```
File created:   23-Mar-2024 22:05:12   {DSK}<home>larry>il>medley>sources>CMLREADTABLE.;2

     edit by:   lmm

  changes to:   (VARS CMLREADTABLECOMS)
                (FUNCTIONS HASH-P)
                (FNS SET-DEFAULT-HASHMACRO-SETTINGS)

previous date:  13-Mar-95 12:41:10  {DSK}<home>larry>il>medley>sources>CMLREADTABLE.;1

  Read Table:   INTERLISP

     Package:   INTERLISP

      Format:    XCCS


(RPAQQ CMLREADTABLECOMS
       ((COMS                                                      ; Common Lisp readtable interface functions
              (FUNCTIONS HASH-LEFT-PAD-INITIAL-CONTENTS CL:SET-SYNTAX-FROM-CHAR CL:GET-DISPATCH-MACRO-CHARACTER
                     CL:GET-MACRO-CHARACTER CL:MAKE-DISPATCH-MACRO-CHARACTER CL:SET-DISPATCH-MACRO-CHARACTER
                     CL:SET-MACRO-CHARACTER)
              (FUNCTIONS DO-DISPATCH-MACRO FIND-MACRO-FUNCTION CL-MACRO-WRAPPED-P CL-UNWRAP-MACRO CL-WRAP-MACRO
                     IL-MACRO-WRAPPED-P IL-UNWRAP-MACRO IL-WRAP-MACRO))
        (COMS                                                      ; hash macro sub functions
              (FUNCTIONS HASH-LEFTPAREN HASH-A HASH-B HASH-BACKSLASH HASH-C HASH-COLON HASH-COMMA HASH-DOT
                     HASH-DOUBLEQUOTE HASH-ILLEGAL-HASH-CHAR HASH-LEFTANGLE HASH-MINUS HASH-NO-PARAMETER-ERROR
                     HASH-O HASH-P HASH-PLUS HASH-QUOTE HASH-R HASH-S HASH-STAR HASH-VBAR HASH-X HASH-EQUAL
                     HASH-NUMBER-SIGN HASH-STRUCTURE-SMASH HASH-STRUCTURE-LOOKUP)
                                                                   ; Temporary
              (VARIABLES *READ-SUPPRESS*))
        [COMS                                                      ; Common Lisp default readtables
              (FNS CMLRDTBL INIT-CML-READTABLES SET-DEFAULT-HASHMACRO-SETTINGS CMLREADSEMI)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (INIT-CML-READTABLES]
        (PROP FILETYPE CMLREADTABLE)))

;; Common Lisp readtable interface functions


(CL:DEFUN HASH-LEFT-PAD-INITIAL-CONTENTS (SIZE IVAL-LIST)
   [LET [(PADLENGTH (- SIZE (LENGTH IVAL-LIST]
        (COND
           [(> PADLENGTH 0)
            (APPEND IVAL-LIST (CL:MAKE-LIST PADLENGTH :INITIAL-ELEMENT (CAR (LAST IVAL-LIST]
           (T (CL:ERROR "Values list too long for #~D()" SIZE])


(CL:DEFUN CL:SET-SYNTAX-FROM-CHAR (TO-CHAR FROM-CHAR &OPTIONAL (TO-READTABLE *READTABLE*)
                                                 (FROM-READTABLE CMLRDTBL))
   (SETSYNTAX (CL:CHAR-CODE TO-CHAR)
          (GETSYNTAX (CL:CHAR-CODE FROM-CHAR)
                 FROM-READTABLE)
          TO-READTABLE))


(CL:DEFUN CL:GET-DISPATCH-MACRO-CHARACTER (DISP-CHAR SUB-CHAR &OPTIONAL (READTABLE *READTABLE*))
   [CDR (ASSOC SUB-CHAR (CDR (ASSOC DISP-CHAR (fetch (READTABLEP DISPATCHMACRODEFS) of READTABLE])

(CL:DEFUN CL:GET-MACRO-CHARACTER (CHAR &OPTIONAL (READTABLE *READTABLE*))

;;; insures entry is Common Lisp form - (MACRO {FIRST,ALWAYS} (LAMBDA (STREAM READTABLE) (FUNCALL <function> '<char> STREAM))))

   [LET ((TABENTRY (GETSYNTAX (CL:CHAR-CODE CHAR)
                          READTABLE))
         NON-TERMINATING-P)
        (AND (CL:CONSP TABENTRY)
             (EQ (CAR TABENTRY)
                 'MACRO)
             (CL:CONSP (CDR TABENTRY))
             (FMEMB (CADR TABENTRY)
                    '(ALWAYS FIRST))
             (SETQ NON-TERMINATING-P (CADR TABENTRY))
             (CL:CONSP (SETQ TABENTRY (CDDR TABENTRY)))
             (NULL (CDR TABENTRY))
             (CL:VALUES (FIND-MACRO-FUNCTION (CAR TABENTRY))
                    (NEQ NON-TERMINATING-P 'ALWAYS])


(CL:DEFUN CL:MAKE-DISPATCH-MACRO-CHARACTER (CHAR &OPTIONAL NON-TERMINATING (READTABLE *READTABLE*))
   (SETSYNTAX (CL:CHAR-CODE CHAR)
          `[MACRO ,(CL:IF NON-TERMINATING
                         'FIRST
