File created: 17-May-90 16:09:40 **{DSK}<usr>local>lde>lispcore>sources>UNDO.;2**

changes to: (VARS UNDOCOMS)

previous date: 8-Jan-88 13:04:47 {DSK}<usr>local>lde>lispcore>sources>UNDO.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1984, 1986, 1988, 1990 by Venue & Xerox Corporation.  All rights reserved.

(RPAQQ UNDOCOMS
       [(FNS SAVESET UNDOSET SAVESETQ SAVESETQQ RPAQQ RPAQ RPAQ? RPLNODE RPLNODE2 NEW/FN UNDOSAVE UNDOLISPX
             UNDOLISPX1 UNDOPRINT UNDOLISPX2 UNDOLISPX3 UNSET /LISPXPUT /PUT-1 /PUT+1 UNDONLSETQ UNDONLSETQ1
             RESETUNDO /DEFINEQ /DEFINE /PRINTLEVEL)
        (INITVARS (%#UNDOSAVES)
               (UNDOSIDE0)
               (TESTMODEFLG))
        (ADDVARS (LISPXFNS (SETQ . SAVESETQ)
                           (SET . SAVESET)
                           (SETQQ . SAVESETQQ)
                           (DEFINEQ . /DEFINEQ)
                           (DEFINE . /DEFINE)
                           (PRINTLEVEL . /PRINTLEVEL))
               (/FNS /ADDPROP /ATTACH /CONTROL /DELETECONTROL /DREMOVE /DREVERSE /DSUBST /ECHOCONTROL /ECHOMODE
                     /LCONC /LISTPUT /LISTPUT1 /MAPCON /MAPCONC /MOVD /NCONC /NCONC1 /PUT /PUTASSOC /PUTD /PUTDQ
                     /PUTHASH /PUTPROP /RADIX /RAISE /REMPROP /RPLACA /RPLACD /RPLNODE /RPLNODE2 /SET /SETA
                     /SETATOMVAL /SETBRK /SETD /SETPROPLIST /SETREADTABLE /SETSEPR /SETSYNTAX /SETTERMTABLE
                     /SETTOPVAL /TCONC))
        (FNS /ADDPROP /ATTACH /CONTROL /DELETECONTROL /DREMOVE /DREVERSE /DSUBST /ECHOCONTROL /ECHOMODE /LCONC
             /LISTPUT /LISTPUT1 /MAPCON /MAPCONC /MOVD /NCONC /NCONC1 /PUT /PUTASSOC /PUTD /PUTDQ /PUTHASH
             /PUTPROP /RADIX /RAISE /REMPROP /RPLACA /RPLACD /RPLNODE /RPLNODE2 /SET /SETA /SETATOMVAL /SETBRK
             /SETD /SETPROPLIST /SETREADTABLE /SETSEPR /SETSYNTAX /SETTERMTABLE /SETTOPVAL /TCONC)
        [P (SETQ LISPXFNS (UNION LISPXFNS (MAPCAR /FNS (FUNCTION (LAMBDA (X Y)
                                                                  (CONS (PACK (CDR (DUNPACK X CHCONLST)))
                                                                        X]
        (P (MOVD? 'RPLNODE 'FRPLNODE)
           (MOVD? 'RPLNODE2 'FRPLNODE2))
        (BLOCKS (NIL UNSET RPLNODE RPLNODE2 /LISPXPUT /PUT-1 /PUT+1 (LINKFNS . T)
                     UNDONLSETQ UNDONLSETQ1 (GLOBALVARS UNDOSTATS CLEARSTKLST DWIMFLG SPELLINGS3 LISPXHISTORY
                                                        %#UNDOSAVES)
                     RESETUNDO UNDOPRINT)
               (NIL RPAQ RPAQQ (LOCALVARS . T))
               (SAVESET SAVESET (LOCALVARS . T)
                        (GLOBALVARS CLEARSTKLST))
               (NIL UNDOSET (GLOBALVARS SPAGHETTIFLG))
               (NIL NEW/FN (GLOBALVARS TESTMODEFLG LISPXFNS CHCONLST /FNS))
               (UNDOLISPXBLOCK UNDOSAVE UNDOLISPX UNDOLISPX1 UNDOLISPX2 UNDOLISPX3 (ENTRIES UNDOSAVE UNDOLISPX
                                                                                            UNDOLISPX1 UNDOLISPX2)
                        (BLKLIBRARY LISPXWATCH)
                        (GLOBALVARS UNDOSAVES UNDOSTATS %#UNDOSAVES DWIMFLG DWIMWAIT LISPXHISTORY CLISPTRANFLG
                                EDITQUIETFLG MAXLEVEL)
                        (LOCALFREEVARS UNDONEFLG))
               (NIL /ADDPROP /ATTACH /CONTROL /DELETECONTROL /DREMOVE /DREVERSE /DSUBST /ECHOCONTROL /ECHOMODE
                    /LCONC /LISTPUT /LISTPUT1 /MAPCON /MAPCONC /MOVD /NCONC /NCONC1 /PRINTLEVEL /PUT /PUTASSOC
                    /PUTD /PUTDQ /PUTHASH /PUTPROP /REMPROP /RPLACA /RPLACD /RPLNODE /RPLNODE2 /SET /SETA
                    /SETBRK /SETD /SETPROPLIST /SETSEPR /SETSYNTAX /SETATOMVAL /SETTOPVAL /TCONC (GLOBALVARS
                                                                                                  UNDOSTATS)
                     (LINKFNS . T)))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA /DEFINEQ SAVESETQ)
                        (NLAML /PUTDQ UNDONLSETQ RPAQ? RPAQ RPAQQ SAVESETQQ)
                        (LAMA /NCONC])

(DEFINEQ

(SAVESET
  [LAMBDA (NAME VALUE TOPFLG FLG)                                    ; Edited  8-Jan-88 12:52 by bvm

    ;; Sets NAME to VALUE, binding used is most recent unless TOPFLG is T in which case always uses top level binding.  The setting is always
    ;; undoable in conventional way.  In addition, if the binding being reset is a top level binding, its value is saved on its property list where it can be
    ;; recovered via UNSET, even outside the scope of the history list, and (NAME RESET) is printed.  If FLG is 'NOPRINT', the printing is suppressed.
    ;; This is the case when called from UNSET.  If FLG is 'NOPROPSAVE', binding is not saved on property list.  This is the case when called from
    ;; /SET.  Note that SET becomes SAVESET in type-ins.  /SET is used when in TESTMODE.  If FLG is NOSTACKUNDO, the call is not undoable
    ;; when the variable in question is bound on the stack.  This is the case on calls from RPAQ, RPAQQ, ADDTOVAR, etc.

    [COND
       ((NOT (LITATOM NAME))
        (LISPERROR "ARG NOT LITATOM" NAME))
       [(NULL NAME)
        (COND
           (VALUE (LISPERROR "ATTEMPT TO SET NIL OR T" VALUE]
       [(EQ NAME T)
```

