File created: 3-Nov-87 13:42:29 **{QV}<LFG>PARSER>LYRIC>DECL.;2**

changes to: (FNS *DECLMAC)

previous date: 23-Feb-87 13:22:27 {QV}<LFG>PARSER>LYRIC>DECL.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ DECLCOMS
       [(* DECLTYPE machinery, declaration translator, and declaration enforcer)
        (LOCALVARS . T)
        (GLOBALVARS FILEPKGFLG CLISPCHANGE CLISPARRAY DWIMESSGAG NOSPELLFLG MSDATABASELST DECLTYPESARRAY
               COMMENTFLG CLISPCHARS DECLATOMS LCASEFLG DECLMESSAGES CLISPRETRANFLG)
        (E (RESETSAVE CLISPIFYPRETTYFLG NIL)
           (RESETSAVE PRETTYPRINTMACROS (APPEND '((DECL . QUOTE)
                                                  (DPROGN . QUOTE)
                                                  (DLAMBDA . QUOTE)
                                                  (DPROG . QUOTE))
                                                PRETTYPRINTMACROS)))
        (COMS (* Interface to file package)
              (FNS DECLTYPE DECLTYPES DUMPDECLTYPES GETDECLDEF)
              (FILEPKGCOMS DECLTYPES IGNOREDECL)
              (PROP ARGNAMES DECLTYPE))
        (* User functions)
        (FNS COVERS GETDECLTYPEPROP SETDECLTYPEPROP SUBTYPES SUPERTYPES)
        (MACROS SELCOVERSQ SELTYPEQ)
        (ALISTS (PRETTYEQUIVLST SELCOVERSQ SELTYPEQ)
               (DWIMEQUIVLST SELCOVERSQ SELTYPEQ))
        [P (SETSYNONYM '(THE TYPE)
                  '(AS A TYPE]
        (* Internal machinery)
        (DECLARE%: DONTCOPY (RECORDS TYPEBLOCK TYPEDEF)
               (ALISTS (PRETTYPRINTYPEMACROS TYPEBLOCK)))
        (INITRECORDS TYPEBLOCK)
        (P (DEFPRINT 'TYPEBLOCK 'TBDEFPRINT))
        (FNS CHECKTYPEXP COLLECTTYPES COVERSCTYPE COVERSTB COVERSTE CREATEFNPROP CREATEFNVAL DECLERROR DELETETB
             FINDDECLTYPE FINDPROP FINDTYPEXP GETCTYPE GETDECLTYPE GETDECLTYPE.NOERROR GETTBPROP INHERITPROP
             INITDECLTYPES LCCTYPE LCC2 MAKECTYPE MAKEDECLTYPE MAKEBINDFN MAKESETFN MAPTYPEUSERS NOTICETB
             PPDTYPE RECDTYPE DECLCHANGERECORD RECDEFTYPE REPROPTB SETTBPROP TBDEFPRINT TETYPE TYPEMSANAL
             TYPEMSANAL1 UNCOMPLETE UNSAVETYPE USERDECLTYPE USESTYPE)
        (BLOCKS (LCCTYPE LCCTYPE LCC2)
               (TYPEMSANAL TYPEMSANAL TYPEMSANAL1))
        (* Test fn creation block)
        (FNS MAKETESTFN MAKETESTFNBLOCK COMBINE.TESTS FUNIFY MKNTHCAR MKNTHCDR OF.TESTFN TUPLE.TESTFN
             WHOSE.TESTFN)
        (BLOCKS (MAKETESTFNBLOCK MAKETESTFNBLOCK COMBINE.TESTS FUNIFY MKNTHCAR MKNTHCDR OF.TESTFN TUPLE.TESTFN
                       WHOSE.TESTFN))
        (* Machinery to compile recursive testfns)
        (FILES (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
               LABEL)
        (* Idioms. Expressed as macros for now)
        (DECLARE%: DONTCOPY EVAL@COMPILE (VARS DefaultBindFn DefaultSetFn)
               (ADDVARS (NLAMA MAKEDECLTYPEQ))
               (MACROS ANYC DECLVARERROR DTYPENAME foreachTB GETCGETD KWOTEBOX LAMBIND LAMVAL MAKEDECLTYPEQ
                      NONEC TESTFORM)
               (FNS TESTFORM)
               (ADDVARS (DONTCOMPILEFNS TESTFORM))
               (TEMPLATES foreachTB MAKEDECLTYPEQ))
        (* Runtime utility functions)
        (FNS EVERYCHAR LARGEP DECLRECURSING SMASHCAR)
        (DECLARE%: EVAL@COMPILE (MACROS LARGEP))
        (DECLARE%: DONTCOPY EVAL@COMPILE (MACROS SMASHCAR))
        (* translator of dprogs and dlambdas)
        (FNS ASSERT ASSERTFAULT ASSERTMAC \*DECL *DECLMAC \CHKINIT CHKINITMAC DECLCONSTANTP DD DECLCLISPTRAN
             DECLMSG DECLDWIMERROR DECLDWIMTESTFN DECLSET DECLSETQ DECLSETQQ DECLTRAN DECLVAR DLAMARGLIST
             DTYPE?TRAN EDITNEWSATLIST FORMUSESTB IGNOREDECL MAKETESTFORM PPDECL PPVARLIST SETQMAC THETRAN
             VALUEERROR \VARASRT VARASRT1 VARSETFN)
        (BLOCKS (DECLTRAN DECLTRAN DECLVAR)
               (PPDECL PPDECL PPVARLIST)
               (\VARASRT \VARASRT VARASRT1))
        (* Declaration database fns)
        (FNS DECLOF DECLOF1 TBOF TYPEBLOCKOF VARDECL)
        (BLOCKS (DECLOFBLK DECLOF DECLOF1 TBOF TYPEBLOCKOF VARDECL (ENTRIES DECLOF TYPEBLOCKOF)))
        (* Enabling and disabling fns)
        (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS FNEQUIVS)
               (MACROS MOVEPROP PUTIFPROP))
        (FNS STARTDECLS DODECLS)
        (FILES (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
               LAMBDATRAN)
        (DECLARE%: EVAL@COMPILE (FILES (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
                                       SIMPLIFY))
        [DECLARE%: EVAL@COMPILE DONTCOPY (* the old NOBOX code.)
               (FNS IBOX FBOX NBOX)
               (P (MOVD? 'LIST 'LBOX)
                  (MOVD? 'CONS 'CBOX))
               (RECORDS FBOX IBOX)
               (MACROS IBOX FBOX NBOX)
               (MACROS CBOX LBOX)
               (I.S.OPRS scratchcollect)
               (ADDVARS (SYSLOCALVARS $$SCCONS $$SCPTR)
                      (INVISIBLEVARS $$SCCONS $$SCPTR))
               (* Definition of WITH. From <SHEIL>WITH.)
               (MACROS WITH)
               (TEMPLATES WITH)
               (P (REMPROP 'WITH 'CLISPWORD)
                  (ADDTOVAR DWIMEQUIVLST (WITH . PROG))
                  (ADDTOVAR PRETTYEQUIVLST (WITH . PROG))]
        [DECLARE%: DOCOPY (DECLARE%: EVAL@LOADWHEN (NEQ (SYSTEMTYPE)
                                                        'D)
                                 (P (OR (GETPROP 'LOADTIMECONSTANT 'FILEDATES)
                                        (PROG ((X (FINDFILE (PACKFILENAME 'NAME 'LOADTIMECONSTANT 'EXTENSION
                                                                      COMPILE.EXT)
                                                            T LISPUSERSDIRECTORIES)))
                                              (COND (X (LOAD X 'SYSLOAD))
                                                    ((NOT (GETPROP 'LOADTIMECONSTANT 'MACRO))
                                                     (PUTPROP 'LOADTIMECONSTANT 'MACRO '((FORM)
                                                                                          (CONSTANT FORM]
        (ADDVARS (OPENFNS \DECLPROGN \CHKVAL \CHKINIT ASSERT \*DECL \VARASRT))
        (PROP CLISPWORD DPROG DPROGN THE the)
        (PROP INFO DLAMBDA DPROG DPROGN)
        (VARS (SATISFIESLIST)
              (CSATISFIESLIST)
              (NEWSATLIST T))
        (INITVARS (DECLMESSAGES)
               (COMPILEIGNOREDECL))
        [ADDVARS (DECLATOMS DLAMBDA DPROG DPROGN)
               (LAMBDASPLST DLAMBDA)
               (SYSLOCALVARS VALUE)
               [DESCRIBELST ("types:    " (GETRELATION FN '(USE TYPE]
               (BAKTRACELST (\DECLPROGN (DPROGN APPLY *PROG*LAM \*DECL *ENV*)
                                    (NIL APPLY *PROG*LAM \*DECL))
                      (PROG (DPROG \DECLPROGN APPLY *PROG*LAM \*DECL]
        (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS SLISTENTRY VARDECL))
        (ALISTS (LAMBDATRANFNS DLAMBDA))
        [DECLARE%: DONTEVAL@LOAD (E (* Declare is so PRETTYPRINTMACROS don't get set up during LOADFROM, when
                                       PPDECL is not being defined. Don't use ALIST for print macros cause
                                       entries are removed while DECL is being dumped))
               (ADDVARS (PRETTYPRINTMACROS (DPROGN . PPDECL)
                               (DECL . PPDECL)
                               (DLAMBDA . PPDECL)
                               (DPROG . PPDECL]
        (PROP INFO ASSERT)
        (MACROS ASSERT .CBIND. \CHKINIT \CHKVAL \*DECL DECL DECLMSGMAC REALSETQ)
        (* MACROS REALSET)
        [P (AND (GETD 'STARTDECLS)
                (STARTDECLS))
           (PROG [(COM (CDR (ASSOC 'DW EDITMACROS]
                 (AND COM (RPLACD COM (CONS (APPEND '(RESETVAR NEWSATLIST (EDITNEWSATLIST))
                                                    (CDR COM]
        (* Builtin DECLOF properties)
        (PROP DECLOF APPEND CONS EQ LIST LISTP NCONC)
        [DECLARE%: EVAL@COMPILE DONTCOPY (P (RESETSAVE DWIMIFYCOMPFLG NIL)
                                            (AND (GETD 'DODECLS)
                                                 (RESETSAVE (DODECLS)
                                                        '(DODECLS T]
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA DECLSETQ DECLMSG DD \CHKINIT \*DECL ASSERT DECLTYPES DECLTYPE)
                      (NLAML DECLSETQQ TYPEMSANAL)
                      (LAMA DECLDWIMERROR])
```

(* * DECLTYPE machinery, declaration translator, and declaration enforcer)

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS FILEPKGFLG CLISPCHANGE CLISPARRAY DWIMESSGAG NOSPELLFLG MSDATABASELST DECLTYPESARRAY COMMENTFLG
       CLISPCHARS DECLATOMS LCASEFLG DECLMESSAGES CLISPRETRANFLG)
)
```

