ;; Post-CR characters go to a separate piece, the CR piece is then paragraph-final
                           (FSETPC (PREVPIECE (\TEDIT.ALIGNEDPIECE (ADD1 CHNO)
                                                                   TEXTOBJ))
                                  PPARALAST T)
                           (CL:WHEN (EQ (CHARCODE LF)
                                        (\TEDIT.TEXTPEEKBIN TSTREAM T))
                                                          ; DO WE EVER WANT TO SEE LF'S ??
                               ;; Linefeed following CR.  Chop it off from whatever follows, and then delete it.
                               (add END -1)               ; One less char to do
                               (\TEDIT.DELETEPIECES (\TEDIT.SELPIECES (ADD1 CHNO)
                                                                      (ADD1 CHNO))
                                                    TEXTOBJ)))
                           NIL)
              repeatuntil (IGEQ CHNO END)))               ; Test END explicitly, because it may get reduced
        (FSETTOBJ TEXTOBJ FORMATTEDP T)
        (\TEDIT.MARK.LINES.DIRTY TEXTOBJ START END))])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \TEDIT.TRANSLATE.ASCII.CHARLOOKS MACRO
          [OPENLAMBDA (TEXTOBJ CLOOKS NEWFONTNAME)
            ;; Macro because CLOOKSLIST is set.  The alist avoids creating and then uniquifying each time we want to make the same translation.
            (CDR (OR (ASSOC CLOOKS CLOOKSLIST)
                     (CAR (PUSH CLOOKSLIST (CONS CLOOKS (\TEDIT.UNIQUIFY.CHARLOOKS
                                                         (create CHARLOOKS using CLOOKS CLFONT _
                                                                 (\TEDIT.FONTCOPY
                                                                  (fetch (CHARLOOKS CLFONT)
                                                                     of CLOOKS)
                                                                  (LIST 'FAMILY NEWFONTNAME)
                                                                  TEXTOBJ)
                                                                 CLNAME _ NEWFONTNAME)
                                                         TEXTOBJ])
)

(DEFINEQ

(\TEDIT.UNIQUIFY.CHARLOOKS
  [LAMBDA (NEWLOOK TEXTOBJ)                                ; Edited 16-Mar-2024 00:32 by rmk
                                                           ; Edited 15-Oct-2023 17:17 by rmk
                                                           ; Edited 18-Aug-2023 21:47 by rmk
                                                           ; Edited 15-Aug-2023 20:57 by rmk
                                                           ; Edited  3-Aug-2023 17:52 by rmk
                                                           ; Edited 30-May-91 21:40 by jds
    ;; Assure that there is only ONE of a given CHARLOOKS in the document--so that all instances of that set of looks share structure. When we get a
    ;; hit, we move it to the front, hopefully more frequent looks will come earlier in the list.
    (CL:WHEN NEWLOOK
        (for LOOKTAIL LOOK PREVTAIL on (FGETTOBJ TEXTOBJ TXTCHARLOOKSLIST)
           do (SETQ LOOK (CAR LOOKTAIL))
              (CL:WHEN (\TEDIT.EQCLOOKS NEWLOOK LOOK)
                  (CL:WHEN PREVTAIL                        ; Not already in first position
                      (RPLACD PREVTAIL (CDR LOOKTAIL))
                      (push (FGETTOBJ TEXTOBJ TXTCHARLOOKSLIST)
                            LOOK))
                  (RETURN LOOK))
              (SETQ PREVTAIL LOOKTAIL)
           finally (push (FGETTOBJ TEXTOBJ TXTCHARLOOKSLIST)
                         NEWLOOK)
                   (RETURN NEWLOOK))])
```

(\TEDIT.UNIQUIFY.PARALOOKS

```
[LAMBDA (NEWLOOK TEXTOBJ)                                   ; Edited 16-Mar-2024 00:30 by rmk
                                                            ; Edited 18-Aug-2023 21:48 by rmk
                                                            ; Edited 30-May-91 21:41 by jds
```

;; Assure that there is only ONE of a given PARALOOKS in the document--so that all instances of that set of looks share structure. When we get a ;; hit, we move it to the front, hopefully more frequent looks will come earlier in the list.

```
(for LOOKTAIL LOOK PREVTAIL on (GETTOBJ TEXTOBJ TXTPARALOOKSLIST)
   do (SETQ LOOK (CAR LOOKTAIL))
      (CL:WHEN (\TEDIT.EQFMTSPEC NEWLOOK LOOK)
          (CL:WHEN PREVTAIL                                 ; Not already in first position
              (RPLACD PREVTAIL (CDR LOOKTAIL))
              (push (GETTOBJ TEXTOBJ TXTPARALOOKSLIST)
                    LOOK))
          (RETURN LOOK))
      (SETQ PREVTAIL LOOKTAIL)
   finally (push (GETTOBJ TEXTOBJ TXTPARALOOKSLIST)
                 NEWLOOK)
           (RETURN NEWLOOK])
```

(\TEDIT.UNIQUIFY.ALL

```
[LAMBDA (TEXTOBJ)                                           ; Edited 16-Mar-2024 10:03 by rmk
                                                            ; Edited 14-Nov-2023 16:20 by rmk
                                                            ; Edited 25-Aug-2023 08:57 by rmk
                                                            ; Edited 15-Aug-2023 22:04 by rmk
                                                            ; Edited  3-Aug-2023 18:44 by rmk
                                                            ; Edited  1-Aug-2023 11:43 by rmk
                                                            ; Edited 13-Jul-2022 22:56 by rmk
   (SETTOBJ TEXTOBJ TXTCHARLOOKSLIST NIL)
   (SETTOBJ TEXTOBJ TXTPARALOOKSLIST NIL)
   (for PC inpieces (\TEDIT.FIRSTPIECE TEXTOBJ) do          ;; Assure that the CHARLOOKS and PARALOOKS of every piece are in the cache.
                                                               (change (PLOOKS PC)
                                                                       (\TEDIT.UNIQUIFY.CHARLOOKS DATUM TEXTOBJ))
                                                               (change (PPARALOOKS PC)
                                                                       (\TEDIT.UNIQUIFY.PARALOOKS DATUM TEXTOBJ)))
   (CL:WHEN (GETTOBJ TEXTOBJ DEFAULTCHARLOOKS)
       (change (GETTOBJ TEXTOBJ DEFAULTCHARLOOKS)
               (\TEDIT.UNIQUIFY.CHARLOOKS DATUM TEXTOBJ)))
   (change (GETTOBJ TEXTOBJ CARETLOOKS)
           (\TEDIT.UNIQUIFY.CHARLOOKS DATUM TEXTOBJ))
   (change (GETTOBJ TEXTOBJ FMTSPEC)
           (\TEDIT.UNIQUIFY.PARALOOKS DATUM TEXTOBJ])
```

(\TEDIT.FLUSH.UNUSED.LOOKS

```
[LAMBDA (TEXTOBJ)                                           ; Edited 16-Mar-2024 10:03 by rmk
                                                            ; Edited 25-Aug-2023 08:03 by rmk
                                                            ; Edited 15-Aug-2023 22:11 by rmk
                                                            ; Edited 30-May-91 21:47 by jds
```

;; Run thru the CHARLOOKS and PARALOOKS lists for this document, and flush any looks that aren't being used in the document itself.

```
(LET ((CHARLOOKS (GETTOBJ TEXTOBJ TXTCHARLOOKSLIST))
      (PARALOOKS (GETTOBJ TEXTOBJ TXTPARALOOKSLIST)))
     ;; Reset the in-use mark in all looks
     (for LOOKS in CHARLOOKS do (replace (CHARLOOKS CLMARK) of LOOKS with NIL))
     (for LOOKS in PARALOOKS do (replace (FMTSPEC FMTMARK) of LOOKS with NIL))
     ;; Run thru the pieces in the document, marking the looks that are really in use.
     (for PC inpieces (\TEDIT.FIRSTPIECE TEXTOBJ) do (replace (CHARLOOKS CLMARK) of (PLOOKS PC) with T)
                                                     (replace (FMTSPEC FMTMARK) of (PPARALOOKS PC)
                                                        with T))
     ;; Keep only those char and para looks that ARE being used.
     (SETTOBJ TEXTOBJ TXTCHARLOOKSLIST (for LOOKS in CHARLOOKS when (fetch (CHARLOOKS CLMARK) of LOOKS)
                                          collect (replace (CHARLOOKS CLMARK) of LOOKS with NIL)
                                                  LOOKS))
     (SETTOBJ TEXTOBJ TXTPARALOOKSLIST (for LOOKS in PARALOOKS when (fetch (FMTSPEC FMTMARK) of LOOKS)
                                          collect (replace (FMTSPEC FMTMARK) of LOOKS with NIL)
                                                  LOOKS])

)

;; Public entries

(DEFINEQ
```

(TEDIT.LOOKS

```
[LAMBDA (TSTREAM NEWLOOKS SELORCH# LEN)                     ; Edited  9-Feb-2024 11:40 by rmk
                                                            ; Edited 23-Dec-2023 14:12 by rmk
                                                            ; Edited 28-May-2023 13:56 by rmk
                                                            ; Edited 24-May-2023 23:12 by rmk
                                                            ; Edited 30-May-91 21:41 by jds
```

```
;; Programmatic interface for character looks in TEdit.  Applies to the LEN characters starting at SELORCH#, or the characters selected by
;; SELORCH# if it is a selection.  Nothing to do if the selection isn't set.  POINT is used only to set the caret looks.

(LET ((TEXTOBJ (TEXTOBJ TSTREAM))
      SEL)
     (CL:UNLESS (\TEDIT.READONLY TEXTOBJ)
         ;; Ignores LEN if SELORCH# is a selection
         [SETQ SEL (COND
                     ((type? SELECTION SELORCH#)
                      SELORCH#)
                     (SELORCH# (TEDIT.SETSEL TSTREAM SELORCH# LEN 'LEFT))
                     (T (FGETTOBJ TEXTOBJ SEL]
         (CL:WHEN (GETSEL SEL SET)
             (if (AND (IGREATERP (GETSEL SEL DCH)
                                 0)
                      (ILEQ (GETSEL SEL CH#)
                            (TEXTLEN TEXTOBJ)))
                 then (\TEDIT.CHANGE.LOOKS TSTREAM NEWLOOKS SEL)
                 else ;; Out of bounds or maybe a point selection, no text to change. Punt out after setting the caret looks.  Old code did not
                      ;; set the history, should we?
                      (TEDIT.CARETLOOKS TSTREAM NEWLOOKS)))))])
```

(TEDIT.GET.LOOKS

```
[LAMBDA (TEXTOBJ CH#ORCHARLOOKS)                                    ; Edited 17-Mar-2024 00:27 by rmk
                                                                    ; Edited 14-Dec-2023 21:00 by rmk
                                                                    ; Edited 21-Jun-2023 11:10 by rmk
                                                                    ; Edited 22-Aug-2022 13:14 by rmk
                                                                    ; Edited 30-May-91 21:44 by jds
    ;; Returns as a property list the looks denoted by CH#ORCHARLOOKS.