```
        (COND
           ((NEQ VALUE T)
            (LISPERROR "ATTEMPT TO SET NIL OR T" VALUE]
       (T (PROG (PTR OLDVAL TEM NEWFLG)
                [SETQ OLDVAL (COND
                                (TOPFLG (GETTOPVAL NAME))
                                ((SETQ PTR (STKSCAN NAME))      ; PTR=NIL means we are working on the top-level value.
                                 (EVALV NAME))
                                (T                              ; This is most efficient for both deep and shallow when we know
                                                                ; there are no bindings.
                                   (GETATOMVAL NAME]
                (COND
                   ((AND (NULL PTR)
                         (EQ DFNFLG 'ALLPROP)
                         (NEQ OLDVAL 'NOBIND))                  ; from LOAD ALLPROP
                    (/PUT NAME 'VALUE VALUE)
                    (AND ADDSPELLFLG (ADDSPELL NAME T))
                    (RETURN VALUE))
                   ([AND PTR (OR (NULL LISPXHIST)
                                 (EQ FLG 'NOSTACKUNDO)
                                 (EQ FLG 'NOUNDO]                ; Bound on stack, but we're not saving, so stop agonizing
                    (SETQ FLG 'NOUNDO)
                    (GO OUT))
                   ([AND [SETQ TEM (SOME (LISTGET1 LISPXHIST 'SIDE)
                                         (FUNCTION (LAMBDA (X)
                                                     (AND (LISTP X)
                                                          (EQ (CAR X)
                                                              'UNDOSET)
                                                          (EQ (CADDR X)
                                                              NAME)
                                                          (EQ (CADR X)
                                                              PTR]
                         (NOT (TAILP TEM (LISTP (EVQ UNDOSIDE0]
                    ;; this variable has already been set, undoably, in this event.  The TAILP check is to make sure it hasn't happened above an
                    ;; UNDONLSETQ now in effect.
                    (SETQ FLG 'NOUNDO)
                    (GO OUT))
                   ((OR PTR (EQ FLG 'NOPROPSAVE)
                        (EQ FLG 'NOSAVE))
                    ;; The first predicate is because SAVESET only works for top level bindings.  The second indicates a call from /SET or
                    ;; /SETQ.  Note that in both cases the variable is NOT added to the spelling list.  (The check for NOSAVE is for backwards
                    ;; compatibility.  the NOPROPSAVE is newer.)
                    (GO OUT))
                   ((EQ (EQUALN OLDVAL VALUE 1000)
                        T)
                    ;; note that we still need to save the undo information because of possibility that we are under an UNDONLSETQ.  e.g. user
                    ;; does (SAVESET --) then several SETQ's than an ERROR! and wants to be sure variable was what it was when he entered
                    ;; the function.

                    )
                   (T                                           ; Variable is being reset.
                      [AND (NEQ DFNFLG T)
                           (COND
                              ([NULL (SETQ NEWFLG (EQ OLDVAL 'NOBIND]
                               (COND
                                  ((NEQ FLG 'NOPRINT)
                                   (EXEC-FORMAT "(~S reset)~%%" NAME)))
                               (/PUT NAME 'VALUE OLDVAL]
                      (MARKASCHANGED NAME 'VARS NEWFLG)))
                (AND ADDSPELLFLG (ADDSPELL NAME T))
         OUT (COND
                [PTR (SET NAME VALUE)
                     (COND
                        ((EQ FLG 'NOUNDO)
                         (RELSTK PTR))
                        (T ;; A stack pointer to the frame of NAME's binding has been created and not released.  This is because it is
                           ;; being saved for possible undoing.  This is exceedingly crufty.
                           (UNDOSAVE (LIST 'UNDOSET PTR NAME OLDVAL)
                                  LISPXHIST]
                (T (COND
                      (TOPFLG
                             ;; Can't just SETATOMVAL, cause if TOPFLG we didn't bother searching for intermediate binding, which
                             ;; would be found by shallow SETATOMVAL
                             (SETTOPVAL NAME VALUE))
                      (T (SETATOMVAL NAME VALUE)))
                   (COND
                      ((AND LISPXHIST (NEQ FLG 'NOUNDO))
                       (UNDOSAVE (LIST 'UNDOSET NIL NAME OLDVAL)
                              LISPXHIST]
   VALUE])
```

(**UNDOSET**

```
  [LAMBDA (PTR NAME VALUE)                                    (* rmk%: " 5-JAN-82 01:35")
    (PROG (TEM)
          (RETURN (COND
                      ((NULL PTR)
                       (AND LISPXHIST (UNDOSAVE (LIST 'UNDOSET NIL NAME (GETTOPVAL NAME))
                                                LISPXHIST))
                       (SETTOPVAL NAME VALUE)
                       T)
                      ((NULL SPAGHETTIFLG)
                       (COND
                          ((EQ (CDR PTR)
                               NAME)
                           [AND LISPXHIST (UNDOSAVE (LIST 'UNDOSET PTR NAME (GETTOPVAL NAME]
                           (SETTOPVAL NAME VALUE)
                           T)))
                      ((SETQ TEM (FRAMESCAN NAME PTR))
                       (AND LISPXHIST (UNDOSAVE (LIST 'UNDOSET PTR NAME (STKARG TEM PTR))
                                                LISPXHIST))
                       (SETSTKARG NAME PTR VALUE)
                       T])
```

(SAVESETQ

```
  [NLAMBDA SETQX                                              (* wt%: "13-JUN-78 23:53")
    (SAVESET (CAR SETQX)
           (APPLY 'PROG1 (CDR SETQX)
                  'INTERNAL])
```

(SAVESETQQ

```
  [NLAMBDA (SETQX SETQY)
    (SAVESET SETQX SETQY])
```

(RPAQQ

```
  [NLAMBDA (X Y)                                              (* rmk%: " 4-JAN-82 13:02")
    (SAVESET X Y T])
```

(RPAQ

```
  [NLAMBDA (RPAQX RPAQY)                                      (* rmk%: " 4-JAN-82 13:03")
    (SAVESET RPAQX (EVAL RPAQY 'INTERNAL)
           T])
```

(RPAQ?

```
  [NLAMBDA (RPAQX RPAQY)                                      (* lmm "23-JUL-83 16:12")
    ;; RPAQ?  and RPAQQ are used by PRETTYDEF to save VARS.
    (OR (NEQ (GETTOPVAL RPAQX)
             'NOBIND)
        (SETTOPVAL RPAQX (EVAL RPAQY])
```

(RPLNODE

```
  [LAMBDA (X A D)
    (AND (NLISTP X)
         (ERRORX (LIST 4 X)))
    (RPLACA X A)
    (RPLACD X D])
```

(RPLNODE2

```
  [LAMBDA (X Y)                                               (* rmk%: " 4-MAR-82 22:07")
    ;; 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
       ((AND Y (NLISTP Y))
        (ERRORX (LIST 4 Y)))
       (T (RPLNODE X (CAR Y)
                   (CDR Y])
```

(NEW/FN

```
  [LAMBDA (FN)                                                (* bvm%: " 1-Jan-84 16:50")
    (PROG (FN1)
          [COND
             [(EQ (CHCON1 FN)
                  (CHARCODE /))
              (SETQ FN1 (PACK (CDR (DUNPACK FN CHCONLST]
             (T (SETQ FN1 FN)
                (SETQ FN (PACK* '/ FN]
          (SETQ /FNS (/NCONC1 /FNS FN))                       ; Used to do this for TESTMODE, but that not implemented any
                                                              ; more: (/PUT FN (QUOTE \DEF) (GETD FN1))
          (SETQ LISPXFNS (/NCONC1 LISPXFNS (CONS FN1 FN)))
          (RETURN FN])
```

(UNDOSAVE

```
  [LAMBDA (UNDOFORM HISTENTRY)                                    (* wt%: 7-JUN-77 0 41)
    (AND (NULL HISTENTRY)
         (SETQ HISTENTRY (EVQ LISPXHIST)))
    (AND HISTENTRY (PROG (Y N)
                         (LISPXWATCH UNDOSAVES)
                         [COND
                            ([NULL (CAR (SETQ Y (CDR (FMEMB 'SIDE HISTENTRY]
                             ;; There could be a property SIDE with value NIL if the user did a FORGET during the execution of the event
                             (NCONC HISTENTRY (LIST 'SIDE (LIST 1 UNDOFORM)))
                             (RETURN))
                            ((EQ (CAR Y)
                                 'NOSAVE)
                             (RETURN))
                            ((EQ (SETQ N (CAAR Y))
                                 -1)                              ; Already gone past #UNDOSAVES and user has confirmed.
                             (GO OUT))
                            (UNDOFORM                             ; can be called with UNDOFORM=NIL just to check on
                                                                  ; #undosaves
                                   (SETQ N (ADD1 N]
                         (COND
                            ([AND %#UNDOSAVES (IGREATERP N (COND
                                                              ((MINUSP %#UNDOSAVES)
                                                               (IMINUS %#UNDOSAVES))
                                                              (T %#UNDOSAVES]
                             (COND
                                ([OR (MINUSP %#UNDOSAVES)
                                     (AND DWIMFLG (NEQ (ASKUSER DWIMWAIT 'N (LIST %#UNDOSAVES
                                                                                  '"undosaves, continue saving")
                                                                                  )
                                                       'Y]
                                 (FRPLACA Y 'NOSAVE)
                                 (RETURN)))
                             (SETQ N -1)))
                      OUT (FRPLACA (SETQ Y (CAR Y))
                                   N)
                         (AND UNDOFORM (FRPLACD Y (CONS UNDOFORM (CDR Y]))
```

(UNDOLISPX

