File created: 30-Jul-2023 17:42:27 **{WMEDLEY}<sources>LLREAD.;105**

edit by: rmk

changes to: (FNS \SUBREAD)

previous date: 17-Jun-2023 13:12:06 {WMEDLEY}<sources>LLREAD.;104

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LLREADCOMS
       [(COMS                                                    ; Reader entrypoints
              (FNS LASTC PEEKC PEEKCCODE RATOM READ READC READCCODE READP SETREADMACROFLG SKIPSEPRCODES
                   SKIPSEPRS SKREAD))
        (COMS                                                    ; CommonLisp read entry points
              (FNS CL:READ CL:READ-PRESERVING-WHITESPACE CL:READ-DELIMITED-LIST CL:PARSE-INTEGER)
              (GLOBALVARS CMLRDTBL))
        (COMS                                                    ; reading strings
              (FNS RSTRING READ-EXTENDED-TOKEN \RSTRING2))
        [COMS                                                    ; Core of the reader
              (FNS \TOP-LEVEL-READ \SUBREAD \SUBREADCONCAT \ORIG-READ.SYMBOL \ORIG-INVALID.SYMBOL
                   \APPLYREADMACRO INREADMACROP)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD? '\ORIG-READ.SYMBOL '\READ.SYMBOL)
                                                 (MOVD? '\ORIG-INVALID.SYMBOL '\INVALID.SYMBOL]
        (COMS                                                    ; Read macro for '
              (FNS READQUOTE))
        (COMS                                                    ; # macro
              (FNS READVBAR READHASHMACRO DEFMACRO-LAMBDA-LIST-KEYWORD-P DIGITBASEP READNUMBERINBASE
                   ESTIMATE-DIMENSIONALITY SKIP.HASH.COMMENT CMLREAD.FEATURE.PARSER))
        (COMS                                                    ; Reading characters with #\
              (FNS CHARACTER.READ CHARCODE.DECODE)
              (FNS HEXNUM? OCTALNUM?)
              (ALISTS (CHARACTERNAMES Page Form FF Rubout Del Null Escape Esc Bell Tab Backspace Bs Newline CR
                             EOL Return Tenexeol Space Sp Linefeed LF)
                      (CHARACTERSETNAMES Meta Function Greek Cyrillic Hira Hiragana Kata Katakana Kanji)))
        (DECLARE%: DOEVAL@COMPILE DONTCOPY (CONSTANTS * READTYPES)
               (MACROS .CALL.SUBREAD. FIXDOT RBCONTEXT PROPRB \RDCONC)
               (SPECVARS *READ-NEWLINE-SUPPRESS* \RefillBufferFn)
               (GLOBALVARS *KEYWORD-PACKAGE* *INTERLISP-PACKAGE*))
        (COMS (INITVARS (*REPLACE-NO-FONT-CODE* T)
                       (*DEFAULT-NOT-CONVERTED-FAT-CODE* 8739))
              (GLOBALVARS *REPLACE-NO-FONT-CODE* *DEFAULT-NOT-CONVERTED-FAT-CODE*))
        (INITVARS (*READ-NEWLINE-SUPPRESS*)
               (\RefillBufferFn (FUNCTION \READCREFILL)))
                                                                 ; Top level val of \RefillBufferFn means act like READC--we must
                                                                 ; be doing a raw BIN (or PEEKBIN?)
        (LOCALVARS . T)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA CL:PARSE-INTEGER
                                                                                    CL:READ-DELIMITED-LIST
                                                                                    CL:READ-PRESERVING-WHITESPACE
                                                                                    CL:READ])
```

;; Reader entrypoints

```
(DEFINEQ
```

(LASTC

```
  [LAMBDA (FILE)                                                 ; Edited  3-May-2021 16:45 by rmk:
    (LET [(LASTCCODE (FETCH (STREAM LASTCCODE) OF (\GETSTREAM FILE 'INPUT]
         (COND
            ((IEQP LASTCCODE 65535)
             NIL)
            (T (FCHARACTER LASTCCODE])
```

(PEEKC

```
  [LAMBDA (FILE FLG)                                             (* rmk%: "10-Apr-85 11:55")
```

;; FLG says to proceed as if Control were T--not implemented correctly here NIL

```
    (LET [(\RefillBufferFn (FUNCTION \PEEKREFILL))
          (STREAM (\GETSTREAM FILE 'INPUT]
         (DECLARE (SPECVARS \RefillBufferFn))
         (FCHARACTER (PEEKCCODE STREAM])
```

(PEEKCCODE

```
  [LAMBDA (FILE NOERROR)                                         ; Edited 19-Jul-2022 23:36 by rmk
```

```
                                                                    ; Edited  3-May-2021 16:47 by rmk:
    (LET ((\RefillBufferFn (FUNCTION \PEEKREFILL)))
         (DECLARE (SPECVARS \RefillBufferFn))
         (\PEEKCCODE.EOLC (\GETSTREAM FILE 'INPUT)
                NOERROR])
```

(RATOM

```
  [LAMBDA (FILE RDTBL)                                              ; Edited 30-Mar-87 17:21 by bvm:
```

;;; Like READ except interpret break characters as single character atoms. I.e., always returns an atom

```
    (SETQ RDTBL (\GTREADTABLE RDTBL))
    (LET ((*READTABLE* RDTBL)
          (*PACKAGE* (if (fetch (READTABLEP USESILPACKAGE) of RDTBL)
                         then *INTERLISP-PACKAGE*
                       else *PACKAGE*))
          (\RefillBufferFn (FUNCTION \RATOM/RSTRING-REFILL)))
         (DECLARE (SPECVARS *READTABLE* *PACKAGE* \RefillBufferFn))
         (WITH-RESOURCE (\PNAMESTRING)
                (\SUBREAD (\GETSTREAM FILE 'INPUT)
                       (fetch (READTABLEP READSA) of *READTABLE*)
                       RATOM.RT \PNAMESTRING (AND (fetch (READTABLEP CASEINSENSITIVE) of *READTABLE*)
                                                  (fetch (ARRAYP BASE) of UPPERCASEARRAY))
                       NIL NIL NIL T])
```

(READ

```
  [LAMBDA (FILE RDTBL FLG)                                          ; Edited 19-Mar-87 18:35 by bvm:
    (LET ((*READTABLE* (\GTREADTABLE RDTBL))
          (*READ-NEWLINE-SUPPRESS* FLG))
         (DECLARE (SPECVARS *READTABLE* *READ-NEWLINE-SUPPRESS*))
```

;; *READ-NEWLINE-SUPPRESS* is used freely by \FILLBUFFER

;; Call reader with PRESERVE-WHITESPACE = T, since that's the semantics Interlisp has always had before (though maybe not explicitly ;; stated).

```
         (\TOP-LEVEL-READ FILE NIL NIL NIL T])
```

(READC

```
  [LAMBDA (FILE RDTBL)                                              ; Edited  6-Aug-2021 21:38 by rmk:
    (SETQ FILE (\GETSTREAM FILE 'INPUT))
    (LET ((*READTABLE* (\GTREADTABLE RDTBL))
          (\RefillBufferFn (FUNCTION \READCREFILL))
          (CODE (\INCCODE.EOLC FILE)))
         (DECLARE (SPECVARS *READTABLE* \RefillBufferFn))
         (CL:WHEN (\CHARCODEP CODE)                                 ; If not a charcode, we must have run off the end with an
                                                                    ; ENDOFSTREAMOP
             (freplace (STREAM LASTCCODE) of FILE with CODE)
             (FCHARACTER CODE))])
```

(READCCODE

```
  [LAMBDA (STREAM RDTBL)                                            ; Edited  6-Aug-2021 21:39 by rmk:
```

;;; returns a 16 bit character code. \INCHAR does the EOL conversion. Saves the character for LASTC as well.

```
    (SETQ STREAM (\GETSTREAM STREAM 'INPUT))
    (LET ((*READTABLE* (\GTREADTABLE RDTBL))
          (\RefillBufferFn (FUNCTION \READCREFILL))
          (CODE (\INCCODE.EOLC STREAM)))
         (DECLARE (SPECVARS *READTABLE* \RefillBufferFn))
         (CL:WHEN (\CHARCODEP CODE)                                 ; If not a charcode, we must have run off the end with an
                                                                    ; ENDOFSTREAMOP
             (freplace (STREAM LASTCCODE) of STREAM with CODE))
        CODE])
```

(READP

```
  [LAMBDA (FILE FLG)                                                (* rmk%: " 5-Apr-85 09:09")
                                                                    ; The 10 does not do the EOL check on the peeked character.
    (LET* ((STREAM (\GETSTREAM FILE 'INPUT))
           (DEVICE (ffetch (STREAM DEVICE) of STREAM)))
          (COND
             ((ffetch (FDEV READP) of DEVICE)
              (FDEVOP 'READP DEVICE STREAM FLG))
             (T (\GENERIC.READP STREAM FLG])
```

(SETREADMACROFLG

```
  [LAMBDA (FLG)                                                     (* rmk%: "25-OCT-83 16:13")
                                                                    ; D doesn't cause the read-macro context error, hence doesn't
                                                                    ; maintain this flag
    NIL])
```

(SKIPSEPRCODES

```
  [LAMBDA (FILE RDTBL)                                              ; Edited 19-Jul-2022 23:36 by rmk
                                                                    ; Edited 18-Jun-2021 11:38 by rmk:
```

;; Passes over non-separators to peek at the first non-separator on FILE. Returns either last peeked character, or NIL if no non-seprs left in the ;; file.

;; Assumes that CR and LF are both seprs so that no EOL processing is needed.

