File created: 11-Sep-2022 20:07:43 **{DSK}<home>larry>medley>sources>IOCHAR.;2**

changes to: (VARS IOCHARCOMS)

previous date: 24-Jul-2022 14:56:20 {DSK}<home>larry>medley>sources>IOCHAR.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ IOCHARCOMS
       [[COMS (FNS CHCON UNPACK DCHCON DUNPACK)
              (FNS UALPHORDER ALPHORDER CONCAT CONCATCODES PACKC PACK PACK* \PACK.ITEM STRPOS)
              (FUNCTIONS XCL:PACK XCL:PACK*)
              (GLOBALVARS \SIGNFLAG \PRINTRADIX)
              (EXPORT (DECLARE%: DONTCOPY (MACROS \CATRANSLATE]
        (COMS (FNS STRPOSL MAKEBITTABLE)
              (DECLARE%: DONTCOPY (RESOURCES \STRPOSLARRAY))
              (INITRESOURCES \STRPOSLARRAY))
        (COMS (FNS CASEARRAY UPPERCASEARRAY)
              (P (MOVD? 'SETA 'SETCASEARRAY)
                 (MOVD? 'ELT 'GETCASEARRAY))
              [DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (\TRANSPARENT (CASEARRAY))
                                                    (UPPERCASEARRAY (UPPERCASEARRAY]
              (DECLARE%: EVAL@COMPILE (PROP GLOBALVAR UPPERCASEARRAY)
                     DONTCOPY
                     (GLOBALVARS \TRANSPARENT)))
        (COMS (FNS FILEPOS FFILEPOS \SETUP.FFILEPOS \SLOWFILEPOS)
              (DECLARE%: EVAL@COMPILE DONTCOPY (RESOURCES \FFDELTA1 \FFDELTA2 \FFPATCHAR)
                     (CONSTANTS (\MAX.PATTERN.SIZE 128)
                            (\MIN.PATTERN.SIZE 3)
                            (FILEPOS.SEGMENT.SIZE 32768)
                            (\MIN.SEARCH.LENGTH 100)))
              (INITRESOURCES \FFDELTA1 \FFDELTA2 \FFPATCHAR))
        [COMS ;; DATE Functions
              (FNS DATE DATEFORMAT GDATE IDATE \IDATESCANTOKEN \IDATE-PARSE-MONTH \OUTDATE \OUTDATE-STRING
                   \RPLRIGHT \UNPACKDATE \PACKDATE \DTSCAN \ISDST? \CHECKDSTCHANGE)
              (OPTIMIZERS DATEFORMAT)
              ;; Default values for \BeginDST and \EndDST are set for (most places in) the U.S., 74 and 312 as of 2021.
              ;;  Note: this might not be relevant to users with local time servers that do the right thing.
              (INITVARS (\TimeZoneComp 8)
                     (\BeginDST 74)
                     (\EndDST 312)
                     (\DayLightSavings T))
              (ADDVARS (TIME.ZONES (8 "PST" "PDT")
                                   (7 "MST" "MDT")
                                   (6 "CST" "CDT")
                                   (5 "EST" "EDT")
                                   (0 "GMT" "BST")
                                   (0 "UT")
                                   (-1 "MET" "MET DST")
                                   (-2 "EET" "EET DST")))
              (DECLARE%: EVAL@COMPILE DONTCOPY (GLOBALVARS \TimeZoneComp \BeginDST \EndDST \DayLightSavings
                                                           TIME.ZONES)
                     (CONSTANTS (\4YearsDays (ADD1 (ITIMES 365 4]
        (LOCALVARS . T)
        (PROP FILETYPE IOCHAR)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA DATEFORMAT)
                                                                               (NLAML)
                                                                               (LAMA PACK* CONCAT])

(DEFINEQ

(CHCON
  [LAMBDA (X FLG RDTBL)                                          (* bvm%: "24-Mar-86 16:29")
    (PROG (BASE OFFST LEN \CHCONLST \CHCONLSTAIL FATP)
          (COND
             (FLG (GO SLOWCASE)))
          (COND
             ((LITATOM X)
              (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
              (SETQ OFFST 1)
              (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X))
              (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X)))
             ((STRINGP X)
              (SETQ BASE (ffetch (STRINGP BASE) of X))
              (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X))
              (SETQ OFFST (ffetch (STRINGP OFFST) of X))
              (SETQ LEN (ffetch (STRINGP LENGTH) of X)))
```

```
            (T (GO SLOWCASE)))
         (RETURN (for I from OFFST to (IPLUS OFFST LEN -1) collect (\GETBASECHAR FATP BASE I)))
      SLOWCASE
         (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)                ; Open code COLLECT
                                 (COND
                                    [\CHCONLSTAIL (FRPLACD \CHCONLSTAIL (SETQ \CHCONLSTAIL (LIST CODE]
                                    (T (SETQ \CHCONLST (SETQ \CHCONLSTAIL (LIST CODE]
                X FLG RDTBL)
         (RETURN \CHCONLST])
```

(UNPACK

```
  [LAMBDA (X FLG RDTBL)                                          (* bvm%: "24-Mar-86 16:29")
    (PROG (BASE OFFST LEN \CHCONLST \CHCONLSTAIL FATP)
          (COND
             (FLG (GO SLOWCASE)))
          (COND
             ((LITATOM X)
              (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
              (SETQ OFFST 1)
              (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X))
              (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X)))
             ((STRINGP X)
              (SETQ BASE (ffetch (STRINGP BASE) of X))
              (SETQ OFFST (ffetch (STRINGP OFFST) of X))
              (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X))
              (SETQ LEN (ffetch (STRINGP LENGTH) of X)))
             (T (GO SLOWCASE)))
          [RETURN (for I from OFFST to (IPLUS OFFST LEN -1) collect (FCHARACTER (\GETBASECHAR FATP BASE I]
      SLOWCASE
          (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                                 (SETQ CODE (FCHARACTER CODE))    ; Open code COLLECT
                                 (COND
                                    [\CHCONLSTAIL (FRPLACD \CHCONLSTAIL (SETQ \CHCONLSTAIL (LIST CODE]
                                    (T (SETQ \CHCONLST (SETQ \CHCONLSTAIL (LIST CODE]
                X FLG RDTBL)
          (RETURN \CHCONLST])
```

(DCHCON

```
  [LAMBDA (X SCRATCHLIST FLG RDTBL)                              ; Edited 24-Dec-86 14:04 by jds
```

;;; Unpack the character codes that make up the print-representation of X into the scratch list SCRATCHLIST. If FLG, use the PRIN2-pname. Do the ;;; printing according to RDTBL readtable, if supplied.

```
    (SCRATCHLIST SCRATCHLIST (PROG (BASE OFFST LEN FATP)
                                   (COND
                                      (FLG (GO SLOWCASE)))
                                   (COND
                                      ((LITATOM X)                ; LITATOM case:  Set up the indexing info for the
                                                                  ; \GETBASECHAR loop below.
                                       (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
                                       (SETQ OFFST 1)
                                       (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X))
                                       (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X)))
                                      ((STRINGP X)                ; STRING case:  Set up the indexing info for the
                                                                  ; \GETBASECHAR loop below.
                                       (SETQ BASE (ffetch (STRINGP BASE) of X))
                                       (SETQ OFFST (ffetch (STRINGP OFFST) of X))
                                       (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X))
                                       (SETQ LEN (ffetch (STRINGP LENGTH) of X)))
                                      (T (GO SLOWCASE)))
                                   [RETURN (for I from OFFST to (IPLUS OFFST LEN -1)
                                              do ;; Copy the characters from the string/atom-pname into the list
                                                 (ADDTOSCRATCHLIST (\GETBASECHAR FATP BASE I]
                                 SLOWCASE

                                 ;; Slow case:  Use \MAPPNAME to generate the characters, and grab onto them.
                                   (RETURN (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                                                                  (ADDTOSCRATCHLIST CODE]
                                                X FLG RDTBL])
```

(DUNPACK

```
  [LAMBDA (X SCRATCHLIST FLG RDTBL)                              (* bvm%: "24-Mar-86 16:30")
    (SCRATCHLIST SCRATCHLIST (PROG (BASE OFFST LEN FATP)
                                   (COND
                                      (FLG (GO SLOWCASE)))
                                   (COND
                                      ((LITATOM X)
                                       (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
                                       (SETQ OFFST 1)
                                       (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X))
                                       (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X)))
                                      ((STRINGP X)
```

```
                                    (SETQ BASE (ffetch (STRINGP BASE) of X))
                                    (SETQ OFFST (ffetch (STRINGP OFFST) of X))
                                    (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X))
                                    (SETQ LEN (ffetch (STRINGP LENGTH) of X)))
                                   (T (GO SLOWCASE)))
                                [RETURN (for I from OFFST to (IPLUS OFFST LEN -1)
                                           do (ADDTOSCRATCHLIST (FCHARACTER (\GETBASECHAR FATP BASE I]
                             SLOWCASE
                                 (RETURN (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                                                                (ADDTOSCRATCHLIST (FCHARACTER CODE]
                                                    X FLG RDTBL])

)

(DEFINEQ

(UALPHORDER
  [LAMBDA (ARG1 B)                                                    (* rmk%: " 2-Apr-85 11:20")
    (ALPHORDER ARG1 B UPPERCASEARRAY])

(ALPHORDER
  [LAMBDA (A B CASEARRAY)                                             (* rmk%: "27-Mar-85 17:43")
    (DECLARE (GLOBALVARS \TRANSPARENT))
    (PROG (CABASE ABASE ALEN AOFFSET AFATP BBASE BLEN BOFFSET BFATP C1 C2)
          [COND
             ((LITATOM A)
              (SETQ ABASE (ffetch (LITATOM PNAMEBASE) of A))
              (SETQ AOFFSET 1)
              (SETQ ALEN (ffetch (LITATOM PNAMELENGTH) of A))
              (SETQ AFATP (ffetch (LITATOM FATPNAMEP) of A)))
             ((STRINGP A)
              (SETQ ABASE (ffetch (STRINGP BASE) of A))
              (SETQ AOFFSET (ffetch (STRINGP OFFST) of A))
              (SETQ ALEN (ffetch (STRINGP LENGTH) of A))
              (SETQ AFATP (ffetch (STRINGP FATSTRINGP) of A)))
             (T (RETURN (COND
                           [(NUMBERP A)                               ; Numbers are less than all other types
                            (OR (NOT (NUMBERP B))
                                (NOT (GREATERP A B]
                           ((OR (NUMBERP B)
                                (LITATOM B)
                                (STRINGP B))
                            NIL)
                           (T T]
          [COND
             ((LITATOM B)
              (SETQ BBASE (ffetch (LITATOM PNAMEBASE) of B))
              (SETQ BOFFSET 1)
              (SETQ BLEN (ffetch (LITATOM PNAMELENGTH) of B))
              (SETQ BFATP (ffetch (LITATOM FATPNAMEP) of B)))
             ((STRINGP B)
              (SETQ BBASE (ffetch (STRINGP BASE) of B))
              (SETQ BOFFSET (ffetch (STRINGP OFFST) of B))
              (SETQ BLEN (ffetch (STRINGP LENGTH) of B))
              (SETQ BFATP (ffetch (STRINGP FATSTRINGP) of B)))
             (T                                                       ; Only numbers are 'less than' atoms and strings
                (RETURN (NOT (NUMBERP B]
          [SETQ CABASE (fetch (ARRAYP BASE) of (SETQ CASEARRAY (\DTEST (OR CASEARRAY \TRANSPARENT)
                                                                       'ARRAYP]
          (RETURN (for I (CAFAT _ (EQ \ST.POS16 (fetch (ARRAYP TYP) of CASEARRAY)))
                     (CASIZE _ (fetch (ARRAYP LENGTH) of CASEARRAY)) from 0
                  do (COND
                        [(IGEQ I ALEN)
                         (RETURN (COND
                                    ((EQ ALEN BLEN)
                                     'EQUAL)
                                    (T 'LESSP]
                        ((IGEQ I BLEN)
                         (RETURN NIL))
                        [(EQ [SETQ C1 (\CATRANSLATE CABASE CASIZE CAFAT (\GETBASECHAR AFATP ABASE
                                                                               (IPLUS I AOFFSET]
                             (SETQ C2 (\CATRANSLATE CABASE CASIZE CAFAT (\GETBASECHAR BFATP BBASE
                                                                               (IPLUS I BOFFSET]
                        ((ILESSP C1 C2)
                         (RETURN 'LESSP))
                        (T                                           ; Greater
                           (RETURN NIL])

