File created: 16-May-90 14:58:17 **{DSK}<usr>local>lde>lispcore>sources>COMMON.;2**

changes to: (VARS COMMONCOMS)

previous date: 30-Mar-87 16:57:29 {DSK}<usr>local>lde>lispcore>sources>COMMON.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ COMMONCOMS
       [[COMS                                                    ; BQUOTE
             (FNS READBQUOTE READBQUOTECOMMA \UNCOMMA \BQUOTE.BREAKRESET)
             (VARIABLES SI::*BACKQUOTE-LEVEL*)
             (FUNCTIONS SI::BQUOTE-EXPANDER SI::BQUOTE-PROCESS-LIST SI::BQUOTE-CONS SI::BQUOTE-APPEND
                    SI::BQUOTE-NCONC SI::COMMA-ERROR-EXPANDER . 'SI::BQUOTE)
             (P (CL:SETF (CL:MACRO-FUNCTION '\,)
                         'SI::COMMA-ERROR-EXPANDER)
                (CL:SETF (CL:MACRO-FUNCTION '\,@)
                         'SI::COMMA-ERROR-EXPANDER)
                (CL:SETF (CL:MACRO-FUNCTION '\,.)
                         'SI::COMMA-ERROR-EXPANDER))
             (DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (\INBQUOTE))
                    (ADDVARS (BREAKRESETFORMS (\BQUOTE.BREAKRESET]
        (COMS                                                    ; Pretty printing of quote and friends
             (FNS QUOTE.WRAPPER BQUOTE.WRAPPER FUNCTION.WRAPPER)
             (PROP PRETTYWRAPPER BQUOTE SI::BQUOTE CL:FUNCTION QUOTE \, . %,.\,@))
        (PROP FILETYPE COMMON)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA])
```

;; BQUOTE

```
(DEFINEQ
```

```
(READBQUOTE
  [LAMBDA (FILE)                                                 ; Edited 19-Mar-87 16:41 by bvm:
    (LET ((\INBQUOTE T))
         (DECLARE (SPECVARS \INBQUOTE))
         (LIST 'BQUOTE (CL:READ FILE T NIL T])
```

```
(READBQUOTECOMMA
  [LAMBDA (FILE)
    (DECLARE (USEDFREE \INBQUOTE))                               ; Edited 19-Mar-87 16:45 by bvm:
    (if (OR (fetch (READTABLEP COMMONLISP) of *READTABLE*)
            \INBQUOTE)
        then                                                     ; Valid context for comma
             (LIST (SELCHARQ (SKIPSEPRCODES FILE)
                             (@ (READCCODE FILE)
                                '\,@)
                             (%. (READCCODE FILE)
                                 '\,.)
                             '\,)
                   (CL:READ FILE T NIL T))
      else ;; Comma outside of backquote context is 'an error'
           ;; In Interlisp read table we want to treat it as though it had been escaped, because files written with old FILERDTBL might have had
           ;; unescaped commas in them.  In Common Lisp, we go ahead and read it as if we were in bquote context, for the benefit of typing
           ;; subexpressions to DEdit
           (LET ((CH (PEEKCCODE FILE)))
                (if (OR (SYNTAXP CH 'BREAK)
                        (SYNTAXP CH 'SEPR))
                    then '%,
                  else (PACK* '%, (READ FILE])
```

```
(\UNCOMMA
  [LAMBDA (X)                                                    (* bvm%: "19-May-86 12:38")
          (* * "Convert an old-style BQUOTE, where the commas were list elements, into the new style, where the commas are
          wrappers")

    (COND
       ((NLISTP X)
        X)
       (T (SELECTQ (CAR X)
                   ((%, %,. %,@ ., %,!)
                        (LET [(TAIL (\UNCOMMA (CDR X]
```

```
                        (CONS (LIST (SELECTQ (CAR X)
                                             (%, '\,)
                                             (%,. '\,.)
                                             '\,@)
                                      (CAR TAIL))
                              (CDR TAIL))))
                (LET [(BCAR (\UNCOMMA (CAR X)))
                      (BCDR (\UNCOMMA (CDR X]
                     (COND
                        ((AND (EQ BCAR (CAR X))
                              (EQ BCDR (CDR X)))
                         X)
                        (T (CONS BCAR BCDR])
```

(**BQUOTE.BREAKRESET**