    (SETQ TEXTOBJ (TEXTOBJ TEXTOBJ))
    (\TEDIT.UNPARSE.CHARLOOKS.LIST (if (type? CHARLOOKS CH#ORCHARLOOKS)
                                       then                         ; Unparse the given looks.
                                            CH#ORCHARLOOKS
                                   elseif (ZEROP (TEXTLEN TEXTOBJ))
                                       then                         ; Empty document, use extant caret looks.
                                            (FGETTOBJ TEXTOBJ CARETLOOKS)
                                   else (PLOOKS (\TEDIT.CHTOPC (OR (FIXP CH#ORCHARLOOKS)
                                                                   (GETSEL (if (type? SELECTION CH#ORCHARLOOKS)
                                                                               then CH#ORCHARLOOKS
                                                                           elseif (NULL CH#ORCHARLOOKS)
                                                                               then (TEXTSEL TEXTOBJ)
                                                                           else (\ILLEGAL.ARG CH#ORCHARLOOKS)
                                                                           )
                                                                   CH#))
                                                TEXTOBJ])
```

(TEDIT.SUBLOOKS

```
[LAMBDA (TEXTSTREAM OLDLOOKSLIST NEWLOOKSLIST)                      ; Edited 17-Mar-2024 17:17 by rmk
                                                                    ; Edited 16-Mar-2024 10:03 by rmk
                                                                    ; Edited 13-Nov-2023 00:26 by rmk
                                                                    ; Edited 18-Apr-2023 23:53 by rmk
                                                                    ; Edited 22-Aug-2022 13:06 by rmk
                                                                    ; Edited 26-Apr-93 14:53 by jds

;;; User entry to substitute one set of looks for another.  Goes through the whole textstream and whenever the looks match the characteristics of
;;; OLDLOOKSLIST which are specified, the characteristics listed in NEWLOOKSLIST are substituted.

    (LET ((TEXTOBJ (TEXTOBJ TEXTSTREAM)))                           ; Turn off the selection, first.
         (CL:UNLESS (ZEROP (FGETTOBJ TEXTOBJ TEXTLEN))
             [for PC CHANGEMADE SEL (OLDLOOKS _ (\TEDIT.PARSE.CHARLOOKS.LIST OLDLOOKSLIST NIL TEXTOBJ))
                 (NEWLOOKS _ (\TEDIT.PARSE.CHARLOOKS.LIST NEWLOOKSLIST NIL TEXTOBJ))
                 (FEATURELIST _ (for A on OLDLOOKSLIST by (CDDR A) collect (CAR A)))
                 (CH# _ 1)
                 inpieces
                 (\TEDIT.FIRSTPIECE TEXTOBJ) as CH# from 1 by (PLEN PC) when (\TEDIT.SAMECLOOKS OLDLOOKS
                                                                                                 (PLOOKS PC)
                                                                                                 FEATURELIST)
              do (CL:UNLESS CHANGEMADE
                     (SETQ CHANGEMADE T)
                     (SETQ SEL (FGETTOBJ TEXTOBJ SEL))
                     (\TEDIT.SHOWSEL SEL NIL)
                     (FSETTOBJ TEXTOBJ \DIRTY T))
                 ;; Note that we may be creating new looks each time, depending on what is there and what is changed.
                 (FSETPC PC PLOOKS (\TEDIT.UNIQUIFY.CHARLOOKS (\TEDIT.PARSE.CHARLOOKS.LIST NEWLOOKSLIST
                                                                                            (PLOOKS PC)
                                                                                            TEXTOBJ)
                                                              TEXTOBJ))
                 (\TEDIT.MARK.LINES.DIRTY TEXTOBJ CH# (+ CH# (PLEN PC)))
              finally (CL:WHEN (FGETTOBJ TEXTOBJ \WINDOW)
                               (\TEDIT.UPDATE.SCREEN TEXTOBJ)       ; Update the screen image
                               (\TEDIT.FIXSEL SEL TEXTOBJ)
```

```
                        (\TEDIT.SHOWSEL SEL T))
                    (RETURN (CL:IF CHANGEMADE
                                'Done
                                'NoChangesMade)])])
```

(TEDIT.FINDLOOKS

```
  [LAMBDA (TEXTSTREAM OLDLOOKSLIST CH#)                    ; Edited 17-Mar-2024 00:27 by rmk
                                                           ; Edited  3-Dec-2023 00:09 by rmk
                                                           ; Edited 13-Nov-2023 00:26 by rmk
                                                           ; Edited 18-Apr-2023 23:53 by rmk
                                                           ; Edited 22-Aug-2022 13:06 by rmk
                                                           ; Edited 26-Apr-93 14:53 by jds
```

;;; Finds and selects the next substring of the text whose looks are a superset of OLDLOOKSLIST.

```
    (LET ((TEXTOBJ (TEXTOBJ TEXTSTREAM)))                  ; Turn off the selection, first.
         (if (AND (FIXP CH#)
                  (IGEQ CH# 1)
                  (ILEQ CH# (FGETTOBJ TEXTOBJ TEXTLEN)))
           elseif (type? SELECTION CH#)
             then (SETQ CH# (TEDIT.GETPOINT TEXTOBJ CH#))
           elseif (NULL CH#)
             then (SETQ CH# (TEDIT.GETPOINT TEXTOBJ (FGETTOBJ TEXTOBJ SEL)))
           else (\ILLEGAL.ARG CH#))
         (CL:UNLESS (ZEROP (FGETTOBJ TEXTOBJ TEXTLEN))
             [for PC PCLAST FOUNDCH# (OLDLOOKS _ (\TEDIT.PARSE.CHARLOOKS.LIST OLDLOOKSLIST NIL TEXTOBJ))
                  (FEATURELIST _ (for A on OLDLOOKSLIST by (CDDR A) collect (CAR A)))
                  inpieces
                  (\TEDIT.CHTOPC CH# TEXTOBJ) when (\TEDIT.SAMECLOOKS OLDLOOKS (PLOOKS PC)
                                                                      FEATURELIST)
                  do [SETQ PCLAST (find PC1 inpieces (NEXTPIECE PC) suchthat (NOT (\TEDIT.SAMECLOOKS OLDLOOKS
                                                                                                    (PLOOKS PC1)
                                                                                                    FEATURELIST]
                     (SETQ PCLAST (CL:IF PCLAST
                                         (PREVPIECE PCLAST)
                                         PC))
                     (SETQ FOUNDCH# (\TEDIT.PCTOCH PC TEXTOBJ))
                     (TEDIT.SETSEL TEXTOBJ FOUNDCH# (IDIFFERENCE (IPLUS (\TEDIT.PCTOCH PCLAST)
                                                                        (PLEN PCLAST))
                                                                 FOUNDCH#)
                           'RIGHT)
                     (TEDIT.NORMALIZECARET TEXTOBJ)
                     (RETURN (\TEDIT.COPYSEL (FGETTOBJ TEXTOBJ SEL])])

)

(DEFINEQ
```

(\TEDIT.CHANGE.LOOKS

```
  [LAMBDA (TSTREAM NEWLOOKS SEL)                           ; Edited 15-Mar-2024 14:23 by rmk
                                                           ; Edited 11-Mar-2024 00:37 by rmk
                                                           ; Edited  9-Mar-2024 11:36 by rmk
                                                           ; Edited 24-Feb-2024 12:33 by rmk
                                                           ; Edited 22-Feb-2024 23:01 by rmk
                                                           ; Edited 23-Dec-2023 15:24 by rmk
                                                           ; Edited 31-Oct-2023 19:40 by rmk
                                                           ; Edited 24-Jul-2023 17:20 by rmk
                                                           ; Edited 28-May-2023 14:38 by rmk
                                                           ; Edited 11-May-2023 12:59 by rmk
                                                           ; Edited 19-Apr-93 14:08 by jds
```

;;; Internal programmatic interface to changing character looks. DOES NOT CHANGE the current selection.

;;; THIS FUNCTION AND \TEDIT.PARSE.CHARLOOKS.LIST MUST TRACK ONE ANOTHER, FOR THE P-LIST FORMAT..

```
    (PROG ((TEXTOBJ (TEXTOBJ TSTREAM))
           FAMILY FONT FACE SIZE PROTECTED SELECTPOINT UNDERLINE OVERLINE STRIKEOUT INVERTED UNBREAKABLE
           SUPERSCRIPT WEIGHT SLOPE SIZEINCREMENT OFFSETINCREMENT EXPANSION SUBSCRIPT INVISIBLE FONTSPEC NEWFONT
           STYLE STYLESET UISET USERINFO START-OF-PIECE SELPIECES)
                                                           ; Construct the set of new looks to apply:
          (if (type? CHARLOOKS NEWLOOKS)
              then (SETQ NEWLOOKS (\TEDIT.UNIQUIFY.CHARLOOKS NEWLOOKS TEXTOBJ))
            elseif (FONTP NEWLOOKS)
              then (SETQ FONT NEWLOOKS)
                   (SETQ NEWLOOKS NIL)                     ; NEWLOOKS is NIL unless it's a CHARLOOKS
            else ;; We got a PList -- extract the various look properties

                                                           (* (for L on NEWLOOKS by CDDR do
                                                           (SET (CAR L) (CADR L))))
                 (SETQ FONT (LISTGET NEWLOOKS 'FONT))
                 (SETQ FAMILY (LISTGET NEWLOOKS 'FAMILY))
                 (SETQ FACE (LISTGET NEWLOOKS 'FACE))
                 (SETQ SIZE (LISTGET NEWLOOKS 'SIZE))
                 (SETQ PROTECTED (LISTGET NEWLOOKS 'PROTECTED))
                 (SETQ SELECTPOINT (LISTGET NEWLOOKS 'SELECTPOINT))
                 (SETQ UNDERLINE (LISTGET NEWLOOKS 'UNDERLINE))
                 (SETQ OVERLINE (LISTGET NEWLOOKS 'OVERLINE))
```