(CONCAT
  [LAMBDA N                                                           (* rmk%: "26-Mar-85 19:08")
    (PROG ((J N)
           (LEN 0)
           (POS 1)
           S NM FATSEENP)
      L1   (COND
```

```
            ((NEQ J 0)
             [COND
                [(STRINGP (SETQ NM (ARG N J)))
                 (OR FATSEENP (SETQ FATSEENP (ffetch (STRINGP FATSTRINGP) of NM]
                [(LITATOM NM)
                 (OR FATSEENP (SETQ FATSEENP (ffetch (LITATOM FATPNAMEP) of NM]
                (T (SETARG N J (SETQ NM (MKSTRING NM)))
                   (OR FATSEENP (SETQ FATSEENP (ffetch (STRINGP FATSTRINGP) of NM]
             (SETQ LEN (IPLUS LEN (NCHARS NM)))
             (SETQ J (SUB1 J))
             (GO L1)))
          (SETQ S (ALLOCSTRING LEN NIL NIL FATSEENP))
      L2  (COND
             ((NEQ J N)
              (SETQ J (ADD1 J))
              (RPLSTRING S POS (ARG N J))
              [SETQ POS (IPLUS POS (NCHARS (ARG N J]
              (GO L2)))
          (RETURN S])
```

(CONCATCODES

```
[LAMBDA (CHARCODES)                                          (* bvm%: " 6-May-84 21:56")
  (PROG [(STR (ALLOCSTRING (LENGTH CHARCODES]
        (for X in CHARCODES as I from 1 do (RPLCHARCODE STR I X))
        (RETURN STR])
```

(PACKC

```
[LAMBDA (X)                                                  ; Edited 11-Nov-2018 12:12 by rmk:
                                                             (* rmk%: "11-Apr-85 15:35")
```

;; Takes character codes in X, stuffs them into the \PNAMESTRING, and then calls \MKATOM.

;; The previous version uses HASFAT as the storage format even if the characters turned out to be all thin. For unknown reasons, this caused ;; existing atoms not to be matched if they had non-ascii thin characters, even

;; though \MKATOM tried to figure out what the truth.

;; But that was a bad optimization, involved an extra pass in every case. Better to start by assuming thin (0-255) characters and store them as ;; bytes, then upgrade the storage format when the first fat code is seen. No extra work for the most common 0-255. If a code is outside of that ;; range (e.g. Japanese), chances are that it will appear early in the sequence, so little work to be done to expand the storage format for previously ;; stored characters.

;; The end-result: the storage format and characters are always consistent, HASFAT is accurate for both, and \MKATOM doesn't have to ;; second-guess.

;; Note: after init, the code for \MKATOM is in PACKAGE-STARTUP

```
  (WITH-RESOURCE (\PNAMESTRING)
         (BIND HASFAT (PBASE _ (ffetch (STRINGP XBASE) of \PNAMESTRING)) for N from 0 as C in X
            do (AND (IGREATERP N \PNAMELIMIT)
                    (LISPERROR "ATOM TOO LONG"))
               (IF HASFAT
                   THEN ;; We already saw a fat, and upgraded the storage format. Continue
                        (\PUTBASEFAT PBASE N C)
                 ELSEIF (ILEQ C \MAXTHINCHAR)
                   THEN ;; Still seeing only thin characters. Continue
                        (\PUTBASETHIN PBASE N C)
                   ELSE ;; First fat, perhaps there are previous thins to convert. Go backwards so we don't smash the early ones
                        (for NN from (SUB1 N) to 0 by -1 DO (\PUTBASEFAT PBASE NN (\GETBASETHIN PBASE NN)))
                        (\PUTBASEFAT PBASE N C)
                        (SETQ HASFAT T))
            finally (RETURN (\MKATOM PBASE 0 N HASFAT])
```

(PACK

```
[LAMBDA (X)                                                  ; Edited 21-Mar-88 15:29 by bvm
  (AND X (NLISTP X)
       (\ILLEGAL.ARG X))
  (DECLARE (SPECVARS PACK.INDEX \PNAMESTRING))
  (WITH-RESOURCE (\PNAMESTRING)
         (PROG ((PACK.INDEX 1)
                ITEM)
           LP   [COND
                   ((NULL X)
                    (RETURN (\MKATOM (fetch (STRINGP XBASE) of \PNAMESTRING)
                                    0
                                    (SUB1 PACK.INDEX)
                                    \FATPNAMESTRINGP]
                (COND
                   ((OR (STRINGP (SETQ ITEM (CAR X)))
                        (LITATOM ITEM))
                    (RPLSTRING \PNAMESTRING (PROG1 PACK.INDEX
                                                   (AND (IGREATERP (add PACK.INDEX (NCHARS ITEM))
                                                                   (ADD1 \PNAMELIMIT))
                                                        (LISPERROR "ATOM TOO LONG")))
                            ITEM))
```

```
                  (T (\PACK.ITEM ITEM)))
               (SETQ X (LISTP (CDR X)))
               (GO LP])
```

(PACK*

```
[LAMBDA U                                                    ; Edited 21-Mar-88 15:29 by bvm
  (DECLARE (SPECVARS PACK.INDEX \PNAMESTRING))
  (WITH-RESOURCE (\PNAMESTRING)
         (PROG ((PACK.INDEX 1)
                (M 1)
                ITEM)
           LP   [COND
                   ((IGREATERP M U)
                    (RETURN (\MKATOM (fetch (STRINGP XBASE) of \PNAMESTRING)
                                    0
                                    (SUB1 PACK.INDEX)
                                    \FATPNAMESTRINGP]
               (SETQ ITEM (ARG U M))
               (COND
                  [(AND (NULL *PACKAGE*)
                        (LITATOM ITEM))
                   ;; If we're in that nasty region of the INIT process before packages have been turned on, then we want to be careful to strip
                   ;; off any pseudo-package prefixes in the symbol's pname.  We use the utility NAMESTRING-CONVERSION-CLAUSE from
                   ;; LLPACKAGE for this search.
                   (LET* ((BASE (ffetch (CL:SYMBOL PNAMEBASE) of ITEM))
                          (LEN (ffetch (CL:SYMBOL PNAMELENGTH) of ITEM))
                          (FATP (ffetch (CL:SYMBOL FATPNAMEP) of ITEM))
                          (CLAUSE (NAMESTRING-CONVERSION-CLAUSE BASE 1 LEN FATP)))
                         (COND
                            ((NULL CLAUSE)                   ; Nothing special to do; this symbol didn't match any of the
                                                             ; conversion clauses.
                             (RPLSTRING \PNAMESTRING (PROG1 PACK.INDEX
                                                            (AND (IGREATERP (add PACK.INDEX (NCHARS ITEM))
                                                                            (ADD1 \PNAMELIMIT))
                                                                 (LISPERROR "ATOM TOO LONG")))
                                        ITEM))
                            (T                               ; The symbol matched a clause.  We should use only that part of
                                                             ; the symbol that comes after the matching prefix.
                               (LET [(PREFIX-LENGTH (ffetch (STRINGP LENGTH)
                                                            (CL:FIRST CLAUSE]
                                    (RPLSTRING \PNAMESTRING (PROG1 PACK.INDEX
                                                                   (AND (IGREATERP (add PACK.INDEX
                                                                                        (IDIFFERENCE (NCHARS
                                                                                                      ITEM)
                                                                                                     PREFIX-LENGTH))
                                                                                   (ADD1 \PNAMELIMIT))
                                                                        (LISPERROR "ATOM TOO LONG")))
                                                (SUBSTRING ITEM (IPLUS 1 PREFIX-LENGTH]
                  ((OR (STRINGP ITEM)
                       (LITATOM ITEM))
                   (RPLSTRING \PNAMESTRING (PROG1 PACK.INDEX
                                                  (AND (IGREATERP (add PACK.INDEX (NCHARS ITEM))
                                                                  (ADD1 \PNAMELIMIT))
                                                       (LISPERROR "ATOM TOO LONG")))
                              ITEM))
                  (T (\PACK.ITEM ITEM)))
               (SETQ M (ADD1 M))
               (GO LP])
```

(\PACK.ITEM

```
[LAMBDA (ITEM)                                               ; Edited 21-Mar-88 15:30 by bvm
  (DECLARE (USEDFREE PACK.INDEX \PNAMESTRING))
```

;;; Slow case for PACK and PACK* -- append characters of ITEM to \PNAMESTRING, updating PACK.INDEX accordingly

```
   (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                        (AND (IGREATERP PACK.INDEX \PNAMELIMIT)
                             (LISPERROR "ATOM TOO LONG"))
                        (\PNAMESTRINGPUTCHAR (fetch (STRINGP BASE) of \PNAMESTRING)
                               (SUB1 PACK.INDEX)
                               CODE)
                        (add PACK.INDEX 1]
         ITEM])
```

(STRPOS

```
[LAMBDA (PAT STRING START SKIP ANCHOR TAIL CASEARRAY BACKWARDSFLG)
                                                             ; Edited  6-Jan-88 12:44 by jds
  (DECLARE (GLOBALVARS \TRANSPARENT))
  (PROG (PATLEN PATBASE PATOFFST STRINGLEN STRINGBASE STRINGOFFST MAXI JMAX 1stPATchar jthPATchar STRFAT
              PATFAT)
        [COND
           ((LITATOM PAT)
            (SETQ PATBASE (fetch (LITATOM PNAMEBASE) of PAT))
```

```
            (SETQ PATOFFST 1)
            (SETQ PATLEN (fetch (LITATOM PNAMELENGTH) of PAT))
            (SETQ PATFAT (fetch (LITATOM FATPNAMEP) of PAT)))
           (T (OR (STRINGP PAT)
                  (SETQ PAT (MKSTRING PAT)))
              (SETQ PATBASE (fetch (STRINGP BASE) of PAT))
              (SETQ PATOFFST (fetch (STRINGP OFFST) of PAT))
              (SETQ PATLEN (fetch (STRINGP LENGTH) of PAT))
              (SETQ PATFAT (fetch (STRINGP FATSTRINGP) of PAT]
        [COND
           ((LITATOM STRING)
            (SETQ STRINGBASE (fetch (LITATOM PNAMEBASE) of STRING))
            (SETQ STRINGOFFST 1)
            (SETQ STRINGLEN (fetch (LITATOM PNAMELENGTH) of STRING))
            (SETQ STRFAT (fetch (LITATOM FATPNAMEP) of STRING)))
           (T (OR (STRINGP STRING)
                  (SETQ STRING (MKSTRING STRING)))
              (SETQ STRINGBASE (fetch (STRINGP BASE) of STRING))
              (SETQ STRINGOFFST (fetch (STRINGP OFFST) of STRING))
              (SETQ STRINGLEN (fetch (STRINGP LENGTH) of STRING))
              (SETQ STRFAT (fetch (STRINGP FATSTRINGP) of STRING]
        (COND
           ([IGEQ 0 (SETQ MAXI (ADD1 (IDIFFERENCE STRINGLEN PATLEN]
                                                              ; Who's he kidding?  The PATTERN length is greater than the
                                                              ; STRING length
            (RETURN)))
        (COND
           [(NULL START)
            (SETQ START (COND
                           (BACKWARDSFLG MAXI)
                           (T 1]
           [(ILESSP START 0)
            (add START (ADD1 STRINGLEN))
            (COND
               ((ILESSP START 1)
                (RETURN]
           ((IGREATERP START MAXI)
            (RETURN)))                                        ; Normalize start to a 1-origin index between 1 and LEN
        [COND
           ((ILEQ PATLEN 0)
            (RETURN (AND TAIL START]                          ; Null pattern matches anything -- but (STRPOS "" "") is NIL
                                                              ; unless TAIL is T.
        (AND SKIP (SETQ SKIP (CHCON1 SKIP)))
        (COND
           ((NULL CASEARRAY)
            (SETQ CASEARRAY \TRANSPARENT))
           ([NOT (AND (ARRAYP CASEARRAY)
                      (OR (EQ \ST.BYTE (fetch (ARRAYP TYP) of CASEARRAY))
                          (EQ \ST.POS16 (fetch (ARRAYP TYP) of CASEARRAY]
            (\ILLEGAL.ARG CASEARRAY)))                        ; Oh, for a LET here!
        (add STRINGOFFST -1)
        (add PATOFFST -1)
        (RETURN (PROG ((CAOFFST (fetch (ARRAYP OFFST) of CASEARRAY))
                       (CABASE (fetch (ARRAYP BASE) of CASEARRAY))
                       (CAFAT (EQ \ST.POS16 (fetch (ARRAYP TYP) of CASEARRAY)))
                       (CASIZE (fetch (ARRAYP LENGTH) of CASEARRAY))
                       [OFFST.I (IPLUS STRINGOFFST START (COND
                                                           (BACKWARDSFLG 1)
                                                           (T -1]
                       [LASTI (IPLUS STRINGOFFST (COND
                                                   (ANCHOR START)
                                                   (BACKWARDSFLG 1)
                                                   (T MAXI]
                       (JSTART (IPLUS PATOFFST 2))
                       (JMAX (IPLUS PATOFFST PATLEN)))        ; Remember! START is a 1-origin index
                                                              ; There will be at least one pass thru the following loop, or else
                                                              ; we would have (RETURN) before now
                      (OR (EQ 0 CAOFFST)
                          (ERROR "CASEARRAY can't be a sub-array: " CASEARRAY))
                      [SETQ 1stPATchar (\CATRANSLATE CABASE CASIZE CAFAT (\GETBASECHAR PATFAT PATBASE
                                                                                      (ADD1 PATOFFST]
                LP    [COND
                         ((COND
                             (BACKWARDSFLG (ILESSP (add OFFST.I -1)
                                                   LASTI))
                             (T (IGREATERP (add OFFST.I 1)
                                           LASTI)))
                          (RETURN))
                         ([AND [OR (EQ 1stPATchar SKIP)
                                   (EQ 1stPATchar (\CATRANSLATE CABASE CASIZE CAFAT (\GETBASECHAR STRFAT
                                                                                                 STRINGBASE OFFST.I]
                               (for J from JSTART to JMAX as K from (ADD1 OFFST.I)
                                  always (OR [EQ SKIP (SETQ jthPATchar (\CATRANSLATE CABASE CASIZE CAFAT
                                                                                     (\GETBASECHAR PATFAT PATBASE J]
                                             (EQ jthPATchar (\CATRANSLATE CABASE CASIZE CAFAT
                                                                          (\GETBASECHAR STRFAT STRINGBASE K]
                          (RETURN (IDIFFERENCE (COND
```