(* * Interface to file package)

```
(DEFINEQ

(DECLTYPE
  [NLAMBDA X                                                             (* bas%: " 7-NOV-79 16:22")
    (USERDECLTYPE (CAR X)
           (CADR X)
           (CDDR X])

(DECLTYPES
  [NLAMBDA DTS                                                           (* bas%: " 7-NOV-79 16:24")
                                                                         (* Defines a list of decltypes)
    (for D in DTS collect (USERDECLTYPE (CAR D)
                                 (CADR D)
                                 (CDDR D])

(DUMPDECLTYPES
  [LAMBDA (TL)                                                           (* rmk%: " 7-SEP-81 04:50")
    (WITH [[TWOFLG (OR (NLISTP TL)
                       (LISTP (CDR TL]
           (FFLG (NEQ T (OUTPUT]

          (* Don't do the plural and extra parens if only one, and don't do the EVAL@COMPILE stuff if going to T=SHOWDEF)

          (if FFLG
              then (printout NIL T "(DECLARE: EVAL@COMPILE" T T))
          (printout NIL (if TWOFLG
                            then "(DECLTYPES"
                          else "(DECLTYPE "))
          (for D in TL do (if TWOFLG
                              then (printout NIL 11 "("))
                          [if (LISTP D)
                              then (printout NIL .P2 (CAR D)
                                          %, .P2 (CAR D)
                                          %, .P2 (CADR D)
                                          %, .PPV (GETDECLTYPEPROP (CAR D)
                                                         (CADR D)))
                            else [SETQ D (CDR (GETDECLDEF D NIL 'NOCOPY]
                                 (printout NIL .P2 (CAR D)
                                        %,)
                                 (for TAIL (POS _ (POSITION)) on (CDDR D) by (CDDR TAIL)
                                    first (PRINTDEF (CADR D)
                                                 POS)
                                    do (printout NIL .TAB POS .P2 (CAR TAIL)
                                              %, .PPF (CADR TAIL]
                          (if TWOFLG
                              then (printout NIL ")")))
          (printout NIL ")" T)
          (if FFLG
              then (printout NIL ")" T])

(GETDECLDEF
  [LAMBDA (NAME FPTYPE OPTIONS)                                          (* bas%: " 9-OCT-79 23:04")

          (* This is the GETDEF function for DECLTYPE. FPTYPE is the file-package-type argument, which we ignore.)

    (WITH ((TB (FINDDECLTYPE NAME))
           (NOCOPYP (EQMEMB 'NOCOPY OPTIONS)))
          (if TB
              then [CONS 'DECLTYPE (CONS NAME (CONS (WITH ((TE (fetch TYPEXP of TB)))
                                                           (if NOCOPYP
                                                               then TE
                                                             else (COPY TE)))
                                                    (WITH ((TP (fetch PROPS of TB)))
                                                           (if NOCOPYP
                                                               then TP
                                                             else (COPY TP]
            elseif (EQMEMB 'NOERROR OPTIONS)
              then NIL
            else (DECLERROR NAME "is not a DECLTYPE"])
)

[PUTDEF 'DECLTYPES 'FILEPKGCOMS '([COM MACRO (X (E (DUMPDECLTYPES 'X]
                                  (TYPE DESCRIPTION "type declarations" GETDEF GETDECLDEF DELDEF
                                        (LAMBDA (NAME)
                                               (DELETETB (OR (FINDDECLTYPE NAME)
                                                             (DECLERROR "Can't delete non-existent type:" NAME]

[PUTDEF 'IGNOREDECL 'FILEPKGCOMS '((COM MACRO (X (DECLARE%: DOEVAL@COMPILE DONTEVAL@LOAD DONTCOPY
                                                         (P (RESETSAVE COMPILEIGNOREDECL 'X]

(PUTPROPS DECLTYPE ARGNAMES (NIL (NAME TYPE PROP1 VAL1 |...|) . X))
```

```
            (* * User functions)

(DEFINEQ

(COVERS
  [LAMBDA (HI LO)                                                  (* bas%: "16-OCT-79 11:22")
    (AND (COVERSTB (GETDECLTYPE HI)
                (GETDECLTYPE LO))
         T])

(GETDECLTYPEPROP
  [LAMBDA (TYPE PROP)                                              (* bas%: " 9-OCT-79 22:56")
    (GETTBPROP (GETDECLTYPE TYPE)
           PROP])

(SETDECLTYPEPROP
  [LAMBDA (NAME PROP VAL)                                          (* rmk%: " 2-AUG-81 08:41")
    (OR (LITATOM NAME)
        (DECLERROR "Can't set property of non-atomic type:" NAME))
    (REPROPTB (OR (FINDDECLTYPE NAME)
                  (DECLERROR "Can't set property of non-existent type:" NAME))
           (LIST PROP VAL))
    (MARKASCHANGED (LIST NAME PROP)
           'DECLTYPES)
    VAL])

(SUBTYPES
  [LAMBDA (NAME)                                                   (* bas%: "16-OCT-79 18:59")
    (PROG (TYPES CT (BT (GETDECLTYPE NAME)))
          (DECLARE (SPECVARS TYPES CT))
          (SETQ CT (GETCTYPE BT))
          (if (NONEC CT)
            elseif (EQ (CAR (fetch TYPEXP of BT))
                       'ONEOF)
              then [RETURN (APPEND (CDR (fetch TYPEXP of BT]
            else [foreachTB S (AND (LITATOM (fetch NAME of S))
                                   (FMEMB CT (GETCTYPE S))
                                   (push TYPES (fetch NAME of S]
                 (RETURN (OR TYPES (LIST 'NONE])

(SUPERTYPES
  [LAMBDA (NAME)                                                   (* bas%: "16-OCT-79 18:20")
    (PROG (TYPES (CN (GETCGETD NAME)))
          (DECLARE (SPECVARS TYPES CN))
          [if (ANYC CN)
              then NIL
            else (foreachTB TB (AND (LITATOM (fetch NAME of TB))
                                    (if (NONEC CN)
                                        then                       (* Very expensive, but kinda wierd)
                                             (NULL (SUBTYPES (fetch NAME of TB)))
                                      else                         (* Any sups will be complete so we dont need to complete here)
                                           (FMEMB (fetch CTYPE of (fetch DEF of TB))
                                                  CN))
                                    (push TYPES (fetch NAME of TB]
          (RETURN TYPES])
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS SELCOVERSQ MACRO [F (LIST [LIST 'LAMBDA '($$TMP)
                                         (CONS 'COND (MAPLIST (CDR F)