                         'ALWAYS)
                 (LAMBDA (STREAM READTABLE Z)
                   (DO-DISPATCH-MACRO ,CHAR STREAM READTABLE]
          READTABLE)
   T)
```

```
(CL:DEFUN CL:SET-DISPATCH-MACRO-CHARACTER (DISP-CHAR SUB-CHAR FUNCTION &OPTIONAL (READTABLE
                                                                                   *READTABLE*))
   (CL:IF (CL:DIGIT-CHAR-P SUB-CHAR)
          (CL:ERROR "Digit ~S illegal as a sub-character for a dispatching macro" SUB-CHAR))
   (SETQ SUB-CHAR (CL:CHAR-UPCASE SUB-CHAR))
   (LET ((DISP-TABLE (OR (ASSOC DISP-CHAR (fetch (READTABLEP DISPATCHMACRODEFS) of READTABLE))
                         (LET ((NEWTABLE (LIST DISP-CHAR)))
                              (push (fetch (READTABLEP DISPATCHMACRODEFS) of READTABLE)
                                    NEWTABLE)
                              NEWTABLE)))
         DISP-CONS)
        (if (SETQ DISP-CONS (ASSOC SUB-CHAR (CDR DISP-TABLE)))
            then (CL:SETF (CDR DISP-CONS)
                          FUNCTION)
          else (push (CDR DISP-TABLE)
                     (CONS SUB-CHAR FUNCTION)))
        T))


(CL:DEFUN CL:SET-MACRO-CHARACTER (CHAR FUNCTION &OPTIONAL NON-TERMINATING (READTABLE *READTABLE*))
   (SETSYNTAX (CL:CHAR-CODE CHAR)
          `[MACRO ,(CL:IF NON-TERMINATING
                          'FIRST
                          'ALWAYS)
                  ,(COND
                      ((IL-MACRO-WRAPPED-P FUNCTION)
                       (IL-UNWRAP-MACRO FUNCTION))
                      (T (CL-WRAP-MACRO FUNCTION CHAR]
          READTABLE)
   T)


(CL:DEFUN DO-DISPATCH-MACRO (CHAR STREAM RDTBL)
   [LET ((*READTABLE* RDTBL)
         [DISP-TABLE (CDR (ASSOC CHAR (fetch (READTABLEP DISPATCHMACRODEFS) of RDTBL]
         INDEX NEXTCHAR)
        (COND
           ((NOT DISP-TABLE)
            (CL:ERROR "~S is not a dispatch macro character" CHAR))
           (T                                                     ; DISPATCHMACRODEFS is a list of A-lists
              [while (DIGITCHARP (SETQ NEXTCHAR (READCCODE STREAM RDTBL)))
                 do                                               ; read the optional numeric arg
                    (SETQ INDEX (+ (TIMES (OR INDEX 0)
                                          10)
                                   (- NEXTCHAR (CHARCODE 0]
              (LET* [(DISP-CHARACTER (CL:CHAR-UPCASE (CL:CODE-CHAR NEXTCHAR)))
                     (DISP-FUNCTION (CDR (ASSOC DISP-CHARACTER DISP-TABLE]
                    (if DISP-FUNCTION
                        then (CL:FUNCALL DISP-FUNCTION STREAM DISP-CHARACTER INDEX)
                      else (CL:IF *READ-SUPPRESS*
                                  (PROGN                          ; Attempt to ignore it
                                         (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
                                         NIL)
                                  (CL:ERROR "Undefined dispatch character ~S for dispatch macro character ~S"
                                         DISP-CHARACTER CHAR))])


(CL:DEFUN FIND-MACRO-FUNCTION (FORM)
   (COND
      ((CL-MACRO-WRAPPED-P FORM)
       (CL-UNWRAP-MACRO FORM))
      ((CL:FUNCTIONP FORM)
       (IL-WRAP-MACRO FORM))))


(CL:DEFUN CL-MACRO-WRAPPED-P (FORM)

;;; Predicate that checks for forms built by CL-WRAP-MACRO

   (AND (CL:CONSP FORM)
        (EQ (CAR FORM)
            'CL:LAMBDA)
        (CL:CONSP (CDR FORM))
        (CL:EQUAL (CADR FORM)
               '(STREAM READTABLE Z))
        (CL:CONSP (CDDR FORM))
        (NULL (CDDDR FORM))
        (CL:CONSP (CADDR FORM))
        (EQ (CAADDR FORM)
            'CL:FUNCALL)))


(CL:DEFUN CL-UNWRAP-MACRO (FORM)

;;; Fetches CL function out wrapped by CL-WRAP-MACRO
```

```
   (CADR (CADR (CADDR FORM))))
```