```
  [LAMBDA (LINE)
```

(* Note%: undoing in order is guaranteed to restore you to the original state.
Undoing out of order is defined as restoring any cells changed in the indicated operation to their original state before the operation was performed. For independent operations, undoing will have the correct effect.
However, for dependent operations, it may have an unforeseen effect.
For example, ATTACH (A X) ATTACH (B X) followed by UNDO A will remove both A and B since the cell changed by the first ATTACH was the first cell in X, and this will be restored to its former state.
In general, operations are always independent if they affect different lists or different sublists
(not TAILS) of the same list. However, because property list functions might be thought of as independent, PUT, REMPROP, and ADDPROP are treated specially. Thus put (FOO PROP1 VAL1) followed by PUTPROP
(FOO PROP2 VAL2) followed by UNDO PROP1 will remove just PROP1 even if both PUT'S resulted in new properties and hence additions to the end of the property list.)

```
    (PROG (UNDONEFLG DWIMCHANGES)
          (SETQ DWIMCHANGES (FMEMB '%: LINE))
          (SETQ LINE (LDIFF LINE DWIMCHANGES))
          (SETQ DWIMCHANGES (CDR DWIMCHANGES))
          [COND
             [LINE (MAPC (LISPXFIND LISPXHISTORY LINE 'ENTRIES T)
                         (FUNCTION (LAMBDA (X)
                                     (SETQ UNDONEFLG (OR (UNDOLISPX1 X NIL DWIMCHANGES)
                                                         UNDONEFLG]
             (T (SOME (CDAR LISPXHISTORY)
                      (FUNCTION (LAMBDA (X)
                                  (SETQ UNDONEFLG (OR (UNDOLISPX1 X T DWIMCHANGES)
                                                      UNDONEFLG]
          (RETURN (COND
                     ((NULL UNDONEFLG)
                      (PRIN1 (COND
                                (DWIMCHANGES '"not found.
                                             ")
                                (T '"nothing saved.
                                   ")))
                             T)
                      (QUOTE))
                     (T UNDONEFLG])
```

(UNDOLISPX1

```
  [LAMBDA (EVENT FLG DWIMCHANGES)
```

;; FLG is T when interpreting a simple UNDO command. In this case, does not UNDO commands already undone, nor other UNDO commands.

```
    (PROG (TEM Y X)
          (COND
```

```
            ([AND FLG (OR (EQ (CAAR EVENT)
                              'UNDO)
                          (EQ (CAAR EVENT)
                              'undo]
             (RETURN NIL)))
          (SETQ TEM (UNDOLISPX2 EVENT NIL DWIMCHANGES))
          [COND
             ((NULL TEM)
              (RETURN))
             ((EQ TEM 'already)
              (COND
                 (FLG                                          ; Searching for last thing to UNDO.
                      (RETURN NIL)))
              (SETQ X TEM))
             ((SETQ Y (FMEMB '*HISTORY* EVENT))
              (SETQ X (CAADR Y)))
             (T (SETQ X (CAR EVENT]
          (COND
             ([COND
                 ((EQ X 'already)
                  (PRIN1 X T))
                 ((NULL DWIMCHANGES)                           ; Messages for DWIMCHANGES are printed in UNDOLISPX3.
                  (SETQ X (UNDOPRINT X EVENT]
              ;; Initially defined as PRIN1.  Separate function so user can advise it to print the 'name' of the event.
              (PRIN1 '" undone.
                     " T)))
          (RETURN X])
```

(UNDOPRINT

```
  [LAMBDA (X EVENT)
    (PRIN2 (COND
              ((NLISTP X)
               X)
              ((LISTP (CAR X))
               (CAAR X))
              (T (CAR X)))
           T T])
```

(UNDOLISPX2

```
  [LAMBDA (X FORGETFLG DWIMCHANGES)
```

;; Searches X for SIDE information. If finds some and is already undone, sets FLG1 to 'ALREADY, otherwise sets FLG1 to T, undoes it, and marks ;; it undone.

;; If FORGETFLG is T, just erases the UNDO information entirely.

```
    (PROG (Y TEM VAL)
          [AND FORGETFLG [MAPC '(ENTERED EDITHIST EDIT)
                               (FUNCTION (LAMBDA (PROP)
                                           (AND (SETQ TEM (CDR (FMEMB PROP X)))
                                                (FRPLACA TEM NIL]
               (COND
                  ((SETQ TEM (CDR (FMEMB 'EDIT X)))
                   (FRPLACA TEM NIL]
          [SETQ VAL (COND
                       ([CDAR (SETQ Y (CDR (FMEMB 'SIDE X]       ; An attempted CLISP correction will leave a side property
                                                                 ; consisting of just (0) the CDAR checks for this.
                        (COND
                           (FORGETFLG (FRPLACA Y NIL))
                           ((NLISTP (SETQ Y (CAR Y)))
                            NIL)
                           ((NULL (CAR Y))
                            'already)
                           (DWIMCHANGES [MAPC DWIMCHANGES (FUNCTION (LAMBDA (DWIMCHANGE)
                                                                      (SETQ VAL (OR (UNDOLISPX3 X DWIMCHANGE)
                                                                                    VAL]
                                        VAL)
                           (T                                    ; (CAR Y) Is the count.
                              [MAPC (CDR Y)
                                    (FUNCTION (LAMBDA (X)
                                                [COND
                                                   ((NLISTP X)   ; a marker
                                                    NIL)
                                                   ((LISTP (CAR X))
                                                    (/RPLNODE (CAR X)
                                                              (CADR X)
                                                              (CDDR X)))
                                                   (T (APPLY (CAR X)
                                                             (CDR X]
                                                (LISPXWATCH UNDOSTATS]
                              (/ATTACH NIL Y)
                              T]
          [COND
             ((SETQ Y (CADR (FMEMB '*GROUP* X)))
              (MAPC (REVERSE Y)
                    (FUNCTION (LAMBDA (X)
```

```
                              (SETQ VAL (OR (UNDOLISPX2 X FORGETFLG DWIMCHANGES)
                                            VAL]
          (RETURN VAL])
```

(UNDOLISPX3