                                                           (FUNCTION (LAMBDA (I)
                                                                       (COND
                                                                          ((CDR I)
                                                                           (CONS (LIST 'COVERS
                                                                                       (KWOTE (CAAR I))
                                                                                       '$$TMP)
                                                                                 (CDAR I)))
                                                                          (T (LIST T (CAR I]
                                    (LIST 'DECLOF (CAR F])

(PUTPROPS SELTYPEQ MACRO (F (APPLYFORM [LIST 'LAMBDA '($$TMP)
                                           (CONS 'COND (MAPLIST (CDR F)
                                                             (FUNCTION (LAMBDA (I)
                                                                         (COND
                                                                            ((CDR I)
                                                                             (CONS (LIST 'TYPE?
                                                                                         (CAAR I)
                                                                                         '$$TMP)
                                                                                   (CDAR I)))
                                                                            (T (LIST T (CAR I]
                                     (CAR F))))
)

(ADDTOVAR PRETTYEQUIVLST (SELCOVERSQ . SELECTQ)
                         (SELTYPEQ . SELECTQ))

(ADDTOVAR DWIMEQUIVLST (SELCOVERSQ . SELECTQ)
                       (SELTYPEQ . SELECTQ))

(SETSYNONYM '(THE TYPE)
       '(AS A TYPE))


          (* * Internal machinery)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(DATATYPE TYPEBLOCK (NAME DEF BF SF TF PROPS)
       [ACCESSFNS TYPEBLOCK ([TYPEXP (fetch TEXP of (fetch DEF of DATUM))
                                     (replace DEF of DATUM with (create TYPEDEF
                                                                       TEXP _ (COPY NEWVALUE]
                             (BINDFN (OR (fetch BF of DATUM)
                                         (MAKEBINDFN DATUM))
                                     (replace BF of DATUM with NEWVALUE))
                             (SETFN (OR (fetch SF of DATUM)
                                        (MAKESETFN DATUM))
                                    (replace SF of DATUM with NEWVALUE))
                             (TESTFN (OR (fetch TF of DATUM)
                                         (MAKETESTFN DATUM))
                                     (replace TF of DATUM with NEWVALUE]
       [CCREATE (PROGN (OR (FASSOC 'NAME FIELDS.IN.CREATE)
                           (HELP "No NAME field in TYPEBLOCK create"))
                       (OR (FASSOC 'TYPEXP FIELDS.IN.CREATE)
                           (FASSOC 'DEF FIELDS.IN.CREATE)
                           (HELP "No type expression in TYPEBLOCK create"))
                       (LIST 'NOTICETB 'DATUM (CADR (FASSOC 'NAME FIELDS.IN.CREATE]
       (SYSTEM))

(RECORD TYPEDEF (TEXP . CTYPE))
)

(/DECLAREDATATYPE 'TYPEBLOCK '(POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '12)

(ADDTOVAR PRETTYPRINTYPEMACROS (TYPEBLOCK . PPDTYPE))
)

(/DECLAREDATATYPE 'TYPEBLOCK '(POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '12)

(DEFPRINT 'TYPEBLOCK 'TBDEFPRINT)

(DEFINEQ

(CHECKTYPEXP
  [LAMBDA (TE)                                                   (* bas%: "18-OCT-79 16:58")
                                                                 (* Checks that a type expression is structurally valid)
    (OR (TETYPE TE)
        (DECLERROR "Invalid type expression" TE])

(COLLECTTYPES
  [LAMBDA (TYPES MERGE)                                          (* bas%: " 9-OCT-79 22:53")

          (* Converts a list of type names into a list of typeblocks merging together those that lie on the same sup-sub chain to the
          highest or lowest member of that chain as specified by MERGE)

    (for SCR I VAL in TYPES eachtime (SETQ I (GETDECLTYPE SCR))
       unless [for J in VAL thereis (OR (EQ I J)
                                        (SELECTQ MERGE
                                            (UP (COVERSTB J I))
                                            (DOWN (COVERSTB I J))
                                            (SHOULDNT]
       do (SETQ VAL (CONS I VAL))                                (* Add to VAL unless dominated by an already existing entry)
       finally                                                   (* Walk back down list throwing out anything that is dominated
                                                                 by a subsequent addition)
             (SETQ SCR VAL)
             (SETQ VAL NIL)
             [while SCR do (SETQ SCR (PROG1 (CDR SCR)
                                           (if (for J in VAL never (SELECTQ MERGE
                                                                       (UP (COVERSTB J (CAR SCR)))
```

```
                                                                        (DOWN (COVERSTB (CAR SCR)
                                                                                        J))
                                                                        (SHOULDNT)))
                                                     then (RPLACD SCR VAL)
                                                          (SETQ VAL SCR)))]
            (RETURN VAL])
```

```
(COVERSCTYPE
  [LAMBDA (H L)                                                        (* bas%: "11-OCT-79 11:35")
                                                                       (* COVERS for CTYPEs)
    (PROG NIL

          (* We use a PROG so we can chase singleton supertypes which are the common case by looping rather than recursion.)

      LP  (if (EQ H L)
              then (RETURN T)
            elseif (NLISTP L)
              then (RETURN (NONEC L))                                 (* Either we had NONE to start or we have arrived at ANY)
            elseif (CDR L)
              then (RETURN (for I in L thereis (COVERSCTYPE H I)))
            else (SETQ L (CAR L))                                      (* Single supertype)
                 (GO LP])

(COVERSTB
  [LAMBDA (H L)                                                        (* bas%: "19-OCT-79 17:40")