```
    (bind PREVC C (STRM _ (\GETSTREAM FILE 'INPUT))
          (SA _ (fetch (READTABLEP READSA) of (\GTREADTABLE RDTBL)))
          (\RefillBufferFn _ '\PEEKREFILL) declare (SPECVARS \RefillBufferFn)
       while [EQ SEPRCHAR.RC (\SYNCODE SA (SETQ C (OR (\PEEKCCODE.EOLC STRM T)
                                                      (RETURN]
       do (SETQ PREVC C)
          (\INCCODE STRM)
       finally (AND PREVC (replace (STREAM LASTCCODE) of STRM with PREVC))
              (RETURN C])
```

(SKIPSEPRS

```
  [LAMBDA (FILE RDTBL)                                              ; Edited 18-Jun-2021 11:39 by rmk:
```

;; Passes over non-separators to peek at the first non-separator on FILE. Returns either last peeked character, or NIL if no non-seprs left in the ;; file.

```
    (LET (C)
         (AND (SETQ C (SKIPSEPRCODES FILE RDTBL))
              (FCHARACTER C])
```

(SKREAD

```
  [LAMBDA (FILE REREADSTRING RDTBL)                                 ; Edited  6-Apr-88 11:06 by amd
    (LET ((*READ-SUPPRESS* 'SKREAD)
          (*READTABLE* (\GTREADTABLE RDTBL))
          (\RBFLG)
          (STRM (\GETSTREAM FILE 'INPUT))
          CH)
         (DECLARE (CL:SPECIAL *READTABLE* *READ-SUPPRESS* \RBFLG))
         [COND
            (REREADSTRING                                           ; REREADSTRING is string of chars already read.
                   (SETQ STRM (CL:MAKE-CONCATENATED-STREAM (CL:MAKE-STRING-INPUT-STREAM (MKSTRING REREADSTRING))
                                     STRM]                          ; Because of return requirements, have to preview stream for
                                                                    ; unbalanced closing bracket/paren
         (if (NULL (SETQ CH (SKIPSEPRCODES STRM)))
             then (\EOF.ACTION STRM)
           else (SELECTC (PROG1 (\SYNCODE (fetch (READTABLEP READSA) of *READTABLE*)
                                          CH)
                                ;; Read in suppressed mode.  Reader sets \Rbflg free if read ended on unbalanced bracket.  Reason we do the
                                ;; READ in all cases is so that we need to consume the unbalanced paren/bracket, just as if we really had read it;
                                ;; however, READ doesn't set \Rbflg for these cases
                                (\TOP-LEVEL-READ STRM NIL NIL NIL T))
                        (RIGHTPAREN.RC                              ; unbalanced right paren
                            '%))
                        (RIGHTBRACKET.RC                            ; unbalanced right bracket
                            '%])
                        (AND \RBFLG '%]])

)

;; CommonLisp read entry points

(DEFINEQ

## (CL:READ

  [CL:LAMBDA (&OPTIONAL (INPUT-STREAM *STANDARD-INPUT*)
                        (EOF-ERROR-P T)
                        EOF-VALUE RECURSIVE-P)                      ; Edited 14-Dec-86 18:48 by bvm
         (COND
            (RECURSIVE-P                                            ; Dive straight into reader using current settings of everything
                   (.CALL.SUBREAD. INPUT-STREAM))
            (T (\TOP-LEVEL-READ INPUT-STREAM (NOT EOF-ERROR-P)
                      EOF-VALUE])

## (CL:READ-PRESERVING-WHITESPACE

  [CL:LAMBDA (&OPTIONAL (STREAM *STANDARD-INPUT*)
                        (EOF-ERRORP T)
                        (EOF-VALUE NIL)
                        (RECURSIVEP NIL))                           ; Edited 19-Mar-87 18:33 by bvm:
```

;; Reads from stream and returns the object read, preserving the whitespace that followed the object.

```
         (COND
            (RECURSIVEP                                             ; Dive straight into reader using current settings of everything
                   (.CALL.SUBREAD. STREAM))
            (T (\TOP-LEVEL-READ STREAM (NOT EOF-ERRORP)
                      EOF-VALUE NIL T])
```

(CL:READ-DELIMITED-LIST

```
  [CL:LAMBDA (CHAR &OPTIONAL (INPUT-STREAM *STANDARD-INPUT*)
                    RECURSIVE-P)                                    ; Edited 14-Dec-86 18:48 by bvm
```

;;; Read a list of elements terminated by CHAR. CHAR must not be a separator char, and ideally should not be a constituent char (if it is, it must be ;;; preceded by whitespace for READ-DELIMITED-LIST to work)

```
          (LET [(ENDCODE (OR (FIXP CHAR)
                             (CL:CHAR-CODE CHAR)))
                (INSTREAM (\GETSTREAM INPUT-STREAM 'INPUT]
               (if RECURSIVE-P
                   then                                             ; Have to dive into reader without disturbing
                                                                    ; *CIRCLE-READ-LIST*
                       (.CALL.SUBREAD. INPUT-STREAM NIL NIL ENDCODE)
                 else (\TOP-LEVEL-READ INPUT-STREAM NIL NIL ENDCODE])
```

(CL:PARSE-INTEGER

```
  [CL:LAMBDA (STRING &KEY START END (RADIX 10)
                     JUNK-ALLOWED)                                  ; Edited 20-Aug-2021 00:02 by rmk:
          (CL:IF (NOT (CL:STRINGP STRING))
              (ERROR "This is not a string : ~S" STRING)
              (PROG ((SA (fetch (READTABLEP READSA) of CMLRDTBL))
                     (BASE (fetch (STRINGP BASE) of STRING))
                     (LEN (fetch (STRINGP LENGTH) of STRING))
                     (OFFST (fetch (STRINGP OFFST) of STRING))
                     (FATP (fetch (STRINGP FATSTRINGP) of STRING))
                     MAXDIGITCODE MAXALPHACODE INDEX STOP CHAR SIGN STARTINT ENDINT ERR)
                    (SETQ RADIX (\CHECKRADIX RADIX))
                    (SETQ INDEX (+ OFFST (if (NULL START)
                                             then 0
                                           elseif (< START 0)
                                             then (\ILLEGAL.ARG START)
                                           else START)))
                    (SETQ STOP (+ OFFST (if (NULL END)
                                            then LEN
                                          elseif (OR (> END LEN)
                                                     (< END 0))
                                            then (\ILLEGAL.ARG END)
                                          else END)))
                    (SETQ MAXDIGITCODE (+ (CHARCODE 0)
                                          RADIX -1))
                    (SETQ MAXALPHACODE (AND (> RADIX 10)
                                            (+ (CHARCODE A)
                                               RADIX -11)))
                    (while (AND (< INDEX STOP)
                                (EQ (\SYNCODE SA (\GETBASECHAR FATP BASE INDEX))
                                    SEPRCHAR.RC))
                       do                                           ; Skip over separators
                          (SETQ INDEX (CL:1+ INDEX)))
                    [COND
                       ((>= INDEX STOP)                             ; no characters remain
                        (RETURN (COND
                                    (JUNK-ALLOWED                   ; don't error
                                           (CL:VALUES NIL STOP))
                                    (T (SETQ ERR "No non-whitespace characters in integer string: ~S")
                                       (GO FAIL]
                ;; Start parsing a number.  Allowed to start with a single sign, then digits in radix, nothing else.  Assume collating sequence is (+, -) <
                ;; digits < uppercase letters < lowercase letters.
                    (do (SETQ CHAR (\GETBASECHAR FATP BASE INDEX))
                        (if (<= CHAR MAXDIGITCODE)
                            then                                    ; sign or digit
                                 (if (>= CHAR (CHARCODE 0))
                                     then                           ;  digit
                                          (OR STARTINT (SETQ STARTINT INDEX))
                                   elseif (AND (NOT SIGN)
                                               (NOT STARTINT))
                                     then                           ; maybe sign.  No good if not at start
                                          (SELCHARQ CHAR
                                                (- (SETQ SIGN '-))
                                                (+ (SETQ SIGN '+))
                                                (RETURN))
                                   else (RETURN))
                          elseif (AND MAXALPHACODE (<= (if (>= CHAR (CHARCODE "a"))
                                                           then     ; uppercase it first
                                                                (- CHAR (- (CHARCODE "a")
                                                                           (CHARCODE "A")))
                                                         else CHAR)
                                                       MAXALPHACODE))
                            then                                    ; is alphabetic digit
                                 (OR STARTINT (SETQ STARTINT INDEX))
                          else (RETURN))
                       repeatwhile (< (add INDEX 1)
                                      STOP))
                    (SETQ ENDINT INDEX)
                    (RETURN (CL:VALUES (COND
```

```
                                        ([AND STARTINT (OR JUNK-ALLOWED (EQ INDEX STOP)
                                                           (do (if (NEQ (\SYNCODE SA CHAR)
                                                                        SEPRCHAR.RC)
                                                                   then
                                                               ; junk found
                                                                   (RETURN NIL)
                                                                elseif (EQ (add INDEX 1)
                                                                           STOP)
                                                                   then
                                                               ; at end of string, win
                                                                   (RETURN T)
                                                                else (SETQ CHAR (\GETBASECHAR FATP BASE INDEX]
                                         (\MKINTEGER BASE STARTINT ENDINT (EQ SIGN '-)
                                                RADIX FATP))
                                        (JUNK-ALLOWED NIL)
                                        ((NULL STARTINT)
                                         (SETQ ERR "There aren't any digits in this integer string: ~S.")
                                         (GO FAIL))
                                        (T (SETQ ERR "There is junk in this integer string: ~S.")
                                           (GO FAIL)))
                                  (- INDEX OFFST)))
              FAIL
                  (CL:ERROR ERR (if (OR START END)
                                    then (CL:SUBSEQ STRING (OR START 0)
                                                   (OR END LEN))
                                  else STRING)))])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS CMLRDTBL)
)

;; reading strings

(DEFINEQ

(RSTRING
  [LAMBDA (FILE RDTBL RSFLG)                                    ; Edited 22-Mar-87 20:53 by bvm:
    (LET ((*READTABLE* (\GTREADTABLE RDTBL))
          (\RefillBufferFn '\RATOM/RSTRING-REFILL)
          (*READ-SUPPRESS* NIL))
         (DECLARE (SPECVARS *READTABLE* \RefillBufferFn *READ-SUPPRESS*))
         ;; It's not clear that *READ-SUPPRESS* is supposed to affect anything other than calls to READ.  So play it safe and force \Rstring2 to
         ;; really read a string.
         (WITH-RESOURCE (\PNAMESTRING)
                (\RSTRING2 (\GETSTREAM FILE 'INPUT)
                       (fetch READSA of *READTABLE*)
                       (OR RSFLG T)
                       \PNAMESTRING])