(CL:DEFUN **CL-WRAP-MACRO** (FN CHAR)

;;; Wraps a form around a CL readmacro to make it acceptable as an IL readmacro

```
   `(CL:LAMBDA (STREAM READTABLE Z)
           (CL:FUNCALL ',FN STREAM ,CHAR)))
```

(CL:DEFUN **IL-MACRO-WRAPPED-P** (FORM)

;;; Predicate that checks for forms built by IL-WRAP-MACRO

```
   (AND (CL:CONSP FORM)
        (EQ (CAR FORM)
            'CL:LAMBDA)
        (CL:CONSP (CDR FORM))
        (EQUAL (CADR FORM)
               '(STREAM CHAR))
        (CL:CONSP (SETQ FORM (CDDR FORM)))
        (NULL (CDR FORM))
        (CL:CONSP (SETQ FORM (CAR FORM)))
        (EQ (CAR FORM)
            'CL:FUNCALL)
        (EQ (CADDR FORM)
            'STREAM)))
```

(CL:DEFUN **IL-UNWRAP-MACRO** (FORM)

```
   (CADR (CADR (CADDR FORM))))
```

(CL:DEFUN **IL-WRAP-MACRO** (FORM)

;;; Wraps a form around an IL readmacro to make it acceptable as a CL readmacro

```
   `(CL:LAMBDA (STREAM CHAR)
           (CL:FUNCALL ',FORM STREAM)))
```

;; hash macro sub functions

(CL:DEFUN **HASH-LEFTPAREN** (STREAM CHAR INDEX)

```
   [LET ((CONTENTS (CL:READ-DELIMITED-LIST #\) STREAM T)))
        (COND
           (*READ-SUPPRESS* NIL)
           [\INBQUOTE
```