                                                                       (* COVERS for type blocks. COVERSTE gets some cases that
                                                                       are difficult from the lattice.)
    (OR (EQ H L)
        (COVERSCTYPE (GETCTYPE H)
               (GETCTYPE L))
        (COVERSTE (fetch TYPEXP of H)
               (fetch TYPEXP of L])

(COVERSTE
  [LAMBDA (H L)                                                        (* bas%: "31-OCT-79 14:34")

          (* COVERS for type expressions. We pick off MEMQ and ONEOFs here because they cannot be efficiently linked into the
          type lattice.)

    (SELECTQ (TETYPE H)
         (ONEOF (SELECTQ (TETYPE L)
                     (ONEOF (for I in (CDR L) always (for J in (CDR H) thereis (COVERS J I))))
                     (for I in (CDR H) thereis (COVERS I L))))
         (MEMQ (SELECTQ (TETYPE L)
                   (MEMQ (for I in (CDR L) always (MEMBER I (CDR H))))
                   NIL))
        NIL])

(CREATEFNPROP
  [LAMBDA (PL PN)                                                      (* bas%: " 7-NOV-79 16:51")

          (* If a value for prop PN appears on PL, CREATEFNVAL it. NIL will indicate that no specification has yet been made.)

    (WITH ((PE (FINDPROP PL PN)))
          (AND PE (CREATEFNVAL (CADR PE)
                         PN])

(CREATEFNVAL
  [LAMBDA (FVAL FNAME)                                                 (* bas%: " 7-NOV-79 16:53")

          (* Gets given a purported FNVAL. If that value is NIL, use the default.
          Dwimify a list value.)

    (DECLARE (USEDFREE DWIMFLG))
    (if FVAL
        then (AND DWIMFLG (LISTP FVAL)
                  (DWIMIFY FVAL T))
             FVAL
      else (SELECTQ FNAME
               (BINDFN (CONSTANT DefaultBindFn))
               (SETFN (CONSTANT DefaultSetFn))
               NIL])

(DECLERROR
  [LAMBDA (MSG1 MSG2)                                                  (* bas%: "25-NOV-78 18:25")
    (if (BOUNDP 'DECLERROR)
        then (SETQ DECLERROR T)
             (ERROR!)
      else (ERROR MSG1 MSG2])
```

```
(DELETETB
  [LAMBDA (TB)                                                   (* rmk%: "19-AUG-81 00:15")
                                                                 (* Dissasociates TB from its name and undoes any
                                                                 dependencies on it)
    (WITH ((NAME (fetch NAME of TB)))
          (SELECTQ NAME
              ((ANY NONE)
                   (DECLERROR "(Futile) attempt to delete" NAME))
              NIL)
          (UNSAVETYPE TB)                                        (* Unsave dependent code)
          (UNCOMPLETE TB)                                        (* Undo the cached values and dependent types)
          (NOTICETB NIL NAME)                                    (* Snap name association)
          (replace NAME of TB with (PACK* 'Deleted NAME])

(FINDDECLTYPE
  [LAMBDA (TE)                                                   (* bas%: "10-OCT-79 01:46")
                                                                 (* Finds the existing typeblock for a type expression if any)
    (COND
       ((LISTP TE)                                               (* TE must be in CLISPARRAY to detect edits of the type
                                                                 expression)
        (AND (GETHASH TE CLISPARRAY)
             (GETHASH TE DECLTYPESARRAY)))
       (T (OR (GETHASH TE DECLTYPESARRAY)
              (RECDTYPE TE])

(FINDPROP
  [LAMBDA (L P)                                                  (* rmk%: "12-Mar-85 08:56")
    (for TAIL on L by (CDR (LISTP (CDR TAIL))) thereis (EQ P (CAR TAIL])

(FINDTYPEXP
  [LAMBDA (TYPE)                                                 (* bas%: "16-OCT-79 14:17")
                                                                 (* Tries to find an equivalent TYPEBLOCK for the expression
                                                                 TYPE)
    (DECLARE (SPECVARS TYPE))
    (foreachTB TB (if (AND (LISTP (fetch NAME of TB))
                           (EQUAL TYPE (fetch TYPEXP of TB)))
                      then (OR (EQUAL TYPE (fetch NAME of TB))
                               (replace NAME of TB with TYPE))   (* NAME has been edited)
                           (NOTICETB TB TYPE)                    (* Remember this path)
                           (RETFROM 'FINDTYPEXP TB)))
    NIL])

(GETCTYPE
  [LAMBDA (TB)                                                   (* rmk%: "29-NOV-81 14:33")
    (OR (fetch CTYPE of (fetch DEF of TB))
        (if (DECLRECURSING 'GETCTYPE TB)
            then (DECLERROR "Invalid recursive type definition" (fetch TYPEXP of TB)))
        (replace CTYPE of (fetch DEF of TB) with (MAKECTYPE (fetch TYPEXP of TB])

(GETDECLTYPE
  [LAMBDA (TE VARNAME)                                           (* bas%: "18-OCT-79 15:38")

            (* Either finds a typeblock with TE as its type expression or creates one.
            We smuggle the name thru in the PROPS field as anyone who specifies a VARNAME is unnamed so neither has nor can
            acquire any properties.)

    (OR (FINDDECLTYPE TE)
        (AND (LISTP TE)
             (OR (FINDTYPEXP TE)
                 (MAKEDECLTYPE TE TE VARNAME)))
        (DECLERROR TE "is not a DECLTYPE"])

(GETDECLTYPE.NOERROR
  [LAMBDA (TE VAR)                                               (* bas%: "19-OCT-79 16:05")
                                                                 (* Makes and completes a typeblock for TE suppressing any
                                                                 DECLERRORs)
    (WITH ((DECLERROR))
          (DECLARE (SPECVARS DECLERROR))
          (OR (CAR (ERSETQ (WITH ((TB (GETDECLTYPE TE VAR)))     (* Force completion so any errors will happen now under the
                                                                 ERSETQ)
                                 (GETCTYPE TB)
                                 (fetch TESTFN of TB)
                                 TB)))
              (COND
                 (DECLERROR NIL)
                 (T (ERROR!])

(GETTBPROP
  [LAMBDA (TB P)                                                 (* bas%: "15-AUG-79 23:49")
    (SELECTQ P
        (BINDFN (fetch BINDFN of TB))
        (SETFN (fetch SETFN of TB))
        (TESTFN (fetch TESTFN of TB))
        (WITH ((PL (FINDPROP (fetch PROPS of TB)
                       P)))
              (if PL
                  then (CADR PL)
                else (INHERITPROP TB P])
```