(READ-EXTENDED-TOKEN
  [LAMBDA (STRM RDTBL ESCAPE-ALLOWED-P)                         ; Edited  6-Aug-2021 21:39 by rmk:
    ;; This is a cross between RSTRING and \SUBREAD.  Read a "token" from STREAM, as defined by the Common Lisp reader and the syntax in
    ;; RDTBL.  EOF terminates as well.  If ESCAPE-ALLOWED-P is true, escapes are honored and if one appears, a second value of T is returned.
    ;; Otherwise, escapes are treated as vanilla chars and the caller can barf on them itself if it desires.
    (SETQ RDTBL (\GTREADTABLE RDTBL))
    (WITH-RESOURCE (\PNAMESTRING)
           (PROG ((CASEBASE (AND (fetch (READTABLEP CASEINSENSITIVE) of RDTBL)
                                 (fetch (ARRAYP BASE) of UPPERCASEARRAY)))
                  (PBASE (ffetch (STRINGP XBASE) of \PNAMESTRING))
                  (J 0)
                  (SA (fetch READSA of RDTBL))
                  CH SNX ANSLIST ANSTAIL ESCAPE-APPEARED ESCAPING FATSEEN)
             LP   (if (\EOFP STRM)
                      then                                      ; end of file terminates string just like a sepr/break
                           (GO FINISH))
                  (SETQ CH (\INCCODE STRM))                     ; NOTE: This should really be (\INCHAR --) --), but eol is usually
                                                                ; a break or sepr and the \BACKNSCHAR doesn't work right.  Fix
                                                                ; this when we unread correctly
                  (SETQ SNX (\SYNCODE SA CH))
                  [COND
                     ((AND ESCAPE-ALLOWED-P (SELECTC SNX
                                                (ESCAPE.RC (SETQ CH (\INCCODE.EOLC STRM))
                                                           (SETQ ESCAPE-APPEARED T))
                                                (MULTIPLE-ESCAPE.RC
                                                     (SETQ ESCAPING (NOT ESCAPING))
                                                     (SETQ ESCAPE-APPEARED T)
                                                     (GO LP))
                                                NIL)))
                     (ESCAPING                                  ; eat chars until next |
                            )
                     ((fetch STOPATOM of SNX)
```

```
                    (\BACKCCODE STRM)
                    (GO FINISH))
                   ((AND CASEBASE (ILEQ CH \MAXTHINCHAR))
                    (SETQ CH (\GETBASEBYTE CASEBASE CH]
                (COND
                   ((EQ J \PNAMELIMIT)                              ; Filled PNSTR so have to save those chars away and start filling
                                                                    ; up a new buffer
                    (SETQ J (\SMASHSTRING (ALLOCSTRING J NIL NIL FATSEEN)
                                  0 \PNAMESTRING J))
                    [COND
                       [ANSLIST (RPLACD ANSTAIL (SETQ ANSTAIL (CONS J NIL]
                       (T (SETQ ANSTAIL (SETQ ANSLIST (CONS J NIL]
                    (SETQ J 0)))
                (\PNAMESTRINGPUTCHAR PBASE J CH)
                (COND
                   ((AND (NOT FATSEEN)
                         (IGREATERP CH \MAXTHINCHAR))
                    (SETQ FATSEEN T)))
                (SETQ J (ADD1 J))
                (GO LP)
            FINISH
                (SETQ J (\SMASHSTRING (ALLOCSTRING J NIL NIL FATSEEN)
                              0 \PNAMESTRING J))
                [COND
                   (ANSLIST (RPLACD ANSTAIL (SETQ ANSTAIL (CONS J NIL)))
                            (SETQ J (CONCATLIST ANSLIST]
                (RETURN (if ESCAPE-APPEARED
                            then                                    ; do it this way because multiple values are slow
                                 (CL:VALUES J T)
                          else J])
```

(\RSTRING2

```
  [LAMBDA (STRM SA RSFLG PNSTR)                                     ; Edited 13-Aug-2021 13:35 by rmk:
```

;;; The main string reader. Reads characters from STREAM according to the syntax table SA and returns a string. PNSTR is an instance of the global ;;; resource \PNAMESTRING, which we can use all to ourselves as a buffer.

;;; If RSFLG is T then the call is from RSTRING, in which case the string is terminated by a break or sepr in SA. If RSFLG is NIL then the string is ;;; terminated by a string delimiter. If RSFLG is SKIP then CR's and the following separator chars are discarded as an otherwise normal string is read

```
    (DECLARE (USEDFREE *READTABLE* *READ-SUPPRESS*))
    (PROG ((EOLC (ffetch EOLCONVENTION of STRM))
           (PBASE (ffetch (STRINGP XBASE) of PNSTR))
           (J 0)
           CH SNX ANSLIST ANSTAIL LASTC FATSEEN SKIPPING)
      RS2LP
          (SETQ CH (\INCCODE.EOLC STRM))
          [COND
             ((EQ CH (CHARCODE EOL))
              ;; We have eaten a CR, LF, or CRLF depending on the EOL convention of STRM, and recognized it as an EOL. If EOL is a stopatom
              ;; character, we terminate the read and backup over the just read character(s) so they can be read again.
              ;; An escaped LF is handled below, stays as LF even from an LF file.
              (COND
                 ([AND (EQ RSFLG T)
                       (fetch STOPATOM of (\SYNCODE SA (CHARCODE EOL]
                  ;; From RSTRING, eol terminates read, but EOL character(s) is/are left to be read again.
                  (\BACKCCODE.EOLC STRM)
                  (GO FINISH]
          (SETQ SNX (\SYNCODE SA CH))
          (SELECTC SNX
               (OTHER.RC                                            ; Normal case, nothing to do
                         )
               (ESCAPE.RC                                           ; Read the escaped character
                          ;; \PRINSTRING puts an escape % before an LF in the string, whether or not it is going to an LF or CR file. An
                          ;; EOL(CR) will be printed as LF on an LF file or CRLF, otherwise left alone. \CHECKEOLC will return EOL for an LF
                          ;; on an LF file, because it doesn't know about escapes. On a CR or an LF file, a CR will come in as an EOL. So the
                          ;; trick here is: don't call \CHECKEOLC on an escaped LF, no matter what the EOL convention of the file..
                          [COND
                             ((fetch ESCAPEFLG of *READTABLE*)
                              (SETQ CH (\INCCODE STRM))
                              (COND
                                 ((EQ CH (CHARCODE LF))             ; An escaped LF stays as an LF, even from a LF file.
                                  (GO PUTCHAR))
                                 (T (SETQ CH (\CHECKEOLC CH EOLC STRM))
                                    (COND
                                       ((AND (EQ RSFLG 'SKIP)
                                             (EQ CH (CHARCODE EOL)))
                                                                    ; Strip leading spaces after escaped returns, too, but leave the
                                                                    ; CR in the string
                                        (SETQ SKIPPING 0)
                                        (GO PUTCHAR])
               (SELECTQ RSFLG
                    (NIL                                            ; end check is dbl quote
```

```
                        (COND
                           ((EQ SNX STRINGDELIM.RC)                    ; Got it
                            (SETQ LASTC CH)
                            (GO FINISH))))
                  (T                                                   ; if called from RSTRING, end check is break or sepr, and we
                                                                       ; must leave delim in stream
                     (COND
                        ((fetch STOPATOM of SNX)
                         (\BACKCCODE STRM)
                         (GO FINISH))))
                  (SKIP                                                ; Like NIL but strip cr's and leading spaces
                        (SELECTC SNX
                            (STRINGDELIM.RC
                                 (SETQ LASTC CH)
                                 (GO FINISH))
                            (SEPRCHAR.RC                               ; Assume that CR is a sepr
                                        (COND
                                           [SKIPPING (COND
                                                       ((EQ CH (CHARCODE EOL))
                                                                       ; Multiple CR's while skipping are kept
                                                        (COND
                                                           ((EQ SKIPPING T)
                                                                       ; Turn previous space back into CR.  Note that J is guaranteed to
                                                                       ; be at least 1
                                                            (\PNAMESTRINGPUTCHAR PBASE (SUB1 J)
                                                                   CH)
                                                            (SETQ SKIPPING 0)))
                                                        (GO PUTCHAR))
                                                       (T     ; Continue skipping seprs
                                                          (GO RS2LP]
                                           ((EQ CH (CHARCODE EOL))
                                                                       ; Turn CR into space and start skipping seprs
                                            (SETQ SKIPPING T)
                                            (SETQ CH (CHARCODE SPACE))
                                            (GO PUTCHAR))))
                                  NIL))
                  (SHOULDNT)))
        (SETQ SKIPPING NIL)
     PUTCHAR
        [COND
           ((NOT *READ-SUPPRESS*)                                      ; Accumulate character
            (COND
               ((EQ J \PNAMELIMIT)                                     ; Filled PNSTR so have to save those chars away and start filling
                                                                       ; up a new buffer
                (SETQ J (\SMASHSTRING (ALLOCSTRING J NIL NIL FATSEEN)
                              0 PNSTR J))
                [COND
                   [ANSLIST (RPLACD ANSTAIL (SETQ ANSTAIL (CONS J NIL]
                   (T (SETQ ANSTAIL (SETQ ANSLIST (CONS J NIL]
                (SETQ J 0)))
            (\PNAMESTRINGPUTCHAR PBASE J CH)
            (SETQ LASTC CH)
            (COND
               ((AND (NOT FATSEEN)
                     (IGREATERP CH \MAXTHINCHAR))
                (SETQ FATSEEN T)))
            (SETQ J (ADD1 J]
        (COND
           ((OR (NEQ RSFLG T)
                (NOT (\EOFP STRM)))                                    ; in RSTRING (RSFLG=T), if we've read something already, then
                                                                       ; end of file terminates string just like a sepr/break
            (GO RS2LP)))
     FINISH
        (AND LASTC (freplace (STREAM LASTCCODE) of STRM with LASTC))
        (RETURN (COND
                   ((NOT *READ-SUPPRESS*)
                    (SETQ J (\SMASHSTRING (ALLOCSTRING J NIL NIL FATSEEN)
                                  0 PNSTR J))
                    (COND
                       (ANSLIST (RPLACD ANSTAIL (SETQ ANSTAIL (CONS J NIL)))
                              (CONCATLIST ANSLIST))
                       (T J])

)

;; Core of the reader

(DEFINEQ

(\TOP-LEVEL-READ
  [LAMBDA (STREAM EOF-SUPPRESS EOF-VALUE CHAR PRESERVE-WHITESPACE)
                                                                       ; Edited 13-Dec-88 16:28 by jds
    ;; Entry to the guts of the reader from a place where you may not be already under the reader.  CHAR is for READ-DELIMITED-LIST -- it is
    ;; charcode to terminate read, in which case we are reading a sequence of things instead of a single thing.  EOF-SUPPRESS is the opposite of
    ;; CL:READ's EOF-ERROR-P arg.