```
  [LAMBDA (EVENT DWIMCHANGE)
    (RESETVARS ((EDITQUIETFLG T)
                (MAXLEVEL 1500)
                (CLISPTRANFLG 'CLISP% ))
               (RETURN (PROG (L (COMS (LIST (LIST 'F DWIMCHANGE T)
                                            1
                                            '(BELOW ^)
                                            'UP))
                                      MARKER L1 L2 TEM)
                             (COND
                                ([NULL (SETQ TEM (LISTGET1 EVENT '*LISPXPRINT*]
                                 (RETURN NIL)))
                             (SETQ L (LIST TEM))
                       LP    (COND
                                ([NULL (AND L (NLSETQ (SETQ L (EDITL L COMS)))
                                              (SETQ MARKER (FASSOC CLISPTRANFLG (CAR L]
                                 ;; The FASSOC looks for the DWIM marker.  If none is found, this message is not associated with a DWIM
                                 ;; correction.
                                 (RETURN))
                                ((NULL (TAILP (CAR L)
                                              (CADADR MARKER)))
                                 ;; The form of MARKER is (CLISP (QUOTE PTR1 PTR2 PTR3)) where PTR1 marks the print list at the
                                 ;; beginning of this DWIM correction, PTR2 the sides at the beginning, and PTR3 the sides at the end.  The
                                 ;; TAILP checks to see that the place where this word was found is inside of the DWIM correction.  If not, it
                                 ;; goes on to look for another instance of this word by starting with the position after the DWIM marker.
                                 (SETQ L (CDR (FMEMB MARKER L)))
                                 (GO LP)))
                             [SETQ L (SETQ L1 (CADDDR (SETQ TEM (CADR MARKER]
                                                                  ; The beginning of the side info.
                             (SETQ L2 (CADDR TEM))
                       LP1   [COND
                                ((EQ L1 L2)
                                 (/RPLNODE L ''PATCHED L2)
                                 (SETQ L1 (CADADR MARKER))
                                 (GO LP2))
                                [(NLISTP (SETQ TEM (CAR L1]
                                ((LISTP (CAR TEM))
                                 (/RPLNODE (CAR TEM)
                                           (CADR TEM)
                                           (CDDR TEM)))
                                (T (APPLY (CAR TEM)
                                          (CDR TEM]
                             (LISPXWATCH UNDOSTATS)
                             (SETQ L1 (CDR L1))
                             (GO LP1)
                       LP2                                        ; Prints the message associated with the DWIM correction.
                             (COND
                                ((EQ (CADR L1)
                                     MARKER)
                                 (LISPXPRIN1 '" undone.
                                             " T)
                                 (RETURN T)))
                             (LISPXPUT '*LISPXPRINT* (LIST (CAR L1))
                                    T LISPXHIST)
                             (LISPXREPRINT (CAR L1))
                             (SETQ L1 (CDR L1))
                             (GO LP2])
```

(UNSET

```
  [LAMBDA (NAME)
    (PROG (X TEM)
          (RETURN (COND
                     ([OR (SETQ X (FMEMB 'VALUE (GETPROPLIST NAME)))
                          (AND DWIMFLG (SETQ TEM (MISSPELLED? NAME 70 SPELLINGS3))
                               (SETQ X (FMEMB 'VALUE (GETPROPLIST (SETQ NAME TEM]
                      ;; Note that UNSET always works for top level bindings in conjuncture with SAVESET: only top level bindings are saved on
                      ;; property lists.
                      (SAVESET NAME (CADR X)
                             T
                             'NOPRINT)
                      NAME)
                     (T (ERROR '"no value saved:" NAME])
```

(/LISPXPUT

```
  [LAMBDA (PROP L ADDFLG LST)
    (PROG (Y)
```

```
          (AND (NULL LST)
               (SETQ LST (CAAR LISPXHISTORY)))                ; Puts property at top level of entry.  Used mostly for calls with
                                                              ; PROP=ERROR.
          [COND
             [(SETQ Y (CDR (FMEMB PROP LST)))
              (/RPLACA Y (COND
                           (ADDFLG (/NCONC (CAR Y)
                                           L))
                           (T L]
             (T (/NCONC LST (LIST PROP L]
          (RETURN L])
```

(/PUT-1

```
  [LAMBDA (ATM PROP)                                          (* removes property and value at PROP)
    (PROG ((X (GETPROPLIST ATM))
           X0)
      LP   (COND
             [(EQ X PROP)
              (UNDOSAVE (LIST '/PUT+1 ATM X0 PROP))
              (COND
                 (X0 (FRPLACD X0 (CDDR PROP)))
                 (T (SETPROPLIST ATM (CDDR PROP]
             ([LISTP (SETQ X (CDR (SETQ X0 X]
              (GO LP])
```

(/PUT+1

```
  [LAMBDA (ATM TAIL PROP)
```

;; CAR and CADR of PROP represent the propety and its value. /PUT+1 resotres CAR and CADR of PROP either at (CDR TAIL) if TAIL is found
;; on the property list of ATM, else at the front of the property list.

```
    (PROG ((X (GETPROPLIST ATM)))
          (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM PROP)
                                   LISPXHIST))
          (COND
             ((NLISTP TAIL)
```

;; TAIL is NIL when the property that was removed was the first one on te property list, i.e. should be attached back at the front.

```
              (GO FRONT)))
      LP  (COND
             ((EQ X TAIL)
              (FRPLACD (CDR PROP)
                       (CDR X))
              (FRPLACD X PROP)
              (RETURN))
             ((LISTP (SETQ X (CDR X)))
              (GO LP)))
      FRONT
          (FRPLACD (CDR PROP)
                   (GETPROPLIST ATM))
          (SETPROPLIST ATM PROP])
```

(UNDONLSETQ

```
  [NLAMBDA (UNDOFORM UNDOFN)                                  (* wt%: 8-JUN-77 1 48)
    (PROG ((LISPXHIST LISPXHIST)
           UNDOSIDE0 UNDOSIDE UNDOTEM)
```

;; A version of NLSETQ that undoes all side effects if an error occurs. There are several situations to 'WORRY' about. First, LISPXHIST may be
;; NIL, but we still want UNDONLSETQ to operate. Second, LISPXHIST may not yet contain a side property. Third, LISPXHIST may contain a
;; side property. In the latter two cases we also have to worry about the number of undosaves exceeding (OR ALEADY HAVING EXCEEDED)
;; #UNDOSAVES. Finally, we want the entire event undoable if the UNDONLSETQ is aborted with a control-d.

```
          [COND
             ([LISTP (SETQ UNDOSIDE (LISTGET1 LISPXHIST 'SIDE]
              (SETQ UNDOSIDE0 (CDR UNDOSIDE))                 ; saves current lst of sides for undoing

              )
             (T (SETQ UNDOSIDE0 UNDOSIDE)                     ; may be NIL or NOSAVE
                (SETQ UNDOSIDE (LIST 0))
                (COND
                   (LISPXHIST (LISTPUT1 LISPXHIST 'SIDE UNDOSIDE))
                   (T (SETQ LISPXHIST (LIST 'SIDE UNDOSIDE]
          (RESETVARS (%#UNDOSAVES)                            ; so saving will continue regardless
                     (SETQ UNDOTEM (ERRORSET UNDOFORM NIL UNDOFN)))
```

;; Note that all side effects are stored onto the higher level LISPXHIST, if any, so that if a control-d is typed, any changes made under the
;; UNDONLSETQ will be undoable.

```
          (COND
             ((EQ UNDOSIDE0 'NOSAVE)
```

;; number of undosaves had already been exceeded before this call to undonlsetq, and user said not to continue saving.

```
              (LISTPUT1 LISPXHIST 'SIDE 'NOSAVE))
             (T (UNDOSAVE)                                    ; to check whether or not to continue saving

                ))
          (COND
```

```
          (UNDOTEM (RETURN UNDOTEM)))
        (UNDONLSETQ1 (CDR UNDOSIDE)
              (LISTP UNDOSIDE0))                    ; undoes the indicated segment.
        (RETURN])
```

(UNDONLSETQ1

```
[LAMBDA (LST TAIL)                                  (* wt%: 23-MAR-77 22 45)
```

;; undoes the side informaton from LST to TAIL and then splices it out by smashing LST appropriately.

```
(AND (NEQ LST TAIL)
     (PROG ((LST1 LST)
            LISPXHIST TEM)
       LP   [COND
              ((EQ LST1 TAIL)
               (FRPLACD LST TAIL)
               (FRPLACA LST ''undonlsetq)
```

;; note that the node TAIL must stay in the list because it might be pointed to as cdr of UNDOSIDE0 for some higher ;; UNDONLSETQ.