(INHERITPROP

```
  [LAMBDA (TB PROP)                                             (* bas%: "19-OCT-79 16:45")

             (* Determines how types inherit their properties on the basis of the way they were formed from other types)

    (WITH ((TE (fetch TYPEXP of TB)))
          (AND (LISTP TE)
               (GETDECLTYPEPROP (SELECTQ (TETYPE TE)
                                    ((ALLOF ONEOF)
                                         (WITH ((V (GETDECLTYPEPROP (CADR TE)
                                                          PROP))
                                                (ANYVAL (GETDECLTYPEPROP 'ANY PROP)))
                                               (RETFROM 'INHERITPROP
                                                      (if [OR (EQ V ANYVAL)
                                                              (for I in (CDDR TE)
                                                                 always (EQ V (GETDECLTYPEPROP I PROP]
                                                          then V
                                                        else ANYVAL))))
                                    ((OF SATISFIES WHOSE)
                                         (CAR TE))
                                    (MEMQ (CONS 'ONEOF (for I in (CDR TE) collect (DTYPENAME I))))
                                    (SHARED (if (EQ PROP 'BINDFN)
                                                then (RETFROM 'INHERITPROP (CONSTANT DefaultBindFn))
                                              else (CADR TE)))
                                    ((SUBTYPE SYNONYM)
                                         (CADR TE))
                                    (TUPLE 'LISTP)
                                    'ANY)
                      PROP])
```