    ;;  I EOF-SUPPRESS, set the stream's EODOFSTREAMOP to retfrom here with EOF-VALUE as its result.
```

```
    (LET ((*PACKAGE* (COND
                         ((fetch (READTABLEP USESILPACKAGE) of (\DTEST *READTABLE* 'READTABLEP))
                          *INTERLISP-PACKAGE*)
                         (T *PACKAGE*)))
          (\RefillBufferFn (FUNCTION \READREFILL))
          (*CIRCLE-READ-LIST* NIL)
          (OLD-EOS-OP (fetch ENDOFSTREAMOP of STREAM)))
         (DECLARE (SPECVARS *PACKAGE* \RefillBufferFn *CIRCLE-READ-LIST* EOF-VALUE))
         (CL:UNWIND-PROTECT
             (PROGN [AND EOF-SUPPRESS (REPLACE ENDOFSTREAMOP OF STREAM WITH #'(LAMBDA (STREAM)
                                                                                 (RETFROM '\TOP-LEVEL-READ
                                                                                    EOF-VALUE]
                    (LET ((RESULT (.CALL.SUBREAD. STREAM EOF-SUPPRESS EOF-VALUE CHAR PRESERVE-WHITESPACE)))
                         (if *CIRCLE-READ-LIST*
                             then                                  ; There were calls to #=, so go fix up all the ## references.
                                  (HASH-STRUCTURE-SMASH RESULT))
                         RESULT))
             (REPLACE ENDOFSTREAMOP OF STREAM WITH OLD-EOS-OP))])
```

(\SUBREAD

```
  [LAMBDA (STRM SA READTYPE PNSTR CASEBASE EOF-SUPPRESS EOF-VALUE CHAR PRESERVE-WHITESPACE)
                                                                  ; Edited 30-Jul-2023 17:42 by rmk
                                                                  ; Edited 19-Jul-2022 23:36 by rmk
                                                                  ; Edited  6-Aug-2021 21:40 by rmk:
```

;; Values of READTYPE are: --- READ.RT for top level of READ, --- NOPROPRB.RT if right-bracket isn't to be propagated -- sublist beginning with ;; left-bracket --- PROPRB.RT if propagation is not suppressed -- sublist beginning with left-paren --- RATOM.RT for call from RATOM

;; PNSTR is an instance of the global resource \PNAMESTRING, acquired in READ and passed on from level to level. It is released during ;; read-macro applications, then reacquired.

;; CASEBASE is base of uppercasearray if read table is case-insensitive.

;; If EOF-SUPPRESS is true, then if we are at end of file we should return EOF-VALUE instead of erroring (we need this because we might actually ;; be sitting before end of file in front of something that reads nothing, e.g., a comment, so caller can't check EOFP itself). Always false on ;; recursive calls.

;; If CHAR is supplied, it is a character code which, when read (in isolation), should terminate this call to read. Never on when at top-level.

;; \RBFLG is propagated for top-level calls, in case they are embedded in read-macros. SKREAD also depends on this.

;; If PRESERVE-WHITESPACE is true, doesn't throw away the whitespace that terminates the read.

```
    (DECLARE (USEDFREE *READTABLE* \RBFLG))
```

;; \RDCONC is a macro that adds a new element as specified by its first argument to the current sublist. Its other arguments will be executed ;; instead if we are the top-level call

```
    (PROG ((TOPLEVELP (SELECTC READTYPE
                              ((LIST READ.RT RATOM.RT)
                                  T)
                              NIL))
           (PBASE (ffetch (STRINGP XBASE) of PNSTR))
           SNX LST END ELT DOTLOC CH J ESCAPEFLG INVALIDFLG PACKAGE NCOLONS AT-EOF EOF-POSSIBILITY EXTRASEGMENTS
           LASTC)
          (if (AND TOPLEVELP (NOT (\INTERMP STRM)))
              then ;; EOF is allowed to terminate tokens on direct READ calls.  Not if reading from terminal, because \FILLBUFFER made sure to
                   ;; put something at the end.
                   (SETQ EOF-POSSIBILITY T))
      NEWTOKEN
```

;; Here ready to scan a new token. First skip over separator characters

```
          (SETQ J 0)
          [SETQ EXTRASEGMENTS (SETQ INVALIDFLG (SETQ ESCAPEFLG (SETQ PACKAGE (SETQ NCOLONS NIL]
          (if (AND EOF-SUPPRESS (NULL (SKIPSEPRCODES STRM)))
              then                                                ; caller specified eof-error-p of NIL.  Happens only on top-level
                                                                  ; calls
                   (RETURN EOF-VALUE))                            ; By Skipping Separator Characters,Happens CHARSET-Mode
                                                                  ; Exchanging. (Solution of AR#114 in FX, edited by tt
                                                                  ; [Jan-22-'90])
          (repeatwhile (EQ [SETQ SNX (\SYNCODE SA (SETQ CH (\INCCODE.EOLC STRM]
                           SEPRCHAR.RC))
          (COND
             ((EQ CH CHAR)                                        ; Read desired terminating char.  TOPLEVELP is always false
                                                                  ; here
              (freplace (STREAM LASTCCODE) of STRM with CH)       ; Save last char for LASTC.
              (RETURN LST))
             ((EQ SNX OTHER.RC)                                   ; Start of an atom
              (COND
                 ([AND (EQ CH (CHARCODE %.))
                       (fetch STOPATOM of (\SYNCODE SA (\PEEKCCODE.EOLC STRM]
                  ;; An isolated, unescaped dot.  This special check on every atom could be eliminated if . had a special SNX code
                  (SETQ DOTLOC END)                               ; DOTLOC points to CONS cell one before the dot, NIL for car of
                                                                  ; list, as desired.

                  ))
              (GO GOTATOMCHAR))
```

```
          [(fetch STOPATOM of SNX)                                  ; This character definitely does not start an atom
           (COND
              ((EQ READTYPE RATOM.RT)
               (GO SINGLECHARATOM))
              (T (GO BREAK]
          ((EQ SNX PACKAGEDELIM.RC)                                 ; Starting a symbol with a package delimiter -- must be a
                                                                    ; keyword
           (SETQ NCOLONS 1)
           (SETQ PACKAGE *KEYWORD-PACKAGE*)
           (SETQ ESCAPEFLG T)
           (GO NEXTATOMCHAR))
          [(AND (SELECTC (fetch MACROCONTEXT of SNX)
                     (FIRST.RMC T)
                     (ALONE.RMC (fetch STOPATOM of (\SYNCODE SA (\PEEKCCODE.EOLC STRM))))
                     NIL)
                (fetch READMACROFLG of *READTABLE*))
           (COND
              ((EQ READTYPE RATOM.RT)
               (GO SINGLECHARATOM))
              (T (GO MACRO]
          (T                                                        ; Some character that starts an atom but has non-trivial syntax
                                                                    ; attributes
             ))
    ATOMLOOP

;; At this point, we are accumulating an atom, and CH does not have syntax OTHER, so we have to check special cases
        (SELECTC SNX
            (ESCAPE.RC                                              ; Take next character to be alphabetic, case exact
                       (COND
                          ((fetch ESCAPEFLG of *READTABLE*)
                           (SETQ CH (\INCCODE.EOLC STRM))           ; No EOFP check needed -- it's an error to have escape char
                                                                    ; with nothing following
                           (SETQ ESCAPEFLG T)
                           (GO PUTATOMCHAR))))
            (MULTIPLE-ESCAPE.RC ;; Take characters up to next multiple escape to be alphabetic, except that single escape chars still
                                ;; escape the next char
                (SETQ ESCAPEFLG T)
                [bind ESCFLG do (SETQ CH (\INCCODE.EOLC STRM))
                                (COND
                                   ([NOT (COND
                                            (ESCFLG (SETQ ESCFLG NIL))
                                            (T (SELECTC (SETQ SNX (\SYNCODE SA CH))
                                                   (MULTIPLE-ESCAPE.RC
                                                                    ; Finished escaped sequence, resume normal processing
                                                        (GO NEXTATOMCHAR))
                                                   (ESCAPE.RC
                                                                    ; Pass the next char thru verbatim
                                                                     (SETQ ESCFLG T))
                                                  NIL]              ; All others are pname chars, quoted
                                    (if (NOT *READ-SUPPRESS*)
                                        then (COND
                                                ((EQ J \PNAMELIMIT)
                                                                    ; if there have been escapes, can't be a number, so ok to error
                                                                    ; now.
                                                 (LISPERROR "ATOM TOO LONG" (\SUBREADCONCAT EXTRASEGMENTS
                                                                                            PBASE J))
                                                 (GO NEWTOKEN)))
                                    (\PNAMESTRINGPUTCHAR PBASE J CH)
                                    (add J 1])
            NIL)
    GOTATOMCHAR

