*File created:* 16-May-90 14:28:05 **{DSK}<usr>local>lde>lispcore>sources>CMLSEQCOMMON.;2**

*changes to:* (VARS CMLSEQCOMMONCOMS)

*previous date:* 12-Nov-86 14:57:08 {DSK}<usr>local>lde>lispcore>sources>CMLSEQCOMMON.;1

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS



```
(RPAQQ CMLSEQCOMMONCOMS ((FUNCTIONS CHECK-SUBSEQ COLLECT-ITEM COPY-VECTOR-SUBSEQ FILL-VECTOR-SUBSEQ
                                MAKE-SEQUENCE-LIKE SEQ-DISPATCH TYPE-SPECIFIER)
                         (FUNCTIONS BACKWARD-LIST-LOOP BACKWARD-VECTOR-LOOP FORWARD-LIST-LOOP
                                FORWARD-VECTOR-LOOP)
                         (PROP FILETYPE CMLSEQCOMMON)
                         (DECLARE%: EVAL@COMPILE DONTCOPY DONTEVAL@LOAD (LOCALVARS . T))))

(DEFMACRO CHECK-SUBSEQ (SEQ START END LENGTH)
   `(CL:IF (NOT (<= 0 ,START ,END ,LENGTH))
        (CL:ERROR "Illegal subsequence for ~S.~%%Start is ~D. End is ~D" ,SEQ ,START ,END)))

(DEFMACRO COLLECT-ITEM (ITEM HEAD TAIL)
   `(CL:IF ,TAIL
        [RPLACD ,TAIL (SETQ ,TAIL (LIST ,ITEM]
        [SETQ ,HEAD (SETQ ,TAIL (LIST ,ITEM)))

(DEFMACRO COPY-VECTOR-SUBSEQ (FROM-VECTOR START-FROM END-FROM TO-VECTOR START-TO END-TO)
   "Copy one vector subsequence to another"
   `(CL:DO ((FROM-INDEX ,START-FROM (CL:1+ FROM-INDEX))
            (TO-INDEX ,START-TO (CL:1+ TO-INDEX)))
           (,(CL:IF END-FROM
                 `(EQL FROM-INDEX ,END-FROM)
                 `(EQL TO-INDEX ,END-TO))
            ,TO-VECTOR)
        (CL:SETF (CL:AREF ,TO-VECTOR TO-INDEX)
               (CL:AREF ,FROM-VECTOR FROM-INDEX))))

(DEFMACRO FILL-VECTOR-SUBSEQ (VECTOR START END NEWVALUE)
   `(CL:DO ((INDEX ,START (CL:1+ INDEX)))
           ((EQL INDEX ,END)
            ,VECTOR)
        (CL:SETF (CL:AREF ,VECTOR INDEX)
               ,NEWVALUE)))

(DEFMACRO MAKE-SEQUENCE-LIKE (SEQUENCE LENGTH)
   "Returns a sequence of the same type as SEQUENCE and the given LENGTH."
   `[LET ((SEQ ,SEQUENCE))
         (CL:ETYPECASE SEQ
             (LIST (CL:MAKE-LIST ,LENGTH))
             (STRING (CL:MAKE-STRING ,LENGTH))
             (CL:VECTOR (MAKE-VECTOR ,LENGTH :ELEMENT-TYPE (CL:ARRAY-ELEMENT-TYPE SEQ))))])

(DEFMACRO SEQ-DISPATCH (SEQUENCE LIST-FORM VECTOR-FORM)
   `(CL:ETYPECASE ,SEQUENCE
        (LIST ,LIST-FORM)
        (CL:VECTOR ,VECTOR-FORM)))

(DEFMACRO TYPE-SPECIFIER (TYPE)
   "Returns the broad class of which TYPE is a specific subclass."
   `(CL:IF (CL:ATOM ,TYPE)
       ,TYPE
       (CAR ,TYPE)))