```
                                                (TAIL (IPLUS OFFST.I PATLEN))
                                                (T OFFST.I))
                                          STRINGOFFST]
                        (GO LP)                                 ; Fall out thru bottom if didn't find it

                    ])

)
```

(CL:DEFUN **XCL:PACK** (NAMES &OPTIONAL (PACKAGE *PACKAGE*))

;;; NAMES should be a list of symbols and strings. A new symbol is created in the given package with a print name equal to the concatenation of the of ;;; the NAMES.

```
   (CL:INTERN (CONCATLIST NAMES)
          PACKAGE))
```

(CL:DEFUN **XCL:PACK*** (&REST NAMES)

;;; NAMES should be a list of symbols and strings. A new symbol is created in the current package with a print name equal to the concatenation of the of ;;; the NAMES.

```
   (CL:INTERN (CONCATLIST NAMES)))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \SIGNFLAG \PRINTRADIX)
)
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \CATRANSLATE MACRO (OPENLAMBDA (CABASE CASIZE CAFAT CHAR)
                                  (COND
                                     ((ILEQ CHAR CASIZE)          ; If it's in the table, use the table value
                                      (\GETBASECHAR CAFAT CABASE CHAR))
                                     (T                           ; Off the end -- assume it's itself
                                        CHAR))))
)
)
```

;; END EXPORTED DEFINITIONS

```
(DEFINEQ
```

(**STRPOSL**

```
  [LAMBDA (A STRING START NEG BACKWARDSFLG)                     (* edited%: "18-Mar-86 17:20")
```

;; Given a list of charcodes, A, find the first one in STRING.

```
    (GLOBALRESOURCE \STRPOSLARRAY (PROG (BASE OFFST LEN I LASTI STRFAT CH)
                                        (OR (type? CHARTABLE A)
                                            (SETQ A (MAKEBITTABLE A NIL \STRPOSLARRAY)))
                                        (if (LITATOM STRING)
                                            then (SETQ BASE (fetch (LITATOM PNAMEBASE) of STRING))
                                                 (SETQ LEN (fetch (LITATOM PNAMELENGTH) of STRING))
                                                 (SETQ OFFST 1)
                                                 (SETQ STRFAT (fetch (LITATOM FATPNAMEP) of STRING))
                                          else (OR (STRINGP STRING)
                                                   (SETQ STRING (MKSTRING STRING)))
                                               (SETQ BASE (fetch (STRINGP BASE) of STRING))
                                               (SETQ LEN (fetch (STRINGP LENGTH) of STRING))
                                               (SETQ OFFST (fetch (STRINGP OFFST) of STRING))
                                               (SETQ STRFAT (fetch (STRINGP FATSTRINGP) of STRING)))
                                        (if (NULL START)
                                            then (SETQ START (if BACKWARDSFLG
                                                                 then LEN
                                                               else 1))
                                          elseif (ILESSP START 0)
                                            then (add START (ADD1 LEN))
                                                 (if (ILESSP START 1)
                                                     then (RETURN))
                                          elseif (IGREATERP START LEN)
                                            then (RETURN))          ; Normalize start to a 1-origin index between 1 and LEN
                                        (add OFFST -1)              ; Bias the OFFST since START is 1-origin and the loop deals in
                                                                    ; 0-origin
                                        (SETQ NEG (if NEG
                                                      then          ; Convert NEG to match the correct value returned by
                                                                    ; \SYNCODE
                                                           0
                                                    else 1))
                                        (SETQ I (IPLUS OFFST START))
```

```
                                    (SETQ LASTI (IPLUS OFFST (if BACKWARDSFLG
                                                                 then (add I 1)
                                                                      1
                                                               else (add I -1)
                                                                    LEN)))
                                                                 ; There will be at least one pass thru the following loop, or else
                                                                 ; we would have (RETURN) before now
                                LP  (if (if BACKWARDSFLG
                                            then (ILESSP (add I -1)
                                                         LASTI)
                                          else (IGREATERP (add I 1)
                                                          LASTI))
                                        then (RETURN)
                                      elseif (EQ NEG (\SYNCODE A (\GETBASECHAR STRFAT BASE I)))
                                        then (RETURN (IDIFFERENCE I OFFST)))
                                    (GO LP])
```

(MAKEBITTABLE

```
  [LAMBDA (L NEG A)                                              ; Edited 29-Apr-91 23:02 by jds
    [COND
       [(type? CHARTABLE A)                                      ; Clear it
        (\ZEROBYTES A 0 \MAXTHINCHAR)
        (if (fetch (CHARTABLE NSCHARHASH) of A)
            then (CLRHASH (fetch (CHARTABLE NSCHARHASH) of A]
       (T (SETQ A (create CHARTABLE]
    (for X in L do (\SETSYNCODE A (OR (SMALLP X)
                                      (CHCON1 X))
                          1))                                    ; Invert 1 and 0 if NEG
    [AND NEG (for I from 0 to \MAXCHAR do (\SETSYNCODE A I (LOGXOR 1 (\SYNCODE A I]
    A])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[PUTDEF '\STRPOSLARRAY 'RESOURCES '(NEW (NCREATE 'CHARTABLE]
)
)

(/SETTOPVAL '\\STRPOSLARRAY.GLOBALRESOURCE NIL)

(DEFINEQ
```

(CASEARRAY

```
  [LAMBDA (OLDAR)                                                (* lmm "20-MAR-81 10:21")
    (COND
       (OLDAR (COPYARRAY OLDAR))
       (T (PROG ((AR (ARRAY 256 'BYTE 0 0)))
                (for I from 0 to 255 do (SETA AR I I))
                (RETURN AR])
```

(UPPERCASEARRAY

```
  [LAMBDA NIL                                                    (* rmk%: " 2-Apr-85 11:22")
    (OR (ARRAYP UPPERCASEARRAY)
        (LET ((CA (CASEARRAY)))
             [for I from (CHARCODE a) to (CHARCODE z) do (SETCASEARRAY CA I (IDIFFERENCE
                                                                               I
                                                                               (CONSTANT (IDIFFERENCE (CHARCODE
                                                                                                        a)
                                                                                                      (CHARCODE A]
             (SETQ UPPERCASEARRAY CA])

)

(MOVD? 'SETA 'SETCASEARRAY)

(MOVD? 'ELT 'GETCASEARRAY)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQ \TRANSPARENT (CASEARRAY))

(RPAQ UPPERCASEARRAY (UPPERCASEARRAY))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS UPPERCASEARRAY GLOBALVAR T)


(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \TRANSPARENT)
```

```
)
)

(DEFINEQ
```

(FILEPOS

```
  [LAMBDA (PATTERN FILE START END SKIP TAIL CASEARRAY)
```

;; Edited 10-Jul-2022 16:51 by rmk

;; Edited 1-Jul-2022 11:55 by rmk

;; Edited 25-Jun-2022 22:51 by rmk: The original version was a byte-level searcher, this upgrades to character searching as determined by the ;; external format of the stream. (It is also a bit faster than the original).

;; This provides accurate results if the stream's external format is stable, wherein each character code has a unique byte representation. If the ;; stream's format is unstable (i.e. XCCS runcoding), then the result is accurate if the stream's initial charset (or other contextual information) is ;; correct for the START byte position.

;; Otherwise, there may be some bad matches and some missing matches. The slow case will be accurate in those cases (and a NIL return for the ;; format's \FORMATBYTESTRING function will kick it into the slow case (about 10 times slower). This always defers to the slow case if SKIP or ;; CASEARRAY are non-NIL.

;; (Original algorithm advanced the pattern (and the stream starting position) over leading skips, presumably to speed up the search. A foolish (and ;; complex) optimization, since it would just mean that you would cover the intervening characters in a different way.)

;; New interface features: TAIL=BOTH means return a dotted pair of the (start . end) of the match

;; CASEARRAY=T forces the slow case, as if CASEARRAY=(CASEARRAU) - transparent.

```
    (PROG ((STREAM (\GETSTREAM FILE 'INPUT))
           STREAMLEN ORGFILEPTR PATSTR PATLEN PATBASE PATLEN FIRSTINDEX LASTINDEX PATFIRSTBYTE STARTBYTEPOS
           ENDBYTEPOS)
```

;;

;; Decode the start and end parameters, set the starting filepointer.

;; Set STARTBYTEPOS and ENDBYTEPOS instead of resetting START and END because vm functions shouldn't change their arguments.

```
          (SETQ ORGFILEPTR (\GETFILEPTR STREAM))
          (SETQ STARTBYTEPOS (COND
                                 (START (CL:UNLESS (AND (FIXP START)
                                                        (IGEQ START 0))
                                                   (LISPERROR "ILLEGAL ARG" START))
                                        (\SETFILEPTR STREAM START)
                                        START)
                                 (T ORGFILEPTR)))
          (SETQ STREAMLEN (\GETEOFPTR STREAM))
          [SETQ ENDBYTEPOS (ADD1 (COND
                                     ((NULL END)                  ; Default is end of file
                                      STREAMLEN)
                                     ((IGEQ END 0)                ; Absolute byte pointer given
                                      (IMIN END STREAMLEN))
                                     (T (IPLUS STREAMLEN END 1]
```

;; STARTBYTEPOS is the position of the first matchable byte = (SETFILEPTR SBP)(BIN)

;; ENDBYTEPOS here is the position one after the last possible start (not the position of the final byte of the last possible match). That is, the ;; match itself can go further than ENDBYTEPOS

```
          (CL:WHEN (IGREATERP STARTBYTEPOS ENDBYTEPOS)           ; nothing to search
              (GO FAILED))
          (CL:WHEN (EQ (NCHARS PATTERN)
                       0)
```

;; Empty string: succed. Already positioned at STARTBYTEPOS

```
              (RETURN STARTBYTEPOS))
          (CL:WHEN [OR CASEARRAY (AND SKIP (STRPOS SKIP PATTERN))
                       (NOT (SETQ PATSTR (\FORMATBYTESTRING STREAM PATTERN]
              (RETURN (OR (\SLOWFILEPOS PATTERN STREAM STARTBYTEPOS ENDBYTEPOS SKIP TAIL CASEARRAY)
                          (GO FAILED))))
```

;;

;; Now we're in the fast case: No SKIP and no CASEARRAY, and we were able to map the search string to a stable sequence of file bytes.