;; CH is a vanilla atom char to accumulate
        [COND
           ((AND CASEBASE (ILEQ CH \MAXTHINCHAR))                   ; Uppercase atom characters
            (SETQ CH (\GETBASEBYTE CASEBASE CH]
    PUTATOMCHAR
        (if (NOT *READ-SUPPRESS*)
            then (COND
                    ((EQ J \PNAMELIMIT)                             ; Symbol is too long.  However, it could just be a bignum, so
                                                                    ; keep accumulating characters until we have to do something.
                     (push EXTRASEGMENTS (\SMASHSTRING (ALLOCSTRING J NIL NIL T)
                                                       0 PNSTR J))
                     (SETQ J 0)))
                (\PNAMESTRINGPUTCHAR PBASE J CH)
                (add J 1)
                (SETQ LASTC CH)                                     ; Save CH for LASTC.
                )
    NEXTATOMCHAR
        (if (AND EOF-POSSIBILITY (SETQ AT-EOF (\EOFP STRM)))
            then                                                    ; EOF terminates atoms at top level
                 (GO FINISHATOM)
          elseif (EQ [SETQ SNX (\SYNCODE SA (SETQ CH (\INCCODE.EOLC STRM]
                     OTHER.RC)
```

```
            then                                                    ; normal case tested first--another vanilla constituent char, so
                                                                    ; keep accumulating atom chars
                 (GO GOTATOMCHAR)
           elseif (fetch STOPATOM of SNX)
            then                                                    ; Terminates atom
                 (GO FINISHATOM)
           elseif (EQ SNX PACKAGEDELIM.RC)
            then (GO GOTPACKAGEDELIM)
           else (GO ATOMLOOP))
     FINISHATOM
```

;; Come here when an atom has been terminated, either by a break/sepr char or by end of file.

```
     (if INVALIDFLG
         then (freplace (STREAM LASTCCODE) of STRM with (OR LASTC CH 65535))
              (\INVALID.SYMBOL PBASE J NCOLONS PACKAGE EXTRASEGMENTS))
     [SETQ ELT (AND (NOT *READ-SUPPRESS*)
                    (if EXTRASEGMENTS
                        then ;; More than \PNAMELIMIT chars were read.  Can't be a symbol, but might be a number.  Pack up all the
                             ;; strings we have into a single string and try to parse it as a number.
                             (SETQ EXTRASEGMENTS (\SUBREADCONCAT EXTRASEGMENTS PBASE J))
                             (OR (AND (NULL (OR PACKAGE ESCAPEFLG NCOLONS))
                                      (\PARSE.NUMBER (fetch (STRINGP BASE) of EXTRASEGMENTS)
                                             (fetch (STRINGP OFFST) of EXTRASEGMENTS)
                                             (fetch (STRINGP LENGTH) of EXTRASEGMENTS)
                                             \FATPNAMESTRINGP))
                                 (LISPERROR "ATOM TOO LONG" EXTRASEGMENTS))
                      else (\READ.SYMBOL PBASE 0 J \FATPNAMESTRINGP PACKAGE (EQ NCOLONS 1)
                                ESCAPEFLG]
     (freplace (STREAM LASTCCODE) of STRM with CH)                  ; Save last READ char for LASTC.
     (if AT-EOF
         then                                                       ; top-level read, atom terminated by EOF
              (RETURN ELT))
     (\RDCONC ELT (PROGN (COND
                           ((OR PRESERVE-WHITESPACE (NEQ SNX SEPRCHAR.RC))
                                                                    ; At top-level, put back the terminating character if preserving
                                                                    ; whitespace or terminator is significant
                            (freplace (STREAM LASTCCODE) of STRM with (OR LASTC CH 65535))
                                                                    ; And LASTC will return the last REAL char read.
                            (\BACKCCODE STRM)))
                         (RETURN ELT)))
     (if (EQ SNX SEPRCHAR.RC)
         then                                                       ; Terminated with sepr, go on to next char
              (GO NEWTOKEN)
       elseif (EQ CH CHAR)
         then                                                       ; read terminates here
              (freplace (STREAM LASTCCODE) of STRM with CH)
              (RETURN LST)
       else                                                         ; Terminated with break, jump into the break char code
            (GO BREAK))
     GOTPACKAGEDELIM
```

;; Come here if CH is a package delimiter. Note that we have already scanned at least one character of the token, so this must be an interior
;; delim

```
     (COND
        (*READ-SUPPRESS*                                            ; Don't care about packages
          )
        [(AND (EQ J 0)
              (NULL EXTRASEGMENTS))
         ;; No chars accumulated, so must be 2 colons in a row.  Note that the case where we've just started scanning a token happens up at
         ;; NEWTOKEN
         (SETQ LASTC CH)
         (COND
            ((AND (EQ NCOLONS 1)
                  (NEQ PACKAGE *KEYWORD-PACKAGE*))                  ; Two colons in a row means internal symbol
             (SETQ NCOLONS 2))
            (T                                                      ; Error, e.g., 'FOO:::BAZ' or '::BAR'
               (SETQ INVALIDFLG T)
               (GO GOTATOMCHAR]
        ((NULL NCOLONS)                                             ; We have just scanned the package name
         (SETQ NCOLONS 1)
         (SETQ LASTC CH)
         [SETQ PACKAGE (COND
                         (EXTRASEGMENTS (LISPERROR "ATOM TOO LONG" (\SUBREADCONCAT EXTRASEGMENTS PBASE J
                                                                                  ))
                                        (SETQ EXTRASEGMENTS NIL))
                         ((\FIND.PACKAGE.INTERNAL PBASE 0 J \FATPNAMESTRINGP))
                         (T                                         ; Error, but don't signal yet -- save name as string for benefit of
                                                                    ; error handlers
                            (\GETBASESTRING PBASE 0 J \FATPNAMESTRINGP]
         (SETQ J 0))
        (T                                                          ; Have alread seen one or more colons, and have scanned more
                                                                    ; symbol.  This colon is an error.
           (SETQ LASTC CH)
```

```
            (SETQ INVALIDFLG T)
            (GO GOTATOMCHAR)))
      (SETQ ESCAPEFLG T)                                    ; Result MUST be a symbol now
      (GO NEXTATOMCHAR)
 SINGLECHARATOM

;; Come here to create a symbol whose single character is CH -- no package stuff to worry about.  This happens mainly for RATOM.  We create
;; the single char atom in IL for backward compatibility.

      (\PNAMESTRINGPUTCHAR PBASE 0 CH)
      (SETQ ELT (\READ.SYMBOL PBASE 0 1 \FATPNAMESTRINGP *INTERLISP-PACKAGE*))
      (freplace (STREAM LASTCCODE) of STRM with CH)
      (\RDCONC ELT (RETURN ELT))
      (GO NEWTOKEN)

;; End of atom scanning code

 BREAK

;; At this point, we have just read a break character, stored in CH

      (freplace (STREAM LASTCCODE) of STRM with CH)
      [SELECTC SNX
           (LEFTPAREN.RC
                          ;; recursively read a list.  If that list (or any of it's non-bracketed sublists) is terminated by a right bracket it
                          ;; terminates our read as well.  PROPRB macro worries about right-bracket propagation: if the subread
                          ;; encounters a right bracket (sets \RBFLG), PROPRB returns true.  In addition, if we were not called by a
                          ;; left-bracket (READTYPE = NOPROPRB.RT) it sets \RBFLG in caller, thereby propagating the bracket upward.
                          (COND
                             ((PROG1 (PROPRB (SETQ ELT (\SUBREAD STRM SA PROPRB.RT PNSTR CASEBASE)))
                                  (\RDCONC ELT (RETURN ELT)))
                              ;; PROG1 is true if the subread encountered a right bracket
                              (FIXDOT)                      ; Fix dotted pair if necessary
                              (RETURN LST))))
           (LEFTBRACKET.RC ;; recursively read a list, terminated by either right paren or right bracket.  In this case, right bracket is not
                           ;; propagated upward--we continue reading elements after it.
                (SETQ ELT (\SUBREAD STRM SA NOPROPRB.RT PNSTR CASEBASE))
                (\RDCONC ELT (RETURN ELT)))
           ((LIST RIGHTPAREN.RC RIGHTBRACKET.RC)
                                ;; Terminate one or more lists, return what we have accumulated so far.  In the case of Right bracket,
                                ;; if caller did not have the matching left bracket, we have to allow the bracket to close more than one
                                ;; list.
                (RETURN (COND
                           (TOPLEVELP
                                ;; Naked right paren/bracket returns NIL.  This is sort of bogus in common lisp, but changing it would
                                ;; be a significant change to Interlisp folks.
                                NIL)
                           (CHAR ;; call from READ-DELIMITED-LIST doesn't want to terminate this way.  Could read as NIL and not
                                 ;; terminate, but seems best to error.
                                 (CL:ERROR "Unmatched ~A encountered while reading to a ~A" (CL:CODE-CHAR
                                                                                             CH)
                                        (CL:CODE-CHAR CHAR))
                                 LST)
                           (T (FIXDOT)
                              (AND (EQ SNX RIGHTBRACKET.RC)
                                   (NEQ READTYPE NOPROPRB.RT)
                                   (SETQ \RBFLG T))
                              LST))))
           (STRINGDELIM.RC ;; Invoke string reader
                (SETQ ELT (\RSTRING2 STRM SA NIL PNSTR))
                (\RDCONC ELT (RETURN ELT)))
           (COND
              ((OR (EQ SNX BREAKCHAR.RC)
                   (NOT (fetch READMACROFLG of *READTABLE*)))  ; A breakchar or a disabled always macro
               (GO SINGLECHARATOM))
              (T (GO MACRO]
      (GO NEWTOKEN)
 MACRO
      (SELECTQ (fetch MACROTYPE of (SETQ SNX (\GETREADMACRODEF CH *READTABLE*)))
           (MACRO (COND
                     ((PROG1 (PROPRB [SETQ ELT (RELEASERESOURCE \PNAMESTRING PNSTR (CL:MULTIPLE-VALUE-LIST
                                                                                    (\APPLYREADMACRO STRM
                                                                                                     SNX]
                                                                                    ; Ignore right-bracket if macro is called at top-level read