```
            (SETQ UNBREAKABLE (LISTGET NEWLOOKS 'UNBREAKABLE))
            (SETQ INVERTED (LISTGET NEWLOOKS 'INVERTED))
            (SETQ STRIKEOUT (LISTGET NEWLOOKS 'STRIKEOUT))
            (SETQ INVISIBLE (LISTGET NEWLOOKS 'INVISIBLE))
            (SETQ SUPERSCRIPT (LISTGET NEWLOOKS 'SUPERSCRIPT))
            (SETQ SUBSCRIPT (LISTGET NEWLOOKS 'SUBSCRIPT))
            (SETQ WEIGHT (LISTGET NEWLOOKS 'WEIGHT))
            (SETQ SLOPE (LISTGET NEWLOOKS 'SLOPE))
            (SETQ EXPANSION (LISTGET NEWLOOKS 'EXPANSION))
            (SETQ SIZEINCREMENT (LISTGET NEWLOOKS 'SIZEINCREMENT))
            (SETQ OFFSETINCREMENT (LISTGET NEWLOOKS 'OFFSETINCREMENT))
            (SETQ STYLE (LISTGET NEWLOOKS 'STYLE))
            (SETQ STYLESET (FMEMB 'STYLE NEWLOOKS))
            (SETQ USERINFO (LISTGET NEWLOOKS 'USERINFO))
            (SETQ UISET (FMEMB 'USERINFO NEWLOOKS))          ; We have extracted all the properties
            (CL:WHEN FAMILY
                (push FONTSPEC 'FAMILY FAMILY))
            (CL:WHEN FONT
                (CL:UNLESS (OR (type? FONTCLASS FONT)
                               (type? FONTDESCRIPTOR FONT))
                    (TEDIT.PROMPTPRINT (CONCAT FONT " isn't a valid font descriptor.")
                           T)
                    (RETURN)))
            (if (OR WEIGHT SLOPE EXPANSION)
                then                                         ; Setting one of these inhibits the FACE parameter
                     (AND WEIGHT (push FONTSPEC 'WEIGHT WEIGHT))
                     (AND SLOPE (push FONTSPEC 'SLOPE SLOPE))
                     (AND EXPANSION (push FONTSPEC 'EXPANSION EXPANSION))
              elseif FACE
                then (push FONTSPEC 'FACE FACE))
            (if SIZE
                then (push FONTSPEC 'SIZE SIZE)
              elseif SIZEINCREMENT
                then (push FONTSPEC 'SIZE 'BOGUSSIZE))
            (SETQ NEWLOOKS NIL))
        (FSETTOBJ TEXTOBJ \DIRTY T)                          ; Mark the document changed.
        (SETQ SELPIECES (\TEDIT.SELPIECES SEL))
        (for PC NEWPCLOOKS OLDLOOKSLIST OLDPCLOOKS (CARETPC _ (\TEDIT.CARETPIECE TEXTOBJ))
            inselpieces SELPIECES
           do (SETQ OLDPCLOOKS (PLOOKS PC))
              (SETQ OLDLOOKSLIST (NCONC1 OLDLOOKSLIST OLDPCLOOKS))
                                                             ; Save old looks for the Undo.
              [COND
                 (NEWLOOKS                                   ; We got a CHARLOOKS in.  Just use it
                        (FSETPC PC PLOOKS NEWLOOKS))
                 (T                                          ; Otherwise, we have to override selectively
                    (SETQ NEWPCLOOKS (create CHARLOOKS using OLDPCLOOKS))
                    (FSETPC PC PLOOKS NEWPCLOOKS)
                    ;; If a size increment is specified, then add to the newspecs arg for fontcopy, the entry with the incremented size from the
                    ;; current font.
                    (SETQ NEWFONT (OR FONT (\TEDIT.FONTCOPY
                                             (fetch (CHARLOOKS CLFONT) of OLDPCLOOKS)
                                             (PROGN (CL:WHEN SIZEINCREMENT
                                                             ; There's a size change requested.  Fix up the size of the font.
                                                      (LISTPUT FONTSPEC 'SIZE
                                                             (IPLUS (FONTPROP (fetch (CHARLOOKS CLFONT)
                                                                                of OLDPCLOOKS)
                                                                              'SIZE)
                                                                    SIZEINCREMENT)))
                                                    FONTSPEC)
                                            TEXTOBJ)))
                    (CL:UNLESS NEWFONT (RETURN))
                    (replace (CHARLOOKS CLFONT) of NEWPCLOOKS with NEWFONT)
                                                             ; Give this piece its new looks
                    [replace (CHARLOOKS CLBOLD) of NEWPCLOOKS with (EQ 'BOLD (FONTPROP NEWFONT 'WEIGHT]
                    [replace (CHARLOOKS CLITAL) of NEWPCLOOKS with (EQ 'ITALIC (FONTPROP NEWFONT 'SLOPE]
                    [AND PROTECTED (replace (CHARLOOKS CLPROTECTED) of NEWPCLOOKS with (EQ PROTECTED
                                                                                           'ON]
                    [AND SELECTPOINT (replace (CHARLOOKS CLSELHERE) of NEWPCLOOKS with (EQ SELECTPOINT
                                                                                           'ON]
                    [AND UNDERLINE (replace (CHARLOOKS CLULINE) of NEWPCLOOKS with (EQ UNDERLINE 'ON]
                    [AND OVERLINE (replace (CHARLOOKS CLOLINE) of NEWPCLOOKS with (EQ OVERLINE 'ON]
                    [AND STRIKEOUT (replace (CHARLOOKS CLSTRIKE) of NEWPCLOOKS with (EQ STRIKEOUT 'ON]
                    [AND INVISIBLE (replace (CHARLOOKS CLINVISIBLE) of NEWPCLOOKS with (EQ INVISIBLE
                                                                                           'ON]
                    (AND SUPERSCRIPT (replace (CHARLOOKS CLOFFSET) of NEWPCLOOKS with SUPERSCRIPT))
                    (AND SUBSCRIPT (replace (CHARLOOKS CLOFFSET) of NEWPCLOOKS with (IMINUS SUBSCRIPT)))
                    (AND STYLESET (replace (CHARLOOKS CLSTYLE) of NEWPCLOOKS with STYLE))
                    (AND UISET (replace (CHARLOOKS CLUSERINFO) of NEWPCLOOKS with USERINFO))
                    [AND UNBREAKABLE (replace (CHARLOOKS CLUNBREAKABLE) of NEWPCLOOKS with (EQ UNBREAKABLE
                                                                                               'ON]
                    (AND OFFSETINCREMENT (replace (CHARLOOKS CLOFFSET) of NEWPCLOOKS
                                                  with (IPLUS (OR (fetch (CHARLOOKS CLOFFSET) of NEWPCLOOKS)
                                                                  0)
                                                              OFFSETINCREMENT)))
                    [AND INVERTED (replace (CHARLOOKS CLINVERTED) of NEWPCLOOKS with (EQ INVERTED 'ON]
```

```
                    (replace (CHARLOOKS CLSIZE) of NEWPCLOOKS with (FONTPROP NEWFONT 'SIZE))
                    ;; Assure that each set of looks appears only once in the world.
                    (replace (PIECE PLOOKS) of PC with (\TEDIT.UNIQUIFY.CHARLOOKS NEWPCLOOKS TEXTOBJ]
                (CL:WHEN (EQ PC CARETPC)
                     (TEDIT.CARETLOOKS TEXTOBJ NEWPCLOOKS))
            finally (\TEDIT.HISTORYADD TEXTOBJ (create TEDITHISTORYEVENT
                                                       THACTION _ :Looks
                                                       THLEN _ (GETSEL SEL DCH)
                                                       THCH# _ (GETSEL SEL CH#)
                                                       THFIRSTPIECE _ (fetch (SELPIECES SPFIRST) of SELPIECES)
                                                       THOLDINFO _ OLDLOOKSLIST))
                    (CL:WHEN (FGETTOBJ TEXTOBJ \WINDOW)
                        (\TEDIT.MARK.LINES.DIRTY TEXTOBJ SEL)
                        (\TEDIT.SHOWSEL (FGETTOBJ TEXTOBJ SEL)
                               NIL)
                        (SELECTQ INVISIBLE
                             (ON ;; Previously visible characters have disappeared, to a point
                                 (\TEDIT.UPDATE.SEL (FGETTOBJ TEXTOBJ SEL)
                                        (GETSEL SEL CH#)
                                        0
                                        'LEFT NIL T))
                             (OFF ;; Previously invisible characters have appeared, to select them
                                  (\TEDIT.UPDATE.SEL (FGETTOBJ TEXTOBJ SEL)
                                         (GETSEL SEL CH#)
                                         (GETSEL SEL DCH)
                                         'RIGHT NIL T))
                             NIL)
                        (\TEDIT.RESET.EXTEND.PENDING.DELETE (FGETTOBJ TEXTOBJ SEL)
                               TEXTOBJ)
                        (\TEDIT.UPDATE.SCREEN TEXTOBJ)            ; Update the screen image
                        (\TEDIT.FIXSEL (FGETTOBJ TEXTOBJ SEL)
                               TEXTOBJ)
                        (\TEDIT.SHOWSEL (FGETTOBJ TEXTOBJ SEL)
                               T))])
```

(\TEDIT.LOOKS

```
  [LAMBDA (TEXTOBJ)                                               ; Edited  8-May-2023 21:21 by rmk
                                                                  ; Edited 30-May-91 21:41 by jds
```

;; Handler for the middle-button menu's LOOKS button. Brings up 3 menus, for font, face, and size. Then calls TEDIT.LOOKS to make the
;; requested changes.

```
  (LET* ((SEL (fetch (TEXTOBJ SEL) of TEXTOBJ))
         (REGION (WINDOWPROP (CAR (fetch (TEXTOBJ \WINDOW) of TEXTOBJ))
                         'REGION))
         (POS (create POSITION
                      XCOORD _ (fetch LEFT of REGION)
                      YCOORD _ (fetch TOP of REGION)))
         FONT FACE SIZE NEWLOOKS)
        (CL:WHEN (ILEQ (fetch (SELECTION CH#) of SEL)
                       (TEXTLEN TEXTOBJ))                         ; Otherwise, nothing to change
            (COND
               ((fetch (SELECTION SET) of SEL)
                (CURSORPOSITION (CREATEPOSITION 0 (fetch HEIGHT of REGION))
                       (CAR (fetch (TEXTOBJ \WINDOW) of TEXTOBJ)))
                (SETQ FONT (MENU (create MENU
                                         TITLE _ "Font:"
                                         ITEMS _ (NCONC1 (COPY TEDIT.KNOWN.FONTS)
                                                         (LIST 'Other (LIST (FUNCTION TEDIT.NEW.FONT)
                                                                            TEXTOBJ)))
                                         CENTERFLG _ T)
                                 POS))                            ; Set the font for the new text.
                (SETQ FACE (SELECTQ (MENU TEDIT.FACE.MENU POS)
                                    (Bold 'BOLD)
                                    (Italic 'ITALIC)
                                    (Bold% Italic 'BOLDITALIC)
                                    (Regular 'STANDARD)
                                    NIL))                         ; Set the face (bold, etc.)
                (SETQ SIZE (MENU TEDIT.SIZE.MENU POS))            ; Set the type size
                                                                  ; Construct the set of new looks to apply:
                (SETQ NEWLOOKS (AND FONT (LIST 'FAMILY FONT)))
                (CL:WHEN FACE
                    (SETQ NEWLOOKS (CONS 'FACE (CONS FACE NEWLOOKS))))
                (CL:WHEN SIZE
                    (SETQ NEWLOOKS (CONS 'SIZE (CONS SIZE NEWLOOKS))))
                (CL:WHEN NEWLOOKS                                 ; There's something to do.
                    (TEDIT.LOOKS TEXTOBJ NEWLOOKS SEL)))
               (T (TEDIT.PROMPTPRINT TEXTOBJ "Please select some text to modify" T)))])
```

(\TEDIT.FONTCOPY

```
  [LAMBDA (FONT NEWSPECS TEXTOBJ)                                 ; Edited 22-Feb-2024 15:35 by rmk
                                                                  ; Edited 12-Nov-2023 23:24 by rmk
                                                                  (* jds "26-Dec-84 16:06")
```

```
    ;; Cloak FONTCOPY in protection for the user from an unavailable font.
    (COND
       ((NULL NEWSPECS)                                          ; No changes specified.  Punt it.
        FONT)
       [(CAR (NLSETQ (FONTCOPY FONT NEWSPECS]
       (T (TEDIT.PROMPTPRINT TEXTOBJ [CONCAT "Can't find font " (OR (LISTGET NEWSPECS 'FAMILY)
                                                                    (FONTPROP FONT 'FAMILY))
                                             " "
                                             (OR (LISTGET NEWSPECS 'SIZE)
                                                 (FONTPROP FONT 'SIZE))
                                             " "
                                             (OR (LISTGET NEWSPECS 'FACE)
                                                 (FONTPROP FONT 'FACE]
             T)
          NIL])

)