```
               (RETURN))
              [(NLISTP (SETQ TEM (CAR LST1]
              ((LISTP (CAR TEM))
               (FRPLACA (CAR TEM)
                        (CADR TEM))
               (FRPLACD (CAR TEM)
                        (CDDR TEM)))
              (T (APPLY (CAR TEM)
                        (CDR TEM]
          (LISPXWATCH UNDOSTATS)
          (SETQ LST1 (CDR LST1))
          (GO LP])
```

(RESETUNDO

```
[LAMBDA (X STOPFLG)                                 (* wt%: 8-JUN-77 1 52)
```

;; this function is a generalization of UNDONLSETQ for use under a RESETLST. When called with X = NIL, it sets up things for undoing, and ;; returns a value which when given back to RESETUNDO undoes the corresponding events. UNDONLSETQ could be written in terms of ;; RESETUNDO as (RESETLST (RESETSAVE (RESETUNDO) (AND (EQ RESETSTATE (QUOTE ERROR)) (RESETUNDO OLDVALUE)) form))

```
(PROG ((UNDOSIDE (CAR X))
       (UNDOSIDE0 (CDR X)))
```

;; note that this function does not reflect the recent change in undonlsetq wherein the undosaves performed under the undonlsetq ARE counted ;; towards the total number

```
      (RETURN (COND
                ((NULL X)                           ; just setup and return.
                 [COND
                    ([LISTP (SETQ UNDOSIDE (LISTGET1 LISPXHIST 'SIDE]
                                                    ; SIDE property may be NOSAVE.
                     (SETQ UNDOSIDE0 (CONS (CAR UNDOSIDE)
                                           (CDR UNDOSIDE))) ; Saves old value of side property.
                     (FRPLACA UNDOSIDE -1)          ; So that will continue saving regardless of number.

                     )
                    (T (SETQ UNDOSIDE (LIST -1))
                       (SETQ LISPXHIST (COND
                                          (LISPXHIST        ; LISTPUT1 is like PUT, except it works with lists.
                                                (LISTPUT1 LISPXHIST 'SIDE UNDOSIDE))
                                          (T (LIST 'SIDE UNDOSIDE]
```

;; Note that all side effects are stored onto the higher level LISPXHIST, if any, so that if a control-d is typed, any changes ;; made under the UNDONLSETQ will be undoable.

```
                 (CONS UNDOSIDE UNDOSIDE0))
                (STOPFLG
```

;; user wants to stop the scope of the resetundo, e.g. he dooes (RESETLST (RESETSAVE (SETQ FOO ;; (RESETUNDO)) (QUOTE (PROGN (RESETUNDO OLDVALUE)))) forms (RESETUNDO FOO T) more-forms) ;; and more-forms will not be affected by tthe RESETUNDO.

```
                 (FRPLACA UNDOSIDE (FLENGTH (CDR UNDOSIDE)))
                 (FRPLACA X (CDAR X))
```

;; CAR of (CAR X) is the number of undosaves for the corresponding segment. The FRPLACA replace (CAR X) ;; by CDR of the corresponing node. Since the firt node of each of these is what gets smashed when new ;; undosaves are added on, this operation protects thee segments from having subsequent undosaves stored in ;; front.

```
                 (FRPLACD X (CDDR X))
                 X)
                ((EQ (CAR UNDOSIDE)
                     -1)
                 (FRPLACA UNDOSIDE (FLENGTH (CDR UNDOSIDE)))
                 (UNDONLSETQ1 (CDR UNDOSIDE)
                        (CDR UNDOSIDE0)))
                (T
```

;; occurs when the scope was stopped by a call to resetundo with stopflg=T. In this case, UNDOSIDE is a tail of a side ;; proprty.

```
                   (UNDONLSETQ1 UNDOSIDE UNDOSIDE0])
```

(/**DEFINEQ**
```
  [NLAMBDA X                                                    (* wt%: "22-JUL-78 18:55")
    (DEFINE X T])
```

(/**DEFINE**
```
  [LAMBDA (X)
    (DEFINE X T])
```

(/**PRINTLEVEL**
```
  [LAMBDA (CARVAL CDRVAL)                                       (* bvm%: " 1-Jan-84 16:18")
    ([LAMBDA (RESULT)
       (UNDOSAVE (LIST (FUNCTION /PRINTLEVEL)
                       RESULT))
       RESULT]
     (PRINTLEVEL CARVAL CDRVAL])
```

)

(RPAQ? **%#UNDOSAVES**)

(RPAQ? **UNDOSIDE0**)

(RPAQ? **TESTMODEFLG**)

```
(ADDTOVAR LISPXFNS (SETQ . SAVESETQ)
                   (SET . SAVESET)
                   (SETQQ . SAVESETQQ)
                   (DEFINEQ . /DEFINEQ)
                   (DEFINE . /DEFINE)
                   (PRINTLEVEL . /PRINTLEVEL))

(ADDTOVAR /FNS /ADDPROP /ATTACH /CONTROL /DELETECONTROL /DREMOVE /DREVERSE /DSUBST /ECHOCONTROL /ECHOMODE /LCONC
                    /LISTPUT /LISTPUT1 /MAPCON /MAPCONC /MOVD /NCONC /NCONC1 /PUT /PUTASSOC /PUTD /PUTDQ
                    /PUTHASH /PUTPROP /RADIX /RAISE /REMPROP /RPLACA /RPLACD /RPLNODE /RPLNODE2 /SET /SETA
                    /SETATOMVAL /SETBRK /SETD /SETPROPLIST /SETREADTABLE /SETSEPR /SETSYNTAX /SETTERMTABLE
                    /SETTOPVAL /TCONC)

(DEFINEQ
```

(/**ADDPROP**
```
  [LAMBDA (ATM PROP NEW FLG)                                    (* wt%: "25-FEB-80 09:40")
    ;; If FLG is T, NEW is consed onto the front, otherwise NCONCED onto the end.
    ;; Value is new PROP value.
    [COND
       [(NULL ATM)
        (ERRORX (LIST 7 (LIST PROP NEW]
       ((NOT (LITATOM ATM))
        (ERRORX (LIST 14 ATM]
    (PROG ((X (GETPROPLIST ATM))
           X0 TEM)
      LOOP
          (COND
             ((NLISTP X)
              (COND
                 ((AND (NULL X)
                       X0)
                  ;; typical case.  property list ran out on an even parity position.  fall through and add property at beginning of property list.
                  [SETQ TEM (LIST PROP (SETQ NEW (LIST NEW]
                  (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                                           LISPXHIST))
                  (FRPLACD (CDR X0)
                           TEM)
                  (RETURN NEW)))
               ;; proprty list was initially NIL or a non-lit, or ele it ended in a non-list following an even parity position, e.g.  (A B . C)

               )
             ((NLISTP (CDR X))
               ;; property list runs out on an odd parity, or else ends in a non-list following an odd parity, e.g.  (A B C) or (A B C . D) fall through and
               ;; add at beginning

               )
             ((EQ (CAR X)
                  PROP)
              [/RPLACA (CDR X)
                       (SETQ NEW (COND
                                    (FLG (CONS NEW (CADR X)))
                                    (T (/NCONC1 (CADR X)
                                                NEW]
              (RETURN NEW))
             (T (SETQ X (CDDR (SETQ X0 X)))
```

```
            (GO LOOP)))                                          ; Add to beginning of property list.
          [SETQ TEM (CONS PROP (CONS (SETQ NEW (LIST NEW))
                                     (GETPROPLIST ATM]
          (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                                   LISPXHIST))
          (SETPROPLIST ATM TEM)
          (RETURN NEW])
```

(/ATTACH

```
[LAMBDA (X LST)                                                  (* wt%: 23-SEP-76 20 55)
  (COND
     [(LISTP LST)
      (/RPLNODE LST X (CONS (CAR LST)
                            (CDR LST]
     ((NULL LST)
      (CONS X))
     (T (ERRORX (LIST 4 LST])
```

(/CONTROL

```
[LAMBDA (FLG TTBL)
  (SETQ FLG (CONTROL FLG TTBL))
  (AND LISPXHIST (UNDOSAVE (LIST '/CONTROL FLG TTBL)
                           LISPXHIST))
  FLG])
```

(/DELETECONTROL