;; We are inside a back-quote - generate ",(coerce ',contents 'vector)"

```
                   (CL:WHEN INDEX (CL:CERROR "Ignore the explicit length" "Explicit length not allowed in
                                         backquoted vectors:~%%#~D~S" INDEX CONTENTS))
                   (LIST '\, `(COERCE ,(LIST 'BQUOTE CONTENTS)
                                     'CL:VECTOR]
           (INDEX (IF (<= (LENGTH CONTENTS)
                          INDEX)
                      THEN (LET [(VEC (CL:MAKE-ARRAY INDEX :INITIAL-ELEMENT (CAR (LAST CONTENTS]
                                 [LET ((XCL-USER::T0 (LENGTH CONTENTS))
                                       (I 0))
                                      (CL:BLOCK NIL
                                          (LET NIL (CL:TAGBODY LOOPTAG0015 (COND
                                                                              ((>= I XCL-USER::T0)
                                                                               (RETURN NIL)))
                                                              (CL:SETF (CL:AREF VEC I)
                                                                     (POP CONTENTS))
                                                              (CL:INCF I)
                                                              (GO LOOPTAG0015))))]
                                VEC)
                    ELSE (CL:ERROR "Values list too long for #~D()" INDEX)))
           (T (CL:MAKE-ARRAY (LENGTH CONTENTS)
                     :INITIAL-CONTENTS CONTENTS])
```

(CL:DEFUN **HASH-A** (STREAM CHAR PARAM)

```
   [LET ((CONTENTS (CL:READ STREAM T NIL T)))
        (COND
           (*READ-SUPPRESS* NIL)
           (T (CL:MAKE-ARRAY (ESTIMATE-DIMENSIONALITY PARAM CONTENTS)
                     :INITIAL-CONTENTS CONTENTS])
```

(CL:DEFUN **HASH-B** (STREAM CHAR PARAM)

```
   (COND
      (*READ-SUPPRESS* (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (READNUMBERINBASE STREAM 2))))


(CL:DEFUN HASH-BACKSLASH (STREAM CHAR PARAM)
   [COND
      (*READ-SUPPRESS* (CHARACTER.READ STREAM)
             NIL)
      (T (CL:IF (OR (NULL PARAM)
                    (AND (TYPEP PARAM 'CL:FIXNUM)
                         (>= PARAM 0)
                         (< PARAM CL:CHAR-FONT-LIMIT)))
             (CHARACTER.READ STREAM)
             (CL:ERROR "Illegal font specifier ~S for #\" PARAM))])


(CL:DEFUN HASH-C (STREAM CHAR PARAM)
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (DESTRUCTURING-BIND (NUM DEN)
                (CL:READ STREAM T NIL T)
                (COMPLEX NUM DEN])


(CL:DEFUN HASH-COLON (STREAM CHAR PARAM)                                ; Uninterned symbol.
   [COND
      (*READ-SUPPRESS* (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (CL:MAKE-SYMBOL (READ-EXTENDED-TOKEN STREAM *READTABLE* T])

(CL:DEFUN HASH-COMMA (STREAM CHAR PARAM)
```

;;; If the compiler is reading, then wrap up the form in a special data object to be noticed by FASL later. If it's not the compiler, then treat exactly like #.

```
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (LET ((FORM (CL:READ STREAM T NIL T)))
              (IF COMPILER::*COMPILER-IS-READING*
                  THEN (COMPILER::MAKE-EVAL-WHEN-LOAD :FORM FORM)
                ELSEIF (FETCH (READTABLEP COMMONLISP) OF *READTABLE*)
                  THEN (CL:EVAL FORM)
                ELSE (EVAL FORM])


(CL:DEFUN HASH-DOT (STREAM CHAR PARAM)
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (COND
            ((fetch (READTABLEP COMMONLISP) of *READTABLE*)
             (CL:EVAL (CL:READ STREAM T NIL T)))
            (T (EVAL (CL:READ STREAM T NIL T])

(CL:DEFUN HASH-DOUBLEQUOTE (STREAM CHAR PARAM)
```

;;; An extension to Common Lisp. This reads a normal string but ignores CR's and any whitespace immediately following them.

```
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (RSTRING STREAM *READTABLE* 'SKIP])


(CL:DEFUN HASH-ILLEGAL-HASH-CHAR (STREAM CHAR PARAM)
   (CL:ERROR "Illegal hash macro character ~S" CHAR))


(CL:DEFUN HASH-LEFTANGLE (STREAM CHAR PARAM)
   (HASH-NO-PARAMETER-ERROR CHAR PARAM)
   (CL:ERROR "Unreadable object #<~A>" (CL:READ STREAM T NIL T)))


(CL:DEFUN HASH-MINUS (STREAM CHAR PARAM)
   ;; When *READ-SUPPRESS* is true, we want to simply skip over the two forms (the feature expression and the controlled expression).  Otherwise,
   ;; we read the feature expression and, when it applies to us, skip over the controlled expression.  In any case, we never return a value.
   [COND
      (*READ-SUPPRESS*                                                  ; Skip two forms.
              (CL:READ STREAM T NIL T)
              (CL:READ STREAM T NIL T))
       (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
          (CL:WHEN (CMLREAD.FEATURE.PARSER (LET ((*PACKAGE* *KEYWORD-PACKAGE*))
                                                (CL:READ STREAM T NIL T)))
              (LET ((*READ-SUPPRESS* T))
                   (CL:READ STREAM T NIL T)))]
   (CL:VALUES))


(CL:DEFUN HASH-NO-PARAMETER-ERROR (CHAR PARAM)
   (CL:WHEN PARAM (CL:ERROR "Parameter ~D not allowed with hash macro ~S" PARAM CHAR)))


(CL:DEFUN HASH-O (STREAM CHAR PARAM)
   (COND
      (*READ-SUPPRESS* (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (READNUMBERINBASE STREAM 8))))


(CL:DEFUN HASH-P (STREAM CHAR PARAM)                          ; Edited 23-Mar-2024 22:01 by lmm
   (PATHNAME (CL:READ STREAM T NIL T)))


(CL:DEFUN HASH-PLUS (STREAM CHAR PARAM)

   ;; When *READ-SUPPRESS* is true, we want to simply skip over the two forms (the feature expression and the controlled expression).  Otherwise,
   ;; we read the feature expression and, unless it applies to us, skip over the controlled expression.  In any case, we never return a value.

   [COND
      (*READ-SUPPRESS*                                        ; Skip two forms.
              (CL:READ STREAM T NIL T)
              (CL:READ STREAM T NIL T))
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (CL:UNLESS (CMLREAD.FEATURE.PARSER (LET ((*PACKAGE* *KEYWORD-PACKAGE*))
                                                 (CL:READ STREAM T NIL T)))
              (LET ((*READ-SUPPRESS* T))
                   (CL:READ STREAM T NIL T)))]
   (CL:VALUES))


(CL:DEFUN HASH-QUOTE (STREAM CHAR PARAM)
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (LIST 'CL:FUNCTION (CL:READ STREAM T NIL T])


(CL:DEFUN HASH-R (STREAM CHAR PARAM)
   (COND
      (*READ-SUPPRESS* (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
             NIL)
      (PARAM (READNUMBERINBASE STREAM PARAM))
      (T (CL:ERROR "No base supplied for #R"))))


(CL:DEFUN HASH-S (STREAM CHAR PARAM)
   [COND
      (*READ-SUPPRESS* (CL:READ STREAM T NIL T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (CREATE-STRUCTURE (CL:READ STREAM T NIL T])


(CL:DEFUN HASH-STAR (STREAM CHAR PARAM)
   (DECLARE (IGNORE CHAR))
   [IF (EQ (PEEKC STREAM)
           '%()
       THEN                                                   ; It's a bitmap.
            (IF *READ-SUPPRESS*
                THEN (CL:READ STREAM NIL NIL T)
                     (CL:READ STREAM NIL NIL T)
              ELSEIF PARAM
                THEN (CL:ERROR "Unexpected parameter ~S given in #* bitmap syntax." PARAM)
              ELSE (FINISH-READING-BITMAP STREAM))
     ELSE                                                     ; It's a bit-vector.
          (LET* ((CONTENTS (READ-EXTENDED-TOKEN STREAM))
                 (LEN (NCHARS CONTENTS)))
                (IF *READ-SUPPRESS*
                    THEN NIL
                  ELSEIF (AND (EQ LEN 0)
                              PARAM
                              (NEQ PARAM 0))
                    THEN (CL:ERROR "No contents specified for bit vector #~A*" PARAM)
                  ELSEIF (AND PARAM (> LEN PARAM))
```

```
                      THEN (CL:ERROR "Bit vector contents longer than specified length in #~A*~A" PARAM CONTENTS)
                    ELSE (LET [(BITARRAY (CL:MAKE-ARRAY (OR PARAM LEN)
                                                :ELEMENT-TYPE
                                                'BIT :INITIAL-ELEMENT (IF (AND PARAM (> PARAM LEN 0))
                                                                          THEN (SELCHARQ (NTHCHARCODE CONTENTS -1
                                                                                                  )
                                                                                   (0 0)
                                                                                   (1 1)
                                                                                   (CL:ERROR "Illegal bit vector
                                                                                          element in #~A*~A"
                                                                                          PARAM CONTENTS))
                                                                        ELSE 0]
                               (CL:DOTIMES (I LEN)
                                   (CL:SETF (CL:AREF BITARRAY I)
                                          (SELCHARQ (NTHCHARCODE CONTENTS (CL:1+ I))
                                               (0 0)
                                               (1 1)
                                               (CL:ERROR "Illegal bit vector element in #~A*~A" PARAM CONTENTS))))
                               BITARRAY])


(CL:DEFUN HASH-VBAR (STREAM CHAR PARAM)
   (OR *READ-SUPPRESS* (HASH-NO-PARAMETER-ERROR CHAR PARAM))
   (LET ((*READ-SUPPRESS* T))
        (SKIP.HASH.COMMENT STREAM *READTABLE*)
        (CL:VALUES)))


(CL:DEFUN HASH-X (STREAM CHAR PARAM)
   (COND
      (*READ-SUPPRESS* (READ-EXTENDED-TOKEN STREAM *READTABLE* T)
             NIL)
      (T (HASH-NO-PARAMETER-ERROR CHAR PARAM)
         (READNUMBERINBASE STREAM 16))))


(CL:DEFUN HASH-EQUAL (STREAM CHAR PARAM)
   (CL:IF *READ-SUPPRESS*
       (CL:VALUES)
       [PROGN (CL:IF (NULL PARAM)
                     (CL:ERROR "#= encountered"))
              (CL:IF (CL:ASSOC PARAM *CIRCLE-READ-LIST*)
                     (CL:ERROR "#~D= seen twice in same context"))
              (LET ((NEWNODE (CONS PARAM NIL)))
                   (CL:PUSH NEWNODE *CIRCLE-READ-LIST*)
                   (CL:SETF (CDR NEWNODE)
                          (CL:READ STREAM T NIL T]))


(CL:DEFUN HASH-NUMBER-SIGN (STREAM CHAR PARAM)
   (CL:IF *READ-SUPPRESS*
       NIL
       [LET ((CIRCLE-PART (CL:ASSOC PARAM *CIRCLE-READ-LIST*)))
            (COND
               (CIRCLE-PART)
               (T (CL:ERROR "#~D# encountered before #~D=" PARAM PARAM]))