;; Paragraph looks functions

(DEFINEQ

(\TEDIT.EQFMTSPEC
  [LAMBDA (PARALOOK1 PARALOOK2)                                  ; Edited  2-Jul-93 21:32 by sybalskY:MV:ENVOS
    ;; Given two sets of FMTSPECS, are they effectively the same?
    (OR (EQ PARALOOK1 PARALOOK2)
        (AND (EQ (fetch (FMTSPEC QUAD) of PARALOOK1)
                 (fetch (FMTSPEC QUAD) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTNEWPAGEBEFORE) of PARALOOK1)
                 (ffetch (FMTSPEC FMTNEWPAGEBEFORE) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTNEWPAGEAFTER) of PARALOOK1)
                 (ffetch (FMTSPEC FMTNEWPAGEAFTER) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTSTYLE) of PARALOOK1)
                 (ffetch (FMTSPEC FMTSTYLE) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTSPECIALX) of PARALOOK1)
                 (ffetch (FMTSPEC FMTSPECIALX) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTSPECIALY) of PARALOOK1)
                 (ffetch (FMTSPEC FMTSPECIALY) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTHEADINGKEEP) of PARALOOK1)
                 (ffetch (FMTSPEC FMTHEADINGKEEP) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTKEEP) of PARALOOK1)
                 (ffetch (FMTSPEC FMTKEEP) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTPARATYPE) of PARALOOK1)
                 (ffetch (FMTSPEC FMTPARATYPE) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTPARASUBTYPE) of PARALOOK1)
                 (ffetch (FMTSPEC FMTPARASUBTYPE) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTHARDCOPY) of PARALOOK1)
                 (ffetch (FMTSPEC FMTHARDCOPY) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTREVISED) of PARALOOK1)
                 (ffetch (FMTSPEC FMTREVISED) of PARALOOK2))
             (EQ (ffetch (FMTSPEC FMTCOLUMN) of PARALOOK1)
                 (ffetch (FMTSPEC FMTCOLUMN) of PARALOOK2))
             (EQP (ffetch (FMTSPEC 1STLEFTMAR) of PARALOOK1)
                  (ffetch (FMTSPEC 1STLEFTMAR) of PARALOOK2))
             (EQP (ffetch (FMTSPEC LEFTMAR) of PARALOOK1)
                  (ffetch (FMTSPEC LEFTMAR) of PARALOOK2))
             (EQP (ffetch (FMTSPEC RIGHTMAR) of PARALOOK1)
                  (ffetch (FMTSPEC RIGHTMAR) of PARALOOK2))
             (EQP (ffetch (FMTSPEC LEADBEFORE) of PARALOOK1)
                  (ffetch (FMTSPEC LEADBEFORE) of PARALOOK2))
             (EQP (ffetch (FMTSPEC LEADAFTER) of PARALOOK1)
                  (ffetch (FMTSPEC LEADAFTER) of PARALOOK2))
             (EQP (ffetch (FMTSPEC LINELEAD) of PARALOOK1)
                  (ffetch (FMTSPEC LINELEAD) of PARALOOK2))
             (EQP (ffetch (FMTSPEC FMTBASETOBASE) of PARALOOK1)
                  (ffetch (FMTSPEC FMTBASETOBASE) of PARALOOK2))
             (EQUAL (ffetch (FMTSPEC FMTUSERINFO) of PARALOOK1)
                    (ffetch (FMTSPEC FMTUSERINFO) of PARALOOK2))
             (EQUAL (ffetch (FMTSPEC FMTCHARSTYLES) of PARALOOK1)
                    (ffetch (FMTSPEC FMTCHARSTYLES) of PARALOOK2))
             (EQUALALL (ffetch (FMTSPEC TABSPEC) of PARALOOK1)
                    (ffetch (FMTSPEC TABSPEC) of PARALOOK2])

(\TEDIT.GET.PARALOOKS
  [LAMBDA (TSTREAM SELORCH#)                                     ; Edited 17-Mar-2024 00:27 by rmk
                                                                 ; Edited 11-Dec-2023 10:12 by rmk
                                                                 ; Edited 22-Jun-2023 00:02 by rmk
                                                                 ; Edited 11-Feb-2023 14:55 by rmk
                                                                 ; Edited 30-May-91 21:44 by jds
    ;; Return a proplist of paragraph formatting information about the characters specified.
    (LET* [(TEXTOBJ (TEXTOBJ TSTREAM))
           (PC (if (type? PIECE SELORCH#)
                   then
```

```
                    ;; An internal call, if we already have the piece.
                    SELORCH#
               else (\TEDIT.CHTOPC (OR (FIXP SELORCH#)
                                       (GETSEL (if (type? SELECTION SELORCH#)
                                                   then SELORCH#
                                                elseif (NULL SELORCH#)
                                                   then (TEXTSEL TEXTOBJ)
                                                   else (\ILLEGAL.ARG SELORCH#))
                                               CH#))
                                   TEXTOBJ)))
             (FMTSPEC (CL:IF PC
                             (PPARALOOKS PC)
                             (fetch (TEXTOBJ FMTSPEC) of TEXTOBJ))]
            (for PROP in (LIST (fetch (FMTSPEC QUAD) of FMTSPEC)
                               (fetch (FMTSPEC 1STLEFTMAR) of FMTSPEC)
                               (fetch (FMTSPEC LEFTMAR) of FMTSPEC)
                               (fetch (FMTSPEC RIGHTMAR) of FMTSPEC)
                               (fetch (FMTSPEC LEADBEFORE) of FMTSPEC)
                               (fetch (FMTSPEC LEADAFTER) of FMTSPEC)
                               (fetch (FMTSPEC LINELEAD) of FMTSPEC)
                               (fetch (FMTSPEC FMTBASETOBASE) of FMTSPEC)
                               (fetch (FMTSPEC TABSPEC) of FMTSPEC)
                               (fetch (FMTSPEC FMTSTYLE) of FMTSPEC)
                               (fetch (FMTSPEC FMTCHARSTYLES) of FMTSPEC)
                               (fetch (FMTSPEC FMTUSERINFO) of FMTSPEC)
                               (fetch (FMTSPEC FMTSPECIALX) of FMTSPEC)
                               (fetch (FMTSPEC FMTSPECIALY) of FMTSPEC)
                               (fetch (FMTSPEC FMTPARATYPE) of FMTSPEC)
                               (fetch (FMTSPEC FMTPARASUBTYPE) of FMTSPEC)
                               (ONOFF (fetch (FMTSPEC FMTNEWPAGEBEFORE) of FMTSPEC))
                               (ONOFF (fetch (FMTSPEC FMTNEWPAGEAFTER) of FMTSPEC))
                               (ONOFF (fetch (FMTSPEC FMTHEADINGKEEP) of FMTSPEC))
                               (fetch (FMTSPEC FMTKEEP) of FMTSPEC)
                               (ONOFF (fetch (FMTSPEC FMTHARDCOPY) of FMTSPEC))
                               (fetch (FMTSPEC FMTREVISED) of FMTSPEC)
                               (fetch (FMTSPEC FMTCOLUMN) of FMTSPEC))
              as PROPNAME
              in '(QUAD 1STLEFTMARGIN LEFTMARGIN RIGHTMARGIN PARALEADING POSTPARALEADING LINELEADING BASETOBASE
                        TABS STYLE CHARSTYLES USERINFO SPECIALX SPECIALY TYPE SUBTYPE NEWPAGEBEFORE NEWPAGEAFTER
                        HEADINGKEEP KEEP HARDCOPY REVISED COLUMN)
              join (LIST PROPNAME PROP])
```

(\TEDIT.PARSE.PARALOOKS.LIST

```
  [LAMBDA (NEWLOOKS OLDLOOKS)                                ; Edited 17-Oct-2023 12:08 by rmk
                                                             ; Edited  9-May-2023 13:20 by rmk
                                                             ; Edited  5-Sep-2022 15:39 by rmk
                                                             ; Edited  3-Jul-93 21:49 by sybalskY:MV:ENVOS
                                                             ; Apply a given format spec to the paragraphs which are included
                                                             ; in this guy.
    (if (type? FMTSPEC NEWLOOKS)
        then                                                 ; if we were given an FMTSPEC really replace the FMTSPEC of
                                                             ; all pieces affected
          NEWLOOKS
      else (LET (1STLEFT LEFT RIGHT LEADB LEADA LLEAD TABSPECC QUADD NLOOKSAVE TYPE SUBTYPE TYPESET SUBTYPESET
                 NEWBEFORESET NEWBEFORE NEWAFTERSET NEWAFTER KEEP KEEPSET HEADINGKEEP BASETOBASE BASESET
                 REVISED REVISEDSET COLUMN COLUMNSET USERINFO USERINFOSET SPECIALX SPECXSET SPECIALY
                 SPECYSET STYLE STYLESET CHARSTYLES CHARSTYLESSET)
                                                             ; create an FMTSPEC from the Plist
            (SETQ 1STLEFT (LISTGET NEWLOOKS '1STLEFTMARGIN))
            (SETQ LEFT (LISTGET NEWLOOKS 'LEFTMARGIN))
            (SETQ RIGHT (LISTGET NEWLOOKS 'RIGHTMARGIN))
            (SETQ LEADB (LISTGET NEWLOOKS 'PARALEADING))
            (SETQ LEADA (LISTGET NEWLOOKS 'POSTPARALEADING))
            (SETQ LLEAD (LISTGET NEWLOOKS 'LINELEADING))
            (SETQ TYPESET (FMEMB 'TYPE NEWLOOKS))
            (SETQ TYPE (LISTGET NEWLOOKS 'TYPE))
            (SETQ SUBTYPESET (FMEMB 'SUBTYPE NEWLOOKS))
            (SETQ SUBTYPE (LISTGET NEWLOOKS 'SUBTYPE))
            (SETQ NEWBEFORESET (FMEMB 'NEWPAGEBEFORE NEWLOOKS))
            (SETQ NEWBEFORE (LISTGET NEWLOOKS 'NEWPAGEBEFORE))
            (SETQ NEWAFTERSET (FMEMB 'NEWPAGEAFTER NEWLOOKS))
            (SETQ NEWAFTER (LISTGET NEWLOOKS 'NEWPAGEAFTER))
            (SETQ HEADINGKEEP (LISTGET NEWLOOKS 'HEADINGKEEP))
                                                             ; Keep for headings
            (SETQ KEEP (LISTGET NEWLOOKS 'KEEP))             ; More general 'Keep-together' spec -- undefined as of 5/22/85
            (SETQ KEEPSET (FMEMB 'KEEP NEWLOOKS))
            (SETQ BASETOBASE (LISTGET NEWLOOKS 'BASETOBASE))
            (SETQ BASESET (FMEMB 'BASETOBASE NEWLOOKS))
            (SETQ REVISED (LISTGET NEWLOOKS 'REVISED))
            (SETQ REVISEDSET (FMEMB 'REVISED NEWLOOKS))
            (SETQ QUADD (LISTGET NEWLOOKS 'QUAD))
            (SETQ COLUMN (LISTGET NEWLOOKS 'COLUMN))
            (SETQ COLUMNSET (FMEMB 'COLUMN NEWLOOKS))
            (SETQ USERINFO (LISTGET NEWLOOKS 'USERINFO))
            (SETQ USERINFOSET (FMEMB 'USERINFO NEWLOOKS))
            (SETQ SPECIALX (LISTGET NEWLOOKS 'SPECIALY))
```

```
(SETQ SPECXSET (FMEMB 'SPECIALY NEWLOOKS))
(SETQ SPECIALY (LISTGET NEWLOOKS 'SPECIALY))
(SETQ SPECYSET (FMEMB 'SPECIALY NEWLOOKS))
(SETQ STYLE (LISTGET NEWLOOKS 'STYLE))
(SETQ STYLESET (FMEMB 'STYLE NEWLOOKS))
(SETQ CHARSTYLES (LISTGET NEWLOOKS 'CHARSTYLES))
(SETQ CHARSTYLESSET (FMEMB 'CHARSTYLES NEWLOOKS))
[SELECTQ QUADD
     ((LEFT RIGHT CENTERED JUSTIFIED NIL)          ; Do nothing -- we got a valid justification spec