```
[LAMBDA (TYPE MESSAGE TTBL)
  (SETQ TTBL (GETTERMTABLE TTBL))
  (AND LISPXHIST MESSAGE (UNDOSAVE (LIST '/DELETECONTROL TYPE (DELETECONTROL TYPE NIL TTBL)
                                         TTBL)
                                   LISPXHIST))
  (DELETECONTROL TYPE MESSAGE TTBL])
```

(/DREMOVE

```
[LAMBDA (X Y)
  (COND
     ((NLISTP Y)
      NIL)
     [(EQ X (CAR Y))
      (COND
         ((CDR Y)
          (/RPLNODE Y (CADR Y)
                 (CDDR Y))
          (/DREMOVE X Y]
     (T (PROG (Z)
              (SETQ Z Y)
          LP  [COND
                 ((NLISTP (CDR Y))
                  (RETURN Z))
                 ((EQ X (CADR Y))
                  (/RPLACD Y (CDDR Y)))
                 (T (SETQ Y (CDR Y]
              (GO LP])
```

(/DREVERSE

```
[LAMBDA (X)
  (PROG (Y Z)
    R1  (COND
           ((NLISTP (SETQ Y X))
            (RETURN Z)))
        (SETQ X (CDR X))
        (SETQ Z (/RPLACD Y Z))
        (GO R1])
```

(/DSUBST

```
[LAMBDA (NEW OLD EXPR)                                           (* wt%: "28-AUG-78 21:55")
  (PROG (B)
        [COND
           ((EQ OLD (SETQ B EXPR))
            (RETURN (COPY NEW]
    LP  [COND
           ((NLISTP EXPR)
            (RETURN B))
           ([COND
               ((LITATOM OLD)
                ;; Most uses involve substitution for an atom, and the check enables avoiding an extra function call (to equal)
                (EQ OLD (CAR EXPR)))
               (T (EQUAL OLD (CAR EXPR]
            (/RPLACA EXPR (COPY NEW)))
           (T (/DSUBST NEW OLD (CAR EXPR]
```

```
          (COND
             ((AND OLD (EQ OLD (CDR EXPR)))
              (/RPLACD EXPR (COPY NEW))
              (RETURN B)))
          (SETQ EXPR (CDR EXPR))
          (GO LP])
```

(/ECHOCONTROL

```
  [LAMBDA (CHAR MODE TTBL)
    (SETQ TTBL (GETTERMTABLE TTBL))
    (AND LISPXHIST MODE (UNDOSAVE (LIST '/ECHOCONTROL CHAR (ECHOCONTROL CHAR NIL TTBL)
                                        TTBL)
                                  LISPXHIST))
    MODE
    (ECHOCONTROL CHAR MODE TTBL])
```

(/ECHOMODE

```
  [LAMBDA (FLG TTBL)
    (SETQ FLG (ECHOMODE FLG TTBL))
    (AND LISPXHIST (UNDOSAVE (LIST '/ECHOMODE FLG TTBL)
                             LISPXHIST))
    FLG])
```

(/LCONC

```
  [LAMBDA (PTR X)
    (PROG (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))
                      (/RPLNODE PTR X XX))
                     (T (/RPLACD (CDR PTR)
                                 X)
                        (/RPLACD PTR XX]
      ERROR
          (ERROR '"bad argument - LCONC" XX])
```

(/LISTPUT

```
  [LAMBDA (LST PROP VAL)                                  ; Like PUT but works on lists.
    (PROG ([X (OR (LISTP LST)
                  (ERRORX (LIST 4 LST]
           X0)
      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
              (/RPLACA (CDR X)
                       VAL)
              (RETURN VAL))
             ([LISTP (SETQ X (CDDR (SETQ X0 X]
              (GO 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.
              (/RPLACD (CDR X0)
                       (LIST PROP VAL))
              (RETURN VAL)))
      ADDFRONT
          [/RPLNODE LST PROP (CONS VAL (CONS (CAR LST)
                                             (CDR LST]
          (RETURN VAL])
```

(/LISTPUT1

```
  [LAMBDA (LST PROP VAL)
    ;; like listput but does one cdr at a time.  inverse of listget1.  used by undonlsetq
    (PROG ((X LST))
      LP  (COND
```

```
            [(NLISTP X)                                    ; Note no checks for LST ending in dotted pairs.
             (RETURN (/NCONC LST (LIST PROP VAL]
            ((EQ (CAR X)
                 PROP)
             [COND
                ((CDR X)
                 (/RPLACA (CDR X)
                          VAL))
                (T (/RPLACD X (LIST VAL]
             (RETURN LST)))
          (SETQ X (CDR X))
          (GO LP])
```

(/MAPCON

```
[LAMBDA (MAPX MAPFN1 MAPFN2)
  (PROG (CL:MAPL MAPE MAPY)
    LP  [COND
           ((NLISTP MAPX)
            (RETURN CL:MAPL))
           ((SETQ MAPY (APPLY* MAPFN1 MAPX))
            [COND
               (MAPE (/RPLACD 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))
           ((SETQ MAPY (APPLY* MAPFN1 (CAR MAPX)))
            [COND
               (MAPE (/RPLACD 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])
```

(/MOVD

```
[LAMBDA (FROM TO FLG)                                      (* rmk%: " 9-JUN-82 21:49")
  (PROG [(NEWFLG (NULL (GETD TO]
        (COND
           ((NULL (GETD FROM))
            (LISPXPRIN1 "****note: " T T)
            (LISPXPRIN2 FROM T T)
            (LISPXPRIN1 " has no definition
                        " T T)))
        [/PUTD TO (COND
                     (FLG (COPY (VIRGINFN FROM)))
                     (T (GETD FROM]
        (AND (EXPRP TO)
             (MARKASCHANGED TO 'FNS NEWFLG))
        (AND ADDSPELLFLG (ADDSPELL TO))
        (RETURN TO])
```

(/NCONC

```
[LAMBDA L
  (PROG (VAL X TEM (N 0))
    LP  (COND
           ((EQ N L)
            (RETURN VAL)))
        [SETQ TEM (ARG L (SETQ N (ADD1 N]
        [COND
           ((LISTP X)
            (/RPLACD (SETQ X (LAST X))
                     TEM))
           (T (SETQ VAL (SETQ X TEM]
        (GO LP])
```

(/**NCONC1**

```
[LAMBDA (LST X)
  (/NCONC LST (FRPLACD (CONS X LST])
```

(/**PUT**

```
[LAMBDA (ATM PROP VAL)
```

;; Now called /PUTPROP but included for backwards compatibility.

```
[COND
   [(NULL ATM)
    (ERRORX (LIST 7 (LIST ATM PROP]
   ((NOT (LITATOM ATM))
    (ERRORX (LIST 14 ATM]
(PROG ((X (GETPROPLIST ATM))
       X0 TEM)
  LOOP
      (COND
         ((NLISTP X)
          (COND
             ((AND (NULL X)
                   X0)                                    ; typical case.  property list ran out on an even parity position.
                                                          ; e.g.  (A B C D)
              (SETQ TEM (LIST PROP VAL))
              (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                                       LISPXHIST))
              (FRPLACD (CDR X0)
                       TEM)
              (RETURN VAL)))
```

;; property list was initially NIL or a non-list, or else it ended in a non-list following an even parity position, e.g. (A B . C) fall through
;; and add new property at beginning

```
          )
          ((NLISTP (CDR X))
```

;; property list runs out on an odd parity, or ends in an odd list following an odd parity, e.g. (A B C) or (A B C . D) fall through and add
;; at beginning.