```
  [LAMBDA (FLG)                                                    (* bvm%: " 6-Jul-85 23:19")
    (PROG1 \INBQUOTE (SETQ \INBQUOTE FLG])
```

)

(CL:DEFVAR **SI::*BACKQUOTE-LEVEL*** 0)

(CL:DEFUN **SI::BQUOTE-EXPANDER** (SI::FORM SI::LEVEL)

```
   [LET ((SI::*BACKQUOTE-LEVEL* SI::LEVEL))
        (CL:IF (CL:ATOM SI::FORM)
            (KWOTE SI::FORM)
            (CASE (CAR SI::FORM)                                   ; form is non-NIL
                ((SI::BQUOTE BQUOTE) (SI::BQUOTE-EXPANDER (SI::BQUOTE-EXPANDER (CADR SI::FORM)
                                                                               (CL:1+ SI::LEVEL))
                                                          SI::LEVEL))
                ((\,) (CADR SI::FORM))                             ; ',foo => foo
                ((\,@) (CL:ERROR ",@ in illegal context: ,@~S" (CADR SI::FORM)))
                ((\,.) (CL:ERROR ",. in illegal context: ,.~S" (CADR SI::FORM)))
                (CL:OTHERWISE (SI::BQUOTE-PROCESS-LIST SI::FORM)))))])
```

(CL:DEFUN **SI::BQUOTE-PROCESS-LIST** (LIST)

;; Expand backquoted list.

```
   (CL:IF (CL:ATOM LIST)
       (KWOTE LIST)                                                ; (KWOTE NIL) => NIL, so this is OK.
       [LET ((SI::ITEM (CAR LIST))
             (SI::TAIL (CDR LIST)))                                ; list has at least one CONS
            (CASE SI::ITEM
                ((\,) (CAR SI::TAIL))
                ((\,@) (CL:ERROR ",@ in illegal context: ,@~S" (CAR SI::TAIL)))
                ((\,.) (CL:ERROR ",. in illegal context: ,.~S" (CAR SI::TAIL)))
                (CL:OTHERWISE (CL:IF (CL:ATOM SI::ITEM)
                                     (SI::BQUOTE-CONS (KWOTE SI::ITEM)
                                                                   ; save a call to bquote-expander.
                                            (SI::BQUOTE-PROCESS-LIST SI::TAIL))
                                     (CASE (CAR SI::ITEM)
                                         ((\,) (SI::BQUOTE-CONS (CADR SI::ITEM)
                                                                (SI::BQUOTE-PROCESS-LIST SI::TAIL)))
                                         ((\,@) (SI::BQUOTE-APPEND (CADR SI::ITEM)
                                                                   (SI::BQUOTE-PROCESS-LIST SI::TAIL)))
                                         ((\,.) (SI::BQUOTE-NCONC (CADR SI::ITEM)
                                                                  (SI::BQUOTE-PROCESS-LIST SI::TAIL)))
                                         (CL:OTHERWISE (SI::BQUOTE-CONS (SI::BQUOTE-EXPANDER SI::ITEM
                                                                                             SI::*BACKQUOTE-LEVEL*)
                                                                        (SI::BQUOTE-PROCESS-LIST SI::TAIL)))))))])
```

(CL:DEFUN **SI::BQUOTE-CONS** (SI::BCAR SI::BCDR)

;; build a call to CONS of bcar and bcdr, optimizing where possible.