         )
     ((JUST J)
         (SETQ QUADD 'JUSTIFIED))
     ((L)
         (SETQQ QUADD LEFT))
     (R (SETQQ QUADD RIGHT))
     ((C CENTER)
         (SETQQ QUADD CENTERED))
     (PROGN                                        ; We got an illegal QUAD value.  Use LEFT.
            (TEDIT.PROMPTPRINT (AND (BOUNDP 'TEXTOBJ)
                                    (EVALV 'TEXTOBJ))
                   (CONCAT "Illegal paragraph quad " QUADD ", replaced with LEFT.")
                   T)
            (SETQ QUADD 'LEFT]
(SETQ TABSPECC (LISTGET NEWLOOKS 'TABS))
```

;; change from the users list to the real tabspec

;; CONS pair of default width and LIST of TAB record instances

```
[COND
   (TABSPECC (SETQ TABSPECC (CONS [OR (CAR TABSPECC)
                                      (AND OLDLOOKS (CAR (fetch (FMTSPEC TABSPEC) of OLDLOOKS]
                                  (for SPEC in (CDR TABSPECC) collect (create TAB
                                                                             TABKIND _ (CDR SPEC)
                                                                             TABX _ (CAR SPEC]
(SETQ NEWLOOKS (create FMTSPEC using (OR OLDLOOKS TEDIT.DEFAULT.FMTSPEC)))
(AND 1STLEFT (replace (FMTSPEC 1STLEFTMAR) of NEWLOOKS with 1STLEFT))
(AND LEFT (replace (FMTSPEC LEFTMAR) of NEWLOOKS with LEFT))
(AND RIGHT (replace (FMTSPEC RIGHTMAR) of NEWLOOKS with RIGHT))
(AND LEADB (replace (FMTSPEC LEADBEFORE) of NEWLOOKS with LEADB))
(AND LEADA (replace (FMTSPEC LEADAFTER) of NEWLOOKS with LEADA))
(AND LLEAD (replace (FMTSPEC LINELEAD) of NEWLOOKS with LLEAD))
(AND TABSPECC (replace (FMTSPEC TABSPEC) of NEWLOOKS with TABSPECC))
(AND QUADD (replace (FMTSPEC QUAD) of NEWLOOKS with QUADD))
(AND TYPESET (replace (FMTSPEC FMTPARATYPE) of NEWLOOKS with TYPE))
(AND SUBTYPESET (replace (FMTSPEC FMTPARASUBTYPE) of NEWLOOKS with SUBTYPE))
(AND NEWBEFORESET (replace (FMTSPEC FMTNEWPAGEBEFORE) of NEWLOOKS with NEWBEFORE))
(AND NEWAFTERSET (replace (FMTSPEC FMTNEWPAGEAFTER) of NEWLOOKS with NEWAFTER))
[AND HEADINGKEEP (replace (FMTSPEC FMTHEADINGKEEP) of NEWLOOKS with (EQ HEADINGKEEP 'ON]
(AND KEEPSET (replace (FMTSPEC FMTKEEP) of NEWLOOKS with KEEP))
(AND BASESET (replace (FMTSPEC FMTBASETOBASE) of NEWLOOKS with BASETOBASE))
(AND REVISEDSET (replace (FMTSPEC FMTREVISED) of NEWLOOKS with REVISED))
(AND COLUMNSET (replace (FMTSPEC FMTCOLUMN) of NEWLOOKS with COLUMN))
(AND SPECXSET (replace (FMTSPEC FMTSPECIALX) of NEWLOOKS with SPECIALX))
(AND SPECYSET (replace (FMTSPEC FMTSPECIALY) of NEWLOOKS with SPECIALY))
(AND STYLESET (replace (FMTSPEC FMTSTYLE) of NEWLOOKS with STYLE))
(AND CHARSTYLESSET (replace (FMTSPEC FMTCHARSTYLES) of NEWLOOKS with CHARSTYLES))
(AND USERINFOSET (replace (FMTSPEC FMTUSERINFO) of NEWLOOKS with USERINFO))
NEWLOOKS])
```

(**TEDIT.PARALOOKS**

```
[LAMBDA (TSTREAM NEWLOOKS SEL LEN)                 ; Edited 16-Mar-2024 21:53 by rmk
                                                   ; Edited 15-Mar-2024 14:23 by rmk
                                                   ; Edited  9-Mar-2024 11:35 by rmk
                                                   ; Edited 24-Feb-2024 12:33 by rmk
                                                   ; Edited  9-Feb-2024 11:41 by rmk
                                                   ; Edited 19-Jan-2024 14:35 by rmk
                                                   ; Edited 29-Dec-2023 15:29 by rmk
                                                   ; Edited 21-Oct-2023 08:55 by rmk
                                                   ; Edited 28-Jul-2023 15:44 by rmk
                                                   ; Edited  6-Jun-2023 21:36 by rmk
                                                   ; Edited 23-May-2023 14:40 by rmk
                                                   ; Edited 21-Apr-93 18:44 by jds
```

;; Apply a given format spec to the paragraphs which are included in this guy. This assumes that paragraph boundaries are aligned with piece
;; boundaries, so no splitting is needed. If we are given a FMTSPEC we replace the FMTSPEC of all pieces in all selected paragraphs. Otherwise,
;; we just override particular values in the selected-paragraph looks.

```
(CL:WHEN (PROG ((TEXTOBJ (TEXTOBJ TSTREAM))
                PARAPIECES REPLACEALLFIELDS 1STLEFT LEFT RIGHT LEADB LEADA BLEAD BLEADSET LLEAD TABSPECC
                QUADD OLDLOOKSLIST TYPE SUBTYPE TYPESET SUBTYPESET SPECIALX SPECIALY NEWBEFORESET NEWBEFORE
                NEWAFTERSET NEWAFTER KEEP KEEPSET HEADINGKEEP BASETOBASE BASESET HCPYMODE HCPYSET USERINFO
                USERSET REVISED REVISEDSET COLUMN COLUMNSET STYLE STYLESET CHARSTYLES CHARSTYLESSET)
               (CL:UNLESS (type? SELECTION SEL)
                   (SETQ SEL (CL:IF (FIXP SEL)
                                 (TEDIT.SETSEL TEXTOBJ SEL LEN 'RIGHT)
                                 (FGETTOBJ TEXTOBJ SEL))))
```

```
(CL:UNLESS (AND (FGETSEL SEL SET)
                (NOT (\TEDIT.READONLY TEXTOBJ)))
       (RETURN NIL))
[COND
   ((type? FMTSPEC NEWLOOKS)                    ; In case it wasn't already uniquified
    (SETQ NEWLOOKS (\TEDIT.UNIQUIFY.PARALOOKS NEWLOOKS TEXTOBJ)))
   (T                                           ; create an FMTSPEC from the Plist
      (SETQ 1STLEFT (LISTGET NEWLOOKS '1STLEFTMARGIN))
      (SETQ LEFT (LISTGET NEWLOOKS 'LEFTMARGIN))
      (SETQ RIGHT (LISTGET NEWLOOKS 'RIGHTMARGIN))
      (SETQ LEADB (LISTGET NEWLOOKS 'PARALEADING))
      (SETQ LEADA (LISTGET NEWLOOKS 'POSTPARALEADING))
      (SETQ LLEAD (LISTGET NEWLOOKS 'LINELEADING))
      (SETQ BLEAD (LISTGET NEWLOOKS 'BASETOBASE))
      (SETQ BLEADSET (FMEMB 'BASETOBASE NEWLOOKS))
      (SETQ QUADD (LISTGET NEWLOOKS 'QUAD))
      (SETQ TYPESET (FMEMB 'TYPE NEWLOOKS))
      (SETQ TYPE (LISTGET NEWLOOKS 'TYPE))
      (SETQ SUBTYPESET (FMEMB 'SUBTYPE NEWLOOKS))
      (SETQ SUBTYPE (LISTGET NEWLOOKS 'SUBTYPE))
      (SETQ SPECIALX (LISTGET NEWLOOKS 'SPECIALX))
      (SETQ SPECIALY (LISTGET NEWLOOKS 'SPECIALY))
      (SETQ NEWBEFORESET (FMEMB 'NEWPAGEBEFORE NEWLOOKS))
      (SETQ NEWBEFORE (LISTGET NEWLOOKS 'NEWPAGEBEFORE))
      (SETQ NEWAFTERSET (FMEMB 'NEWPAGEAFTER NEWLOOKS))
      (SETQ NEWAFTER (LISTGET NEWLOOKS 'NEWPAGEAFTER))
      (SETQ HEADINGKEEP (LISTGET NEWLOOKS 'HEADINGKEEP))
      (SETQ KEEP (LISTGET NEWLOOKS 'KEEP))      ; More general 'Keep-together' spec -- undefined as of 5/22/85
      (SETQ KEEPSET (FMEMB 'KEEP NEWLOOKS))
      (SETQ BASETOBASE (LISTGET NEWLOOKS 'BASETOBASE))
      (SETQ BASESET (FMEMB 'BASETOBASE NEWLOOKS))
      (SETQ HCPYMODE (LISTGET NEWLOOKS 'HARDCOPY))
      (SETQ HCPYSET (FMEMB 'HARDCOPY NEWLOOKS))
      (SETQ USERINFO (LISTGET NEWLOOKS 'USERINFO))
      (SETQ USERSET (FMEMB 'USERINFO NEWLOOKS))
      (SETQ REVISED (LISTGET NEWLOOKS 'REVISED))
      (SETQ REVISEDSET (FMEMB 'REVISED NEWLOOKS))
      (SETQ TABSPECC (LISTGET NEWLOOKS 'TABS))
      (SETQ STYLE (LISTGET NEWLOOKS 'STYLE))
      (SETQ STYLESET (FMEMB 'STYLE NEWLOOKS))
      (SETQ CHARSTYLES (LISTGET NEWLOOKS 'CHARSTYLES))
      (SETQ CHARSTYLESSET (FMEMB 'CHARSTYLES NEWLOOKS))
      (SETQ COLUMN (LISTGET NEWLOOKS 'COLUMN))
      (SETQ COLUMNSET (FMEMB 'COLUMN NEWLOOKS))
      (SETQ STYLE (LISTGET NEWLOOKS 'STYLE))
      (SETQ STYLESET (FMEMB 'STYLE NEWLOOKS]
```

;; The new format specification has been decoded into the different variables.

;; Apply it to the piece that begins the paragraph containing the first selected character, the piece that ends the paragraph containing
;; the last piece of the selection, and all pieces in between.