(INITDECLTYPES

```
  [LAMBDA NIL                                                   (* rmk%: "14-SEP-82 22:17")
                                                                (* Initializes DECLTYPES hash array)
    [COND
       ((BOUNDP 'DECLTYPESARRAY)
        (CLRHASH DECLTYPESARRAY))
       (T (SETQ DECLTYPESARRAY (CONS (HARRAY 128)
                                     128]
    (FILEPKGCHANGES 'DECLTYPES NIL)                             (* Make FILEPKG forget about any types it may have noticed.)
    (RESETVARS (FILEPKGFLG)
              [for I in '(ANY NONE) do (create TYPEBLOCK
                                              NAME _ I
                                              DEF _ (create TYPEDEF
                                                           TEXP _ I
                                                           CTYPE _ I)
                                              BINDFN _ (CONSTANT DefaultBindFn)
                                              SETFN _ (CONSTANT DefaultSetFn)
                                              TESTFN _ (LAMVAL (EQ I 'ANY]
                                                                (* ANY and NONE are created complete)
              (MAKEDECLTYPEQ ARRAYP (SUBTYPE ANY)
                     (TESTFN ARRAYP))
              (MAKEDECLTYPEQ HARRAYP (SUBTYPE ARRAYP)
                     (TESTFN HARRAYP))
              (MAKEDECLTYPEQ LISTP (SUBTYPE ANY)
                     (TESTFN LISTP EVERYFN EVERY))
              [MAKEDECLTYPEQ HASHARRAY (ONEOF HARRAYP (LISTP (WHOSE (CAR HARRAYP]
              (MAKEDECLTYPEQ READTABLEP (SUBTYPE ARRAYP)
                     (TESTFN READTABLEP))
              (MAKEDECLTYPEQ ATOM (SUBTYPE ANY)
                     (TESTFN ATOM))
              (MAKEDECLTYPEQ LITATOM (SUBTYPE ATOM)
                     (TESTFN LITATOM))
              (MAKEDECLTYPEQ BOOL (MEMQ NIL T))
              (MAKEDECLTYPEQ NUMBERP (SUBTYPE ATOM)
                     (TESTFN NUMBERP))
              (MAKEDECLTYPEQ FIXP (SUBTYPE NUMBERP)
                     (TESTFN FIXP))
              [MAKEDECLTYPEQ CARDINAL (FIXP (SATISFIES (IGEQ VALUE 0]
              (MAKEDECLTYPEQ SMALLP (SUBTYPE FIXP)
                     (TESTFN SMALLP))
              (MAKEDECLTYPEQ LARGEP (SUBTYPE FIXP)
                     (TESTFN LARGEP))
              (MAKEDECLTYPEQ FLOATP (SUBTYPE NUMBERP)
                     (TESTFN FLOATP))
              (MAKEDECLTYPEQ FUNCTION (SUBTYPE ANY)
                     (TESTFN FNTYP))
```

```
                (MAKEDECLTYPEQ NIL (MEMQ NIL)
                       (TESTFN NULL))
                (MAKEDECLTYPEQ LST (ONEOF LISTP NIL)
                       (EVERYFN EVERY))
                (MAKEDECLTYPEQ ALIST (LST OF LISTP))
                (MAKEDECLTYPEQ STACKP (SUBTYPE ANY)
                       (TESTFN STACKP))
                (MAKEDECLTYPEQ STRINGP (SUBTYPE ANY)
                       (TESTFN STRINGP EVERYFN EVERYCHAR])
```