(CL:DEFUN HASH-STRUCTURE-SMASH (THING)
   (CL:TYPECASE THING
       (CONS
          (CL:IF (HASH-STRUCTURE-LOOKUP (CAR THING))
              (CL:SETF (CAR THING)
                     (CDAR THING))
              (HASH-STRUCTURE-SMASH (CAR THING)))
          (CL:IF (HASH-STRUCTURE-LOOKUP (CDR THING))
              (CL:SETF (CDR THING)
                     (CDDR THING))
              (HASH-STRUCTURE-SMASH (CDR THING))))
       ((CL:ARRAY T) [LET* ((ASIZE (CL:ARRAY-TOTAL-SIZE THING))
                            (VARRAY (CL:IF (> (CL:ARRAY-RANK THING)
                                                  1)
                                           (CL:MAKE-ARRAY ASIZE :DISPLACED-TO THING)
                                           THING))
                            SLOTCONTENTS)
                           (CL:DOTIMES (X ASIZE)
                               (CL:SETQ SLOTCONTENTS (CL:AREF VARRAY X))
                               (CL:IF (HASH-STRUCTURE-LOOKUP SLOTCONTENTS)
                                   (CL:SETF (CL:AREF VARRAY X)
                                          (CDR SLOTCONTENTS))
                                   (HASH-STRUCTURE-SMASH SLOTCONTENTS)))])
       (CL::STRUCTURE-OBJECT [LET (SLOTCONTENTS)
                                  (CL:DOLIST (DESCR (CL::STRUCTURE-POINTER-SLOTS (CL:TYPE-OF THING)))
                                      (CL:SETQ SLOTCONTENTS (FETCHFIELD DESCR THING))
                                      (CL:IF (HASH-STRUCTURE-LOOKUP SLOTCONTENTS)
                                          (REPLACEFIELD DESCR THING (CDR SLOTCONTENTS))
                                          (HASH-STRUCTURE-SMASH SLOTCONTENTS)))]))))
```

```
(CL:DEFUN HASH-STRUCTURE-LOOKUP (SLOTCONTENTS)
   (AND (CL:CONSP SLOTCONTENTS)
        (MEMQ SLOTCONTENTS *CIRCLE-READ-LIST*)))

;; Temporary


(CL:DEFVAR *READ-SUPPRESS* NIL)

;; Common Lisp default readtables

(DEFINEQ

(CMLRDTBL
  [LAMBDA NIL                                                   (* bvm%: "14-Oct-86 16:01")

    ;; Creates a vanilla common-lisp read table

    (PROG [(TBL (COPYREADTABLE 'ORIG]

     ;; First reset the table

          (for I from 0 to \MAXTHINCHAR do (SETSYNTAX I 'OTHER TBL))

     ;; Install the goodies

          (SETSEPR (CHARCODE (SPACE CR ^L LF TAB))
                 1 TBL)
          (SETSYNTAX (CHARCODE "'")
                 '(MACRO ALWAYS READQUOTE)
                 TBL)

     ;; Note that in cml, most of these macros are terminating, even though it would be nicer for us if they were not

          (SETSYNTAX (CHARCODE ";")
                 '(MACRO ALWAYS CMLREADSEMI)
                 TBL)
          (SETSYNTAX (CHARCODE ")")
                 'RIGHTPAREN TBL)
          (SETSYNTAX (CHARCODE "(")
                 'LEFTPAREN TBL)
          (READTABLEPROP TBL 'CASEINSENSITIVE T)
          (READTABLEPROP TBL 'COMMONLISP T)
          (READTABLEPROP TBL 'COMMONNUMSYNTAX T)
          (READTABLEPROP TBL 'USESILPACKAGE NIL)
          (READTABLEPROP TBL 'ESCAPECHAR (CHARCODE "\"))
          (READTABLEPROP TBL 'MULTIPLE-ESCAPECHAR (CHARCODE "|"))
          (if *PACKAGE*
              then (READTABLEPROP TBL 'PACKAGECHAR (CHARCODE ":")))
          (SET-DEFAULT-HASHMACRO-SETTINGS TBL)
          (SETSYNTAX (CHARCODE %")
                 'STRINGDELIM TBL)
          (SETSYNTAX (CHARCODE "`")
                 '(MACRO ALWAYS READBQUOTE)
                 TBL)
          (SETSYNTAX (CHARCODE ",")