```
          )
          ((EQ (CAR X)
               PROP)
           (/RPLACA (CDR X)
                    VAL)
           (RETURN VAL))
          (T (SETQ X (CDDR (SETQ X0 X)))
             (GO LOOP)))
      [SETQ TEM (CONS PROP (CONS VAL (GETPROPLIST ATM]
      (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                               LISPXHIST))
      (SETPROPLIST ATM TEM)
      (RETURN VAL])
```

(/**PUTASSOC**

```
[LAMBDA (KEY VAL ALST)                                    (* lmm%: 5 SEP 75 119)
  (PROG [(X (OR (LISTP ALST)
                (ERRORX (LIST 4 ALST]
    LP  (COND
           ((EQ (CAR (OR (LISTP (CAR X))
                         (GO NEXT)))
                KEY)
            (/RPLACD (CAR X)
                     VAL)
            (RETURN VAL)))
    NEXT
      [SETQ X (OR (LISTP (CDR X))
                  (PROGN (/RPLACD X (LIST (CONS KEY VAL)))
                         (RETURN VAL]
      (GO LP])
```

(/**PUTD**

```
[LAMBDA (FN DEF FLG)                                      (* lmm "11-FEB-82 14:46")
  (PROG ((TEM (GETD FN)))
        (PUTD FN DEF FLG)
```

;; The reason for doing the PUTD first is to avoid storing any undo information if the PUTD should cause an error --- e.g. if FN is non-atomic. If
;; undo information were stored, then undoing the event would cause an error --- thereby preventing the rest of the undo information (if any) from
;; being undone.

```
        (AND LISPXHIST (UNDOSAVE (LIST '/PUTD FN TEM)
                                 LISPXHIST))
        (RETURN DEF])
```

(/**PUTDQ**

```
  [NLAMBDA (X Y)
    (/PUTD X Y)
    X])
```

(/PUTHASH

```
  [LAMBDA (ITEM VAL ARRAY)
    (AND LISPXHIST (UNDOSAVE (LIST '/PUTHASH ITEM (GETHASH ITEM ARRAY)
                                   ARRAY)
                             LISPXHIST))
    (PUTHASH ITEM VAL ARRAY])
```

(/PUTPROP

```
  [LAMBDA (ATM PROP VAL)
    [COND
       [(NULL ATM)
        (ERRORX (LIST 7 (LIST ATM PROP]
       ((NOT (LITATOM ATM))
        (ERRORX (LIST 14 ATM]
    (PROG ((X (GETPROPLIST ATM))
           X0 TEM)
      LOOP
          (COND
             ((NLISTP X)
              (COND
                 ((AND (NULL X)
                       X0)                                  ; typical case.  property list ran out on an even parity position.
                                                            ; e.g.  (A B C D)
                  (SETQ TEM (LIST PROP VAL))
                  (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                                           LISPXHIST))
                  (FRPLACD (CDR X0)
                           TEM)
                  (RETURN VAL)))
                 ;; propety list was initially NIL or a non-list, or else it ended in a non-list following an even parity position, e.g.  (A B . C) fall through
                 ;; and add new property at beginning

                 )
             ((NLISTP (CDR X))
                 ;; property list runs out on an odd parity, or ends in an odd list following an odd parity, e.g.  (A B C) or (A B C . D) fall through and add
                 ;; at beginning.

                 )
             ((EQ (CAR X)
                  PROP)
              (/RPLACA (CDR X)
                       VAL)
              (RETURN VAL))
             (T (SETQ X (CDDR (SETQ X0 X)))
                (GO LOOP)))
          [SETQ TEM (CONS PROP (CONS VAL (GETPROPLIST ATM]
          (AND LISPXHIST (UNDOSAVE (LIST '/PUT-1 ATM TEM)
                                   LISPXHIST))
          (SETPROPLIST ATM TEM)
          (RETURN VAL])
```

(/RADIX

```
  [LAMBDA (N)                                               (* wt%: "16-MAY-79 19:15")
    (COND
       (N (SETQ N (RADIX N))
          (AND LISPXHIST (UNDOSAVE (LIST '/RADIX N)
                                   LISPXHIST))
          N)
       (T (RADIX])
```

(/RAISE

```
  [LAMBDA (FLG TTBL)                                        (* wt%: "16-MAY-79 19:13")
    (SETQ FLG (RAISE FLG TTBL))
    (AND LISPXHIST (UNDOSAVE (LIST '/RAISE FLG TTBL)
                             LISPXHIST))
    FLG])
```

(/REMPROP

```
  [LAMBDA (ATM PROP)
    [COND
       ((NULL (LITATOM ATM))
        (ERRORX (LIST 14 ATM]
    (PROG ((X (GETPROPLIST ATM))
           X0 VAL)
      LP  [COND
             ((OR (NLISTP X)
                  (NLISTP (CDR X)))
```

```
             (RETURN VAL))
            ((EQ (CAR X)
                 PROP)
             (SETQ VAL PROP)
             (AND LISPXHIST (UNDOSAVE (LIST '/PUT+1 ATM (CDR X0)
                                            X)
                                      LISPXHIST))
             [COND
                (X0 (FRPLACD (CDR X0)
                             (CDDR X)))
                (T (SETPROPLIST ATM (CDDR X]
             (SETQ X (CDDR X)))
            (T (SETQ X (CDDR (SETQ X0 X]
         (GO LP])
```

(/**RPLACA**

```
[LAMBDA (LST Y)                                          (* wt%: 20-OCT-76 5 10)
  (COND
     ((LISTP LST)
      (AND LISPXHIST (UNDOSAVE (LIST '/RPLACA LST (CAR LST))
                               LISPXHIST))
      (RPLACA LST Y))
     [(NULL LST)
      (AND Y (ERRORX (LIST 7 Y]
     (T (AND (LITATOM LST)
             (PRIN1 '"Use SETTOPVAL to 'set' a top level value
                    " T))
        (ERRORX (LIST 4 LST])
```

(/**RPLACD**

```
[LAMBDA (LST Y)                                          (* wt%: 20-OCT-76 5 11)
  (COND
     ((LISTP LST)
      (AND LISPXHIST (UNDOSAVE (LIST '/RPLACD LST (CDR LST))
                               LISPXHIST))
      (RPLACD LST Y))
     [(NULL LST)
      (AND Y (ERRORX (LIST 7 Y]
     (T (AND (LITATOM LST)
             (PRIN1 '"Use SETPROPLIST to 'set' a property list
                    " T))
        (ERRORX (LIST 4 LST])
```

(/**RPLNODE**

```
[LAMBDA (X A D)
```

(* Coombines action of /RPLACA and /RPLACD. In this case, it takes only 3 cells to save the undo informaion whereasa /RPLACA and /RPLACD take eight. However, even where only /RPLACA or /RPLACD is being performed, an equivalent /RPLNODE is still cheaper, 3 cells to four.)

```
  (COND
     ((LISTP X)
      (AND LISPXHIST (UNDOSAVE (CONS X (CONS (CAR X)
                                             (CDR X)))
                               LISPXHIST))
      (FRPLACA X A)
      (FRPLACD X D))
     (T (ERRORX (LIST 4 X])
```

(/**RPLNODE2**

```
[LAMBDA (X Y)                                            (* rmk%: " 4-MAR-82 22:07")
  (COND
     ((AND Y (NLISTP Y))
      (ERRORX (LIST 4 Y)))
     (T (/RPLNODE X (CAR Y)
                  (CDR Y])
```

(/**SET**

```
[LAMBDA (NAME VALUE)
  (SAVESET NAME VALUE NIL 'NOPROPSAVE])
```

(/**SETA**

```
[LAMBDA (A N V)                                          (* lmm%: 29-NOV-75 21%:21)
  [AND LISPXHIST (UNDOSAVE (LIST '/SETA A N (ELT A N]
  (SETA A N V])
```

(/**SETATOMVAL**

```
[LAMBDA (ATM VAL)                                        (* lmm "12-FEB-82 21:54")
  (COND
     [(NULL ATM)
```

```
        (AND VAL (ERRORX (LIST 6 VAL]
       [(EQ ATM T)
        (OR (EQ VAL T)
            (ERRORX (LIST 6 VAL]
       ((LITATOM ATM)
        [AND LISPXHIST (UNDOSAVE (LIST '/SETATOMVAL ATM (GETATOMVAL ATM]
        (SETATOMVAL ATM VAL))
       (T (ERRORX (LIST 14 ATM])
```

(/SETBRK