(LCCTYPE

```
  [LAMBDA (TL)                                          (* bas%: "18-SEP-79 17:24")
                                                        (* Returns the lowest common ctype for the type names in TL)

    (WITH [(C1 (GETCGETD (CAR TL]
           (if (CDR TL)
               then (LCC2 C1 (LCCTYPE (CDR TL)))
             else C1])
```

(LCC2

```
  [LAMBDA (A B)                                         (* bas%: "10-OCT-79 19:12")
                                                        (* Returns the lcd of A and B)

    (if (COVERSCTYPE A B)
        then A
      elseif (COVERSCTYPE B A)
        then B
      else (for I C in A do (WITH ((D (LCC2 I B)))
                                  (if (OR (NULL C)
                                          (COVERSCTYPE C D))
                                      then (SETQ C D)))
              finally (RETURN C])
```

(MAKECTYPE

```
  [LAMBDA (TE)                                          (* bas%: "31-OCT-79 16:44")
                                                        (* Computes the real sup types of TE)

    (SELECTQ (TETYPE TE)
         (ALLOF [WITH [(S (COLLECTTYPES (CDR TE)
                                  'DOWN]
                      (if (CDR S)
                          then (SMASHCAR S (FUNCTION GETCTYPE))
                        else                            (* They are all on the same path)
                             (GETCTYPE (CAR S])
         (ONEOF [WITH [(S (COLLECTTYPES (CDR TE)
                                  'UP]

                (* Rather than having the subtypes point to this new ctype, we pick that case up in COVERS to avoid making the supertype
                structure bushy.)

                      (if (CDR S)
                          then (LIST (LCCTYPE (CDR TE)))
                        else                            (* All on the same path)
                             (GETCTYPE (CAR S])
         ((SHARED SYNONYM)
              (GETCGETD (CADR TE)))
         (LIST (SELECTQ (TETYPE TE)
                    (MEMQ (LCCTYPE (for I in (CDR TE)
                                      scratchcollect
                                      (DTYPENAME I))))
                    (GETCGETD (SELECTQ (TETYPE TE)
                                  ((OF SATISFIES WHOSE)
                                       (CAR TE))
                                  (SUBTYPE (CADR TE))
                                  (TUPLE (if (CDR TE)
                                             then 'LISTP
                                           else 'NIL))
                                  (SHOULDNT])
```

(MAKEDECLTYPE

```
  [LAMBDA (NAME DECL PROPS)                             (* bas%: " 7-NOV-79 16:33")
                                                        (* Defines the type specified by the type expression DECL)
    (CHECKTYPEXP DECL)                                  (* Provides an early check on well formedness)
    (WITH [(TB (create TYPEBLOCK
                      NAME _ NAME
                      TYPEXP _ DECL
                      PROPS _ (COPY PROPS]
          (if (LISTP PROPS)
              then (replace BINDFN of TB with (CREATEFNPROP PROPS 'BINDFN))
                   (replace SETFN of TB with (CREATEFNPROP PROPS 'SETFN))
                   (replace TESTFN of TB with (CREATEFNPROP PROPS 'TESTFN))
                   (CREATEFNPROP PROPS 'EVERYFN))
          TB])
```

(MAKEBINDFN

```
  [LAMBDA (TB)                                                  (* bas%: "18-OCT-79 18:17")
                                                                (* Finds a BINDFN for TB)

    (replace BINDFN of TB with (INHERITPROP TB 'BINDFN])
```

(MAKESETFN

```
  [LAMBDA (TB)                                                  (* bas%: "18-OCT-79 21:17")
                                                                (* Finds a SETFN for TB)

    (replace SETFN of TB with (INHERITPROP TB 'SETFN])
```

(MAPTYPEUSERS

```
  [LAMBDA (NAME FN)                                             (* bas%: "28-AUG-79 22:18")
    (DECLARE (SPECVARS . T))
    (foreachTB TB (AND (USESTYPE NAME (fetch TYPEXP of TB))
                       (APPLY* FN TB])
```

(NOTICETB

```
  [LAMBDA (TBLOCK TEXP)                                         (* rmk%: " 7-SEP-81 03:26")
                                                                (* Enters hash links so TBLOCK can be found given type
                                                                expression TEXP)

    (if (LISTP TEXP)
        then (PUTHASH TEXP TEXP CLISPARRAY)                     (* Access name is also in CLISPARRAY to detect changes))
    (PUTHASH TEXP TBLOCK DECLTYPESARRAY])
```