```
   [LET (SI::CONSTA SI::CONSTD)
        (COND
           [(AND (CL:SETQ SI::CONSTA (CONSTANTEXPRESSIONP SI::BCAR))
                 (CL:SETQ SI::CONSTD (CONSTANTEXPRESSIONP SI::BCDR)))
            (KWOTE (CONS (CL:FIRST SI::CONSTA)
                         (CL:FIRST SI::CONSTD]
           ((NULL SI::BCDR)
            ;; (cons x nil) => (list x)
            (LIST 'LIST SI::BCAR))
           [(CL:LISTP SI::BCDR)
            (CASE (CL:FIRST SI::BCDR)
                ((CONS LIST*)
                   ;; (cons x (cons . rest)) => (list* x . rest)
                   ;; (cons x (list* . rest)) => (list* x . rest)
                   (LIST* 'LIST* SI::BCAR (CL:REST SI::BCDR)))
                (LIST
```

```
                    ;; (cons x (list . rest)) => (list x . rest)
                    (LIST* 'LIST SI::BCAR (CL:REST SI::BCDR)))
                (CL:OTHERWISE (LIST 'CONS SI::BCAR SI::BCDR)))]
            (T (LIST 'CONS SI::BCAR SI::BCDR])
```

(CL:DEFUN **SI::BQUOTE-APPEND** (SI::HEAD SI::TAIL)

;; create a call to APPEND of head and tail, optimizing where possible.

```
[COND
   ((NULL SI::HEAD)
    (CL:IF (CL:ZEROP SI::*BACKQUOTE-LEVEL*)
        SI::TAIL
        (LIST 'CL:APPEND SI::TAIL)))
   ((NULL SI::TAIL)
    (CL:IF (CL:ZEROP SI::*BACKQUOTE-LEVEL*)
        SI::HEAD
        (LIST 'CL:APPEND SI::HEAD)))
   (T (CASE (CAR (LISTP SI::HEAD))
          (CONS
             ;; (append (cons x y) z) => (cons x (append y z))
             (SI::BQUOTE-CONS (CL:SECOND SI::HEAD)
                    (SI::BQUOTE-APPEND (CL:THIRD SI::HEAD)
                           SI::TAIL)))
          (LIST
             ;; (append (list ...) x) => (list* ... x)
             (CONS 'LIST* (APPEND (CL:REST SI::HEAD)
                                  (LIST SI::TAIL))))
          (LIST*
             ;; (append (list* ... x) y) => (list* ... (append x y))
             [CONS 'LIST* (CL:APPEND (CL:BUTLAST (CL:REST SI::HEAD))
                                  (LIST (SI::BQUOTE-APPEND (CAR (LAST SI::HEAD))
                                                SI::TAIL])
          ((CL:APPEND NCONC)
             ;; (append (append ...) x) => (append ... x)
             [CONS 'CL:APPEND (APPEND (CL:REST SI::HEAD)
                                      (CL:IF (EQ (CL:FIRST (LISTP SI::TAIL))
                                                 'CL:APPEND)
                                          (CL:REST SI::TAIL)
                                          (LIST SI::TAIL))])
          (CL:OTHERWISE (CL:IF (EQ (CL:FIRST (LISTP SI::TAIL))
                                   'CL:APPEND)
                               (LIST* 'CL:APPEND SI::HEAD (CL:REST SI::TAIL))
                               (LIST 'CL:APPEND SI::HEAD SI::TAIL)))))])
```

(CL:DEFUN **SI::BQUOTE-NCONC** (SI::HEAD SI::TAIL)

;; create a call to NCONC of head and tail, optimizing where possible.

```
[COND
   ((NULL SI::HEAD)
    (CL:IF (CL:ZEROP SI::*BACKQUOTE-LEVEL*)
        SI::TAIL
        (LIST 'NCONC SI::TAIL)))
   ((NULL SI::TAIL)
    (CL:IF (CL:ZEROP SI::*BACKQUOTE-LEVEL*)
        SI::HEAD
        (LIST 'NCONC SI::HEAD)))
   (T (CASE (CL:FIRST (LISTP SI::HEAD))
          (CONS
             ;; (nconc (cons x y) z) => (cons x (nconc y z))
             (SI::BQUOTE-CONS (CL:SECOND SI::HEAD)
                    (SI::BQUOTE-NCONC (CL:THIRD SI::HEAD)
                           SI::TAIL)))
          ((LIST LIST* CL:APPEND)
             ;; since you're splicing new structure, may as well use append and it's optimizations.
             (SI::BQUOTE-APPEND SI::HEAD SI::TAIL))
          (NCONC
             ;; (nconc (nconc ...) y) => (nconc ... y)
             [CONS 'NCONC (APPEND (CL:REST SI::HEAD)
                                  (CL:IF (EQ (CAR (LISTP SI::TAIL))
                                             'NCONC)
                                      (CL:REST SI::TAIL)
                                      (LIST SI::TAIL))])
          (CL:OTHERWISE (CL:IF (EQ (CAR (LISTP SI::TAIL))
                                   'NCONC)
                               (LIST* 'NCONC SI::HEAD (CL:REST SI::TAIL))
                               (LIST 'NCONC SI::HEAD SI::TAIL)))))])
```