```
(SETQ PARAPIECES (\TEDIT.PARAPIECES SEL NIL TEXTOBJ))
```

;; Presumably all the pieces within a paragraph have the same looks, and maybe a sequence of paragraphs will have the same looks.
;; Testing LASTFMTSPEC will typically avoid repeated calculation of the same NEWFMTSPEC

```
(for PC LASTFMTSPEC NEWFMTSPEC inselpieces PARAPIECES
   do (push OLDLOOKSLIST (PPARALOOKS PC))     ; Save the old looks of each piece for undoing.
      (if (type? FMTSPEC NEWLOOKS)
          then (FSETPC PC PPARALOOKS NEWLOOKS)
        else (CL:UNLESS (EQ (PPARALOOKS PC)
                            LASTFMTSPEC)
                   ;; We need to instantiate new looks for this piece.
                   (SETQ LASTFMTSPEC (PPARALOOKS PC))
                   (SETQ NEWFMTSPEC (create FMTSPEC using LASTFMTSPEC))
                   (AND 1STLEFT (freplace (FMTSPEC 1STLEFTMAR) of NEWFMTSPEC with 1STLEFT))
                   (AND LEFT (freplace (FMTSPEC LEFTMAR) of NEWFMTSPEC with LEFT))
                   (AND RIGHT (freplace (FMTSPEC RIGHTMAR) of NEWFMTSPEC with RIGHT))
                   (AND LEADB (freplace (FMTSPEC LEADBEFORE) of NEWFMTSPEC with LEADB))
                   (AND LEADA (freplace (FMTSPEC LEADAFTER) of NEWFMTSPEC with LEADA))
                   (AND BLEADSET (freplace (FMTSPEC FMTBASETOBASE) of NEWFMTSPEC with BLEAD))
                   (AND LLEAD (freplace (FMTSPEC LINELEAD) of NEWFMTSPEC with LLEAD))
                   (CL:WHEN TABSPECC
                       ;; change from the users list to the real tabspec --- CONS pair of default width and LIST of TAB
                       ;; record instances
                       [SETQ TABSPECC (CONS [OR (COND
                                                   ((AND (CAR TABSPECC)
                                                         (ZEROP (CAR TABSPECC)))
                                                    1)
                                                   (T (CAR TABSPECC)))
                                                (CAR (fetch (FMTSPEC TABSPEC) of (PPARALOOKS
                                                                                   PC]
                                            (for SPEC in (CDR TABSPECC)
                                               collect (create TAB
                                                               TABKIND _ (CDR SPEC)
                                                               TABX _ (CAR SPEC]
                       (freplace (FMTSPEC TABSPEC) of NEWFMTSPEC with TABSPECC))
```

```
                                (AND QUADD (freplace (FMTSPEC QUAD) of NEWFMTSPEC with QUADD))
                                (AND TYPESET (freplace (FMTSPEC FMTPARATYPE) of NEWFMTSPEC with TYPE))
                                (AND SUBTYPESET (freplace (FMTSPEC FMTPARASUBTYPE) of NEWFMTSPEC with SUBTYPE))
                                (AND SPECIALX (freplace (FMTSPEC FMTSPECIALX) of NEWFMTSPEC with SPECIALX))
                                (AND SPECIALY (freplace (FMTSPEC FMTSPECIALY) of NEWFMTSPEC with SPECIALY))
                                (AND NEWBEFORESET (freplace (FMTSPEC FMTNEWPAGEBEFORE) of NEWFMTSPEC
                                                     with NEWBEFORE))
                                (AND NEWAFTERSET (freplace (FMTSPEC FMTNEWPAGEAFTER) of NEWFMTSPEC with
                                                                                              NEWAFTER
                                                      ))
                                [AND HEADINGKEEP (freplace (FMTSPEC FMTHEADINGKEEP) of NEWFMTSPEC
                                                    with (EQ HEADINGKEEP 'ON]
                                (AND KEEPSET (freplace (FMTSPEC FMTKEEP) of NEWFMTSPEC with KEEP))
                                (AND BASESET (freplace (FMTSPEC FMTBASETOBASE) of NEWFMTSPEC with BASETOBASE))
                                (AND HCPYSET (freplace (FMTSPEC FMTHARDCOPY) of NEWFMTSPEC with HCPYMODE))
                                (AND USERSET (freplace (FMTSPEC FMTUSERINFO) of NEWFMTSPEC with USERINFO))
                                (AND REVISEDSET (freplace (FMTSPEC FMTREVISED) of NEWFMTSPEC with REVISED))
                                (AND STYLESET (freplace (FMTSPEC FMTSTYLE) of NEWFMTSPEC with STYLE))
                                (AND CHARSTYLESSET (freplace (FMTSPEC FMTCHARSTYLES) of NEWFMTSPEC with
                                                                                            CHARSTYLES
                                                        ))
                                (AND COLUMNSET (freplace (FMTSPEC FMTCOLUMN) of NEWFMTSPEC with COLUMN))
                                (AND STYLESET (replace (FMTSPEC FMTSTYLE) of NEWFMTSPEC with STYLE)))
                            (SETQ NEWFMTSPEC (\TEDIT.UNIQUIFY.PARALOOKS NEWFMTSPEC TEXTOBJ))
                            (FSETPC PC PPARALOOKS NEWFMTSPEC)))
                (\TEDIT.HISTORYADD TEXTOBJ (create TEDITHISTORYEVENT
                                               THACTION _ :ParaLooks
                                               THLEN _ (fetch (SELPIECES SPLEN) of PARAPIECES)
                                               THCH# _ (fetch (SELPIECES SPFIRSTCHAR) of PARAPIECES)
                                               THFIRSTPIECE _ (fetch (SELPIECES SPFIRST) of PARAPIECES)
                                               THOLDINFO _ (DREVERSE OLDLOOKSLIST)))
           ;; Pieces have been updated.  Now update any visible lines.

                (CL:WHEN (FGETTOBJ TEXTOBJ \WINDOW)
                    (\TEDIT.SHOWSEL SEL NIL)                  ; Turn off the sel before updating the screen
                    (\TEDIT.MARK.LINES.DIRTY TEXTOBJ PARAPIECES)
                    (CL:UNLESS (AND (LISTP NEWLOOKS)
                                    (EQ 'HARDCOPY (CAR NEWLOOKS))
                                    (NULL (CDDR NEWLOOKS)))

                        ;; The document is "dirty" for the titlebar and saving only if something other than hardcopy-display mode was
                        ;; changed

                        (FSETTOBJ TEXTOBJ \DIRTY T))          ; Save this action for undo/redo
                    (\TEDIT.RESET.EXTEND.PENDING.DELETE SEL TEXTOBJ)
                    (\TEDIT.UPDATE.SCREEN TEXTOBJ)            ; Update the screen image, showing the original selection
                    (\TEDIT.FIXSEL SEL TEXTOBJ)
                    (\TEDIT.SHOWSEL SEL T))])
```

(TEDIT.COPY.PARALOOKS

```
  [LAMBDA (TSTREAM SOURCE DEST)                               ; Edited 17-Mar-2024 00:27 by rmk
                                                              ; Edited  9-Feb-2024 11:39 by rmk
                                                              ; Edited 18-Apr-2023 23:53 by rmk
                                                              ; Edited 22-Oct-2022 15:29 by rmk
                                                              ; Edited 22-Aug-2022 13:15 by rmk
                                                              ; Edited 30-May-91 21:44 by jds

    ;; Copy the PARAGRAPH LOOKS from one place to another

    (PROG ((TEXTOBJ (TEXTOBJ TSTREAM))
           LOOKS LEN)                                         ; get the paragraph looks of the first character of SOURCE
          [SETQ LOOKS (PPARALOOKS (if (FIXP SOURCE)
                                      then (\TEDIT.CHTOPC SOURCE TEXTOBJ)
                                    elseif (type? SELECTION SOURCE)
                                      then (\TEDIT.CHTOPC (fetch (SELECTION CH#) of SOURCE)
                                                  (fetch (SELECTION SELTEXTOBJ) of SOURCE))
                                    else (\ILLEGAL.ARG SOURCE]
          (COND
             ((type? SELECTION DEST)                          ; make sure that the destination selection is in this document
              (CL:UNLESS (EQ TEXTOBJ (fetch (SELECTION SELTEXTOBJ) of DEST))
                     (\LISPERROR "Destination selection is not in stream " TSTREAM)))
             (T                                               ; set the LEN arg for TEDIT.PARALOOKS to be 1 since we just
                                                              ; have a char pos.
                (SETQ LEN 1)))
          (TEDIT.PARALOOKS TEXTOBJ LOOKS DEST LEN])
```

(\TEDIT.PARABOUNDS

```
  [LAMBDA (TEXTOBJ CH#)                                       ; Edited 17-Mar-2024 00:27 by rmk
                                                              ; Edited 26-Mar-2023 12:54 by rmk
                                                              ; Edited 20-Feb-2023 13:55 by rmk
                                                              ; Edited 25-Oct-2022 14:50 by rmk
                                                              ; Edited 22-Aug-2022 13:17 by rmk
                                                              ; Edited 21-Apr-93 18:22 by jds

    ;; Returns the first and last character number of the paragraph that brackets CH#

    (if (ZEROP (TEXTLEN TEXTOBJ))
        then                                                  ; Empty document
```

```
            (CONS 0 0)
      else (LET (CHPIECE START-OF-PIECE START END)
                (DECLARE (SPECVARS START-OF-PIECE))
                (SETQ CHPIECE (\TEDIT.CHTOPC (IMIN CH# (TEXTLEN TEXTOBJ))
                                    TEXTOBJ T))
                (SETQ START START-OF-PIECE)                    ; Find the paragraph's first char
                [for PC backpieces (PREVPIECE CHPIECE) until (PPARALAST PC) do (add START (IMINUS (PLEN PC]
                (SETQ END (SUB1 START-OF-PIECE))               ; Find the paragraph's last char
                (for PC inpieces CHPIECE do (add END (PLEN PC)) repeatuntil (PPARALAST PC))
                (CONS START END])

)

;; For making paragraph-looks substitutions.

(DEFINEQ
```

(TEDIT.SUBPARALOOKS

```
  [LAMBDA (TEXTSTREAM OLDLOOKSLIST NEWLOOKSLIST)               ; Edited 17-Mar-2024 00:27 by rmk
                                                               ; Edited 15-Mar-2024 14:23 by rmk
                                                               ; Edited 18-Apr-2023 23:54 by rmk
                                                               ; Edited 22-Aug-2022 13:13 by rmk
                                                               ; Edited 26-Apr-93 15:13 by jds
```

;;; User entry to substitute one set of looks for another. Goes through the whole textstream and whenever the looks match the characteristics of ;;; OLDLOOKSLIST which are specified, the characteristics listed in NEWLOOKSLIST are substituted.