                                     )
                             [COND
                                ((NULL ELT)                 ; Macro returned zero values, read as nothing

                                 )
                                (T (SETQ ELT (CAR ELT))
                                   (\RDCONC ELT (RETURN ELT])
```

```
                        (FIXDOT)                                    ; Encountered right bracket if we get here -- return what we have
                        (RETURN LST))))
            (INFIX      ;; We give macro TCONC list of what we've accumulated so far--it gets to modify it as it pleases and return it.  We continue
                        ;; from there.
                        (COND
                           ((PROG1 [PROPRB (SETQ ELT (RELEASERESOURCE \PNAMESTRING PNSTR
                                                                  (\APPLYREADMACRO STRM SNX (AND LST (CONS LST END]
                                 [COND
                                    [TOPLEVELP                      ; What does INFIX mean at top level??  See IRM
                                           (COND
                                              ((AND (LISTP ELT)
                                                    (CDR ELT))      ; Result is in TCONC format, so it's returnable
                                               (RETURN (COND
                                                         ((EQ (CDR ELT)
                                                              (CAR ELT))
                                                                    ; TCONC list of one element--return the element.  This is how
                                                                    ; INFIX top level macro can return a non-list.
                                                          (CAAR ELT))
                                                         (T (CAR ELT]
                                    (T                              ; Reading sublist.  Take apart TCONC list and continue.
                                       (SETQ LST (CAR ELT))
                                       (SETQ END (CDR ELT])
                           (FIXDOT)                                 ; Macro hit right bracket if we got to here
                           (RETURN LST))))
            (SPLICE     ;; Macro returns arbitrary number of values to be spliced inline.
                        [RBCONTEXT (SETQ ELT (RELEASERESOURCE \PNAMESTRING PNSTR (\APPLYREADMACRO STRM SNX]
                                                                    ; Note: we don't care if there was terminating right-bracket
                                                                    ; Why? -bvm
                        (COND
                           ((OR (NULL ELT)
                                TOPLEVELP)
                            ;; On the 10, it actually returns ELT if it is a list and the next token is a closing paren or bracket.  Hard to see how to
                            ;; get that behavior--rmk
                            (GO NEWTOKEN))
                           ((NLISTP ELT)                            ; The 10 throws initial non-lists away (What if LST/END aren't
                                                                    ; set?)
                            (SETQ ELT (AND LST (LIST '%. ELT)))
                            (SETQ DOTLOC END)))
                        [COND
                           ((NOT *READ-SUPPRESS*)
                            (COND
                               (LST (RPLACD END ELT))
                               (T (SETQ LST ELT)))
                            (SETQ END (LAST ELT))
                            (COND
                               ((CDR END)                           ; A dotted pair
                                (SETQ DOTLOC END)
                                (RPLACD END (CONS '%. (SETQ END (CONS (CDR END])
            (SHOULDNT))
        (GO NEWTOKEN])
```

(**SUBREADCONCAT**

```
[LAMBDA (EXTRASEGMENTS PBASE J)                                     ; Edited 16-Jan-87 15:08 by bvm:
```

;; Produces a string consisting of all the characters \SUBREAD has been buffering up into a token. Last J chars are stored at PBASE.
;; EXTRASEGMENTS is a list of strings in reverse order in the case that more characters were scanned than the pname string accommodates.

```
(SETQ PBASE (\GETBASESTRING PBASE 0 J \FATPNAMESTRINGP))
(if EXTRASEGMENTS
    then (CONCATLIST (NCONC1 (REVERSE EXTRASEGMENTS)
                             PBASE))
  else PBASE])
```

(**ORIG-READ.SYMBOL**

```
[LAMBDA (BASE OFFSET LEN FATP PACKAGE EXTERNALP NONNUMERICP)    (* bvm%: " 3-Aug-86 15:25")
```

;;; Read a number or symbol from the string defined by BASE OFFSET LEN FATP PACKAGE is NIL if no package was specified, a package object or a
;;; string if an unknown package was typed (causes error). EXTERNALP is true if symbol was typed with one colon, which requires that the symbol exist
;;; and be external. NONNUMERICP is true if we know the symbol is not a number, e.g., some characters in it were escaped.

;;; For now a dummy definition

```
    (COND
       (PACKAGE                                                     ; For debugging
              (CONCAT PACKAGE (COND
                                 (EXTERNALP ":")
                                 (T "::"))
                      (\GETBASESTRING BASE OFFSET LEN FATP)))
       (T (OR (AND (NOT NONNUMERICP)
                   (\PARSE.NUMBER BASE OFFSET LEN FATP))
              (\MKATOM BASE OFFSET LEN FATP T])
```

(\ORIG-INVALID.SYMBOL

```
  [LAMBDA (BASE LEN NCOLONS PACKAGE EXTRASEGMENTS)                ; Edited 15-Jan-87 17:33 by bvm:

;;; Called when scanning a symbol that has more than 2 colons, or more than 1 non-consecutive colon.  If return from here, will read the symbol as
;;; though the extra colons were escaped.

    (CL:CERROR "Treat the extra colon(s) as if they were escaped" "Invalid symbol syntax in %"~A%""
           (CONCAT (if (AND PACKAGE (NEQ PACKAGE *KEYWORD-PACKAGE*))
                       then (if (STRINGP PACKAGE)
                                then PACKAGE
                              else (CL:PACKAGE-NAME PACKAGE))
                     else "")
                   (SELECTQ NCOLONS
                       (1 ":")
                       (2 "::")
                       "")
                   (\SUBREADCONCAT EXTRASEGMENTS BASE LEN])
```

(\APPLYREADMACRO

```
  [LAMBDA (STREAM MACDEF ANSCELL)                                 (* bvm%: " 4-May-86 16:38")
                                                                  ; INREADMACROP searches for this framename
    (DECLARE (USEDFREE *READTABLE*))
    (APPLY* (fetch MACROFN of MACDEF)
           STREAM *READTABLE* ANSCELL])
```

(INREADMACROP

```
  [LAMBDA NIL                                                     (* edited%: "26-MAY-79 00:12")
    (PROG (TEM (\READDEPTH -1))
          (DECLARE (SPECVARS \READDEPTH))
          (COND
             ([NULL (SETQ TEM (STKPOS '\APPLYREADMACRO]
              (RETURN NIL)))
          (MAPDL [FUNCTION (LAMBDA (NM POS)
                           (COND
                              ((EQ NM '\SUBREAD)
                               (SETQ \READDEPTH (ADD1 \READDEPTH]
                 TEM)
          (RELSTK TEM)
          (RETURN \READDEPTH])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD? '\ORIG-READ.SYMBOL '\READ.SYMBOL)

(MOVD? '\ORIG-INVALID.SYMBOL '\INVALID.SYMBOL)
)

;; Read macro for '

(DEFINEQ
```

(READQUOTE

```
  [LAMBDA (FILE)                                                  ; Edited 19-Mar-87 16:10 by bvm:
    (LIST 'QUOTE (CL:READ FILE T NIL T])

)

;; # macro

(DEFINEQ
```

(READVBAR

```
  [LAMBDA (STREAM RDTBL)                                          (* bvm%: "14-May-86 17:31")

;;; Read Interlisp's | macro.  Originally this char was just a sepr in FILERDTBL but was then extended in various hokey ways, because it was the only
;;; character plausibly available for redefinition.  Today it is extended still further to be Common Lisp # in all the cases not already taken by some other
;;; meaning

    (SELCHARQ (PEEKCCODE STREAM)
          (%'                                                     ; commonlisp defines #'X to mean (FUNCTION X), but here it's
                                                                  ; BQUOTE
              (READCCODE STREAM)
              (READBQUOTE STREAM RDTBL))
          ((%( { ^)                                               ; Used by HPRINT
              (HREAD STREAM))
          (%# (READCCODE STREAM)                                  ; |# = Common Lisp #
              (READHASHMACRO STREAM RDTBL))
          ((EOL TAB SPACE)                                        ; CR or tab, treat as separator
              (CL:VALUES))
          (PROGN                                                  ; Everything else not already preempted by old-style | is
                                                                  ; interpreted as Common Lisp
                 (READHASHMACRO STREAM RDTBL])
```

```
(READHASHMACRO
  [LAMBDA (STREAM RDTBL INDEX)                                    (* amd "15-Oct-86 16:36")
```

;;; Implements the standard # macro dispatch -- reads next character to find out what to do. Can return zero values if we just want to skip something.