```
(CL:DEFUN SI::COMMA-ERROR-EXPANDER (SI::FORM SI::ENV)
   (CL:ERROR "Tried to evaluate ~A~S outside of a backquote scope.~%%Probably a missing backquote or too many
          commas." (CL:ECASE (CL:FIRST SI::FORM)
                       ((\,) ",")
                       ((\,@) ",@")
                       ((\,.) ",."))
          (CL:SECOND SI::FORM)))


(DEFMACRO BQUOTE (FORM)
   (SI::BQUOTE-EXPANDER (\UNCOMMA FORM)
          0))


(DEFMACRO SI::BQUOTE (SI::FORM)
   (SI::BQUOTE-EXPANDER (\UNCOMMA SI::FORM)
          0))

(CL:SETF (CL:MACRO-FUNCTION '\,)
       'SI::COMMA-ERROR-EXPANDER)

(CL:SETF (CL:MACRO-FUNCTION '\,@)
       'SI::COMMA-ERROR-EXPANDER)

(CL:SETF (CL:MACRO-FUNCTION '\,.)
       'SI::COMMA-ERROR-EXPANDER)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQQ \INBQUOTE NIL)

(ADDTOVAR BREAKRESETFORMS (\BQUOTE.BREAKRESET))
)

;; Pretty printing of quote and friends

(DEFINEQ

(QUOTE.WRAPPER
  [LAMBDA (E FILE)                                          (* bvm%: " 4-Jun-86 18:19")
    (LET [(SYN (GETSYNTAX '%']
         (AND (LISTP SYN)
              (EQ (CAR (LAST SYN))
                  'READQUOTE)
              "'"])


(BQUOTE.WRAPPER
  [LAMBDA (E FILE)                                          (* bvm%: " 4-Jun-86 18:20")

          (* * "To print bquote wrappers in their original syntax")

    (AND [MEMB 'READBQUOTE (LISTP (GETSYNTAX '%`]
         (SELECTQ (CAR E)
              (BQUOTE "`")
              (\, ",")
              (\,. ",.")
              (\,@ ",@")
             NIL])


(FUNCTION.WRAPPER
  [LAMBDA (E FILE)                                          (* bvm%: "18-Apr-86 16:36")
    (AND (EQ (fetch (READTABLEP HASHMACROCHAR) of (\GTREADTABLE NIL))
             (CHARCODE %#))
         "#'"])

)

(PUTPROPS BQUOTE PRETTYWRAPPER BQUOTE.WRAPPER)

(PUTPROPS SI::BQUOTE PRETTYWRAPPER BQUOTE.WRAPPER)

(PUTPROPS CL:FUNCTION PRETTYWRAPPER FUNCTION.WRAPPER)

(PUTPROPS QUOTE PRETTYWRAPPER QUOTE.WRAPPER)

(PUTPROPS \, PRETTYWRAPPER BQUOTE.WRAPPER)

(PUTPROPS \,. PRETTYWRAPPER BQUOTE.WRAPPER)

(PUTPROPS \,@ PRETTYWRAPPER BQUOTE.WRAPPER)

(PUTPROPS COMMON FILETYPE COMPILE-FILE)

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

```
(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(PUTPROPS COMMON COPYRIGHT ("Venue & Xerox Corporation" 1985 1986 1987 1990))
```

FUNCTION INDEX

PROPERTY INDEX

VARIABLE INDEX

MACRO INDEX