```
    (LET* ((OLDLOOKS (\TEDIT.PARSE.PARALOOKS.LIST OLDLOOKSLIST))
           (NEWLOOKS (\TEDIT.PARSE.PARALOOKS.LIST NEWLOOKSLIST))
           (TEXTOBJ (TEXTOBJ TEXTSTREAM))
           (SEL (fetch (TEXTOBJ SEL) of TEXTOBJ))
           (FIRSTPC (\TEDIT.CHTOPC 1 TEXTOBJ))
           (FEATURELIST (for A on OLDLOOKSLIST by (CDDR A) collect (CAR A)))
           CHANGEMADE)
          (\TEDIT.SHOWSEL SEL NIL)                             ; Turn off the selection, first.
          [OR (ZEROP (fetch (TEXTOBJ TEXTLEN) of TEXTOBJ))
              (bind (CH# _ 1) for (PC _ FIRSTPC) while PC by (fetch (PIECE NEXTPIECE) of PC)
                 do (COND
                       ((SAMEPARALOOKS OLDLOOKS (fetch (PIECE PPARALOOKS) of PC)
                               FEATURELIST)
                        (replace (TEXTOBJ \DIRTY) of (TEXTOBJ TEXTSTREAM) with T)
                        (freplace (PIECE PPARALOOKS) of PC with (\TEDIT.UNIQUIFY.PARALOOKS
                                                                 (\TEDIT.PARSE.PARALOOKS.LIST NEWLOOKSLIST
                                                                        (fetch (PIECE PPARALOOKS) of PC))
                                                                 (TEXTOBJ TEXTSTREAM)))
                        (\TEDIT.MARK.LINES.DIRTY TEXTOBJ CH# (+ CH# (fetch (PIECE PLEN) of PC)))
                        (SETQ CHANGEMADE T)))
                    (add CH# (fetch (PIECE PLEN) of PC]
          (COND
             ((fetch (TEXTOBJ \WINDOW) of TEXTOBJ)
              (\TEDIT.UPDATE.SCREEN TEXTOBJ)                   ; Update the screen image
              (\TEDIT.FIXSEL SEL TEXTOBJ)
              (\TEDIT.SHOWSEL SEL T)))
          (COND
             (CHANGEMADE 'Done)
             (T 'NoChangesMade])
```

(SAMEPARALOOKS

```
  [LAMBDA (PARALOOK1 PARALOOK2 FEATURES)                       ; Edited  8-Dec-92 00:44 by jds
    ;; Predicate to determine if CLOOK1 and CLOOK2 are the same in all the characteristics listed in FEATURES
    (for F in FEATURES always (SELECTQ F
                                  (STYLE (EQUAL (fetch (FMTSPEC FMTSTYLE) of PARALOOK1)
                                                (fetch (FMTSPEC FMTSTYLE) of PARALOOK2)))
                                  (LEFTMARGIN (IEQP (fetch (FMTSPEC LEFTMAR) of PARALOOK1)
                                                    (fetch (FMTSPEC LEFTMAR) of PARALOOK2)))
                                  (1STLEFTMARGIN
                                       (IEQP (fetch (FMTSPEC 1STLEFTMAR) of PARALOOK1)
                                             (fetch (FMTSPEC 1STLEFTMAR) of PARALOOK2)))
                                  (RIGHTMARGIN (IEQP (fetch (FMTSPEC RIGHTMAR) of PARALOOK1)
                                                     (fetch (FMTSPEC RIGHTMAR) of PARALOOK2)))
                                  (QUAD (EQ (fetch (FMTSPEC QUAD) of PARALOOK1)
                                            (fetch (FMTSPEC QUAD) of PARALOOK2)))
                                  (POSTPARALEADING
                                       (IEQP (fetch (FMTSPEC LEADBEFORE) of PARALOOK1)
                                             (fetch (FMTSPEC LEADBEFORE) of PARALOOK2)))
                                  (PARALEADING (IEQP (fetch (FMTSPEC LEADBEFORE) of PARALOOK1)
                                                     (fetch (FMTSPEC LEADBEFORE) of PARALOOK2)))
                                  (LINELEADING (IEQP (fetch (FMTSPEC LINELEAD) of PARALOOK1)
                                                     (fetch (FMTSPEC LINELEAD) of PARALOOK2)))
                                  (TABS (EQUAL (fetch (FMTSPEC TABSPEC) of PARALOOK1)
                                               (fetch (FMTSPEC TABSPEC) of PARALOOK2)))
                                  (NEWPAGEBEFORE
                                       (EQ (fetch (FMTSPEC FMTNEWPAGEBEFORE) of PARALOOK1)
                                           (fetch (FMTSPEC FMTNEWPAGEBEFORE) of PARALOOK2)))
                                  (NEWPAGEAFTER (EQ (fetch (FMTSPEC FMTNEWPAGEAFTER) of PARALOOK1)
```

```
                                        (fetch (FMTSPEC FMTNEWPAGEAFTER) of PARALOOK2)))
                (SPECIALX (IEQP (fetch (FMTSPEC FMTSPECIALX) of PARALOOK1)
                                (fetch (FMTSPEC FMTSPECIALX) of PARALOOK2)))
                (SPECIALY (IEQP (fetch (FMTSPEC FMTSPECIALY) of PARALOOK1)
                                (fetch (FMTSPEC FMTSPECIALY) of PARALOOK2)))
                (HEADINGKEEP (EQ (fetch (FMTSPEC FMTHEADINGKEEP) of PARALOOK1)
                                 (fetch (FMTSPEC FMTHEADINGKEEP) of PARALOOK2)))
                (ERROR (CONCAT F " is an unknown feature of paragraph looks.  Detected in
                               SAMEPARALOOKS"])

)

;; UNDO & History List stuff

(DEFINEQ

(\TEDIT.UNDO.LOOKS
  [LAMBDA (TEXTOBJ EVENT)                                    ; Edited 15-Mar-2024 14:23 by rmk
                                                             ; Edited 19-Feb-2024 11:32 by rmk
                                                             ; Edited 14-Dec-2023 21:01 by rmk
                                                             ; Edited 30-May-2023 22:56 by rmk
                                                             ; Edited 28-May-2023 00:31 by rmk
                                                             ; Edited  4-May-2023 14:35 by rmk
                                                             ; Edited 18-Apr-2023 23:56 by rmk
                                                             ; Edited 30-May-91 21:44 by jds
    ;; The loop is controlled by the looks, since the pieces are still chained through the text.
    (for PC (CARETPC _ (\TEDIT.CARETPIECE TEXTOBJ))
        (SEL _ (FGETTOBJ TEXTOBJ SEL))
        inpieces
        (fetch THFIRSTPIECE of EVENT) as OLDLOOKS in (GETTH EVENT THOLDINFO)
       collect                                               ; Remember this for the undo.
                                                             ; Give this piece its old looks
              (CL:WHEN (EQ PC CARETPC)
                  (FSETTOBJ TEXTOBJ CARETLOOKS (\TEDIT.CARETLOOKS.VERIFY TEXTOBJ OLDLOOKS)))
              (PROG1 (PLOOKS PC)
                     (FSETPC PC PLOOKS OLDLOOKS))
       finally (SETTH EVENT THOLDINFO $$VAL)                 ; Remember the other looks in case we UNDO the UNDO.
              (\TEDIT.MARK.LINES.DIRTY TEXTOBJ (GETTH EVENT THCH#)
                     (SUB1 (GETTH EVENT THCHLIM)))
              (\TEDIT.UPDATE.SCREEN TEXTOBJ)
              (\TEDIT.SET.SEL.LOOKS SEL 'NORMAL)
              (\TEDIT.FIXSEL SEL TEXTOBJ)
              (\TEDIT.SHOWSEL SEL T))
    (\TEDIT.HISTORYADD TEXTOBJ EVENT])

(\TEDIT.UNDO.PARALOOKS
  [LAMBDA (TEXTOBJ EVENT)                                    ; Edited 15-Mar-2024 14:23 by rmk
                                                             ; Edited 19-Feb-2024 11:32 by rmk
                                                             ; Edited 11-Dec-2023 11:10 by rmk
                                                             ; Edited 21-Sep-2023 23:51 by rmk
                                                             ; Edited 30-May-2023 22:55 by rmk
                                                             ; Edited  4-May-2023 14:35 by rmk
                                                             ; Edited 18-Apr-2023 23:57 by rmk
                                                             ; Edited 30-May-91 21:44 by jds
    ;; Undo the setting of paragraph looks.
    (for PC (SEL _ (GETTOBJ TEXTOBJ SEL))
        inpieces
        (fetch THFIRSTPIECE of EVENT) as OLDLOOKS in (fetch THOLDINFO of EVENT) do (FSETPC PC PPARALOOKS OLDLOOKS)

                                                             ; Give this piece its old looks
       finally ;; Remember the current looks in case we UNDO the UNDO.
              (replace THOLDINFO of EVENT with $$VAL)
              (\TEDIT.MARK.LINES.DIRTY TEXTOBJ (fetch THCH# of EVENT)
                     (IPLUS (fetch THCH# of EVENT)
                            (fetch THLEN of EVENT)
                            -1))
              (\TEDIT.UPDATE.SCREEN TEXTOBJ)
              (\TEDIT.SET.SEL.LOOKS SEL 'NORMAL)
              (\TEDIT.FIXSEL SEL TEXTOBJ)
              (\TEDIT.SHOWSEL SEL T))
    (\TEDIT.HISTORYADD TEXTOBJ EVENT])

)

;; Revision-mark support

(DEFINEQ

(\TEDIT.MARK.REVISION
  [LAMBDA (TEXTOBJ FMTSPEC IMAGESTREAM LINE)                 ; Edited 27-May-2023 12:12 by rmk
                                                             ; Edited 30-May-91 21:38 by jds
    (LET ((SCALE (DSPSCALE NIL IMAGESTREAM)))
```

```
         (BLTSHADE BLACKSHADE IMAGESTREAM (IPLUS (GETLD LINE RIGHTMARGIN LINE)
                                                (FIXR (ITIMES 12 SCALE)))
                (GETLD LINE YBOT)
                (FIXR SCALE)
                (GETLD LINE LHEIGHT)
                'PAINT])

)

;; Added by yabu.fx, for SUNLOADUP without DWIM

(DEFINEQ

(\CREATE.TEDIT.DEFAULT.FMTSPEC
  [LAMBDA NIL                                                    ; Edited 24-Aug-2023 23:31 by rmk
    (create FMTSPEC
            QUAD _ 'LEFT
            1STLEFTMAR _ 0
            LEFTMAR _ 0
            RIGHTMAR _ 0
            LEADBEFORE _ 0
            LEADAFTER _ 0
            LINELEAD _ 0
            FMTSPECIALX _ 0
            FMTSPECIALY _ 0
            TABSPEC _ (CONS DEFAULTTAB NIL])

(\CREATE.TEDIT.FACE.MENU
  [LAMBDA NIL
    (create MENU
            ITEMS _ '(Bold Italic Bold% Italic Regular)
            CENTERFLG _ T
            TITLE _ "Face:"])

(\CREATE.TEDIT.SIZE.MENU
  [LAMBDA NIL
    (create MENU
            ITEMS _ '(6 7 8 9 10 11 12 14 18 24 30 36)
            CENTERFLG _ T
            MENUROWS _ 4
            TITLE _ "Type Size:"])

)

;; Style-sheet support

(DEFINEQ

(\TEDIT.APPLY.STYLES
  [LAMBDA (LOOKS PC TSTREAM)                                     ; Edited 12-Nov-2023 16:08 by rmk
                                                                 ; Edited 18-Mar-2023 21:45 by rmk
                                                                 ; Edited 25-Sep-2022 13:28 by rmk
                                                                 ; Edited 11-Sep-2022 14:45 by rmk
                                                                 ; Edited  4-Jul-93 01:02 by sybalskY:MV:ENVOS
    ;; Given a set of looks, return the looks with the proper styles expanded out.
    (SETQ TSTREAM (TEXTSTREAM TSTREAM))
    (OR (CDR (ASSOC LOOKS *TEDIT-CURRENTPARA-CACHE*))
        (CDR (ASSOC LOOKS *TEDIT-PARASTYLE-CACHE*))
        (LET* ((TEXTOBJ (TEXTOBJ TSTREAM))
               (STYLE (fetch (CHARLOOKS CLSTYLE) of LOOKS))
               (STYLE-SHEET (OR (FGETTOBJ TEXTOBJ TXTSTYLESHEET)
                                TEDIT.STYLES))
               NOSTYLE CHARSTYLES CHARSTYLE IN-PARA FMTSPEC)
              (SETQ STYLE (COND
                             ((NULL STYLE)                       ; STYLE of NIL means don't bother.  Just use the looks we got.
                              (SETQ NOSTYLE T)
                              LOOKS)
                             ((AND [SETQ CHARSTYLES (AND (fetch (TEXTSTREAM CURRENTPARALOOKS) of TSTREAM)
                                                         (fetch (FMTSPEC FMTCHARSTYLES)
                                                            of (fetch (TEXTSTREAM CURRENTPARALOOKS) of TSTREAM]
                                   (SETQ CHARSTYLE (FASSOC STYLE CHARSTYLES)))
                                                                 ; If the paragraph we're in has character styles, and this is one of
                                                                 ; them, use it.
                              (SETQ IN-PARA T)
                              CHARSTYLE)
                             ((CDR (SASSOC STYLE STYLE-SHEET)))
                             ((AND (LITATOM STYLE)
                                   (DEFINEDP STYLE))             ; Call the guy's function to find the new looks
                              (APPLY* STYLE LOOKS PC TEXTOBJ))
                             (T                                  ; If all else fails, return the original set of looks
                                (SETQ NOSTYLE T)
                                LOOKS)))
              (SETQ STYLE (COND
```

```
                        ((LISTP STYLE)
                         (\TEDIT.PARSE.CHARLOOKS.LIST (APPEND STYLE '(STYLE NIL))
                                LOOKS TEXTOBJ))
                        (T STYLE)))
```

;; Cache the looks->styled-looks mapping, either in the cache for this kind of paragraph (which gets wiped when we hit a new para ;; type), or in the global cache.