                 '(MACRO ALWAYS READBQUOTECOMMA)
                 TBL)
          (RETURN TBL])

(INIT-CML-READTABLES
  [LAMBDA NIL                                                   ; Edited 16-Jan-87 15:47 by bvm:
    (DECLARE (GLOBALVARS CMLRDTBL *COMMON-LISP-READ-ENVIRONMENT* READ-LINE-RDTBL))
    (READTABLEPROP (SETQ CMLRDTBL (CMLRDTBL))
           'NAME "LISP")
    (SETQ *COMMON-LISP-READ-ENVIRONMENT* (MAKE-READER-ENVIRONMENT (CL:FIND-PACKAGE "USER")
                                                CMLRDTBL 10))
    (LET ((FILETBL (COPYREADTABLE CMLRDTBL)))                   ; Make one for files that has font indicators as seprs
         (for I from 1 to 26 do (SETSYNTAX I 'SEPRCHAR FILETBL))
         (READTABLEPROP FILETBL 'NAME "XCL"))
    (PROGN                                                      ; Read table to make READ-LINE work easily
           (SETQ READ-LINE-RDTBL (COPYREADTABLE 'ORIG))
           (for I from 0 to \MAXTHINCHAR do (SETSYNTAX I 'OTHER READ-LINE-RDTBL))
           (SETBRK (CHARCODE (EOL))
                  NIL READ-LINE-RDTBL])

(SET-DEFAULT-HASHMACRO-SETTINGS
  [LAMBDA (RDTBL)                                               ; Edited 23-Mar-2024 21:57 by lmm
                                                                (* jrb%: "10-Nov-86 15:46")
    (READTABLEPROP RDTBL 'HASHMACROCHAR (CHARCODE "#"))
    (CL:MAKE-DISPATCH-MACRO-CHARACTER #\# T RDTBL)
    (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\( 'HASH-LEFTPAREN RDTBL)
    (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\' 'HASH-QUOTE RDTBL)
    (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\. 'HASH-DOT RDTBL)
    (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\, 'HASH-COMMA RDTBL)
    (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\\ 'HASH-BACKSLASH RDTBL)
```

```
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\* 'HASH-STAR RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\: 'HASH-COLON RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\O 'HASH-O RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\B 'HASH-B RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\X 'HASH-X RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\R 'HASH-R RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\A 'HASH-A RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\S 'HASH-S RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\C 'HASH-C RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\P 'HASH-P RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\+ 'HASH-PLUS RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\- 'HASH-MINUS RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\| 'HASH-VBAR RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\< 'HASH-LEFTANGLE RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\" 'HASH-DOUBLEQUOTE RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\= 'HASH-EQUAL RDTBL)
     (CL:SET-DISPATCH-MACRO-CHARACTER #\# #\# 'HASH-NUMBER-SIGN RDTBL)
     RDTBL])

(CMLREADSEMI
  [LAMBDA (STREAM RDTBL)                                         ; Edited  9-Mar-95 13:41 by sybalsky:mv:envos

;;; Read and discard through end of line

     (until (FMEMB (READCCODE STREAM)
                   (CHARCODE (LF NEWLINE)))
        do NIL)
     (CL:VALUES])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(INIT-CML-READTABLES)
)

(PUTPROPS CMLREADTABLE FILETYPE CL:COMPILE-FILE)
```

FUNCTION INDEX

PROPERTY INDEX

VARIABLE INDEX