```
          (SETQ PATLEN (ffetch (STRINGP LENGTH) of PATSTR))
          (CL:WHEN (IGREATERP (SUB1 PATLEN)
                              (IDIFFERENCE STREAMLEN ENDBYTEPOS))
```

;; EOF guard; needed to wait for actual pattern length

```
              (SETQ ENDBYTEPOS (IDIFFERENCE STREAMLEN (SUB1 PATLEN)))
              (CL:WHEN (IGEQ STARTBYTEPOS ENDBYTEPOS)
                       (GO FAILED)))
          (SETQ PATBASE (ffetch (STRINGP BASE) of PATSTR))
          (SETQ FIRSTINDEX (ffetch (STRINGP OFFST) of PATSTR))
          (SETQ LASTINDEX (IPLUS FIRSTINDEX (SUB1 PATLEN)))
          (SETQ PATFIRSTBYTE (\GETBASEBYTE PATBASE FIRSTINDEX))
          (ADD FIRSTINDEX 1)                                     ; Start at the second byte when the first one matched.
```

;;

;; The stream keeps track of its byte position, but we must count down ourselves so that we don't go beyond ENDBYTEPOS (would be nice if we ;; could construct a substream). Use hi/lo arithmetic to avoid large integers on big byte regions.

;; A loop of nomatch-match sequences

```
          (BIND (NBYTES _ (IDIFFERENCE ENDBYTEPOS STARTBYTEPOS))
```

```
            NBYTESHI NBYTESLO FIRST (SETQ NBYTESHI (FOLDLO NBYTES FILEPOS.SEGMENT.SIZE))
                              (SETQ NBYTESLO (IMOD NBYTES FILEPOS.SEGMENT.SIZE))
         DO (DO ;; Find next FIRSTBYTE
                (CL:WHEN (ILEQ NBYTESLO 0)                    ; Finished this segment
                    (CL:WHEN (EQ NBYTESHI 0)
                           (GO FAILED))                       ; Roll over to a new segment
                    (add NBYTESLO FILEPOS.SEGMENT.SIZE)
                    (add NBYTESHI -1))
                (ADD NBYTESLO -1)                             ; Decrement the byte count
               REPEATUNTIL (EQ PATFIRSTBYTE (\BIN STREAM)))
            ;;
            ;; Found PATFIRSTBYTE, enter match loop.
            (FOR I FROM FIRSTINDEX TO LASTINDEX DO (CL:UNLESS (EQ (\GETBASEBYTE PATBASE I)
                                                                  (\BIN STREAM))
                                                       ;; Match failed: Go back to second position and try again
                                                       (\INCFILEPTR STREAM (SUB1 (IDIFFERENCE FIRSTINDEX
                                                                                              I)))
                                                       (RETURN))
               FINALLY
                    ;; Ran off the end: complete match, get out of the outer loop
                    (GO FOUNDIT)))
    FOUNDIT
```

;; The stream's charset should be set to the charset corresponding to the return byte-position. We haven't been tracking it, but if we are returning
;; the tail pointer, then the stream's character set must be the same as the character set of the last character o fPATTERN.

;; Getting the character set for the start of the match is a little trickier. We know the character set at the byte that starts the beginning of the match
;; (= character set of PATTERN's first character. If we set the stream to that charset, then back up one character, that should get it right.

;; This should only be necessary for an unstable format, maybe don't bother if it isn't XCCS. There is another special case here for XCCS: if the
;; charset is 255 at the start (=2 byte encoding), then we assume that it didn't change, and nothing to worry about.

```
    (RETURN (IF TAIL
                THEN (CL:UNLESS (EQ NSCHARSETSHIFT (ffetch (STREAM CHARSET) of STREAM))
                         (freplace (STREAM CHARSET) of STREAM with (\CHARSET (NTHCHARCODE PATTERN -1))))
                     (CL:IF (EQ TAIL 'BOTH)
                         (CONS (IDIFFERENCE (\GETFILEPTR STREAM)
                                            PATLEN)
                               (\GETFILEPTR STREAM))
                         (\GETFILEPTR STREAM))
                ELSE ;; Fileptr wants to be where the match started, PATLEN back from where the match ended
                     (\INCFILEPTR STREAM (IMINUS PATLEN))
                     (SETQ STARTBYTEPOS (\GETFILEPTR STREAM))
                     (CL:UNLESS (EQ NSCHARSETSHIFT (ffetch (STREAM CHARSET) of STREAM))
                         (freplace (STREAM CHARSET) of STREAM with (\CHARSET (CHCON1 PATTERN)))
                         (\BACKCCODE STREAM)                  ; Should fix the charset
                         (\SETFILEPTR STREAM STARTBYTEPOS))
                     STARTBYTEPOS))
FAILED
    (\SETFILEPTR STREAM ORGFILEPTR)                           ; return the fileptr to its initial position. We didn't jigger the
                                                              ; original charset
    (RETURN NIL])
```

(FFILEPOS

```
[LAMBDA (PATTERN FILE START END SKIP TAIL CASEARRAY)
```

;; Edited 10-Jul-2022 10:17 by rmk

;; Edited 1-Jul-2022 11:55 by rmk

;; Edited 23-Jun-2022 08:50 by rmk: CASEARRAY is now also a slow (FFILEPOS) case. Fast case now works for arbitrary external formats

;; Edited 10-Aug-2020 21:44 by rmk:

;; RMK: Added coercion from internal XCCS string to UTF8 if searching a UTF8 file (* Pavel "12-Oct-86 15:20")

```
(PROG ((STREAM (\GETSTREAM FILE 'INPUT))
       BYTEPATTERN BPATBASE BPATOFFSET BPATLEN ORGFILEPTR STARTBYTEPOS ENDBYTEPOS BIGENDOFFSET STARTSEG
       ENDSEG EOF)
      (CL:WHEN [OR SKIP CASEARRAY (NOT (fetch PAGEMAPPED of (fetch (STREAM DEVICE) of STREAM)))
                   (NULL (SETQ BYTEPATTERN (\FORMATBYTESTRING STREAM PATTERN]
                                                              ; Slow case--use FILEPOS
          (GO TRYFILEPOS))                                    ; calculate start addr and set file ptr.
      (SETQ BPATBASE (fetch (STRINGP BASE) of BYTEPATTERN))
      (SETQ BPATOFFSET (fetch (STRINGP OFFST) of BYTEPATTERN))
      (SETQ BPATLEN (fetch (STRINGP LENGTH) of BYTEPATTERN))
      (COND
         ((OR (IGREATERP BPATLEN \MAX.PATTERN.SIZE)
              (ILESSP BPATLEN \MIN.PATTERN.SIZE))
          (GO TRYFILEPOS)))
      (SETQ ORGFILEPTR (\GETFILEPTR STREAM))
      (SETQ STARTBYTEPOS (IPLUS (COND
                                   (START (COND
                                             ((NOT (AND (FIXP START)
                                                        (IGEQ START 0)))
```

```
                                        (LISPERROR "ILLEGAL ARG" START)))
                                   START)
                              (T ORGFILEPTR))
                             (SUB1 BPATLEN)))            ; STARTBYTEPOS is the address of the character
                                                         ; corresponding to the last character of PATTERN.
    (SETQ EOF (\GETEOFPTR STREAM))                       ; calculate the character address of the character after the last
                                                         ; possible match.
    [SETQ ENDBYTEPOS (COND
                        ((NULL END)                      ; Default is end of file
                         EOF)
                        (T (IMIN (IPLUS (COND
                                           ((ILESSP END 0)
                                            (IPLUS EOF END 1))
                                           (T END))
                                        BPATLEN)
                                 EOF]
```

;; use STARTBYTEPOS and ENDBYTEPOS instead of START and END because vm functions shouldn't change their arguments.

```
    (COND
       ((IGEQ STARTBYTEPOS ENDBYTEPOS)                   ; nothing to search
        (RETURN))
       ((ILESSP (IDIFFERENCE ENDBYTEPOS STARTBYTEPOS)
                \MIN.SEARCH.LENGTH)                      ; too small to make FFILEPOS worthwhile
        (GO TRYFILEPOS)))
    (\SETFILEPTR STREAM STARTBYTEPOS)
    [RETURN (GLOBALRESOURCE (\FFDELTA1 \FFDELTA2 \FFPATCHAR)
                  (PROG ((DELTA1 (fetch (ARRAYP BASE) of \FFDELTA1))
                         (DELTA2 (fetch (ARRAYP BASE) of \FFDELTA2))
                         (PATCHAR (fetch (ARRAYP BASE) of \FFPATCHAR))
                         (MAXPATINDEX (SUB1 BPATLEN))
                         CHAR CURPATINDEX LASTCHAR INC)
```

;; Use Boyer-Moore string search algorithm. Use two auxiliary tables, DELTA1 and DELTA2, to tell how far ahead to ;; move in the file when a partial match fails. DELTA1 contains, for each character code, the distance of that ;; character from the right end of the pattern, or PATLEN if the character does not occur in the pattern. DELTA2 ;; contains, for each character position in the pattern, how far ahead to move such that the partial substring ;; discovered to the right of the position now matches some other substring (to the left) in the pattern. PATCHAR is ;; just PATTERN translated thru CASEARRAY

```
                        (\SETUP.FFILEPOS BPATBASE BPATOFFSET BPATLEN PATCHAR DELTA1 DELTA2)
                        [COND
                           ((SMALLP ENDBYTEPOS)
                            (SETQ STARTSEG (SETQ ENDSEG 0)))
                           (T
```

;; The search will be in the large integers at least part of the time, so split the start and end fileptrs into ;; hi and lo parts. The 'segment' size we choose is smaller than 2^16 so that we are still smallp near ;; the boundary. Note that STARTBYTEPOS and ENDBYTEPOS are never actually used as file ptrs, ;; just for counting.

```
                              (SETQ ENDSEG (FOLDLO ENDBYTEPOS FILEPOS.SEGMENT.SIZE))
                              (SETQ BIGENDOFFSET (MOD ENDBYTEPOS FILEPOS.SEGMENT.SIZE))
                              (SETQ STARTSEG (FOLDLO STARTBYTEPOS FILEPOS.SEGMENT.SIZE))
                              (SETQ STARTBYTEPOS (MOD STARTBYTEPOS FILEPOS.SEGMENT.SIZE))
                              (SETQ ENDBYTEPOS (COND
                                                  ((EQ STARTSEG ENDSEG)
                                                   BIGENDOFFSET)
                                                  (T
```

;; In different segments, so we'll have to search all the way to the end of this seg; hence, ;; 'end' is currently as big as it gets

```
                                                     FILEPOS.SEGMENT.SIZE]
                        (SETQ LASTCHAR (GETBASEBYTE PATCHAR MAXPATINDEX))
                     FIRSTCHARLP
                        (COND
                           [(IGEQ STARTBYTEPOS ENDBYTEPOS)
                                                         ; End of this chunk
                            (COND
                               ((EQ STARTSEG ENDSEG)     ; failed
                                (GO FAILED))
                               (T                        ; Finished this segment, roll over into new one
                                  (add STARTSEG 1)
                                  (SETQ STARTBYTEPOS (IDIFFERENCE STARTBYTEPOS FILEPOS.SEGMENT.SIZE))
                                  (COND
                                     ((EQ STARTSEG ENDSEG)
                                      (SETQ ENDBYTEPOS BIGENDOFFSET)))
                                  (GO FIRSTCHARLP]
                           ((NEQ (SETQ CHAR (\BIN STREAM))
                                 LASTCHAR)
                            (add STARTBYTEPOS (SETQ INC (GETBASEBYTE DELTA1 CHAR)))
                            (OR (EQ INC 1)
                                (\INCFILEPTR STREAM (SUB1 INC)))
                                                         ; advance file pointer accordingly (\BIN already advanced it one)
                            (GO FIRSTCHARLP)))
                        (SETQ CURPATINDEX (SUB1 MAXPATINDEX))
                     MATCHLP
                        (COND
                           ((ILESSP CURPATINDEX 0)
                            (GO FOUNDIT)))
                        (\DECFILEPTR STREAM 2)            ; back up to read previous char
```

```
                        (COND
                           ((NEQ (SETQ CHAR (\BIN STREAM))
                                 (GETBASEBYTE PATCHAR CURPATINDEX))
                                                    ; Mismatch, advance by greater of delta1 and delta2
                            (add STARTBYTEPOS (IDIFFERENCE (SETQ INC (IMAX (GETBASEBYTE DELTA1 CHAR)
                                                                           (GETBASEBYTE DELTA2
                                                                                  CURPATINDEX)))
                                                           (IDIFFERENCE MAXPATINDEX CURPATINDEX)))
                            (OR (EQ INC 1)
                                (\INCFILEPTR STREAM (SUB1 INC)))
                            (GO FIRSTCHARLP)))
                        (SETQ CURPATINDEX (SUB1 CURPATINDEX))
                        (GO MATCHLP)
                    FOUNDIT