(DEFMACRO BACKWARD-LIST-LOOP (SEQUENCE START END LOCAL-VARS RETURN-FORM &REST BODY)
   [LET ((INDEX-VAR (CAR LOCAL-VARS))
         (CURRENT-ELEMENT-VAR (CADR LOCAL-VARS))
         (OTHER-VARS (CDDR LOCAL-VARS)))
        `(CL:DO ((,INDEX-VAR (CL:1- ,END)
                             (CL:1- ,INDEX-VAR))
                 %%SUBSEQ
                 ,CURRENT-ELEMENT-VAR
                 ,@OTHER-VARS)
                ((< ,INDEX-VAR ,START)
```

```
               ,RETURN-FORM)
            (SETQ %%SUBSEQ (CL:NTHCDR ,INDEX-VAR ,SEQUENCE))
            (SETQ ,CURRENT-ELEMENT-VAR (CAR %%SUBSEQ))
            ,@BODY)])

(DEFMACRO BACKWARD-VECTOR-LOOP (SEQUENCE START END LOCAL-VARS RETURN-FORM &REST BODY)
   [LET ((INDEX-VAR (CAR LOCAL-VARS))
         (CURRENT-ELEMENT-VAR (CADR LOCAL-VARS))
         (OTHER-VARS (CDDR LOCAL-VARS)))
        `(CL:DO ((,INDEX-VAR (CL:1- ,END)
                        (CL:1- ,INDEX-VAR))
                 ,CURRENT-ELEMENT-VAR
                 ,@OTHER-VARS)
                ((< ,INDEX-VAR ,START)
                 ,RETURN-FORM)
            (SETQ ,CURRENT-ELEMENT-VAR (CL:AREF ,SEQUENCE ,INDEX-VAR))
            ,@BODY)])

(DEFMACRO FORWARD-LIST-LOOP (SEQUENCE START END LOCAL-VARS RETURN-FORM &REST BODY)
   [LET ((INDEX-VAR (CAR LOCAL-VARS))
         (CURRENT-ELEMENT-VAR (CADR LOCAL-VARS))
         (OTHER-VARS (CDDR LOCAL-VARS)))
        `(CL:DO ((%%SUBSEQ (CL:NTHCDR ,START ,SEQUENCE)
                        (CDR %%SUBSEQ))
                 (,INDEX-VAR ,START (CL:1+ ,INDEX-VAR))
                 ,CURRENT-ELEMENT-VAR
                 ,@OTHER-VARS)
                ((EQL ,INDEX-VAR ,END)
                 ,RETURN-FORM)
            (SETQ ,CURRENT-ELEMENT-VAR (CAR %%SUBSEQ))
            ,@BODY)])

(DEFMACRO FORWARD-VECTOR-LOOP (SEQUENCE START END LOCAL-VARS RETURN-FORM &REST BODY)
   "Canonical forward loop for vectors"
   [LET ((INDEX-VAR (CAR LOCAL-VARS))
         (CURRENT-ELEMENT-VAR (CADR LOCAL-VARS))
         (OTHER-VARS (CDDR LOCAL-VARS)))
        `(CL:DO ((,INDEX-VAR ,START (CL:1+ ,INDEX-VAR))
                 ,CURRENT-ELEMENT-VAR
                 ,@OTHER-VARS)
                ((EQL ,INDEX-VAR ,END)
                 ,RETURN-FORM)
            (SETQ ,CURRENT-ELEMENT-VAR (CL:AREF ,SEQUENCE ,INDEX-VAR))
            ,@BODY)])

(PUTPROPS CMLSEQCOMMON FILETYPE CL:COMPILE-FILE)

(DECLARE%: EVAL@COMPILE DONTCOPY DONTEVAL@LOAD

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)

(PUTPROPS CMLSEQCOMMON COPYRIGHT ("Venue & Xerox Corporation" 1986 1990))
```

## MACRO INDEX



## PROPERTY INDEX