```
[OR NOSTYLE (CL:IF IN-PARA
                   (push *TEDIT-CURRENTPARA-CACHE* (CONS LOOKS STYLE))
                   (push *TEDIT-PARASTYLE-CACHE* (CONS LOOKS STYLE)))]
STYLE])
```

(\TEDIT.APPLY.PARASTYLES

```
[LAMBDA (PARALOOKS PC TEXTOBJ)                                  ; Edited 4-Mar-2023 22:23 by rmk
                                                                ; Edited 25-Sep-2022 13:26 by rmk
                                                                ; Edited 3-Jul-93 23:15 by sybalskY:MV:ENVOS
```

;; Given a set of looks, return the looks with the proper styles expanded out.

```
(\TEDIT.CHECK (type? FMTSPEC PARALOOKS))                        ; Incoming thing has to be a LOOKS.
(OR (CDR (ASSOC PARALOOKS *TEDIT-PARASTYLE-CACHE*))
    (LET* [NOSTYLE (STYLE-SHEET (OR (fetch (TEXTOBJ TXTSTYLESHEET) of TEXTOBJ)
                                    TEDIT.STYLES))
                   (STYLE (COND
                             ((NULL (fetch (FMTSPEC FMTSTYLE) of PARALOOKS))
                              (SETQ NOSTYLE T)
                              PARALOOKS)
                             ((CDR (SASSOC (fetch (FMTSPEC FMTSTYLE) of PARALOOKS)
                                           STYLE-SHEET)))
                             ((AND (LITATOM (fetch (FMTSPEC FMTSTYLE) of PARALOOKS))
                                   (DEFINEDP (fetch (FMTSPEC FMTSTYLE) of PARALOOKS)))
                                                                ; Call the guy's function to find the new looks
                              (APPLY* (fetch (FMTSPEC FMTSTYLE) of PARALOOKS)
                                      PARALOOKS PC TEXTOBJ))
                             (T (SETQ NOSTYLE T)
                                PARALOOKS]
          (SETQ STYLE (COND
                         ((LISTP STYLE)
                          (\TEDIT.PARSE.PARALOOKS.LIST (APPEND STYLE '(STYLE NIL))
                                 PARALOOKS))
                         (T STYLE)))
          (CL:UNLESS NOSTYLE
              (push *TEDIT-PARASTYLE-CACHE* (CONS PARALOOKS STYLE)))
          STYLE])
```

(TEDIT.STYLESHEET

```
[LAMBDA (SHEET TEXTSTREAM)                                      ; Edited 3-Jul-93 23:19 by sybalskY:MV:ENVOS
```

;; Put a new stylesheet into force. This REPLACES any existing style sheets, and forgets any pushed sheets.

```
(LET [(TEXTOBJ (AND TEXTSTREAM (TEXTOBJ TEXTSTREAM]
     (COND
        (TEXTOBJ (SETQ *TEDIT-PARASTYLE-CACHE* NIL)             ; Clear the cache, to force reformatting
                 (replace (TEXTOBJ TXTSTYLESHEET) of TEXTOBJ with SHEET))
        (T ;; No specific document given; change the global style sheet TEDIT.STYLES
           (SETQ *TEDIT-PARASTYLE-CACHE* NIL)                   ; Clear the cache, to force reformatting
           (SETQ TEDIT.STYLES SHEET)
           (SETQ *TEDIT-STYLESHEET-SAVE-LIST* (LIST TEDIT.STYLES])
```

(TEDIT.POP.STYLESHEET

```
[LAMBDA NIL                                                     ; Edited 3-Jul-93 17:42 by sybalskY:MV:ENVOS
```

;; Go back to an earlier stylesheet, by popping the stack of saved sheets. You can't pop back to no sheet -- you'll always bottom out at the original ;; style sheet.

```
(SETQ *TEDIT-PARASTYLE-CACHE* NIL)                              ; Clear the cache, to force reformatting
(SETQ TEDIT.STYLES (OR (CL:POP *TEDIT-STYLESHEET-SAVE-LIST*)
                       TEDIT.STYLES])
```

(TEDIT.PUSH.STYLESHEET

```
[LAMBDA (SHEET)                                                 ; Edited 3-Jul-93 17:40 by sybalskY:MV:ENVOS
```

;; Add more style definitions to the current style sheet, and remember how to get back to the old one. Think of this as PUSHING onto a stack of ;; stylesheets, with the new sheet being a composition of SHEET and the existing styles.

```
(SETQ *TEDIT-PARASTYLE-CACHE* NIL)                              ; Clear the cache, to force reformatting
(SETQ TEDIT.STYLES (APPEND SHEET TEDIT.STYLES))
(CL:PUSH TEDIT.STYLES *TEDIT-STYLESHEET-SAVE-LIST*])
```

(TEDIT.ADD.STYLESHEET

```
[LAMBDA (SHEET)                                                 ; Edited 3-Jul-93 17:38 by sybalskY:MV:ENVOS
```

;; Add more style definitions to the current style sheet. This ADDS entries, without remembering that there was an earlier sheet.

```
(SETQ *TEDIT-PARASTYLE-CACHE* NIL)                              ; Clear the cache, to force reformatting
(SETQ TEDIT.STYLES (APPEND SHEET TEDIT.STYLES))
```

```
    (SETQ *TEDIT-STYLESHEET-SAVE-LIST* (LIST TEDIT.STYLES])

)
```

;; *TEDIT-PARASTYLE-CACHE* is an ALIST of original char/para looks to styled char/para looks. It is used to cache stylings, and is reset when the ;; main stylesheet changes, and when we change paragraph looks, given paras that have private char styles.

;; *TEDIT-CURRENTPARA-CACHE* is NIL if we're not in a para that has private char styles, or is the FMTSPEC (styled!) for that para, if we are. Used ;; to decide when we have to flush *TEDIT-PARASTYLE-CACHE* at paragraph boundaries. Mostly, this'll be NIL and not interesting.

;; *TEDIT-STYLESHEET-SAVE-LIST* is a list of points inside TEDIT.STYLES, so we can "push" new style sheets on the front, and "pop" them off ;; sensibly. This is the push-stack, in effect. Used by TEDIT.ADD.STYLESHEET, TEDIT.PUSH.STYLESHEET, and TEDIT.POP.STYLESHEET

```
(RPAQ? TEDIT.STYLES )
```

;; RMK 2023: Maybe this should be one of the later ones? Only partly implemented

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.STYLES)
)

(RPAQ? *TEDIT-PARASTYLE-CACHE* )

(RPAQ? *TEDIT-CURRENTPARA-CACHE* )

(RPAQ? *TEDIT-STYLESHEET-SAVE-LIST* )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *TEDIT-PARASTYLE-CACHE* *TEDIT-CURRENTPARA-CACHE* *TEDIT-STYLESHEET-SAVE-LIST*)
)
```

FUNCTION INDEX

CHARLOOKS.FROM.FONT 4
SAMEPARALOOKS 22
TEDIT.ADD.STYLESHEET 25
TEDIT.CARETLOOKS 6
TEDIT.COPY.LOOKS 6
TEDIT.COPY.PARALOOKS 21
TEDIT.FINDLOOKS 14
TEDIT.GET.LOOKS 13
TEDIT.GET.PARALOOKS 17
TEDIT.LOOKS 12
TEDIT.MODIFYLOOKS 7
TEDIT.NEW.FONT 7
TEDIT.PARALOOKS 19
TEDIT.POP.STYLESHEET 25
TEDIT.PUSH.STYLESHEET 25
TEDIT.STYLESHEET 25
TEDIT.SUBLOOKS 13
TEDIT.SUBPARALOOKS 22
\CREATE.TEDIT.DEFAULT.FMTSPEC ...24
\CREATE.TEDIT.FACE.MENU 24
\CREATE.TEDIT.SIZE.MENU 24
\TEDIT.APPLY.PARASTYLES 25
\TEDIT.APPLY.STYLES 24
\TEDIT.CARETLOOKS.VERIFY 7
\TEDIT.CHANGE.LOOKS 14
\TEDIT.CHARLOOKS.DEFPRINT 3
\TEDIT.CONVERT.TO.FORMATTED 10
\TEDIT.EQCLOOKS 5
\TEDIT.EQFMTSPEC 17
\TEDIT.FLUSH.UNUSED.LOOKS 12
\TEDIT.FMTSPEC.DEFPRINT 4
\TEDIT.FONTCOPY 16
\TEDIT.GET.INSERT.CHARLOOKS 7
\TEDIT.GET.TERMSA.WIDTHS 8
\TEDIT.LOOKS 16
\TEDIT.MARK.REVISION 23
\TEDIT.PARABOUNDS 21
\TEDIT.PARSE.CHARLOOKS.LIST 8
\TEDIT.PARSE.PARALOOKS.LIST 18
\TEDIT.SAMECLOOKS 5
\TEDIT.TRANSLATE.ASCIICHARS 9
\TEDIT.UNDO.LOOKS 23
\TEDIT.UNDO.PARALOOKS 23
\TEDIT.UNIQUIFY.ALL 12
\TEDIT.UNIQUIFY.CHARLOOKS 11
\TEDIT.UNIQUIFY.PARALOOKS 12
\TEDIT.UNPARSE.CHARLOOKS.LIST 6

VARIABLE INDEX

TEDIT-CURRENTPARA-CACHE 26
TEDIT-PARASTYLE-CACHE 26
TEDIT-STYLESHEET-SAVE-LIST 26
FONTVARS 4
TEDIT.CHARLOOKS.FEATURES 4
TEDIT.DEFAULT.FMTSPEC 4
TEDIT.DEFAULT.FOLIO 4
TEDIT.FACE.MENU 4
TEDIT.KNOWN.FONTS 4
TEDIT.SIZE.MENU 4
TEDIT.STYLES 26
TEDIT.TERMSA.FONTS 4

MACRO INDEX

ONOFF 3
\TEDIT.TRANSLATE.ASCII.CHARLOOKS 11
\WORDSETA 3

RECORD INDEX

CHARLOOKS 2
FMTSPEC 2