```
[LAMBDA (LST FLG RDTBL)
  (AND LISPXHIST (UNDOSAVE (LIST '/SETBRK (GETBRK RDTBL)
                                  NIL RDTBL)
                           LISPXHIST))
  (SETBRK LST FLG RDTBL])
```

(/SETD

```
[LAMBDA (A N V)                                          (* lmm%: 29-NOV-75 20%:43)
  [AND LISPXHIST (UNDOSAVE (LIST '/SETD A N (ELTD A N]
  (SETD A N V])
```

(/SETPROPLIST

```
[LAMBDA (ATM LST)                                        (* lmm "15-Apr-84 13:03")
  [AND LISPXHIST (UNDOSAVE (LIST '/SETPROPLIST ATM (GETPROPLIST ATM]
  (SETPROPLIST ATM LST])
```

(/SETREADTABLE

```
[LAMBDA (RDTBL)                                          (* wt%: " 7-FEB-79 22:59")
  (SETQ RDTBL (SETREADTABLE RDTBL))
  (AND LISPXHIST (UNDOSAVE (LIST '/SETREADTABLE RDTBL)))
  RDTBL])
```

(/SETSEPR

```
[LAMBDA (LST FLG RDTBL)
  (AND LISPXHIST (UNDOSAVE (LIST '/SETSEPR (GETSEPR RDTBL)
                                  NIL RDTBL)
                           LISPXHIST))
  (SETSEPR LST FLG RDTBL])
```

(/SETSYNTAX

```
[LAMBDA (CH CLASS TABLE)
  (COND
     (LISPXHIST (PROG (OLDCLASS OLDCH)
                      [SELECTQ CLASS
                           ((CHARDELETE DELETECHAR LINEDELETE DELETELINE RETYPE CTRLV CNTRLV EOL NONE)
                                    ;; Reason for this is currently setsyntax doesnt return enough information to enable restoring, e.g. if
                                    ;; you do (SETSYNTAX 1 'LINEDELETE), value is 17, but you dont know what 1 WAS.  however,
                                    ;; cant do a GETSYNTAX because that defaults to readtable when table is NIL.  When setsyntax is
                                    ;; fixed, this shuld be changed, since it will not work correctly if any terminal classes are added or
                                    ;; changed.
                                (SETQ TABLE (GETTERMTABLE TABLE)))
                           (AND (NULL TABLE)
                                (SETQ TABLE (GETREADTABLE TABLE]
                      (SETQ OLDCLASS (GETSYNTAX CH TABLE))
                      (UNDOSAVE (LIST '/SETSYNTAX CH OLDCLASS TABLE)
                                LISPXHIST)
                      (COND
                         ((NUMBERP (SETQ OLDCH (SETSYNTAX CH CLASS TABLE)))
                          ;; Says that CLASS specified one of the unique classes, and oldch was the character that prviously had this
                          ;; CLASS.
                          (UNDOSAVE (LIST '/SETSYNTAX OLDCH (GETSYNTAX CH TABLE)
                                          TABLE)
                                    LISPXHIST)
                          ;; Restores the character that previously was this unique class.  e.g.  if you say (SETSYNTAX 1 17
                          ;; (GETTERMTABLE)), this will restore 17 to LINEDELETE.  The GETSYNTAX is necessary because in this
                          ;; example nowhere did the user mention LINEDELETE

                          ))
                      (RETURN OLDCH)))
     (T (SETSYNTAX CH CLASS TABLE])
```

(/SETTERMTABLE

```
[LAMBDA (TTBL)                                           (* wt%: " 7-FEB-79 22:59")
  (SETQ TTBL (SETTERMTABLE TTBL))
  (AND LISPXHIST (UNDOSAVE (LIST '/SETTERMTABLE TTBL)))
  TTBL])
```

```
(/SETTOPVAL
  [LAMBDA (ATM VAL)                                          (* lmm "12-FEB-82 21:54")
    (COND
       [(NULL ATM)
        (AND VAL (ERRORX (LIST 6 VAL]
       [(EQ ATM T)
        (OR (EQ VAL T)
            (ERRORX (LIST 6 VAL]
       ((LITATOM ATM)
        [AND LISPXHIST (UNDOSAVE (LIST '/SETTOPVAL ATM (GETTOPVAL ATM]
        (SETTOPVAL ATM VAL))
       (T (ERRORX (LIST 14 ATM])


(/TCONC
  [LAMBDA (PTR X)
    (PROG (XX)
          (RETURN (COND
                     ((NULL PTR)
                      (CONS (SETQ XX (CONS X NIL))
                            XX))
                     ((NLISTP PTR)
                      (ERROR '"bad argument - TCONC" PTR))
                     ((NULL (CDR PTR))
                      (/RPLNODE PTR (SETQ XX (CONS X NIL))
                             XX))
                     (T ;; The (FRPLACD (CONS)) is just a kluge to get the cons on the same page as (CDR PTR).  can be taken out when we
                        ;; eliminate that part of cons algorithm
                        (/RPLACD PTR (CDR (/RPLACD (CDR PTR)
                                                   (FRPLACD (CONS X (CDR PTR])

)

[SETQ LISPXFNS (UNION LISPXFNS (MAPCAR /FNS (FUNCTION (LAMBDA (X Y)
                                                        (CONS (PACK (CDR (DUNPACK X CHCONLST)))
                                                              X]

(MOVD? 'RPLNODE 'FRPLNODE)

(MOVD? 'RPLNODE2 'FRPLNODE2)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: NIL UNSET RPLNODE RPLNODE2 /LISPXPUT /PUT-1 /PUT+1 (LINKFNS . T)
       UNDONLSETQ UNDONLSETQ1 (GLOBALVARS UNDOSTATS CLEARSTKLST DWIMFLG SPELLINGS3 LISPXHISTORY %#UNDOSAVES)
       RESETUNDO UNDOPRINT)

(BLOCK%: NIL RPAQ RPAQQ (LOCALVARS . T))

(BLOCK%: SAVESET SAVESET (LOCALVARS . T)
       (GLOBALVARS CLEARSTKLST))

(BLOCK%: NIL UNDOSET (GLOBALVARS SPAGHETTIFLG))

(BLOCK%: NIL NEW/FN (GLOBALVARS TESTMODEFLG LISPXFNS CHCONLST /FNS))

(BLOCK%: UNDOLISPXBLOCK UNDOSAVE UNDOLISPX UNDOLISPX1 UNDOLISPX2 UNDOLISPX3 (ENTRIES UNDOSAVE UNDOLISPX
                                                                                    UNDOLISPX1 UNDOLISPX2)
       (BLKLIBRARY LISPXWATCH)
       (GLOBALVARS UNDOSAVES UNDOSTATS %#UNDOSAVES DWIMFLG DWIMWAIT LISPXHISTORY CLISPTRANFLG EDITQUIETFLG
              MAXLEVEL)
       (LOCALFREEVARS UNDONEFLG))

(BLOCK%: NIL /ADDPROP /ATTACH /CONTROL /DELETECONTROL /DREMOVE /DREVERSE /DSUBST /ECHOCONTROL /ECHOMODE /LCONC
       /LISTPUT /LISTPUT1 /MAPCON /MAPCONC /MOVD /NCONC /NCONC1 /PRINTLEVEL /PUT /PUTASSOC /PUTD /PUTDQ /PUTHASH
       /PUTPROP /REMPROP /RPLACA /RPLACD /RPLNODE /RPLNODE2 /SET /SETA /SETBRK /SETD /SETPROPLIST /SETSEPR
       /SETSYNTAX /SETATOMVAL /SETTOPVAL /TCONC (GLOBALVARS UNDOSTATS)
       (LINKFNS . T))
)

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

(ADDTOVAR NLAMA /DEFINEQ SAVESETQ)

(ADDTOVAR NLAML /PUTDQ UNDONLSETQ RPAQ? RPAQ RPAQQ SAVESETQQ)

(ADDTOVAR LAMA /NCONC)
)

(PUTPROPS UNDO COPYRIGHT ("Venue & Xerox Corporation" 1984 1986 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX