File created: 30-Aug-91 09:56:27 **{DSK}<new>sources>lispcore>sources>MISC.;2**

changes to: (FNS TAILP LAST)
(VARS MISCCOMS)

previous date: 16-May-90 20:40:30 {DSK}<new>sources>lispcore>sources>MISC.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ MISCCOMS
       [(FNS ADD1VAR ADDTOVAR APPENDTOVAR APPEND \APPEND2 ASSOC ATTACH CHANGEPROP CONCATLIST COPY DEFINEQ
             DEFLIST DREMOVE DREVERSE DSUBST EQLENGTH EVERY GETLIS INTERSECTION KWOTE LAST LASTN LCONC LDIFF
             LDIFFERENCE LENGTH LISTGET LISTGET1 LISTPUT LISTPUT1 LSUBST MAP MAP2C MAP2CAR MAPC MAPCAR MAPCON
             MAPCONC MAPLIST MEMBER NLEFT NOTANY NOTEVERY NTH PUTASSOC RATOMS REMOVE REVERSE RPT RPTQ FRPTQ
             SASSOC SAVEDEF SAVEDEF1 SELECT SELECT1 SELECTC SETQQ SOME STRMEMB SUB1VAR SUBSET SUBST TAILP TCONC
             UNION)
        (COMS                                                    ; ERRORSET stuff
              (FNS ERSETQ NLSETQ XNLSETQ RESETLST RESETSAVE RESETFORM RESETVARS RESETVAR SI::RESETUNWIND)
              (FNS SI::NLSETQHANDLER)
              (INITVARS (SI::*NLSETQFLAG*)
                     (RESETSTATE))
              (PROP INFO RESETTOPVALS))
        (COMS (FNS GENSYM GENSYM? \GS.INITBUF)
                                                                 ; GENSYM garbage
              (DECLARE%: EVAL@COMPILE DONTCOPY (CONSTANTS (\GS.BUFSIZE 100)))
              (INITVARS (GENNUM 0)
                     (\GS.OGENNUM -1)
                     (\GS.NUMLEN 0)
                     (\GS.BUF NIL)
                     (\GS.STR (ALLOCSTRING 0)))
              (GLOBALVARS GENNUM \GS.OGENNUM \GS.NUMLEN \GS.BUF \GS.STR))
        (ALISTS (PRETTYEQUIVLST SELECTC)
               (DWIMEQUIVLST SELECTC))
        (LOCALVARS . T)
        [P (CL:PROCLAIM '(GLOBAL MAKESYSDATE MAKESYSNAME]
        (PROP FILETYPE MISC)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA RESETVARS RESETFORM RESETSAVE RESETLST NLSETQ ERSETQ SELECTC SELECT FRPTQ RPTQ
                               DEFINEQ APPENDTOVAR ADDTOVAR)
                      (NLAML RESETVAR XNLSETQ SUB1VAR SETQQ ADD1VAR)
                      (LAMA APPEND])

(DEFINEQ

(ADD1VAR
  [NLAMBDA (ADD1X)
    (SET ADD1X (ADD1 (EVAL ADD1X])

(ADDTOVAR
  [NLAMBDA X                                                     ; Edited  8-Jan-88 12:50 by bvm
    (LET*
     ((VAR (CAR X))
      [VAL (OR (AND (EQ DFNFLG 'ALLPROP)
                    (GETPROP VAR 'VALUE))
               (LISTP (EVALV VAR]
      TYPE)
     [if [AND (NEQ DFNFLG 'ALLPROP)
              (SETQ TYPE (GETPROP VAR 'VARTYPE))
              (SETQ TYPE (OR (LISTGET1 (LISTP TYPE)
                                       'ALIST)
                             (EQ TYPE 'ALIST]
         then
         ;; The variable appears to be an A-list.  Treat it as such unless we see evidence to the contrary.
         (for PAIR in (CDR X) BIND ADDED-NONLIST?
            do [if (NLISTP PAIR)
                   then ;; This is evidence to the contrary.  We arrange for the variable itself to be marked as changed below.
                        (SETQ VAL (CONS PAIR VAL))
                        (SETQ ADDED-NONLIST? T)
                   else (LET [(OLDENTRY (if (EQ TYPE 'USERMACROS)
                                            then [find OP in VAL suchthat (AND (EQ (CAR OP)
                                                                                   (CAR PAIR))
                                                                               (EQ (NULL (CADR OP))
                                                                                   (NULL (CADR PAIR]
                                            else (FASSOC (CAR PAIR)
                                                         VAL]
                             (if (NOT (EQUAL OLDENTRY PAIR))
```

```
                              then (if (AND OLDENTRY (NEQ DFNFLG T))
                                       then (EXEC-FORMAT "(new ~S entry for ~S)~%" VAR (CAR PAIR)))
                                   (MARKASCHANGED (LIST VAR (CAR PAIR))
                                          'ALISTS
                                          (NULL OLDENTRY))
                                   (SETQ VAL (CONS PAIR (if OLDENTRY
                                                            then (/DREMOVE OLDENTRY VAL)
                                                         else VAL]
          FINALLY (if ADDED-NONLIST?
                      then (SAVESET VAR VAL NIL 'NOSTACKUNDO)
                    else (/SET VAR VAL)))
    else ;; The variable doesn't appear to be an A-list.
         (LET ((DFNFLG (if (EQ DFNFLG 'ALLPROP)
                           then 'PROP
                         else DFNFLG)))
              (DECLARE (SPECVARS DFNFLG))
              (if (OR VAL (CDR X))
                  then (SAVESET VAR (UNION (CDR X)
                                           VAL)
                               NIL
                               'NOSTACKUNDO)
                elseif (EQ (EVALV VAR)
                           'NOBIND)
                  then ;; The semantics of (ADDVARS (FOO)) are to initialize FOO to NIL if it is NOBIND, otherwise leave it alone.
                       (/SET VAR NIL]
    VAR])
```

(APPENDTOVAR

```
[NLAMBDA X                                                    ; Edited 9-Mar-87 15:48 by Pavel
  (LET*
   ((VAR (CAR X))
    [VAL (OR (AND (EQ DFNFLG 'ALLPROP)
                  (GETPROP VAR 'VALUE))
             (LISTP (EVALV VAR]
    TYPE)
   [IF [AND (NEQ DFNFLG 'ALLPROP)
            (SETQ TYPE (GETPROP VAR 'VARTYPE))
            (SETQ TYPE (OR (LISTGET1 (LISTP TYPE)
                                     'ALIST)
                           (EQ TYPE 'ALIST]
       THEN
       ;; The variable appears to be an A-list.  Treat it as such unless we see evidence to the contrary.
       (LET
        ((ADDED-NONLIST? NIL))
        [FOR PAIR IN (CDR X)
           DO (IF (NLISTP PAIR)
                  THEN ;; This is evidence to the contrary.  We arrange for the variable itself to be marked as changed below.
                       (SETQ VAL (APPEND VAL (LIST PAIR)))
                       (SETQ ADDED-NONLIST? T)
                ELSE (LET [(OLDENTRY (IF (EQ TYPE 'USERMACROS)
                                         THEN [FIND OP IN VAL SUCHTHAT (AND (EQ (CAR OP)
                                                                                (CAR PAIR))
                                                                            (EQ (NULL (CADR OP))
                                                                                (NULL (CADR PAIR]
                                         ELSE (FASSOC (CAR PAIR)
                                                      VAL]
                          (IF (NOT (EQUAL OLDENTRY PAIR))
                              THEN (IF (AND OLDENTRY (NEQ DFNFLG T))
                                       THEN (EXEC-FORMAT "(new ~S entry for ~S)~%" VAR (CAR PAIR)))
                                   (MARKASCHANGED (LIST VAR (CAR PAIR))
                                          'ALISTS
                                          (NULL OLDENTRY))
                                   (SETQ VAL (APPEND (IF OLDENTRY
                                                         THEN (/DREMOVE OLDENTRY VAL)
                                                      ELSE VAL)
                                                     (LIST PAIR]
       (IF ADDED-NONLIST?
           THEN (SAVESET VAR VAL NIL 'NOPRINT)
         ELSE (/SET VAR VAL)))
    ELSE ;; The variable doesn't appear to be an A-list.
         (LET ((DFNFLG (IF (EQ DFNFLG 'ALLPROP)
                           THEN 'PROP
                         ELSE DFNFLG)))
              (DECLARE (SPECVARS DFNFLG))
              (IF (OR VAL (CDR X))
                  THEN (SAVESET VAR (APPEND VAL (LDIFFERENCE (CDR X)
                                                             VAL))
                               NIL
                               'NOPRINT)
                ELSEIF (EQ (EVALV VAR)
                           'NOBIND)
```

```
                    THEN ;; The semantics of (ADDVARS (FOO)) are to initialize FOO to NIL if it is NOBIND, otherwise leave it alone.
                         (/SET VAR NIL]
    VAR])
```

(APPEND

```
[LAMBDA L                                                    (* lmm "30-Jun-84 00:37")
                                                             ; fixed bug so that (APPEND (QUOTE (A B . C))) was (QUOTE
                                                             ; (A B . C))
  (COND
     ((EQ L 0)
      NIL)
     ((EQ L 1)
      (\APPEND2 (ARG L 1)
             NIL))
     (T (bind (VAL _ (ARG L L))
           (N _ L) while (IGREATERP (add N -1)
                                    0)
           do (SETQ VAL (\APPEND2 (ARG L N)
                                  VAL))
           finally (RETURN VAL])
```

(\APPEND2

```
[LAMBDA (L1 L2)                                              (* lmm "30-Jun-84 00:30")
  (COND
     ((LISTP L1)
      (PROG ((VAL (CONS (CAR L1)
                        L2))
             TAIL)
            (SETQ TAIL VAL)
        LP  [FRPLACD TAIL (SETQ TAIL (LIST (CAR (OR (LISTP (SETQ L1 (CDR L1)))
                                                    (PROGN (FRPLACD TAIL (OR L2 L1))
                                                           (RETURN VAL]
            (GO LP)))
     ((NLISTP L2)
      L1)
     (T L2])
```

(ASSOC

```
[LAMBDA (KEY ALST)                                           (* bvm%: "20-FEB-81 14:58")
  (PROG NIL
    LP  [COND
           ((NLISTP ALST)
            (RETURN))
           ((AND (LISTP (CAR ALST))
                 (EQ (CAAR ALST)
                     KEY))
            (RETURN (CAR ALST]
        (SETQ ALST (CDR ALST))
        (GO LP])
```

(ATTACH

```
[LAMBDA (X L)
  (COND
     ((LISTP L)
      (FRPLACA (FRPLACD L (CONS (CAR L)
                                (CDR L)))
               X))
     ((NULL L)
      (CONS X))
     (T (ERRORX (LIST 4 L])
```

(CHANGEPROP

```
[LAMBDA (X PROP1 PROP2)                                      (* wt%: "31-MAY-79 22:28")
  (PROG [(Z (COND
              ((LITATOM X)
               (GETPROPLIST X))
              (T (ERRORX (LIST 14 X]
    LP  (RETURN (COND
                  ((NLISTP Z)
                   NIL)
                  ((EQ (CAR Z)
                       PROP1)
                   (FRPLACA Z PROP2)
                   X)
                  (T [SETQ Z (CDR (LISTP (CDR Z]
                     (GO LP])
```

(CONCATLIST

```
[LAMBDA (L)                                                  ; Edited 24-Nov-86 17:37 by jop:
  (PROG (STR FATP)                                           ; Try to pre-determine FATP, at least for strings and litatoms,
                                                             ; where it is easy to tell.
```

```
          (SETQ STR (ALLOCSTRING (for X in L sum [OR FATP (COND
                                                             ((STRINGP X)
                                                              (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X)))
                                                             ((LITATOM X)
                                                              (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X]
                                                    (NCHARS X))
                                 NIL NIL FATP))
          (for X in L as I from 1 by (NCHARS X) do (RPLSTRING STR I X))
          (RETURN STR])
```

(COPY

```
  [LAMBDA (X)                                                    (* lmm "16-FEB-82 22:07")
    (COND
       ((NLISTP X)
        X)
       (T (PROG [TAIL (VAL (LIST (COPY (CAR X]
                (SETQ TAIL VAL)
            LP  (COND
                   ((NLISTP (SETQ X (CDR X)))
                    (AND X (FRPLACD TAIL X))
                    (RETURN VAL)))
                [FRPLACD TAIL (SETQ TAIL (CONS (COPY (CAR X]
                (GO LP])
```

(DEFINEQ

```
  [NLAMBDA X
    (DEFINE X])
```

(DEFLIST

```
  [LAMBDA (L PROP)
    (PROG NIL
      LP  (COND
             ((NLISTP L)
              (RETURN)))
          (PUTPROP (CAAR L)
                 PROP
                 (CADAR L))                                      ; NOTE: this call to PUTPROP is changed to /PUTPROP later in
                                                                 ; the loadup.
          (SETQ L (CDR L))
          (GO LP])
```

(DREMOVE

```
  [LAMBDA (X L)
    (COND
       ((NLISTP L)
        NIL)
       [(EQ X (CAR L))
        (COND
           ((CDR L)
            (FRPLACA L (CADR L))
            (FRPLACD L (CDDR L))
            (DREMOVE X L]
       (T (PROG (Z)
                (DECLARE (LOCALVARS Z))
                (SETQ Z L)
             LP  [COND
                    ((NLISTP (CDR L))
                     (RETURN Z))
                    ((EQ X (CADR L))
                     (FRPLACD L (CDDR L)))
                    (T (SETQ L (CDR L]
                (GO LP])
```

(DREVERSE

```
  [LAMBDA (L)
    (PROG (Y Z)
          (DECLARE (LOCALVARS Y Z))
      R1  (COND
             ((NLISTP (SETQ Y L))
              (RETURN Z)))
          (SETQ L (CDR L))
          (SETQ Z (FRPLACD Y Z))
          (GO R1])
```

(DSUBST

```
  [LAMBDA (NEW OLD EXPR)                                         (* lmm "16-FEB-82 22:10")
    (PROG (B)
          [COND
             ((EQ OLD (SETQ B EXPR))
              (RETURN (COPY NEW]
      LP  [COND
```

```
            ((NLISTP EXPR)
             (RETURN B))
            ((EQUAL OLD (CAR EXPR))
             (FRPLACA EXPR (COPY NEW)))
            (T (DSUBST NEW OLD (CAR EXPR]
         (COND
            ((AND OLD (EQ OLD (CDR EXPR)))
             (FRPLACD EXPR (COPY NEW))
             (RETURN B)))
         (SETQ EXPR (CDR EXPR))
         (GO LP])
```

(EQLENGTH

```
[LAMBDA (X N)                                          (* bvm%: "14-Feb-85 00:34")
```

;; Generated by paatern match. INcluded so user can load code that has been dwimified and or compiled into a nonclisp system and run it.

```
(COND
   ((ILESSP N 0)
    NIL)
   ((EQ N 0)
    (NLISTP X))
   (T (AND (LISTP (SETQ X (NTH X N)))
           (NLISTP (CDR X])
```

(EVERY

```
[LAMBDA (EVERYX EVERYFN1 EVERYFN2)
```

;; Note that EVERY does not compile open, although SOME does.

```
(PROG NIL
  CL:LOOP
       (COND
          ((NLISTP EVERYX)
           (RETURN T))
          ((NULL (APPLY* EVERYFN1 (CAR EVERYX)
                         EVERYX))
           (RETURN NIL)))
       [SETQ EVERYX (COND
                       (EVERYFN2 (APPLY* EVERYFN2 EVERYX))
                       (T (CDR EVERYX]
       (GO CL:LOOP])
```

(GETLIS

```
[LAMBDA (X PROPS)                                      (* wt%: "31-MAY-79 22:25")
  (PROG [(Z (COND
                ((LITATOM X)
                 (GETPROPLIST X))
                (T X]
    LP  (RETURN (COND
                   ((NLISTP Z)
                    NIL)
                   ((FMEMB (CAR Z)
                           PROPS)
                    Z)
                   (T [SETQ Z (CDR (LISTP (CDR Z]
                      (GO LP])
```

(INTERSECTION

```
[LAMBDA (X Y)
  (PROG ((R (CONS))
         S)
        (DECLARE (LOCALVARS R S))
    LP  (COND
           ((NLISTP X)
            (RETURN (CAR R)))
           ([COND
               [(LITATOM (SETQ S (CAR X)))
                (AND (FMEMB S Y)
                     (NULL (FMEMB S (CAR R]
               (T (AND (MEMBER S Y)
                       (NULL (MEMBER S (CAR R]
            (TCONC R S)))
        (SETQ X (CDR X))
        (GO LP])
```

(KWOTE

```
[LAMBDA (X)                                            (* dcl%: 15 SEP 75 15%:25)
  (COND
     ((OR (NULL X)
          (EQ X T)
          (NUMBERP X))
      X)
     (T (LIST 'QUOTE X])
```

(LAST

```
  [LAMBDA (X N)                                          ; Edited 30-Aug-91 09:54 by jrb:
    (PROG (XX)
          (DECLARE (LOCALVARS XX))
     ;; Added argument N for CLtL2 compatibility
          (COND
             (N (GO BL)))
      L   (COND
             ((NLISTP X)
              (RETURN XX)))
          (SETQ XX X)
          (SETQ X (CDR X))
          (GO L)
      BL  (SETQ XX (IDIFFERENCE (LENGTH X)
                                N))
          (COND
             ((ILESSP XX 0)
              (RETURN X))
             ((ILESSP N 0)
              (ERROR "LAST: N is negative" N))
             (T (RETURN (NTH X (ADD1 XX])
```

(LASTN

```
  [LAMBDA (L N)
    (PROG (X Y)
          (DECLARE (LOCALVARS X Y))
          (COND
             ((NLISTP L)
              (RETURN NIL))
             ((NULL (SETQ X (FNTH L N)))
              (RETURN)))
      LP  [COND
             ((NULL (SETQ X (CDR X)))
              (RETURN (CONS Y L]
          (SETQ Y (NCONC1 Y (CAR L)))
          (SETQ L (CDR L))
          (GO LP])
```

(LCONC

```
  [LAMBDA (PTR X)
    (PROG (XX)
          (DECLARE (LOCALVARS XX))
          [RETURN (COND
                     ((NULL X)
                      PTR)
                     ([OR (NLISTP X)
                          (CDR (SETQ XX (LAST X]
                      (SETQ XX X)
                      (GO ERROR))
                     ((NULL PTR)
                      (CONS X XX))
                     ((NLISTP PTR)
                      (SETQ XX PTR)
                      (GO ERROR))
                     ((NULL (CAR PTR))
                      (FRPLACA (FRPLACD PTR XX)
                               X))
                     (T (FRPLACD (CDR PTR)
                                 X)
                        (FRPLACD PTR XX]
      ERROR
          (ERROR '"bad argument - LCONC" XX])
```

(LDIFF

```
  [LAMBDA (X Y Z)
    (COND
       ((EQ X Y)
        Z)
       ((AND (NULL Y)
             (NULL Z))
        X)
       (T (PROG (V)
                [COND
                   [Z (SETQ V (CDR (FRPLACD (SETQ V (FLAST Z))
                                            (FRPLACD (CONS (CAR X)
                                                           V]
                   (T (SETQ V (SETQ Z (CONS (CAR X]
          CL:LOOP
                (SETQ X (CDR X))
                [COND
                   ((EQ X Y)
```

```
                    (RETURN Z))
                   ((NULL X)
                    (RETURN (ERROR '"LDIFF: not a tail" Y]
                [SETQ V (CDR (FRPLACD V (FRPLACD (CONS (CAR X)
                                                       V]
                (GO CL:LOOP])
```

(LDIFFERENCE

```
[LAMBDA (X Y)                                          (* lmm "27-Mar-84 16:26")
  (for Z in X when (NOT (MEMBER Z Y)) collect Z])
```

(LENGTH

```
[LAMBDA (X)
  (PROG ((N 0))
        (DECLARE (LOCALVARS N))
    LP  (COND
           ((NLISTP X)
            (RETURN N))
           (T (SETN N (ADD1 N))
              (SETQ X (CDR X))
              (GO LP])
```

(LISTGET

```
[LAMBDA (LST PROP)                                     ; Edited  3-Sep-87 12:18 by bvm:
```

;; like GETPROP but works on lists, searching them two cdrs at a time.

```
(PROG NIL
  LP  [COND
         ((NLISTP LST)
          (RETURN))
         ((EQ (CAR LST)
              PROP)
          (RETURN (CADR LST]
      [SETQ LST (CDR (LISTP (CDR LST]
      (GO LP])
```

(LISTGET1

```
[LAMBDA (LST PROP)
```

;; Used to be called GET. Like LISTGET but only searches one cdr at a time.

```
(PROG NIL
  LP  [COND
         ((NLISTP LST)
          (RETURN))
         ((EQ (CAR LST)
              PROP)
          (RETURN (CADR LST]
      (SETQ LST (CDR LST))
      (GO LP])
```

(LISTPUT

```
[LAMBDA (LST PROP VAL)
```

;; Like PUT but works on lists. Inverse of LISTGET

```
(PROG ([X (OR (LISTP LST)
              (ERRORX (LIST 4 LST]
       X0)
  CL:LOOP
      (COND
         ((NLISTP (CDR X))                             ; Odd parity;  either (A B C) or (A B C . D) --- drop thru and add
                                                       ; at beginning

          )
         ((EQ (CAR X)
              PROP)                                    ; found it
          (FRPLACA (CDR X)
                   VAL)
          (RETURN VAL))
         ([LISTP (SETQ X (CDDR (SETQ X0 X]
          (GO CL:LOOP))
         ((NULL X)
```

;; Ran out without finding PROP on even parity. add at end If X is not NIL, means ended in a non-list following even parity, e.g. (A B . ;; C) so drop through and add at front.

```
          (FRPLACD (CDR X0)
                   (LIST PROP VAL))
          (RETURN VAL)))
  ADDFRONT
      [FRPLNODE LST PROP (CONS VAL (CONS (CAR LST)
                                         (CDR LST]
      (RETURN VAL])
```

(LISTPUT1

```
[LAMBDA (LST PROP VAL)                                  (* lmm "22-Oct-85 16:44")
  ;; Used to be called PUTL.  Like LISTPUT but only searches one cdr at a time.  Inverse of LISTGET1
  (PROG ((X LST))
    LP  (COND
           [(NLISTP X)                                  ; Note no checks for lists ending in dotted pairs.
            (RETURN (NCONC LST (LIST PROP VAL]
           ((EQ (CAR X)
                PROP)
            [COND
               ((CDR X)
                (FRPLACA (CDR X)
                         VAL))
               (T (FRPLACD X (LIST VAL]
            (RETURN LST)))
        (SETQ X (CDR X))
        (GO LP])
```

(LSUBST

```
[LAMBDA (NEW OLD EXPR)                                  (* lmm "16-FEB-82 22:11")
  ;; Substitutes X as a segment for Y in Z.  E.g. LSUBST ((A B) Y (X Y Z)) is (X A B Z) not meaningful for Y an atom and CDR of a list.  if X is NIL,
  ;; operation effectively deletes Y, i.e. produces a copy without Y in it.
  (COND
     ((NULL EXPR)
      NIL)
     ((NLISTP EXPR)
      (COND
         ((EQ OLD EXPR)
          NEW)
         (T EXPR)))
     [(EQUAL OLD (CAR EXPR))
      (NCONC (COPY NEW)
             (LSUBST NEW OLD (CDR EXPR]
     (T (CONS (LSUBST NEW OLD (CAR EXPR))
              (LSUBST NEW OLD (CDR EXPR])
```

(MAP

```
[LAMBDA (MAPX MAPFN1 MAPFN2)
  (PROG NIL
    LP  (COND
           ((NLISTP MAPX)
            (RETURN)))
        (APPLY* MAPFN1 MAPX)
        [SETQ MAPX (COND
                      (MAPFN2 (APPLY* MAPFN2 MAPX))
                      (T (CDR MAPX]
        (GO LP])
```

(MAP2C

```
[LAMBDA (MAPX MAPY MAPFN1 MAPFN2)
  (PROG NIL
    LP  (COND
           ((OR (NLISTP MAPX)
                (NLISTP MAPY))
            (RETURN)))
        (APPLY* MAPFN1 (CAR MAPX)
                (CAR MAPY))
        [COND
           (MAPFN2 (SETQ MAPX (APPLY* MAPFN2 MAPX))
                   (SETQ MAPY (APPLY* MAPFN2 MAPY)))
           (T (SETQ MAPX (CDR MAPX))
              (SETQ MAPY (CDR MAPY]
        (GO LP])
```

(MAP2CAR

```
[LAMBDA (MAPX MAPY MAPFN1 MAPFN2)
  (PROG (CL:MAPL MAPE)
    LP  (COND
           ((OR (NLISTP MAPX)
                (NLISTP MAPY))
            (RETURN CL:MAPL)))
        (SETQ MAPE (CONS (APPLY* MAPFN1 (CAR MAPX)
                                        (CAR MAPY))
                         MAPE))
        (COND
           (CL:MAPL (FRPLACD (CDR MAPE)
                             (FRPLACD MAPE)))
           (T (SETQ CL:MAPL MAPE)))
        [COND
           (MAPFN2 (SETQ MAPY (APPLY* MAPFN2 MAPY))
```

```
                  (SETQ MAPX (APPLY* MAPFN2 MAPX)))
            (T (SETQ MAPY (CDR MAPY))
               (SETQ MAPX (CDR MAPX]
         (GO LP])
```

(MAPC

```
 [LAMBDA (MAPX MAPFN1 MAPFN2)
   (PROG NIL
     LP  (COND
            ((NLISTP MAPX)
             (RETURN)))
         (APPLY* MAPFN1 (CAR MAPX))
         [SETQ MAPX (COND
                       (MAPFN2 (APPLY* MAPFN2 MAPX))
                       (T (CDR MAPX]
         (GO LP])
```

(MAPCAR

```
 [LAMBDA (MAPX MAPFN1 MAPFN2)
   (PROG (CL:MAPL MAPE)
     LP  (COND
            ((NLISTP MAPX)
             (RETURN CL:MAPL)))
         (SETQ MAPE (CONS (APPLY* MAPFN1 (CAR MAPX))
                          MAPE))
         (COND
            (CL:MAPL (FRPLACD (CDR MAPE)
                              (FRPLACD MAPE)))
            (T (SETQ CL:MAPL MAPE)))
         [SETQ MAPX (COND
                       (MAPFN2 (APPLY* MAPFN2 MAPX))
                       (T (CDR MAPX]
         (GO LP])
```

(MAPCON

```
 [LAMBDA (MAPX MAPFN1 MAPFN2)
   (PROG (CL:MAPL MAPE MAPY)
     LP  [COND
            ((NLISTP MAPX)
             (RETURN CL:MAPL))
            ((LISTP (SETQ MAPY (APPLY* MAPFN1 MAPX)))
             [COND
                (MAPE (FRPLACD MAPE MAPY))
                (T (SETQ CL:MAPL (SETQ MAPE MAPY]
             (PROG NIL
               LP  (COND
                      ((SETQ MAPY (CDR MAPE))
                       (SETQ MAPE MAPY)
                       (GO LP]
         [SETQ MAPX (COND
                       (MAPFN2 (APPLY* MAPFN2 MAPX))
                       (T (CDR MAPX]
         (GO LP])
```

(MAPCONC

```
 [LAMBDA (MAPX MAPFN1 MAPFN2)
   (PROG (CL:MAPL MAPE MAPY)
     LP  [COND
            ((NLISTP MAPX)
             (RETURN CL:MAPL))
            ([LISTP (SETQ MAPY (APPLY* MAPFN1 (CAR MAPX]
             [COND
                (MAPE (FRPLACD MAPE MAPY))
                (T (SETQ CL:MAPL (SETQ MAPE MAPY]
             (PROG NIL
               LP  (COND
                      ((SETQ MAPY (CDR MAPE))
                       (SETQ MAPE MAPY)
                       (GO LP]
         [SETQ MAPX (COND
                       (MAPFN2 (APPLY* MAPFN2 MAPX))
                       (T (CDR MAPX]
         (GO LP])
```

(MAPLIST

```
 [LAMBDA (MAPX MAPFN1 MAPFN2)
   (PROG (CL:MAPL MAPE)
     LP  (COND
            ((NLISTP MAPX)
             (RETURN CL:MAPL)))
         (SETQ MAPE (CONS (APPLY* MAPFN1 MAPX)
                          MAPE))
```

```
          (COND
             (CL:MAPL (FRPLACD (CDR MAPE)
                               (FRPLACD MAPE)))
             (T (SETQ CL:MAPL MAPE)))
          [SETQ MAPX (COND
                        (MAPFN2 (APPLY* MAPFN2 MAPX))
                        (T (CDR MAPX]
          (GO LP])
```

(MEMBER

```
  [LAMBDA (X Y)
    (PROG NIL
      LP  (RETURN (COND
                     ((NLISTP Y)
                      NIL)
                     ([COND
                         ((LITATOM X)
                          (EQ X (CAR Y)))
                         (T (EQUAL X (CAR Y]
                      Y)
                     (T (SETQ Y (CDR Y))
                        (GO LP])
```

(NLEFT

```
  [LAMBDA (L N TAIL)                                    (* bvm%: "14-Feb-85 00:35")
```

;; Returns TAIL of L containing N elements more than TAIL, e.g. if TAIL is NIL (the usual case) NLEFT ((A B C D E) 2) is (D E). If FOO is (A B C D ;; E) and FIE is (CDDDR FOO), (NLEFT FOO 1 FIE) is (C D E).

```
    (PROG ((X L)
           (Y L))
      LP  (COND
             ((EQ N 0)
              (GO LP1))
             ((OR (EQ X TAIL)
                  (NLISTP X))
              (RETURN NIL)))
          (SETQ X (CDR X))
          (SUB1VAR N)
          (GO LP)
      LP1 (COND
             ((OR (EQ X TAIL)
                  (NLISTP X))
              (RETURN Y)))
          (SETQ X (CDR X))
          (SETQ Y (CDR Y))
          (GO LP1])
```

(NOTANY

```
  [LAMBDA (SOMEX SOMEFN1 SOMEFN2)
    (NULL (SOME SOMEX SOMEFN1 SOMEFN2])
```

(NOTEVERY

```
  [LAMBDA (EVERYX EVERYFN1 EVERYFN2)
    (NULL (EVERY EVERYX EVERYFN1 EVERYFN2])
```

(NTH

```
  [LAMBDA (X N)
    (COND
       ((IGREATERP 1 N)
        (CONS NIL X))
       (T (PROG NIL
            LP  (COND
                   ((NOT (IGREATERP N 1))
                    (RETURN X))
                   ((NLISTP X)
                    (RETURN NIL)))
                (SETQ X (CDR X))
                (SETQ N (SUB1 N))
                (GO LP])
```

(PUTASSOC

```
  [LAMBDA (KEY VAL ALST)                                (* lmm%: 5 SEP 75 119)
    (PROG [(X (OR (LISTP ALST)
                  (ERRORX (LIST 4 ALST]
          (DECLARE (LOCALVARS X))
      LP  (COND
             ((EQ (CAR (OR (LISTP (CAR X))
                           (GO NEXT)))
                  KEY)
              (FRPLACD (CAR X)
                       VAL)
```

```
            (RETURN VAL)))
      NEXT
          [SETQ X (OR (LISTP (CDR X))
                      (PROGN (FRPLACD X (LIST (CONS KEY VAL)))
                             (RETURN VAL]
          (GO LP])
```

(RATOMS

```
  [LAMBDA (A FILE RDTBL)
    (PROG (L X)
      B   (COND
             ((EQ (SETQ X (RATOM FILE RDTBL))
                  A)
              (RETURN (CAR L)))
             ((SETQ L (TCONC L X))
              (GO B])
```

(REMOVE

```
  [LAMBDA (X L)
    (COND
       ((NLISTP L)
        NIL)
       ((EQUAL X (CAR L))
        (REMOVE X (CDR L)))
       (T (CONS (CAR L)
                (REMOVE X (CDR L])
```

(REVERSE

```
  [LAMBDA (L)
    (PROG (U)
          (DECLARE (LOCALVARS U))
      CL:LOOP
          (COND
             ((NLISTP L)
              (RETURN U)))
          (SETQ U (CONS (CAR L)
                        U))
          (SETQ L (CDR L))
          (GO CL:LOOP])
```

(RPT

```
  [LAMBDA (RPTN RPTF)
    (DECLARE (SPECVARS RPTN)
             (LOCALVARS RPTF))                          ; Edited 6-Apr-87 13:57 by Pavel
    (PROG (RPTV)
          (DECLARE (LOCALVARS RPTV))
      LP  (COND
             ((IGREATERP RPTN 0)
              (SETQ RPTV (EVAL RPTF 'INTERNAL))
              (SETQ RPTN (SUB1 RPTN))
              (GO LP))
             (T (RETURN RPTV])
```

(RPTQ

```
  [NLAMBDA RPTZ
    (PROG ((RPTN (EVAL (CAR RPTZ)
                       'INTERNAL))
           RPTV)
          (DECLARE (SPECVARS RPTN))
      RPTQLOOP
          (COND
             ((IGREATERP RPTN 0)
              (SETQ RPTV (APPLY (FUNCTION PROGN)
                                (CDR RPTZ)
                                'INTERNAL))
              (SETQ RPTN (SUB1 RPTN))
              (GO RPTQLOOP)))
          (RETURN RPTV])
```

(FRPTQ

```
  [NLAMBDA RPTZ
    (DECLARE (LOCALVARS . T))
    (PROG ((RPTN (EVAL (CAR RPTZ)
                       'INTERNAL))
           RPTV)
      RPTQLOOP
          (COND
             ((IGREATERP RPTN 0)
              (SETQ RPTV (APPLY (FUNCTION PROGN)
                                (CDR RPTZ)
                                'INTERNAL))
```

```
              (SETQ RPTN (SUB1 RPTN))
              (GO RPTQLOOP)))
          (RETURN RPTV])
```

(SASSOC

```
  [LAMBDA (KEY ALST)
    (PROG NIL
      LP  [COND
             ((NLISTP ALST)
              (RETURN NIL))
             ((EQUAL (CAAR ALST)
                     KEY)
              (RETURN (CAR ALST]
          (SETQ ALST (CDR ALST))
          (GO LP])
```

(SAVEDEF

```
  [LAMBDA (X)
    (COND
       ((ATOM X)
        (SAVEDEF1 X))
      (T (MAPCAR X (FUNCTION SAVEDEF1])
```

(SAVEDEF1

```
  [LAMBDA (X)
    (PROG ((DF (GETD X)))
          (RETURN (COND
                     (DF (PUTPROP X [SETQ X (SELECTQ (FNTYP X)
                                                     ((SUBR SUBR* FSUBR FSUBR*)
                                                         'SUBR)
                                                     ((EXPR EXPR* FEXPR FEXPR*)
                                                         'EXPR)
                                                     ((CEXPR CEXPR* CFEXPR CFEXPR*)
                                                         'CODE)
                                                     (COND
                                                        ((EXPRP X)
                                                         'EXPR)
                                                        (T 'LIST]
                                    DF)                                  ; NOTE: this call to PUTPROP is changed to /PUTPROP later in
                                                                         ; the loadup.
                                X])
```

(SELECT

```
  [NLAMBDA .SELEC.
    (DECLARE (LOCALVARS . T))                                            (* dcl%: 12 Dec 78 09%:08)
    (APPLY 'PROGN (SELECT1 (EVAL (CAR .SELEC.)
                                 'SELECTQ)
                           (CDR .SELEC.))
           'SELECTQ])
```

(SELECT1

```
  [LAMBDA (M L)
    (DECLARE (LOCALVARS . T))                                            (* edited%: 8 Dec 78 13%:53)
    (PROG (C A)
      LP  (SETQ C L)
          (COND
             ((NULL (SETQ L (CDR L)))
              (RETURN C))
             ([NLISTP (CAR (SETQ C (CAR C]
              (AND (EQ M (EVAL (CAR C)
                               'INTERNAL))
                   (RETURN (CDR C)))
              (GO LP)))
          (SETQ A (CAR C))
      L2  (COND
             ((EQ M (EVAL (CAR A)
                          'INTERNAL))
              (RETURN (CDR C)))
             ((LISTP (SETQ A (CDR A)))
              (GO L2))
             (T (GO LP])
```

(SELECTC

```
  [NLAMBDA SELCQ                                                         (* lmm "28-FEB-82 16:07")
    (DECLARE (LOCALVARS . T))
    (APPLY 'PROGN ([LAMBDA (M L)
                     (PROG (C TL)
                       LP  (SETQ C L)
                           [COND
                              ((NULL (SETQ L (CDR L)))
                               (RETURN C))
```

```
                              ((OR (EQ (SETQ TL (EVAL (CAR (SETQ C (CAR C)))
                                                      'INTERNAL))
                                       M)
                                   (AND (LISTP TL)
                                        (FMEMB M TL)))
                               (RETURN (CDR C]
                         (GO LP]
                   (EVAL (CAR SELCQ)
                         'SELECTQ)
                   (CDR SELCQ))
          'SELECTQ])
```

(SETQQ

```
  [NLAMBDA (X Y)
    (SET X Y])
```

(SOME

```
  [LAMBDA (SOMEX SOMEFN1 SOMEFN2)                          ; SOME compiles open.
    (PROG NIL
      CL:LOOP
          (COND
             ((NLISTP SOMEX)
              (RETURN NIL))
             ((APPLY* SOMEFN1 (CAR SOMEX)
                      SOMEX)
              (RETURN SOMEX)))
          [SETQ SOMEX (COND
                        (SOMEFN2 (APPLY* SOMEFN2 SOMEX))
                        (T (CDR SOMEX]
          (GO CL:LOOP])
```

(STRMEMB

```
  [LAMBDA (X Y)                                            (* rmk%: " 6-JUN-82 15:08")
    (PROG (C N)
          (DECLARE (LOCALVARS C N))
          (SETQ Y (SUBSTRING Y 1))
      B   (SETQ N 1)
      A   (COND
             ((NULL (SETQ C (NTHCHARCODE X N)))
              (RETURN Y)))
          (COND
             ((EQ C (NTHCHARCODE Y N))
              (SETQ N (ADD1 N))
              (GO A)))
          (COND
             ((NULL (GNC Y))
              (RETURN))
             (T (GO B])
```

(SUB1VAR

```
  [NLAMBDA (SUB1X)
    (SET SUB1X (SUB1 (EVAL SUB1X])
```

(SUBSET

```
  [LAMBDA (MAPX MAPFN1 MAPFN2)
    (DECLARE (LOCALVARS . T))
    (PROG (RESULT TAIL)
      LP  [COND
             ((NLISTP MAPX)
              (RETURN RESULT))
             ((APPLY* MAPFN1 (CAR MAPX))
              (COND
                 [(NULL RESULT)
                  (SETQ RESULT (SETQ TAIL (CONS (CAR MAPX]
                 (T [SETQ TAIL (CDR (FRPLACD TAIL (FRPLACD (CONS (CAR MAPX)
                                                                 TAIL]
                                                          ; Eseentially an open TCONC.

                    ]
          [SETQ MAPX (COND
                       (MAPFN2 (APPLY* MAPFN2 MAPX))
                       (T (CDR MAPX]
          (GO LP])
```

(SUBST

```
  [LAMBDA (NEW OLD EXPR)                                   (* lmm "16-FEB-82 22:11")
    (COND
       ((NULL EXPR)
        NIL)
       ((NLISTP EXPR)
        (COND
```

```
          ((EQ OLD EXPR)
           (COPY NEW))
          (T EXPR)))
      (T (CONS [COND
                 ((EQUAL OLD (CAR EXPR))
                  (COPY NEW))
                 (T (SUBST NEW OLD (CAR EXPR]
               (SUBST NEW OLD (CDR EXPR])
```

(TAILP

```
[LAMBDA (X Y)                                          ; Edited 29-Aug-91 21:23 by jrb:
  ;; True if X is A tail of Y X and Y non-null.       ; Included with editor for block compilation purposes.
  (AND X (PROG NIL
           LP  (COND
                  ((EQ X Y)
                   (RETURN X))
                  ((NLISTP Y)
                   (RETURN NIL)))
               (SETQ Y (CDR Y))
               (GO LP])
```

(TCONC

```
[LAMBDA (PTR X)
  (PROG (XX)
        (DECLARE (LOCALVARS XX))
        (RETURN (COND
                   ((NULL PTR)
                    (CONS (SETQ XX (CONS X NIL))
                          XX))
                   ((NLISTP PTR)
                    (ERROR '"bad argument - TCONC" PTR))
                   ((NULL (CDR PTR))
                    (FRPLACA PTR (CONS X NIL))
                    (FRPLACD PTR (CAR PTR)))
                   (T (FRPLACD PTR (CDR (FRPLACD (CDR PTR)
                                                 (FRPLACD (CONS X (CDR PTR])
```

(UNION

```
[LAMBDA (X Y)
  (DECLARE (LOCALVARS . T))                            (* bvm%: "30-Jun-86 16:59")
```

;;; Defined explicitly to be Y prepended with any elements of X not in Y

```
  (for ELT in X bind HEAD TAIL unless (COND
                                        ((LITATOM ELT)          ; Optimize MEMBER for a common case
                                         (FMEMB ELT Y))
                                        (T (MEMBER ELT Y)))
     do [COND
          [TAIL (RPLACD TAIL (SETQ TAIL (CONS ELT NIL]
          (T (SETQ HEAD (SETQ TAIL (CONS ELT NIL]
     finally (RETURN (COND
                       (TAIL (RPLACD TAIL Y)
                             HEAD)
                       (T Y])

)
```

;; ERRORSET stuff

```
(DEFINEQ
```

(ERSETQ

```
[NLAMBDA ERSETX                                        (* bvm%: "14-Oct-86 11:42")
  (ERRORSET (CONS 'PROGN ERSETX)
         T])
```

(NLSETQ

```
[NLAMBDA NLSETX                                        (* bvm%: "14-Oct-86 11:41")
  (ERRORSET (CONS 'PROGN NLSETX)
         NIL])
```

(XNLSETQ

```
[NLAMBDA (XNLSETQX XNLSETFLG XNLSETFN)
  (ERRORSET XNLSETQX XNLSETFLG XNLSETFN])
```

(RESETLST

```
[NLAMBDA RESETX                                        (* bvm%: "11-Nov-86 22:26")
  ;; RESETLST and RESETSAVE together permit the user to combine the effects of several RESETVAR's and RESETFORM's under one function.
  ;; RESETLST acts like an ERRORSET which takes an indefinite number of forms, i.e. like PROGN, and errorset protects them, and restores all
  ;; RESETSAVE's performed while inside of RESETLST.  RESETLST compiles open.
```

```
    (RESETLST
        (\EVPROGN RESETX))])
```

(RESETSAVE

```
  [NLAMBDA RESETX                                                    (* wt%: "23-JUL-79 21:08")
    (DECLARE (LOCALVARS . T))
```

;; for use under a RESETLST.

```
    (SETQ SI::*RESETFORMS* (CONS [COND
                                    [(AND (CAR RESETX)
                                          (LITATOM (CAR RESETX)))
```

;; This is the (RESETSAVE var value) form

```
                                     (PROG1 (CONS (CAR RESETX)
                                                  (GETTOPVAL (CAR RESETX)))
                                            (SETTOPVAL (CAR RESETX)
                                                   (\EVAL (CADR RESETX))))]
                                    [(CDR RESETX)
```

;; This is the (RESETSAVE savingform restore-form). CADR of the entry we save is the value of the
;; saving form. The variable OLDVALUE is bound to this value during restoration. This makes it more
;; convenient for the restoration to be conditional, e.g. the user can perform (RESETSAVE (FOO
;; mumble) '(AND pred (FIE OLDVALUE)))

```
                                     (LIST (\EVAL (CADR RESETX))
                                           (\EVAL (CAR RESETX]
                                    (T
```

;; This is the (RESETSAVE (fn arg)) form, a special case of the above. Save (fn oldval) as the
;; restoration expression.

```
                                       (LET ((FORM (CAR RESETX)))
                                            (LIST (LIST (COND
                                                        ((EQ (CAR FORM)
                                                             'SETQ)
```

;; Silly special case: in (RESETSAVE (SETQ var (fn arg))) ignore the
;; SETQ for restoration purposes.

```
                                                         (CAR (CADDR FORM)))
                                                        (T (CAR FORM)))
                                                  (\EVAL FORM]
                                  SI::*RESETFORMS*])
```

(RESETFORM

```
  [NLAMBDA RESETZ                                                    ; Edited  3-Sep-87 12:15 by bvm:
```

;; Similar to RESETVAR. Permits evaluation of a form while resetting a system state, and provides for the system to be returned to that state after
;; evaluation. RESETX is a form, e.g. (OUTPUT T), (PRINTLEVEL 2) etc. RESETX is evaluated and its value saved. Then RESETY is evaluated
;; under errorset protection and then (CAR RESETX) is applied to the result of the evaluation of X. If an error occurs during the evaluation of
;; FORM, the effect of RESETX is still 'undone'

```
    (LET [(SI::*RESETFORMS* (LIST (LIST (LIST (CAAR RESETZ)
                                              (\EVAL (CAR RESETZ]
          (DECLARE (SPECVARS SI::*RESETFORMS*))
          (CL:UNWIND-PROTECT
              (\EVPROGN (CDR RESETZ))
              (SI::RESETUNWIND))])
```

(RESETVARS

```
  [NLAMBDA RESETX                                                    ; Edited 25-Nov-86 23:16 by bvm:
    (LET [(SI::*RESETFORMS* (PROGN                                   ; Initialize *RESETFORMS* to list of vars and old values
                                  (for V in (CAR RESETX) collect (if (LISTP V)
                                                                     then (SETQ V (CAR V)))
                                                                 (CONS V (GETTOPVAL V]
          (DECLARE (LOCALVARS . T)
                   (SPECVARS SI::*RESETFORMS*))
          (CL:UNWIND-PROTECT
              (PROGN                                                 ; Set the variables to new values, execute prog body
                     (for V in (CAR RESETX) do (if (LISTP V)
                                                   then (SETTOPVAL (CAR V)
                                                                   (\EVPROG1 (CDR V)))
                                                 else                ; initial value NIL
                                                      (SETTOPVAL V NIL)))
                     (APPLY 'PROG (CONS NIL (CDR RESETX))
                            'INTERNAL))
              (SI::RESETUNWIND))])
```

(RESETVAR

```
  [NLAMBDA (RESETX RESETY RESETZ)                                    ; Edited 19-Mar-87 16:06 by jrb:
```

;; Permits evaluation of a form while resetting a top level variable, and provides for the variable to be automatcally restored after valuation. In this
;; way, the user pays when he wants to 'rebind' a globalvariable, but does not have to pay for the possiblity, as would be the case if variables such
;; as DFNFLG, LISPXHISTORY, etc. were not global, i.e. were looked up. In the event of a control-D, or control-C reenter, the variabes will still be
;; restored by EVALQT. Note that STKEVALs will not do the right t on variables reset by RESETVAR.

```
    (LET [(SI::*RESETFORMS* (LIST (CONS RESETX (GETTOPVAL RESETX]
          (DECLARE (SPECVARS SI::*RESETFORMS*))
```

```
         (CL:UNWIND-PROTECT
             (PROGN (SETTOPVAL RESETX (\EVAL RESETY))
                    (\EVAL RESETZ))
             (SI::RESETUNWIND))])
```

(SI::RESETUNWIND

```
  [LAMBDA (NORMALP)                                                    (* bvm%: " 4-Nov-86 16:53")
    (while (LISTP SI::*RESETFORMS*) bind OLDVALUE RESETZ do (SETQ RESETZ (pop SI::*RESETFORMS*))
                                                            (if (LISTP (CAR RESETZ))
                                                                then
                                                                ; RESETSAVE and RESETFORM do this
                                                                   (SETQ OLDVALUE (if (CDR RESETZ)
                                                                                      then
;; occurs for RESETSAVE's when second argument is specified.  In this case, (CADR RESETZ) is the
;; value of the saving form, i.e. the first argument to RESETSAVE.
                                                                                                    (CADR RESETZ)
                                                                                          else (CADAR RESETZ)))
                                                                   (APPLY (CAAR RESETZ)
                                                                          (CDAR RESETZ))
                                                            else
                                                                ; RESETSAVE of a symbol sets its value
                                                                 (SETTOPVAL (CAR RESETZ)
                                                                        (CDR RESETZ])

)

(DEFINEQ
```

(SI::NLSETQHANDLER

```
  [LAMBDA (C)                                                          (* bvm%: "16-Sep-86 19:19")
    (if (AND SI::*NLSETQFLAG* NLSETQGAG)
        then (ABORT C])

)

(RPAQ? SI::*NLSETQFLAG* )

(RPAQ? RESETSTATE )

(PUTPROPS RESETTOPVALS INFO (EVAL BINDS))

(DEFINEQ
```

(GENSYM

```
  [LAMBDA (PREFIX NUMSUFFIX OSTRBUFFER NEW? CHARCODE)                  (* bvm%: "25-Aug-86 16:03")
```

;;; Create a unique SYMBOL with the given prefix.

```
    (OR (NULL PREFIX)
        (STRINGP PREFIX)
        (LITATOM PREFIX)
        (CL:STRINGP PREFIX)
        (\ILLEGAL.ARG PREFIX))                                         ; The prefix has to be something string-like
    (OR (NULL NUMSUFFIX)
        (FIXP NUMSUFFIX)
        (\ILLEGAL.ARG NUMSUFFIX))                                      ; Any number-suffix better be numeric
    (OR (NULL OSTRBUFFER)
        (STRINGP OSTRBUFFER)
        (\ILLEGAL.ARG OSTRBUFFER))                                     ; Any buffer you supply better be an Interlisp string
    (OR (NULL CHARCODE)
        (CHARCODEP CHARCODE)
        (\ILLEGAL.ARG CHARCODE))                                       ; Any charcode better really be one
    (PROG ((BUFSIZE \GS.BUFSIZE)
           (NUMLEN \GS.NUMLEN)
           [BUF (OR (STRINGP \GS.BUF)
                    (SETQ \GS.BUF (ALLOCSTRING \GS.BUFSIZE]
           (PREFIXLEN 0)
           BEG.I ATOM)
          (COND
             [(OR (NULL PREFIX)
                  (EQ (SETQ PREFIXLEN (NCHARS PREFIX))
                      0))
              (SETQ PREFIX)
              (COND
                 ((NULL CHARCODE)                                      ; Here's the default case
                  (SETQ CHARCODE (CHARCODE A]
             ((IGREATERP PREFIXLEN (IDIFFERENCE BUFSIZE 10))
              (ERROR PREFIX "Too long")))
          (COND
             ((COND
                 (OSTRBUFFER (COND
                                ((NULL NUMSUFFIX)
                                 (HELP "OSTRBUFFER supplied without NUMSUFFIX"))
                                ((ILESSP (SETQ BUFSIZE (NCHARS OSTRBUFFER))
                                         (IPLUS 12 PREFIXLEN))
```

```
                                (ERROR OSTRBUFFER "Too short")))
                            T)
                    (NUMSUFFIX                                  ; Insulate the normal \GS.BUF from random intrusions
                           [SETQ OSTRBUFFER (ALLOCSTRING (SETQ BUFSIZE (IPLUS PREFIXLEN 12]
                           T))
                (SETQ BUF OSTRBUFFER)))
      A     (UNINTERRUPTABLY
                [COND
                   [(COND
                       (OSTRBUFFER                              ; Use the user-supplied buffer, or a freshly cons'd one if he
                                                                ; supplied NUMSUFFIX without OSTRBUFFER
                             T)
                        ((NOT (FIXP GENNUM))                    ; Disaster recovery
                         (SETQ GENNUM 0)
                         T))
                    (SETQ NUMLEN (\GS.INITBUF BUF BUFSIZE (OR NUMSUFFIX GENNUM]
                   (T ;; In this case, we have kept account of the contents of \GS.BUF so we don't have to call \GS.INITBUF afresh, but rather
                      ;; merely 'patch up' the effect of adding 1 to GENNUM
                      [COND
                         ((COND
                             ((NOT (IEQP GENNUM \GS.OGENNUM))   ; User perhaps has reset GENNUM
                              (COND
                                 ((ILESSP GENNUM 0)
                                  (SETQ GENNUM 0)))
                              T)
                             ((IGEQ GENNUM MAX.FIXP)            ; Sigh, two's complement wrap-around
                              (SETQ GENNUM 0)
                              T))
                          (SETQ NUMLEN (\GS.INITBUF BUF BUFSIZE GENNUM]
                                                                ; Increment the GENNUM counter and the string buffer buffer.
                      [COND
                         ((for CNT C to NUMLEN as I from BUFSIZE by -1 do
                                                                ; Simulates a BCD type add in the gensym string
                                                                       (SETQ C (NTHCHARCODE \GS.BUF I))
                                                                       [COND
                                                                          ((ILEQ (add C 1)
                                                                                 (CHARCODE 9))
                                                                                                ; ha, carry stops here
                                                                           (RPLCHARCODE BUF I C)
                                                                           (RETURN))
                                                                          (T (RPLCHARCODE BUF I (CHARCODE 0]
                             finally (RETURN T))                ; Sigh, we have to extend the numerical part
                          (RPLCHARCODE BUF (IDIFFERENCE BUFSIZE NUMLEN)
                                 (CHARCODE 1))
                          (SETQ NUMLEN (add \GS.NUMLEN 1]
                      (SETQ \GS.OGENNUM (add GENNUM 1]          ; BEG.I will be the beginning index, in the buffer, for the atom
                (SETQ BEG.I (ADD1 (IDIFFERENCE BUFSIZE NUMLEN)))
                (COND
                   (CHARCODE (RPLCHARCODE BUF (add BEG.I -1)
                                  CHARCODE)))
                (COND
                   (PREFIX (RPLSTRING BUF (SETQ BEG.I (IDIFFERENCE BEG.I PREFIXLEN))
                                 PREFIX)))
                (SETQ \GS.STR (SUBSTRING BUF BEG.I BUFSIZE \GS.STR))
                (SETQ ATOM (MKATOM \GS.STR)))
             (COND
                ((NUMBERP ATOM)
                 (\ILLEGAL.ARG PREFIX)))
             (RETURN ATOM])
```

(**GENSYM?**

```
[LAMBDA (X)                                                     (* lmm " 1-JUN-81 08:30")
   (AND (LITATOM X)
        (EQ (NTHCHARCODE X -5)
            (CHARCODE A))
        (FIXP (NTHCHAR X -4))
        (FIXP (NTHCHAR X -3))
        (FIXP (NTHCHAR X -2))
        (FIXP (NTHCHAR X -1))
        T])
```

(**\GS.INITBUF**

```
[LAMBDA (BUF BUFSIZE N)                                         (* lmm "14-Apr-85 20:36")
```

;; Initializes BUF (which must be a stringp of length BUFSIZE) with the digits of N right-justified and left-0 padded up to a minimum of 4 digits.
;; Returns the decimal length of N

```
   (PROG (NUMLEN)
         (RPLSTRING BUF [IDIFFERENCE BUFSIZE (if (ILESSP N 10000)
                                                 then           ; Trick to get leading zeros
                                                      (SETQ N (IPLUS N 10000))
                                                      (SETQ NUMLEN 4)
                                                 else (SUB1 (SETQ NUMLEN (NCHARS N]
                N)
         (AND (EQ BUF \GS.BUF)
```

```
                (SETQ \GS.NUMLEN NUMLEN))
          (RETURN NUMLEN])

)
```

;; GENSYM garbage

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \GS.BUFSIZE 100)

(CONSTANTS (\GS.BUFSIZE 100))
)
)

(RPAQ? GENNUM 0)

(RPAQ? \GS.OGENNUM -1)

(RPAQ? \GS.NUMLEN 0)

(RPAQ? \GS.BUF NIL)

(RPAQ? \GS.STR (ALLOCSTRING 0))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS GENNUM \GS.OGENNUM \GS.NUMLEN \GS.BUF \GS.STR)
)

(ADDTOVAR PRETTYEQUIVLST (SELECTC . SELECTQ))

(ADDTOVAR DWIMEQUIVLST (SELECTC . SELECTQ))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(CL:PROCLAIM '(GLOBAL MAKESYSDATE MAKESYSNAME))

(PUTPROPS MISC FILETYPE :BCOMPL)

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

(ADDTOVAR NLAMA RESETVARS RESETFORM RESETSAVE RESETLST NLSETQ ERSETQ SELECTC SELECT FRPTQ RPTQ DEFINEQ
                        APPENDTOVAR ADDTOVAR)

(ADDTOVAR NLAML RESETVAR XNLSETQ SUB1VAR SETQQ ADD1VAR)

(ADDTOVAR LAMA APPEND)
)

(PUTPROPS MISC COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1991))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

CONSTANT INDEX