                    ;; Unlike FILEPOS, it appears that the file is now positioned just after the first byte of the match.  See note there
                    ;; about charsets.
                        (RETURN (IF TAIL
                                    THEN (CL:UNLESS (EQ NSCHARSETSHIFT (ffetch (STREAM CHARSET)
                                                                           of STREAM))
                                                    (freplace (STREAM CHARSET) of STREAM
                                                       with (\CHARSET (NTHCHARCODE PATTERN -1))))
                                                (\INCFILEPTR STREAM (SUB1 BPATLEN))
                                                (SETQ ENDBYTEPOS (\GETFILEPTR STREAM))
                                                (CL:IF (EQ TAIL 'BOTH)
                                                    (CONS (IDIFFERENCE ENDBYTEPOS BPATLEN)
                                                          ENDBYTEPOS)
                                                    ENDBYTEPOS)
                                    ELSE ;; Fileptr wants to be where the match started, 1 back from where the match ended
                                                (\INCFILEPTR STREAM -1)
                                                (SETQ STARTBYTEPOS (\GETFILEPTR STREAM))
                                                (CL:UNLESS (EQ NSCHARSETSHIFT (ffetch (STREAM CHARSET) of STREAM))
                                                    (freplace (STREAM CHARSET) of STREAM with (\CHARSET (CHCON1
                                                                                                         PATTERN)))
                                                    (\BACKCCODE STREAM)
                                                                    ; Should fix the charset
                                                    (\SETFILEPTR STREAM STARTBYTEPOS))
                                                STARTBYTEPOS))
                    FAILED
                        (\SETFILEPTR STREAM ORGFILEPTR)    ; return the fileptr to its initial position.
                        (RETURN NIL]
    TRYFILEPOS
        (RETURN (FILEPOS PATTERN STREAM START END SKIP TAIL CASEARRAY])
```

(\SETUP.FFILEPOS

```
  [LAMBDA (PATBASE PATOFFSET PATLEN PATCHAR DELTA1 DELTA2)
```

;; Edited 24-Jun-2022 16:32 by rmk: Removing CASE argument. That forces the \SLOWFILEPOS, because the the alternative stream matches
;; can't be anticipated.

(* jop%: "25-Sep-86 11:44")

;;; Set up PATCHAR, DELTA1 and DELTA2 arrays from string. This is a separate function currently so I can gather stats on it

```
    (PROG ((PATLEN,PATLEN (IPLUS (LLSH PATLEN BITSPERBYTE)
                                 PATLEN))
           (MAXPATINDEX (SUB1 PATLEN))
           CHAR)
          (for I from 0 to (FOLDLO \MAXCHAR BYTESPERWORD) do (PUTBASE DELTA1 I PATLEN,PATLEN))
```

;; DELTA1 initially all PATLEN, the default for chars not in the pattern. I assume array is word-aligned

```
          (for I from 0 to MAXPATINDEX do (SETQ CHAR (GETBASEBYTE PATBASE (IPLUS PATOFFSET I)))
                                          (PUTBASEBYTE PATCHAR I CHAR)
                                          (PUTBASEBYTE DELTA1 CHAR (IDIFFERENCE MAXPATINDEX I))
                                                                    ; DELTA1 = how far ahead to move when we mismatch with this
                                                                    ; char
                                          )
```

;; Now set up DELTA2. Scan pattern backwards. For each character, we want to find the rightmost reoccurrence of the substring consisting of
;; the chars to the right of the current char. This is slightly different than Boyer-Moore, in that we do not insist that it be the rightmost reoccurrence
;; that is not preceded by the current char. Small difference, noticeable only in patterns that contain multiple occurrences of tails of the pattern.
;; The following loop calculates DELTA2 in almost the obvious way, using the observation that DELTA2 is strictly increasing (by our definition) as
;; the pattern index decreases. This algorithm is potentially quadratic, as it amounts to searching a string (PATTERN, backwards) for a given
;; substring in the 'dumb' way; fortunately, it is rarely so in practice for 'normal' patterns

```
          (for P from (SUB1 MAXPATINDEX) to 0 by -1 bind (LASTD2 _ 1)
                                                         (LASTMATCHPOS _ MAXPATINDEX)
             do (PUTBASEBYTE DELTA2 P
                             (SETQ LASTD2
                              (COND
                                 ([OR (IGEQ LASTD2 PATLEN)
                                      (EQ (GETBASEBYTE PATCHAR (IDIFFERENCE MAXPATINDEX LASTD2))
                                          (GETBASEBYTE PATCHAR (ADD1 P]
```

;; The last time around we matched a terminal substring somehow, and now the next char matches the char before
;; that substring, so DELTA2 is just one more, i.e. the match continues. Once we've overflowed the pattern, the
;; 'match' continues trivially

```
                        (ADD1 LASTD2))
                       (T [do (SETQ LASTMATCHPOS (SUB1 LASTMATCHPOS))
                              repeatuntil (for I from MAXPATINDEX to (ADD1 P) by -1 as J from LASTMATCHPOS
                                             to 0 by -1 always (EQ (GETBASEBYTE PATCHAR I)
                                                                   (GETBASEBYTE PATCHAR J]
                                                                ; Substring from P+1 onward matches substring that ends at
                                                                ; LASTMATCHPOS
                          (IPLUS (IDIFFERENCE MAXPATINDEX LASTMATCHPOS)
                                 (IDIFFERENCE MAXPATINDEX P])
```

(\SLOWFILEPOS

```
  [LAMBDA (PATTERN STREAM STARTBYTEPOS ENDBYTEPOS SKIP TAIL CASEARRAY)
```

;; Edited 24-Jul-2022 14:56 by rmk

;; Edited 10-Jul-2022 16:50 by rmk

;; Edited 1-Jul-2022 10:51 by rmk

;; Edited 29-Jun-2022 13:43 by rmk: The slow case when either SKIP or TAIL is specified. Those operate only on character codes, not on
;; individual bytes of the external format, so the file has to be decoded with generic character functions.

;; CASEARRAY is assumed only to map ASCII, but that is independent of the logic here.

```
    (PROG ((SKIPCODE (CL:WHEN SKIP (CHCON1 SKIP)))
           PATBASE PATLEN PATFATP FIRSTINDEX LASTINDEX SKIPCODE PATFIRSTCODE NFIRSTCODEBYTES NPBYTES CABASE
           CASIZE CAFAT STARTCHARSET (ORGCHARSET (ffetch (STREAM CHARSET) of STREAM)))
          (DECLARE (SPECVARS NFIRSTCODEBYTES NPBYTES))
          (CL:WHEN (AND CASEARRAY (NEQ T CASEARRAY))
              (CL:UNLESS [AND (ARRAYP CASEARRAY)
                              (OR (EQ \ST.BYTE (ffetch (ARRAYP TYP) of CASEARRAY))
                                  (SETQ CAFAT (EQ \ST.POS16 (ffetch (ARRAYP TYP) of CASEARRAY]
                     (LISPERROR "ILLEGAL ARG" CASEARRAY))
              (SETQ CABASE (FETCH (ARRAYP BASE) OF CASEARRAY))
              (SETQ CASIZE (FETCH (ARRAYP LENGTH) OF CASEARRAY))
              (SETQ PATTERN (CONCAT PATTERN))                    ; Map all STR characters thru the case array
              (FOR C INSTRING PATTERN AS I FROM 1 DO (RPLCHARCODE PATTERN I (\CATRANSLATE CABASE CASIZE CAFAT
                                                                             C))))
    ;;
    ;; PATSTR now has case-mapped characters
          [COND
             ((LITATOM PATTERN)
              (SETQ PATBASE (ffetch (LITATOM PNAMEBASE) of PATTERN))
              (SETQ PATLEN (ffetch (LITATOM PNAMELENGTH) of PATTERN))
              (SETQ FIRSTINDEX 1)
              (SETQ PATFATP (ffetch (LITATOM FATPNAMEP) of PATTERN)))
             (T (CL:UNLESS (STRINGP PATTERN)
                    (SETQ PATTERN (MKSTRING PATTERN)))
                (SETQ PATBASE (ffetch (STRINGP BASE) of PATTERN))
                (SETQ PATLEN (ffetch (STRINGP LENGTH) of PATTERN))
                (SETQ FIRSTINDEX (ffetch (STRINGP OFFST) of PATTERN))
                (SETQ PATFATP (ffetch (STRINGP FATSTRINGP) of PATTERN]
          (SETQ LASTINDEX (IPLUS FIRSTINDEX (SUB1 PATLEN)))
          (SETQ PATFIRSTCODE (\GETBASECHAR PATFATP PATBASE FIRSTINDEX))
          (ADD FIRSTINDEX 1)                                     ; Start at the second character after the first one matched.
    ;;
    ;; A loop of nomatch-match sequences
    ;; EOFGUARD saves a little testing, assumes no character in any encoding takes more than 10 bytes.
          (BIND STREAMCODE NBYTESHI NBYTESLO SECONDCHARSET *BYTECOUNTER* (NBYTES _ (IDIFFERENCE ENDBYTEPOS
                                                                                    STARTBYTEPOS))
              (EOLC _ (FFETCH (STREAM EOLCONVENTION) OF STREAM))
              (INCCODEFN _ (FFETCH (STREAM INCCODEFN) OF STREAM)) DECLARE (SPECVARS *BYTECOUNTER*)
             FIRST (SETQ NBYTESHI (FOLDLO NBYTES FILEPOS.SEGMENT.SIZE))
                   (SETQ NBYTESLO (IMOD NBYTES FILEPOS.SEGMENT.SIZE))
             DO (DO ;; Find next FIRSTCHAR
                    (CL:WHEN (ILEQ NBYTESLO 0)                   ; Finished this segment
                        (CL:WHEN (EQ NBYTESHI 0)
                               (GO FAILED))                      ; Roll over to a new segment
                        (add NBYTESLO FILEPOS.SEGMENT.SIZE)
                        (add NBYTESHI -1))
                    ;; Guard \INCCODE against EOF, only when we are getting close
                    (CL:WHEN (AND (EQ NBYTESHI 0)
                                  (ILEQ NBYTESLO 10)
                                  (NULL (\PEEKCCODE STREAM T)))
                           (GO FAILED))
                    (SETQ STARTCHARSET (ffetch (STREAM CHARSET) of STREAM))
                    (PROGN ;; Open coding of \INCCODE.EOLC (with *BYTECOUNTER* and EOLC bindings above.
                           (SETQ STREAMCODE (\CHECKEOLC (CL:FUNCALL INCCODEFN STREAM T)
                                                        EOLC STREAM NIL T))
                           ;; Make negative because that's the \INCCODE convention
                           (SETQ NFIRSTCODEBYTES (IMINUS *BYTECOUNTER*)))
                                                                 ; Decrement the character's byte count
```

```
                    (ADD NBYTESLO NFIRSTCODEBYTES)
                    (CL:WHEN (EQ PATFIRSTCODE SKIPCODE)          ; Pattern starts with skip
                        (RETURN))
                    (CL:WHEN CABASE
                        (SETQ STREAMCODE (\CATRANSLATE CABASE CASIZE CAFAT STREAMCODE)))
                  REPEATUNTIL (EQ STREAMCODE PATFIRSTCODE))
                (SETQ SECONDCHARSET (ffetch (STREAM CHARSET) of STREAM))
                ;;
                ;; Found PATFIRSTCODE, match the rest
                ;; The matching loop must fail at EOF, otherwise either match or return to firstchar loop.
                ;; The EOF guard is \PEEKCCODE (no error), we only want to bother when we might be getting close.
                (SETQ NPBYTES 0)
                (FOR I PATCODE (EOFGUARD _ (AND (EQ NBYTESHI 0)
                                                (ILEQ NBYTESLO 10)))
                  FROM FIRSTINDEX TO LASTINDEX DO (CL:WHEN (AND EOFGUARD (NULL (\PEEKCCODE STREAM T)))
                                                           (GO FAILED))
                                                  (SETQ PATCODE (\GETBASECHAR PATFATP PATBASE I))
                                                  (SETQ STREAMCODE (\INCCODE.EOLC STREAM NIL 'NPBYTES NPBYTES
                                                                                  ))
                                                  (CL:UNLESS (EQ PATCODE SKIPCODE)
                                                      (CL:WHEN CABASE
                                                          (SETQ STREAMCODE (\CATRANSLATE CABASE CASIZE CAFAT
                                                                                         STREAMCODE)))
                                                      (CL:UNLESS (EQ STREAMCODE PATCODE)
                                                          ;; Match failed: Go back to second position and try again
                                                          (\INCFILEPTR STREAM NPBYTES)
                                                          (freplace (STREAM CHARSET) of STREAM with
                                                                                                SECONDCHARSET
                                                                  )
                                                          (RETURN)))
                  FINALLY (GO FOUNDIT)))
      FOUNDIT

      ;; The CHARSET should be accurate in the tail case.  We have to adjust for the start case.  NPBYTES is positive, given that we opencoded the
      ;; \INCCODE.EOLC in the first-char loop.
          (RETURN (SELECTQ TAIL
                       (NIL                                  ; Fileptr wants to be where the match started
                            (freplace (STREAM CHARSET) of STREAM with STARTCHARSET)
                            (\INCFILEPTR STREAM (IPLUS NFIRSTCODEBYTES NPBYTES))
                            (\GETFILEPTR STREAM))
                       (BOTH (CONS (IPLUS (\GETFILEPTR STREAM)
                                          NFIRSTCODEBYTES NPBYTES)
                                   (\GETFILEPTR STREAM)))
                       (\GETFILEPTR STREAM)))
      FAILED
          (freplace (STREAM CHARSET) of STREAM with ORGCHARSET)
          (RETURN NIL])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

[PUTDEF '\FFDELTA1 'RESOURCES '(NEW (ARRAY (ADD1 \MAXCHAR)
                                           'BYTE]

[PUTDEF '\FFDELTA2 'RESOURCES '(NEW (ARRAY \MAX.PATTERN.SIZE 'BYTE]

[PUTDEF '\FFPATCHAR 'RESOURCES '(NEW (ARRAY \MAX.PATTERN.SIZE 'BYTE]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \MAX.PATTERN.SIZE 128)

(RPAQQ \MIN.PATTERN.SIZE 3)

(RPAQQ FILEPOS.SEGMENT.SIZE 32768)

(RPAQQ \MIN.SEARCH.LENGTH 100)

(CONSTANTS (\MAX.PATTERN.SIZE 128)
       (\MIN.PATTERN.SIZE 3)
       (FILEPOS.SEGMENT.SIZE 32768)
       (\MIN.SEARCH.LENGTH 100))
)
)

(/SETTOPVAL '\\FFDELTA1.GLOBALRESOURCE NIL)

(/SETTOPVAL '\\FFDELTA2.GLOBALRESOURCE NIL)
```

```
(/SETTOPVAL '\\FFPATCHAR.GLOBALRESOURCE NIL)
```

;; DATE Functions

```
(DEFINEQ

(DATE
  [LAMBDA (FORMAT)                                          (* raf "16-Oct-86 17:16")
    (\OUTDATE (\UNPACKDATE)
           FORMAT])


(DATEFORMAT
  [NLAMBDA FORMAT                                           (* raf "16-Oct-86 17:17")
    (CONS 'DATEFORMAT FORMAT])


(GDATE
  [LAMBDA (DATE FORMAT STRPTR)                              (* raf "16-Oct-86 17:17")
    (\OUTDATE (\UNPACKDATE DATE)
           FORMAT STRPTR])


(IDATE
  [LAMBDA (STR DEFAULTTIME)                                 ; Edited 17-Apr-2018 10:05 by rmk:
                                                            ; Edited  4-May-89 18:22 by bvm
    ;; RMK:  Fixed so that year < 100 heuristic is changed to add 2000 if < 50, 1900 if >= 50.  Y2K guess for 2-digit years
    (if (NULL STR)
        then (DAYTIME)
      else
       (PROG ((*STR* (MKSTRING STR))
              (*POS* 1)
              MONTH DAY YEAR HOUR MINUTES SECONDS N1 N2 CH DLS TIMEZONE)
             (DECLARE (CL:SPECIAL *STR* *POS*))
         TOP (OR (SETQ N1 (\IDATESCANTOKEN))
                 (RETURN NIL))
             (SELCHARQ (NTHCHARCODE *STR* *POS*)
                  ((/ - SPACE)                              ; Okay to put inside date
                       (add *POS* 1))
                  ("," (if (LISTP N1)
                           then                             ; Assume str was something like Mon, Apr 1....  Trash the day.
                                (add *POS* 1)
                                (GO TOP)))
                  ("." (if (LISTP N1)
                           then                             ; Abbreviated month?
                                (add *POS* 1)))
                  NIL)
             (OR (SETQ N2 (\IDATESCANTOKEN))
                 (RETURN NIL))
             (SELCHARQ (NTHCHARCODE *STR* *POS*)
                  ((/ - SPACE %,)
                       (add *POS* 1))
                  ("." (if (LISTP N2)
                           then                             ; Abbreviated month?
                                (add *POS* 1)))
                  NIL)
             (if [NOT (FIXP (SETQ YEAR (\IDATESCANTOKEN]
                 then (RETURN NIL)
               elseif (< YEAR 100)
                 then                                       ; Y2K heuristic
                      (add YEAR (if (< YEAR 50)
                                    THEN 2000
                                  ELSE 1900))
               elseif (OR (< YEAR 1900)
                          (> YEAR 2037))
                 then                                       ; out of range
                      (RETURN NIL))                         ; Now figure out day and month
             (if (FIXP N2)
                 then                                       ; Must be month-day
                      (SETQ DAY N2)
                      (SETQ MONTH N1)
               elseif (FIXP (SETQ DAY N1))
                 then                                       ; day-month
                      (SETQ MONTH N2)
               else (RETURN NIL))
             (if (FIXP MONTH)
                 then (if (OR (< MONTH 1)
                              (> MONTH 12))
                          then                              ; invalid month
                               (RETURN NIL))
               elseif (SETQ MONTH (\IDATE-PARSE-MONTH MONTH))
               else (RETURN NIL))
             (if (OR (< DAY 1)
                     (> DAY (SELECTQ MONTH
                                 ((9 4 6 11)                ; 30 days hath September...
                                      30)
```

```
                              (2 (if (EVENP YEAR 4)
                                     then 29
                                   else 28))
                             31)))
                then (RETURN NIL))
           (while (EQ (SETQ CH (NTHCHARCODE *STR* *POS*))
                      (CHARCODE SPACE))
              do                                                ; Skip spaces
                 (add *POS* 1))
           (SELCHARQ (NTHCHARCODE *STR* *POS*)
                ("," (add *POS* 1))                             ; Ok to terminate date with comma
                (NIL                                            ; No time.  Ok if DEFAULTTIME passed in
                     (if (NULL DEFAULTTIME)
                         then (RETURN NIL))
                     (SETQ SECONDS (IREMAINDER DEFAULTTIME 60))
                     (SETQ MINUTES (IREMAINDER (SETQ DEFAULTTIME (IQUOTIENT DEFAULTTIME 60))
                                               60))
                     (SETQ HOUR (IQUOTIENT DEFAULTTIME 60))
                     (GO DONE))
                NIL)
;; Now scan time
           (if [NOT (FIXP (SETQ HOUR (\IDATESCANTOKEN]
               then (RETURN NIL))
           (if (EQ (SETQ CH (NTHCHARCODE *STR* *POS*))
                   (CHARCODE %:))
               then                                             ; hh:mm
                    (add *POS* 1)
                    (OR (FIXP (SETQ MINUTES (\IDATESCANTOKEN)))
                        (RETURN NIL))
                    (if (EQ (SETQ CH (NTHCHARCODE *STR* *POS*))
                            (CHARCODE %:))
                        then                                    ; hh:mm:ss
                             (add *POS* 1)
                             (OR (FIXP (SETQ SECONDS (\IDATESCANTOKEN)))
                                 (RETURN NIL))
                             (SETQ CH (NTHCHARCODE *STR* *POS*)))
             else                                               ; break apart time given without colon
                  (SETQ MINUTES (IREMAINDER HOUR 100))
                  (SETQ HOUR (IQUOTIENT HOUR 100)))
           [if CH
               then                                             ; There's more
                    [while (EQ CH (CHARCODE SPACE)) do          ; Skip spaces
                                                       (SETQ CH (NTHCHARCODE *STR* (add *POS* 1]
                    [if [AND (FMEMB CH (CHARCODE (A P a p)))
                             (FMEMB (NTHCHARCODE *STR* (ADD1 *POS*))
                                    (CHARCODE (M m)))
                             (FMEMB (NTHCHARCODE *STR* (+ *POS* 2))
                                    (CHARCODE (SPACE - NIL]
                        then                                    ; AM or PM appended
                             (if (NOT (< HOUR 13))
                                 then                           ; bogus
                                      (RETURN NIL))
                             (if (EQ HOUR 12)
                                 then                           ; wrap to zero
                                      (SETQ HOUR 0))
                             (if (FMEMB CH (CHARCODE (P p)))
                                 then                           ; PM = 12 hours later
                                      (add HOUR 12))
                             (SETQ CH (NTHCHARCODE *STR* (add *POS* 2)))
                             (while (EQ CH (CHARCODE SPACE)) do ; Skip spaces
                                                       (SETQ CH (NTHCHARCODE *STR* (add *POS* 1]
                    ;; Now check for time zone
                    [if [AND (EQ CH (CHARCODE -))
                             (ALPHACHARP (NTHCHARCODE *STR* (ADD1 *POS*]
                        then                                    ; Some obsolete date forms gave time zone separated from time
                                                                ; by hyphen
                             (SETQ CH (NTHCHARCODE *STR* (add *POS* 1]
                    (SELCHARQ CH
                         ((+ -)                                 ; Explicit offset +-hhmm from GMT
                              (add *POS* 1)
                              (if [NOT (FIXP (SETQ TIMEZONE (\IDATESCANTOKEN]
                                  then (RETURN NIL))
                              (CL:MULTIPLE-VALUE-BIND (H M)
                                  (CL:TRUNCATE TIMEZONE 100)
                                [SETQ TIMEZONE (if (EQ M 0)
                                                   then H
                                                 else           ; Non-hour timezone.  Use ratios.
                                                      (+ H (/ M 60])
                              (if (EQ CH (CHARCODE +))
                                  then                          ; we represent time zones the other way around, so have to
                                                                ; negate
                                       (SETQ TIMEZONE (- TIMEZONE))))
                         (if (AND CH (ALPHACHARP CH))
                             then                               ; Perhaps symbolic time zone
```

```
                                (PROG ((START *POS*))
                                  LP  (if [NULL (SETQ CH (NTHCHARCODE *STR* (add *POS* 1]
                                        elseif (ALPHACHARP CH)
                                          then (GO LP)
                                        elseif (EQ CH (CHARCODE SPACE))
                                          then                    ; Space may terminate, except that some time zones have space
                                                                  ; in middle, e.g., EET DST.
                                               (if (AND (SETQ CH (NTHCHARCODE *STR* (ADD1 *POS*)))
                                                        (ALPHACHARP CH))
                                                   then (add *POS* 1)
                                                        (GO LP))
                                        else                      ; Non-alphabetic in timezone
                                             (RETURN NIL))

                                ;; Potential time zone from START to before POS

                                     (SETQ TIMEZONE (SUBSTRING *STR* START (SUB1 *POS*)))
                                     (RETURN (SETQ TIMEZONE
                                              (for ZONE in TIME.ZONES bind DST
                                                 do (if (STRING-EQUAL TIMEZONE (CADR ZONE))
                                                        then (RETURN (CAR ZONE))
                                                      elseif (AND (SETQ DST (CADDR ZONE))
                                                                  (STRING-EQUAL TIMEZONE DST))
                                                        then ; The daylight equivalent is off by one hour
                                                             (RETURN (SUB1 (CAR ZONE]
    DONE
        (RETURN (AND (< HOUR 24)
                     (< MINUTES 60)
                     (OR (NOT SECONDS)
                         (< SECONDS 60))
                     (\PACKDATE YEAR (SUB1 MONTH)
                            DAY HOUR MINUTES (OR SECONDS 0)
                            TIMEZONE])
```

(\IDATESCANTOKEN

```
[LAMBDA NIL                                                       ; Edited  4-May-89 15:20 by bvm
  (DECLARE (CL:SPECIAL *STR* *POS*))

  ;; Returns next token in STR, starting at POS.  Is either an integer or list of alphabetic charcodes.  Skips blanks

  (PROG (RESULT CH)
    LP  (SETQ CH (NTHCHARCODE *STR* *POS*))
        (RETURN (COND
                   ((NULL CH)
                    NIL)
                   ((EQ CH (CHARCODE SPACE))                      ; Skip leading spaces
                    (add *POS* 1)
                    (GO LP))
                   ((DIGITCHARP CH)
                    (SETQ RESULT (- CH (CHARCODE 0)))
                    [while (AND (SETQ CH (NTHCHARCODE *STR* (add *POS* 1)))
                                (DIGITCHARP CH))
                       do (SETQ RESULT (+ (- CH (CHARCODE 0))
                                          (TIMES RESULT 10]
                    RESULT)
                   ((ALPHACHARP CH)
                    (CONS (UCASECODE CH)
                          (while (AND (SETQ CH (NTHCHARCODE *STR* (add *POS* 1)))
                                      (ALPHACHARP CH))
                             collect (UCASECODE CH])
```

(\IDATE-PARSE-MONTH

```
[LAMBDA (MONTH)                                                   ; Edited  4-May-89 14:54 by bvm

  ;; MONTH is a list of upper case character codes.  Figure out which month (1-12) we mean.  We require that MONTH be at least 3 characters long
  ;; and a prefix of month name

  ;; These ugly macros produce code, essentially a decision tree, that walks down the list of char codes looking for exactly the right ones.

  (CL:MACROLET
   [[DISCRIMINATE (FORMS)

           ;; The entry -- start MINCHARS at 3 and turn the month names into char codes.  FORMS is quoted list to workaround masterscope
           ;; stupidity

           `(DISCRIMINATE-1 3 ,@(FOR F IN (CADR FORMS) COLLECT (CONS (CHCON (CAR F))
                                                                     (CDR F]

    [DISCRIMINATE-1 (MINCHARS &BODY FORMS)
           (IF (NULL (CDR FORMS))
               THEN                                               ; only one case
                    `[COND
                        ((DISCRIMINATE-2 ,MINCHARS ,(CAAR FORMS))
                         ,@(CDAR FORMS]
             ELSE                                                 ; Discriminate on the first code and recur on the tails
                  (LIST* 'CASE `(CAR CODEVAR)
                         (WHILE FORMS BIND REST C
                            COLLECT (SETQ REST (CL:REMOVE (SETQ C (CAAAR FORMS))
                                                          FORMS :KEY 'CAAR))
                                    `(,C (SETQ CODEVAR (CDR CODEVAR))
                                         (DISCRIMINATE-1 ,(SUB1 MINCHARS)
```

```
                                                ,@(FOR F IN (CL:SET-DIFFERENCE FORMS (SETQ FORMS REST))
                                                     COLLECT (CONS (CDAR F)
                                                                   (CDR F]
      (DISCRIMINATE-2 (MINCHARS MATCHLST)
             ;; True if codes match MATCHLST, with prefix at least MINCHARS long.
             (IF (NULL MATCHLST)
                 THEN `(NULL CODEVAR)
                 ELSE (LET [(CODE `(AND (EQ (CAR CODEVAR)
                                            ,(POP MATCHLST))
                                        (PROGN (SETQ CODEVAR (CDR CODEVAR))
                                               (DISCRIMINATE-2 ,(SUB1 MINCHARS)
                                                               ,MATCHLST]
                           (IF (<= MINCHARS 0)
                               THEN                                    ; Ok to match null
                                    `(OR (NULL CODEVAR)
                                         ,CODE)
                               ELSE                                    ; Must match exactly so far
                                    CODE]
      (LET ((CODEVAR MONTH))                                           ; This LET is solely to allow more compact code (PVAR_ is one
                                                                       ; byte less than IVARX_)
           (DISCRIMINATE '(("JANUARY" 1)
                           ("FEBRUARY" 2)
                           ("MARCH" 3)
                           ("APRIL" 4)
                           ("MAY" 5)
                           ("JUNE" 6)
                           ("JULY" 7)
                           ("AUGUST" 8)
                           ("SEPTEMBER" 9)
                           ("OCTOBER" 10)
                           ("NOVEMBER" 11)
                           ("DECEMBER" 12])
```

(\OUTDATE

```
  [LAMBDA (UD FORMAT STRING)                                           ; Edited  3-May-2018 00:02 by rmk:
    (DESTRUCTURING-BIND
     (YEAR MONTH DAY HOUR MINUTE SECOND DST WDAY)
     UD
     (LET ((SEPR (CHARCODE -))
           (HOUR.LENGTH 2)
           SIZE S N NO.DATE NO.TIME NO.LEADING.SPACES TIME.ZONE TIME.ZONE.LENGTH YEAR.LENGTH MONTH.LENGTH
           DAY.LENGTH WDAY.LENGTH NO.SECONDS NUMBER.OF.MONTH MONTH.LONG MONTH.LEADING YEAR.LONG DAY.OF.WEEK
           DAY.SHORT CIVILIAN.TIME)
          (if (NOT FORMAT)
              then NIL
            elseif (NEQ (CAR (LISTP FORMAT))
                        'DATEFORMAT)
              then (LISPERROR "ILLEGAL ARG" FORMAT)
            else (for TOKEN in FORMAT
                    do (SELECTQ TOKEN
                               (NO.DATE (SETQ NO.DATE T))
                               (NO.TIME (SETQ NO.TIME T))
                               (NUMBER.OF.MONTH
                                   (SETQ NUMBER.OF.MONTH T))
                               (YEAR.LONG (SETQ YEAR.LONG T))
                               (MONTH.LONG (SETQ MONTH.LONG T))
                               (MONTH.LEADING
                                   (SETQ MONTH.LEADING T))
                               (SLASHES (SETQ SEPR (CHARCODE /)))
                               (SPACES (SETQ SEPR (CHARCODE SPACE)))
                               (NO.LEADING.SPACES
                                   (SETQ NO.LEADING.SPACES T))
                               (TIME.ZONE (SETQ TIME.ZONE (OR [LISTP (CDR (if (FIXP \TimeZoneComp)
                                                                              then (ASSOC \TimeZoneComp TIME.ZONES)
                                                                            else
                                                                                                  ; Ugh, not a small integer
                                                                                   (CL:ASSOC \TimeZoneComp TIME.ZONES
                                                                                          :TEST '=]
                                                                         \TimeZoneComp)))
                               (NO.SECONDS (SETQ NO.SECONDS T))
                               (DAY.OF.WEEK (SETQ DAY.OF.WEEK T))
                               (DAY.SHORT (SETQ DAY.SHORT T))
                               (CIVILIAN.TIME
                                   (SETQ CIVILIAN.TIME T))
                              NIL)))
          (SETQ YEAR.LONG T)                                           ; RMK: Y2K
          [SETQ SIZE
           (+ (if NO.DATE
                  then 0
                else (+ (if MONTH.LEADING
                            then (SETQ SEPR (CHARCODE SPACE))
                                 (SETQ NUMBER.OF.MONTH NIL)            ; Will use a comma
                                 1
                          else 0)
                        (SETQ MONTH.LENGTH
```

```
                    (if NUMBER.OF.MONTH
                        then                                        ; Month input is zero-based
                             (if (AND (< (add MONTH 1)
                                         10)
                                      NO.LEADING.SPACES)
                                 then 1
                                 else 2)
                          else [SETQ MONTH
                                (CL:NTH MONTH
                                       '("January" "February" "March" "April" "May" "June" "July" "August"
                                         "September" "October" "November" "December"]
                                (if MONTH.LONG
                                    then (NCHARS MONTH)
                                    else 3)))
                    (SETQ DAY.LENGTH (if (AND (OR NO.LEADING.SPACES MONTH.LEADING)
                                              (< DAY 10))
                                         then 1
                                         else 2))
                    (SETQ YEAR.LENGTH (if (OR YEAR.LONG (> YEAR 1999))
                                          then 4
                                          else (SETQ YEAR (IREMAINDER YEAR 100))
                                               2))
                    (if DAY.OF.WEEK
                        then [SETQ DAY.OF.WEEK (CL:NTH WDAY '("Monday" "Tuesday" "Wednesday" "Thursday"
                                                              "Friday" "Saturday" "Sunday"]
                             [+ 3 (SETQ WDAY.LENGTH (if DAY.SHORT
                                                        then    ; 3 letters plus " ()"
                                                             3
                                                        else (NCHARS DAY.OF.WEEK]
                          else 0)
                    2))
          (if NO.TIME
              then 0
            else (+ (if NO.DATE
                        then 5
                      else 6)
                    (if NO.SECONDS
                        then 0
                      else 3)
                    (if CIVILIAN.TIME
                        then                                        ; Use AM/PM
                             (SETQ CIVILIAN.TIME (if (> HOUR 11)
                                                     then       ; PM
                                                          (if (> HOUR 12)
                                                              then (add HOUR -12))
                                                          (CHARCODE p)
                                                   else (if (EQ HOUR 0)
                                                            then (SETQ HOUR 12))
                                                        (CHARCODE a)))
                             (if (AND (< HOUR 10)
                                      NO.LEADING.SPACES)
                                 then (SETQ HOUR.LENGTH 1)
                                 else 2)
                          else 0)
                    (if (NULL TIME.ZONE)
                        then 0
                      elseif (NUMBERP TIME.ZONE)
                        then                                        ; Use the -0800 format
                             6
                      else                                          ; Depends on dst: (normal dst).  If missing, we are forced to use
                                                                    ; numeric format
                          (SETQ TIME.ZONE (OR (if DST
                                                  then (CADR TIME.ZONE)
                                                else (CAR TIME.ZONE))
                                              \TimeZoneComp))
                          (ADD1 (SETQ TIME.ZONE.LENGTH (NCHARS TIME.ZONE]
    (SETQ S (ALLOCSTRING SIZE (CHARCODE SPACE)))
    (if (NOT NO.DATE)
        then (if MONTH.LEADING
                 then                                               ; Month day, year
                      (RPLSTRING S 1 MONTH)
                      (SETQ N MONTH.LENGTH)
                      (RPLCHARCODE S (add N 1)
                             SEPR)
                      (\RPLRIGHT S (add N (if (< DAY 10)
                                              then 1
                                              else 2))
                             DAY 1)
                      (RPLCHARCODE S (add N 1)
                             (CHARCODE ","))
               else                                                 ; Day<sepr>month<sepr>year
                    (\RPLRIGHT S (SETQ N DAY.LENGTH)
                           DAY 1)
                    (RPLCHARCODE S (add N 1)
                           SEPR)
                    (if NUMBER.OF.MONTH
                        then (\RPLRIGHT S (add N MONTH.LENGTH)
```

```
                                    MONTH MONTH.LENGTH)
                           else (\OUTDATE-STRING S N MONTH (NOT MONTH.LONG))
                                (add N MONTH.LENGTH)))
                    (RPLCHARCODE S (add N 1)
                           SEPR)
                    (\RPLRIGHT S (add N YEAR.LENGTH)
                           YEAR 2)
                    (OR NO.TIME (add N 1))
                    [if DAY.OF.WEEK
                        then                                        ; Day of week at very end in parens
                             (LET [(START (SUB1 (- SIZE WDAY.LENGTH]
                                  (RPLCHARCODE S START (CHARCODE "("))
                                  (\OUTDATE-STRING S START DAY.OF.WEEK DAY.SHORT)
                                  (RPLCHARCODE S SIZE (CHARCODE ")"]
              else (SETQ N 0))
             [if (NOT NO.TIME)
                 then (\RPLRIGHT S (add N HOUR.LENGTH)
                           HOUR
                           (if CIVILIAN.TIME
                               then 1
                             else 2))
                    (RPLCHARCODE S (ADD1 N)
                           (CHARCODE %:))
                    (\RPLRIGHT S (add N 3)
                           MINUTE 2)
                    (if (NOT NO.SECONDS)
                        then (RPLCHARCODE S (ADD1 N)
                                    (CHARCODE %:))
                             (\RPLRIGHT S (add N 3)
                                    SECOND 2))
                    (if CIVILIAN.TIME
                        then (RPLCHARCODE S (ADD1 N)
                                    CIVILIAN.TIME)
                             (RPLCHARCODE S (add N 2)
                                    (CHARCODE m)))
                    (if TIME.ZONE
                        then (if (NUMBERP TIME.ZONE)
                                 then                               ; +0800 etc
                                      (if DST
                                          then                      ; Daylight savings is in effect, so time zone is off by an hour
                                               (SETQ TIME.ZONE (SUB1 TIME.ZONE)))
                                      (RPLCHARCODE S (+ N 2)
                                             (if (<= TIME.ZONE 0)
                                                 then               ; East of GMT, which is denoted + in this notation
                                                      (SETQ TIME.ZONE (- TIME.ZONE))
                                                      (CHARCODE +)
                                               else (CHARCODE -)))
                                      (if (FIXP TIME.ZONE)
                                          then                      ; integral number of hours
                                               (\RPLRIGHT S (+ N 4)
                                                      TIME.ZONE 2)
                                               (RPLSTRING S (+ N 5)
                                                      "00")
                                        else (CL:MULTIPLE-VALUE-BIND (H M)
                                                 (CL:TRUNCATE TIME.ZONE)
                                               (\RPLRIGHT S (+ N 4)
                                                      H 2)
                                               (\RPLRIGHT S (+ N 6)
                                                      (ROUND (TIMES M 60))
                                                      2)))
                               else (RPLSTRING S (+ N 2)
                                           TIME.ZONE]
             (if STRING
                 then (SUBSTRING S 1 -1 STRING)
               else S])
```

(\OUTDATE-STRING

```
  [LAMBDA (S N STRING SHORTP)                                       ; Edited 18-May-89 18:38 by bvm
    ;; Append STRING to S, using only the first 3 chars if SHORTP is true.  N is the index of the last char appended to S.  Returns new N
    (if SHORTP
        then                                                        ; Use only first 3 chars
             (for I from 1 to 3 do (RPLCHARCODE S (+ N I)
                                          (NTHCHARCODE STRING I)))
      else (RPLSTRING S (ADD1 N)
                  STRING])
```

(\RPLRIGHT

```
  [LAMBDA (S AT N MINDIGITS)                                        (* bvm%: "21-NOV-83 17:19")
    (RPLCHARCODE S AT (IPLUS (CHARCODE 0)
                             (IREMAINDER N 10)))
    (COND
       ((OR (IGREATERP MINDIGITS 1)
            (IGEQ N 10))
        (\RPLRIGHT S (SUB1 AT)
```

```
              (IQUOTIENT N 10)
              (SUB1 MINDIGITS])
```

(\UNPACKDATE

```
[LAMBDA (D)                                                    ; Edited  4-May-89 18:18 by bvm
```

;; Converts an internal Lisp date D into a list of integers (Year Month Day Hours Minutes Seconds daylightp DayOfWeek). D defaults to current ;; date. --- DayOfWeek is zero for Monday --- --- D is first converted to the alto standard, a 32-bit unsigned integer, representing the number of ;; seconds since jan 1, 1901-Gmt. We have to be a little tricky in our computations to avoid the sign bit.

```
(SETQ D (OR D (DAYTIME)))
(PROG ((CHECKDLS \DayLightSavings)
       (DQ (IQUOTIENT (LRSH (LISP.TO.ALTO.DATE D)
                            1)
                      30))
       MONTH SEC HR DAY4 YDAY WDAY YEAR4 TOTALDAYS MIN DLS FRAC)
                                                               ; DQ is number of minutes since day 0, getting us past the sign
                                                               ; bit problem.
      (SETQ SEC (IMOD [+ D (CONSTANT (- 60 (IMOD MIN.FIXP 60]
                      60))
      (SETQ MIN (IREMAINDER DQ 60))
```

;; Now we can adjust to the current time zone. Since this might cause DQ to go negative, first add in 4 years worth of hours, making the base ;; date be Jan 1, 1897

```
      [LET ((ZONE \TimeZoneComp))
           (if (NOT (FIXP ZONE))
               then                                            ; Gack, a non-hour offset.  Use the integer here, then adjust the
                                                               ; minutes, etc.
                    (CL:MULTIPLE-VALUE-SETQ (ZONE FRAC)
                           (CL:FLOOR ZONE)))
           (SETQ HR (IREMAINDER (SETQ DQ (- (+ (IQUOTIENT DQ 60)
                                               (CONSTANT (ITIMES 24 \4YearsDays)))
                                            ZONE))
                                24))
           (if FRAC
               then (SETQ FRAC (ROUND (TIMES FRAC -60)))       ; Minutes to add (time zones are never below the minute offset)
                    (CL:MULTIPLE-VALUE-SETQ (FRAC MIN)
                           (CL:FLOOR (+ MIN FRAC)
                                     60))
                    (if (NEQ FRAC 0)
                        then                                   ; Adjust the hours
                             (CL:MULTIPLE-VALUE-SETQ (FRAC HR)
                                    (CL:FLOOR (+ HR FRAC)
                                              24]
      (SETQ TOTALDAYS (IQUOTIENT DQ 24))
      (if FRAC
          then                                                 ; For non-integral time zones, here's the last of the leftover.
               (add TOTALDAYS FRAC))
 DTLOOP
      (SETQ DAY4 (IREMAINDER TOTALDAYS \4YearsDays))           ; DAY4 = number of days since last leap year day 0
      [SETQ DAY4 (+ DAY4 (CDR (\DTSCAN DAY4 '((789 . 3)
                                              (424 . 2)
                                              (59 . 1)
                                              (0 . 0]          ; pretend every year is a leap year, adding one for days after Feb
                                                               ; 28
      (SETQ YEAR4 (IQUOTIENT TOTALDAYS \4YearsDays))           ; YEAR4 = number of years til that last leap year / 4
      (SETQ YDAY (IREMAINDER DAY4 366))                        ; YDAY is the ordinal day in the year (jan 1 = zero)
      (SETQ WDAY (IREMAINDER (+ TOTALDAYS 3)
                             7))
      (if (AND CHECKDLS (SETQ DLS (\ISDST? YDAY HR WDAY)))
          then ;; This date is during daylight savings, so add 1 hour.  Third arg is day of the week, which we determine by taking days mod 7
               ;; plus offset.  Monday = zero in this scheme.  Jan 1 1897 was actually a Friday (not Thursday=3), but we're cheating--1900
               ;; was not a leap year
               (if (> (SETQ HR (ADD1 HR))
                      23)
                   then ;; overflowed into the next day.  This case is too hard (we might have overflowed the month, for example), so just go
                        ;; back and recompute
                        (SETQ TOTALDAYS (ADD1 TOTALDAYS))
                        (SETQ HR 0)
                        (SETQ CHECKDLS NIL)
                        (GO DTLOOP)))
      [SETQ MONTH (\DTSCAN YDAY '((335 . 11)
                                  (305 . 10)
                                  (274 . 9)
                                  (244 . 8)
                                  (213 . 7)
                                  (182 . 6)
                                  (152 . 5)
                                  (121 . 4)
                                  (91 . 3)
                                  (60 . 2)
                                  (31 . 1)
                                  (0 . 0]                      ; Now return year, month, day, hr, min, sec
      (RETURN (LIST (+ 1897 (ITIMES YEAR4 4)
                       (IQUOTIENT DAY4 366))
```

```
                        (CDR MONTH)
                        (ADD1 (- YDAY (CAR MONTH)))
                        HR MIN SEC DLS WDAY])
```

(\PACKDATE

```
[LAMBDA (YR MONTH DAY HR MIN SEC TIMEZONE)                    ; Edited 22-Mar-88 05:33 by jds
```

;; Packs indicated date into a single integer in Lisp date format. Returns NIL on errors.

```
(PROG (YDAY DAYSSINCEDAY0)
      (COND
         ((NOT (AND YR MONTH DAY HR MIN SEC))                 ; Values missing
          (RETURN)))
      (SETQ DAYSSINCEDAY0 (+ (SETQ YDAY (+ (SELECTQ MONTH
                                                (0 0)
                                                (1 31)
                                                (2 59)
                                                (3 90)
                                                (4 120)
                                                (5 151)
                                                (6 181)
                                                (7 212)
                                                (8 243)
                                                (9 273)
                                                (10 304)
                                                (11 334)
                                                NIL)
                                           (SUB1 DAY)))
                             (TIMES 365 (SETQ YR (- YR 1901)))
                             (IQUOTIENT YR 4)))
      [COND
         ((> MONTH 1)                                         ; After February 28
          (add YDAY 1)                                        ; Day-of-year for dst is based on 366-day year
          (COND
             ((AND (EQ 3 (IREMAINDER YR 4))
                   (NEQ YR -1))                               ; It is a leap year, so real day count also incremented. Note that
                                                              ; YR is years since 1901 at this point
              (add DAYSSINCEDAY0 1]
      (COND
         ((OR (< DAYSSINCEDAY0 -1)
              (< (add HR (TIMES 24 DAYSSINCEDAY0)
                        (COND
                           (TIMEZONE)
                           ((AND \DayLightSavings (\ISDST? YDAY HR (IREMAINDER (+ DAYSSINCEDAY0 1)
                                                                               7)))
                            ;; Subtract one to go from daylight to standard time. This time we computed weekday based on day 0 = Jan 1,
                            ;; 1901, which was a Tuesday = 1
                            (SUB1 \TimeZoneComp))
                           (T \TimeZoneComp)))
                 0))
          ;; Earlier than day 0 -- second check is needed because day 0 west of GMT is sometime during Dec 31, 1900
          (RETURN)))
      (RETURN (+ SEC (PROGN ;; Add the seconds to the converted date, rather than the raw one, and use LLSH instead of multiplying by
                            ;; 60, to avoid creating a bignum
                            (ALTO.TO.LISP.DATE (LLSH (TIMES 30 (+ MIN (TIMES 60 HR)))
                                                     1])
```

(\DTSCAN

```
[LAMBDA (X L)                                                 (* lmm%: 22 NOV 75 1438)
  (PROG NIL
    LP  (COND
           ((IGREATERP (CAAR L)
                       X)
            (SETQ L (CDR L))
            (GO LP)))
        (RETURN (CAR L])
```

(\ISDST?

```
[LAMBDA (YDAY HOUR WDAY)                                      ; Edited 27-Oct-87 18:51 by bvm:
```

;; Returns true if YDAY, HOUR is during the daylight savings period. WDAY is day of week, zero = Monday. YDAY is the ordinal day of the year, ;; pretending it is a leap year, with zero = Jan 1.

;; Unfortunately, \BeginDST and \EndDST are 1-based and so documented, so we have to convert to zero base inside here.

```
(AND (\CHECKDSTCHANGE (add YDAY 1)
             HOUR WDAY \BeginDST)
     (NOT (\CHECKDSTCHANGE YDAY HOUR WDAY \EndDST])
```

(\CHECKDSTCHANGE

```
[LAMBDA (YDAY HOUR WDAY DSTDAY)                               (* bvm%: " 2-NOV-80 15:34")
```

;; Tests to see if YDAY, HOUR is after the start of daylight (or standard) time. WDAY is the day of the week, Monday=zero. DSTDAY is the last ;; day of the month in which time changes, as a YDAY, usually Apr 30 or Oct 31

```
    (COND
       ((IGREATERP YDAY DSTDAY)                                    ; Day is in the next month already
        T)
       ((ILESSP YDAY (IDIFFERENCE DSTDAY 6))                       ; day is at least a week before end of month, so time hasn't
                                                                   ; changed yet
        NIL)
       ((EQ WDAY 6)
        ;; It's Sunday, so time changes today at 2am.  Check for hour being past that.  Note that there is a hopeless ambiguity when the time is
        ;; between 1:00 and 2:00 am the day that DST goes into effect, as that hour happens twice
        (IGREATERP HOUR 1))
       (T                                                          ; okay if last Monday (YDAY-WDAY) is less than a week before
                                                                   ; end of month
          (IGREATERP (IDIFFERENCE YDAY WDAY)
                 (IDIFFERENCE DSTDAY 6])

)


(DEFOPTIMIZER DATEFORMAT (&REST X)
                         (KWOTE (CONS 'DATEFORMAT X)))

;; Default values for \BeginDST and \EndDST are set for (most places in) the U.S., 74 and 312 as of 2021.
;;  Note: this might not be relevant to users with local time servers that do the right thing.

(RPAQ? \TimeZoneComp 8)

(RPAQ? \BeginDST 74)

(RPAQ? \EndDST 312)

(RPAQ? \DayLightSavings T)

(ADDTOVAR TIME.ZONES
          (8 "PST" "PDT")
          (7 "MST" "MDT")
          (6 "CST" "CDT")
          (5 "EST" "EDT")
          (0 "GMT" "BST")
          (0 "UT")
          (-1 "MET" "MET DST")
          (-2 "EET" "EET DST"))

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \TimeZoneComp \BeginDST \EndDST \DayLightSavings TIME.ZONES)
)

(DECLARE%: EVAL@COMPILE

(RPAQ \4YearsDays (ADD1 (ITIMES 365 4)))

[CONSTANTS (\4YearsDays (ADD1 (ITIMES 365 4]
)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(PUTPROPS IOCHAR FILETYPE CL:COMPILE-FILE)

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

(ADDTOVAR NLAMA DATEFORMAT)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA PACK* CONCAT)
)

(PUTPROPS IOCHAR COPYRIGHT ("Venue & Xerox Corporation" 1981 1982 1983 1984 1985 1986 1987 1988 1990 1991 2018
                                2020))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

PROPERTY INDEX

MACRO INDEX

OPTIMIZER INDEX