```
    (LET ([READFN (COND
                      ((fetch (READTABLEP COMMONLISP) of RDTBL)
                       ;; Kludge: if we have to recursively read something that will not end up as the resulting list structure, use the reader that
                       ;; passes thru CMLTRANSLATE
                       (FUNCTION CL:READ))
                      (T (FUNCTION READ]
           NEXTCHAR READVAL)
          [while (DIGITCHARP (SETQ NEXTCHAR (PEEKCCODE STREAM RDTBL)))
             do (SETQ INDEX (PLUS (TIMES (OR INDEX 0)
                                         10)
                                  (DIFFERENCE (READCCODE STREAM RDTBL)
                                         (CHARCODE 0]
          (SELCHARQ NEXTCHAR
               ("(" [LET ((CONTENTS (APPLY* READFN STREAM)))
                         (COND
                            (INDEX (FILL-VECTOR (CL:MAKE-ARRAY INDEX)
                                                CONTENTS))
                            (T (CL:MAKE-ARRAY (LENGTH CONTENTS)
                                              :INITIAL-CONTENTS CONTENTS])
               (PROGN                                                 ; Those cases we left the dispatching char in buffer for
                                                                      ; convenience of the next read.  Now eat it
                      (SELCHARQ (READCCODE STREAM RDTBL)
                           (%' (LIST 'FUNCTION (READ STREAM RDTBL)))
                           (%. (EVAL (APPLY* READFN STREAM)))
                           (%, (LIST 'LOADTIMECONSTANT (READ STREAM RDTBL)))
                           (\ (CHARACTER.READ STREAM))
                           ("*"                                       ; Read bit vector
                                [LET [(CONTENTS (while (MEMQ (PEEKCCODE STREAM RDTBL)
                                                             (CHARCODE (0 1)))
                                                   collect (IDIFFERENCE (READCCODE STREAM RDTBL)
                                                                        (CHARCODE 0]
                                     (COND
                                        (INDEX (FILL-VECTOR (CL:MAKE-ARRAY INDEX :ELEMENT-TYPE 'BIT)
                                                            CONTENTS))
                                        (T (CL:MAKE-ARRAY (LENGTH CONTENTS)
                                                          :INITIAL-CONTENTS CONTENTS :ELEMENT-TYPE 'BIT])
                           (":" ;; The same thing HASH-COLON does.
                                (CL:MAKE-SYMBOL (READ-EXTENDED-TOKEN STREAM RDTBL)))
                           ((O o)
                                (READNUMBERINBASE STREAM 8))
                           ((B b)
                                (READNUMBERINBASE STREAM 2))
                           ((X x)
                                (READNUMBERINBASE STREAM 16))
                           ((R r)
                                (READNUMBERINBASE STREAM INDEX))
                           ((A a)
                                (LET ((CONTENTS (APPLY* READFN STREAM)))
                                     (CL:MAKE-ARRAY (ESTIMATE-DIMENSIONALITY INDEX CONTENTS)
                                            :INITIAL-CONTENTS CONTENTS)))
                           ((S s)
                                (CREATE-STRUCTURE (APPLY* READFN STREAM)))
                           ((C c)
                                (DESTRUCTURING-BIND (NUM DEN)
                                       (APPLY* READFN STREAM)
                                       (COMPLEX NUM DEN)))
                           (+                                         ; Skip expression if feature not present
                              (COND
                                 ((NOT (CMLREAD.FEATURE.PARSER (READ STREAM RDTBL)))
                                  (CL:READ STREAM RDTBL)))
                              (CL:VALUES))
                           (-                                         ; Skip expression if feature IS present
                              (COND
                                 ((CMLREAD.FEATURE.PARSER (READ STREAM RDTBL))
                                  (CL:READ STREAM RDTBL)))
                              (CL:VALUES))
                           ("|"                                       ; special comment
                                (SKIP.HASH.COMMENT STREAM RDTBL)
                                (CL:VALUES))
                           (< (ERROR "#< construct is un-READ-able" (READ)))
                           ((SPACE TAB NEWLINE PAGE RETURN %))
                                (ERROR "Illegal read syntax " (CHARCODE.UNDECODE NEXTCHAR)))
                           (%"                                        ; An extension -- read string without cr's and leading spaces
                               (RSTRING STREAM RDTBL 'SKIP))
                           (APPLY* (OR (GET (CHARACTER NEXTCHAR)
                                            'HASHREADMACRO)
                                       (ERROR "Undefined hashmacro char" NEXTCHAR))
                                  STREAM RDTBL])
```

(DEFMACRO-LAMBDA-LIST-KEYWORD-P

```
[LAMBDA (S)                                                    (* bvm%: " 3-Nov-86 15:12")
  (AND (FMEMB S '(&OPTIONAL &REST &KEY &ALLOW-OTHER-KEYS &AUX &BODY &WHOLE))
       T])
```

(DIGITBASEP

```
[LAMBDA (CODE RADIX)                                           (* lmm "11-Jun-85 00:54")
  (COND
     ((AND (GEQ CODE (CHARCODE 0))
           (LESSP CODE (PLUS (CHARCODE 0)
                             RADIX)))
      (DIFFERENCE CODE (CHARCODE 0)))
     ((GREATERP RADIX 10)
      [COND
         ((AND (GEQ CODE (CHARCODE a))
               (LEQ CODE (CHARCODE z)))
          (add CODE (DIFFERENCE (CHARCODE A)
                                (CHARCODE a]
      (COND
         ((AND (GEQ CODE (CHARCODE A))
               (LEQ CODE (CHARCODE Z)))
          [SETQ CODE (PLUS 10 (DIFFERENCE CODE (CHARCODE A]
          (COND
             ((LESSP CODE RADIX)
              CODE])
```

(READNUMBERINBASE

```
[LAMBDA (STREAM RADIX)                                         (* bvm%: " 4-Nov-86 21:34")
  (PROG ((BODY (READ-EXTENDED-TOKEN STREAM))
         (I 1)
         CH VAL NUMERATOR SIGN BASE)                           ; First check for leading sign
        (if *READ-SUPPRESS*
            then                                               ; work is done
                 (RETURN NIL))
        (SELCHARQ (SETQ CH (NTHCHARCODE BODY 1))
             (+ (GO NEXTCH))
             (- (SETQ SIGN T)
                (GO NEXTCH))
             NIL)
   LP   (if (SETQ BASE (DIGITBASEP CH RADIX))
            then (SETQ VAL (+ (TIMES (OR VAL 0)
                                     RADIX)
                              BASE))
          elseif (EQ CH (CHARCODE "/"))
            then                                               ; Ratio marker
                 (if (OR NUMERATOR (NULL VAL))
                     then (GO MALFORMED))
                 (SETQ NUMERATOR VAL)
                 (SETQ VAL NIL)
          else                                                 ; Terminated by a character that is not a token delimiter
               (GO MALFORMED))
   NEXTCH
        (if (SETQ CH (NTHCHARCODE BODY (add I 1)))
            then (GO LP)
          else                                                 ; end of token, fall thru
               )
   DONE
        (if (NULL VAL)
            then (GO MALFORMED))
        (if NUMERATOR
            then (SETQ VAL (%%/ NUMERATOR VAL)))
        (RETURN (if SIGN
                    then (- VAL)
                  else VAL))
   MALFORMED
        (RETURN (CL:ERROR "Malformed base ~D rational ~S" RADIX BODY])
```

(ESTIMATE-DIMENSIONALITY

```
[LAMBDA (RANK CONTENTS)                                        (* bvm%: " 9-May-86 16:06")
  (COND
     ((NULL RANK)
      (ERROR "No rank found while reading array" NIL))
     ((EQ RANK 0)
      NIL)
     (T (to RANK as (D _ CONTENTS) by (CAR D) collect (LENGTH D])
```

(SKIP.HASH.COMMENT

```
[LAMBDA (STREAM RDTBL)                                         (* bvm%: "12-Sep-86 21:02")
  (PROG NIL
```

;; a tiny fsm that recognizes #| ... |# with possible nestings of itself

```
      LP  (SELCHARQ (READCCODE STREAM RDTBL)
                ("#" (GO SHARP))
                ("|" (GO VBAR))
                (GO LP))
      SHARP
          (SELCHARQ (READCCODE STREAM RDTBL)
                ("|"                                          ; #| -- recursively skip nested section
                     (SKIP.HASH.COMMENT STREAM RDTBL)
                     (GO LP))
                ("#" (GO SHARP))
                (GO LP))
      VBAR
          (SELCHARQ (READCCODE STREAM RDTBL)
                ("|" (GO VBAR))
                ("#"                                          ; found closing |#
                     (RETURN))
                (GO LP])
```

(CMLREAD.FEATURE.PARSER

```
  [LAMBDA (EXPR)                                              (* bvm%: " 3-Nov-86 15:07")
    (COND
       ((CL:CONSP EXPR)
        (SELECTQ (CAR EXPR)
             ((:AND AND)
                  (EVERY (CDR EXPR)
                         (FUNCTION CMLREAD.FEATURE.PARSER)))
             ((:OR OR)
                  (SOME (CDR EXPR)
                        (FUNCTION CMLREAD.FEATURE.PARSER)))
             ((:NOT NOT)
                  (NOT (CMLREAD.FEATURE.PARSER (CADR EXPR))))
             (ERROR "Bad feature expression" EXPR)))
       ((FMEMB EXPR *FEATURES*)
        T])

)

;; Reading characters with #\

(DEFINEQ

(CHARACTER.READ
  [LAMBDA (STREAM)                                            (* bvm%: " 4-Nov-86 21:50")

;;; Called by the #\ macro -- reads a character object consisting of the thing next named

    (LET ((NEXTCHAR (READCCODE STREAM))
          CH)
         (COND
            ((OR (NULL (SETQ CH (PEEKCCODE STREAM T)))
                 (fetch STOPATOM of (\SYNCODE (fetch READSA of *READTABLE*)
                                              CH)))          ; Terminates next, so it's just this char
             (CL:CODE-CHAR NEXTCHAR))
            (*READ-SUPPRESS*                                  ; don't try to decode it, could be illegal
                   (READ-EXTENDED-TOKEN STREAM)
                   NIL)
            (T                                                ; Read a whole name, up to the next break/sepr
               (CL:CODE-CHAR (CHARCODE.DECODE (CONCAT (ALLOCSTRING 1 NEXTCHAR)
                                                      (READ-EXTENDED-TOKEN STREAM])
```

(CHARCODE.DECODE

```
  [LAMBDA (C NOERROR)                                         ; Edited 24-Aug-2021 10:03 by rmk:
                                                              ; Edited 18-Feb-87 22:03 by bvm:
    (DECLARE (GLOBALVARS CHARACTERNAMES CHARACTERSETNAMES))
    ;; RMK 2020: Added hexstring decoding for Unicode: no commas or other delimiters
    ;; RMK 2021: Moved single chars above atom test to be more precise about digits.
    ;; Moved Unicode up, out of comma testing, allowed lower-case u.
    ;; Also disallowed unknown junk in the parse-integer strings and substrings so we know what's happening
    (COND
       ((NOT C)
        NIL)
       ((LISTP C)
        (CONS (CHARCODE.DECODE (CAR C)
                       NOERROR)
              (CHARCODE.DECODE (CDR C)
                       NOERROR)))
       ((EQ (NCHARS C)
            1)                                                ; Includes singleton digits 0-9, the only FIXP's allowed. 0 is 0, not
                                                              ; 48
        (CHCON1 C))
       ((NOT (OR (LITATOM C)
                 (STRINGP C)))                                ; LITATOM instead of ATOM stops numbers right here.
```

```
        (AND (NOT NOERROR)
             (ERROR "BAD CHARACTER SPECIFICATION" C)))
       ((HEXNUM? C T))
       (T (SELCHARQ (CHCON1 C)
              (^ (AND (SETQ C (CHARCODE.DECODE (SUBSTRING C 2 -1)
                                               NOERROR))
                      (LOGAND C (LOGNOT 96))))
              (%# ;; We use IPLUS instead of LOGOR here because some people want ##char to read as Xerox Meta, i.e., 1,char
                  ;; RMK:  I don't understand that comment: "X,#a" would map to the high panel corresponding to "a" in any character set X,
                  ;; including Meta or Function, wherever they happen to be.  Won't adding and orring be the same?
                  (AND (SETQ C (CHARCODE.DECODE (SUBSTRING C 2 -1)
                                                NOERROR))
                       (IPLUS C 128)))
              (for X in CHARACTERNAMES when (STRING.EQUAL (CAR X)
                                                          C)
                 do (RETURN (OR (NUMBERP (CADR X))
                                (CHARCODE.DECODE (CADR X)
                                                 NOERROR)))
                 finally (RETURN (LET ([POS (find I from 1 suchthat (FMEMB (OR (NTHCHARCODE C I)
                                                                               (RETURN))
                                                                           (CHARCODE (%, - %. %|]
                                            CH CSET SSTR)              ; In the form charset,char
                                           ;; Don't use STRPOSL because CHARTABLE is not available in loadup sequence.
                                           ;; The character set loop is like the character loop with a different search list and no recursion for
                                           ;; character sets.
                                           (COND
                                              ((AND POS (SETQ CH (OR [OCTALNUM? (SETQ SSTR (SUBSTRING C (ADD1 POS]
                                                                     (CHARCODE.DECODE SSTR NOERROR)))
                                                    (< CH 256)
                                                    (>= CH 0)
                                                    (SETQ CSET (OR [OCTALNUM? (SETQ SSTR (SUBSTRING C 1 (SUB1 POS]
                                                                   (CADR (find PAIR in CHARACTERSETNAMES
                                                                               suchthat
                                                                               ;; No recursion. If not a number the list is bad
                                                                               ;; even if C is OK
                                                                               (STRING.EQUAL (CAR PAIR)
                                                                                             SSTR)))
                                                                   (HEXNUM? SSTR T)))
                                                    (< CSET 256)
                                                    (>= CSET 0))       ; parsed the charset part as an octal,  standard charset name, or
                                                                       ; hex
                                               (LOGOR (LLSH CSET 8)
                                                      CH))
                                              ((NOT NOERROR)
                                               (ERROR "BAD CHARACTER SPECIFICATION" C])

)

(DEFINEQ

(HEXNUM?
  [LAMBDA (STR PREFIXED?)                                              ; Edited 24-Aug-2021 08:31 by rmk:
    ;; Returns the number encoded as a hex representation in STR, NIL if it is not an unsigned hex string.  The hex digits can be upper or lower case.
    ;; If PREFIXED?, then hex ending must follow one of 0x, 0X, u+, U+ prefixes
    ;; CL:PARSE-INTEGER with JUNK-ALLOWED would also return NIL, but it would trim commonlisp seprs...and also depends on CHARTABLE
    ;; which is not available at the right place in the loadup.
    (CL:WHEN [OR (NOT PREFIXED?)
                 (AND (SELCHARQ (CHCON1 STR)
                                (0                                     ; Hex?  0X or 0x
                                   (FMEMB (NTHCHARCODE STR 2)
                                          (CHARCODE (x X))))
                                ((U u)                                 ; Unicode  U+ or u+
                                     (EQ (NTHCHARCODE STR 2)
                                         (CHARCODE +)))
                                NIL)
                      (SETQ STR (SUBSTRING STR 3 NIL (CONSTANT (CONCAT]
       (FOR I C (NUM _ 0) FROM 1 WHILE (SETQ C (NTHCHARCODE STR I))
          DO [SETQ C (IDIFFERENCE C (IF (AND (IGEQ C (CHARCODE 0))
                                             (ILEQ C (CHARCODE 9)))
                                        THEN (CHARCODE 0)
                                      ELSEIF (IF (AND (IGEQ C (CHARCODE a))
                                                      (ILEQ C (CHARCODE f)))
                                                 THEN (IDIFFERENCE (CHARCODE a)
                                                                   10)
                                               ELSEIF (AND (IGEQ C (CHARCODE A))
                                                           (ILEQ C (CHARCODE F)))
                                                 THEN (IDIFFERENCE (CHARCODE A)
                                                                   10))
                                      ELSE (RETURN NIL]
             (SETQ NUM (IPLUS (LLSH NUM 4)
                              C))
```

```
            FINALLY (RETURN NUM))])
```

(OCTALNUM?

```
  [LAMBDA (STR)                                          ; Edited 24-Aug-2021 08:25 by rmk:
    ;; Returns the number encoded as an octal representation in STR, NIL if it is not an unsigned octal string.
    ;; CL:PARSE-INTEGER with JUNK-ALLOWED would also return NIL, but it would trim commonlisp seprs...and also depends on CHARTABLE
    ;; which is not available at the right place in the loadup.
    (FOR I C (NUM _ 0) FROM 1 WHILE (SETQ C (NTHCHARCODE STR I))
       DO (IF (AND (IGEQ C (CHARCODE 0))
                   (ILEQ C (CHARCODE 7)))
              THEN [SETQ NUM (IPLUS (LLSH NUM 3)
                                    (IDIFFERENCE C (CHARCODE 0]
            ELSE (RETURN NIL))
       FINALLY (RETURN NUM])

)

(ADDTOVAR CHARACTERNAMES
          (Page 12)
          (Form 12)
          (FF 12)
          (Rubout 127)
          (Del 127)
          (Null 0)
          (Escape 27)
          (Esc 27)
          (Bell 7)
          (Tab 9)
          (Backspace 8)
          (Bs 8)
          (Newline 13)
          (CR 13)
          (EOL 13)
          (Return 13)
          (Tenexeol 31)
          (Space 32)
          (Sp 32)
          (Linefeed 10)
          (LF 10))

(ADDTOVAR CHARACTERSETNAMES (Meta 1)
                            (Function 2)
                            (Greek 38)
                            (Cyrillic 39)
                            (Hira 36)
                            (Hiragana 36)
                            (Kata 37)
                            (Katakana 37)
                            (Kanji 48))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(RPAQQ READTYPES (READ.RT RATOM.RT NOPROPRB.RT PROPRB.RT))

(DECLARE%: EVAL@COMPILE

(RPAQQ READ.RT NIL)

(RPAQQ RATOM.RT 1)

(RPAQQ NOPROPRB.RT T)

(RPAQQ PROPRB.RT 0)

(CONSTANTS READ.RT RATOM.RT NOPROPRB.RT PROPRB.RT)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS .CALL.SUBREAD. MACRO ((STREAM EOF-SUPPRESS EOF-VALUE CHAR PRESERVE-WHITESPACE)
                                (WITH-RESOURCE (\PNAMESTRING)
                                       (\SUBREAD (\GETSTREAM STREAM 'INPUT)
                                              (fetch (READTABLEP READSA) of *READTABLE*)
                                              (COND
                                                 (CHAR -1)
                                                 (T READ.RT))
                                              \PNAMESTRING
                                              (AND (fetch (READTABLEP CASEINSENSITIVE) of *READTABLE*)
                                                   (fetch (ARRAYP BASE) of UPPERCASEARRAY))
                                             EOF-SUPPRESS EOF-VALUE CHAR PRESERVE-WHITESPACE))))

(PUTPROPS FIXDOT MACRO [NIL (PROGN                       ; Fix a non-first dot followed by a singleton
                                   (AND DOTLOC (CDDR DOTLOC)
                                        (NULL (CDDDR DOTLOC))
                                        (RPLACD DOTLOC (CADDR DOTLOC])
```

```
(PUTPROPS RBCONTEXT MACRO ((X . Y)
                           ([LAMBDA (\RBFLG)
                              (DECLARE (SPECVARS \RBFLG))
                              (PROGN X . Y)
                              \RBFLG]
                            NIL)))

(PUTPROPS PROPRB MACRO [(X . Y)                              ; Propagates the right-bracket flag
                        (AND (RBCONTEXT X . Y)
                             (OR (EQ READTYPE NOPROPRB.RT)
                                 (SETQ \RBFLG T])

(PUTPROPS \RDCONC MACRO [(ELT . TOPFORMS)
                         ;; Add ELT to the accumulating list to be returned by \SUBREAD.  If at top level and no list accumulated, then run
                         ;; TOPFORMS
                         (COND
                            [LST (RPLACD END (SETQ END (CONS ELT]
                            (TOPLEVELP . TOPFORMS)
                            ((NOT *READ-SUPPRESS*)           ; Don't bother consing the result if it's going to be thrown away
                             (SETQ END (SETQ LST (CONS ELT])
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS *READ-NEWLINE-SUPPRESS* \RefillBufferFn)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *KEYWORD-PACKAGE* *INTERLISP-PACKAGE*)
)
)

(RPAQ? *REPLACE-NO-FONT-CODE* T)

(RPAQ? *DEFAULT-NOT-CONVERTED-FAT-CODE* 8739)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *REPLACE-NO-FONT-CODE* *DEFAULT-NOT-CONVERTED-FAT-CODE*)
)

(RPAQ? *READ-NEWLINE-SUPPRESS* )

(RPAQ? \RefillBufferFn (FUNCTION \READCREFILL))

;; Top level val of \RefillBufferFn means act like READC--we must be doing a raw BIN (or PEEKBIN?)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA CL:PARSE-INTEGER CL:READ-DELIMITED-LIST CL:READ-PRESERVING-WHITESPACE CL:READ)
)

(PUTPROPS LLREAD COPYRIGHT ("Venue & Xerox Corporation" 1981 1982 1983 1984 1985 1986 1987 1988 1990 1991 1993
                            2021))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

CONSTANT INDEX