(PPDTYPE

```
  [LAMBDA (TYPE)                                                (* bas%: "18-OCT-79 17:57")
                                                                (* PPs typeblock, completing unless NOCOMPFLG)
    (WITH [(LM (IPLUS 4 (POSITION)))
           (TB (if (type? TYPEBLOCK TYPE)
                   then TYPE
                 else (GETDECLTYPE TYPE]
          (printout NIL "DECLTYPE: " (fetch NAME of TB)
                 " = "
                 (OR (fetch TYPEXP of TB)
                     "No type expression"))
          (printout NIL .TAB LM "Suptypes: ")
          (if (fetch CTYPE of (fetch DEF of TB))
              then (for I in (GETCTYPE TB) declare (SPECVARS I)
                      do (printout NIL .TAB0 (IPLUS LM 10))     (* Start each new suptype list on a new line)
                         (foreachTB S (AND (EQ I (fetch CTYPE of (fetch DEF of S)))
                                           (printout NIL (fetch NAME of S)
                                                  %,)))         (* Dont force a completion to get the CTYPE)
                         )
            else (printout NIL "... not completed..."))
          (if (fetch BF of TB)
              then (printout NIL .TAB LM "Bindfn:   " .PPF (fetch BF of TB)))
          (if (fetch SF of TB)
              then (printout NIL .TAB LM "Setfn:    " .PPF (fetch SF of TB)))
          (if (fetch TF of TB)
              then (printout NIL .TAB LM "Testfn:   " .PPF (fetch TF of TB)))
          [if (fetch PROPS of TB)
              then (printout NIL .TAB LM "Property: ")
                   (for P on (fetch PROPS of TB) by (CDDR P) do (printout NIL .TAB0 (IPLUS LM 10)
                                                                       (CAR P)
                                                                       " = " .P2 (CADR P]
          (TERPRI)
          TB])
```

(RECDTYPE

```
  [LAMBDA (R)                                                   (* rmk%: " 6-SEP-81 04:29")
    (WITH [RDECL TB (TST (LIST 'type? R (CONS 'NILL]

          (* The CONS produces a unique, dwim-immune object to give RECORDTRAN to dwimify.
          We can then substitute for it to build the testfn.)

          (COND
             ([RESETVARS (CLISPCHANGE (DWIMESSGAG T))           (* CLISPCHANGE bound cause RECORDTRAN sets it)

          (* If the record package translation bombs, simply return NIL to GETDECLTYPE, which might then print an error message.)

                         (RETURN (NLSETQ (RECORDTRAN TST 'DTYPE?TRAN]
              (SETQ RDECL (RECLOOK R))
              [SETQ TB (create TYPEBLOCK
                              NAME _ R
                              TYPEXP _ (LIST 'SUBTYPE (RECDEFTYPE RDECL]

          (* Use SETTBPROP to store TESTFN rather than doing it in the create, so that it also shows up on the property list.
          Then the decltype will print with all its info.)

              [SETTBPROP TB 'TESTFN (LAMVAL (LIST COMMENTFLG 'ASSERT%: (LIST 'RECORD R))
                                           (SUBST 'VALUE (CADDR TST)
                                                  (PROG1 (GETHASH TST CLISPARRAY)
                                                         (PUTHASH TST NIL CLISPARRAY]
                                                        (* The record package stores the DECL form in 9th car of the
                                                        translation)
            (for X on (CDAR (FNTH (GETHASH RDECL CLISPARRAY)
                                  9))
               by (CDDR X) do (SETTBPROP TB (CAR X)
                                         (CADR X)))
            TB])

(DECLCHANGERECORD
  [LAMBDA (RNAME RFIELDS OLDFLG)                        (* rmk%: " 7-SEP-81 04:17")

          (* CHANGERECORD is the default value of RECORDCHANGEFN, which is applied by RECREDECLARE.
          This makes sure that a record change wipes out a dependent decltype)

    (REALCHANGERECORD RNAME RFIELDS OLDFLG)
    (AND OLDFLG (WITH (TEMP (TB (GETHASH RNAME DECLTYPESARRAY)))

          (* This is a marginal guess at the dependency%: we would be wrong if the user had, e.g., dumped a record-derived decltype
          and loaded it into a system without the record.)

                      (if (AND TB (SETQ TEMP (fetch TESTFN of TB))
                               [EQ COMMENTFLG (CAR (SETQ TEMP (CADDR TEMP]
                               (EQ (CADR TEMP)
                                   'ASSERT%:)
                               (EQ (CAR (SETQ TEMP (CADDR TEMP)))
                                   'RECORD)
                               (EQ RNAME (CADR TEMP)))
                          then (DELETETB TB])

(RECDEFTYPE
  [LAMBDA (RD)                                          (* bas%: "21-SEP-79 14:53")
                                                        (* Computes the DECLOF type corresponding to a record
                                                        package type expression)
    (SELECTQ (CAR RD)
         (ACCESSFNS (WITH ((CRF (FASSOC 'CREATE RD)))
                          (if CRF
                              then (DECLOF (CADR CRF))
                            else 'ANY)))
         (ARRAYRECORD 'ARRAYP)
         (ASSOCRECORD 'ALIST)
         (ATOMRECORD 'LITATOM)
         (DATATYPE (if (LISTP (CADR RD))
                       then (CADADR RD)
                     else 'ANY))
         (HASHLINK 'HARRAYP)
         (PROPRECORD 'LST)
         (RECORD (WITH ((FLDS (CADDR RD)))
                       (if (LISTP FLDS)
                           then 'LST
                         elseif [AND FLDS (EQ FLDS (CADR (FASSOC 'SUBRECORD RD]
                           then                         (* The declaration has a top-level field equal to the subrecord
                                                        name)
                                FLDS
                         else 'ANY)))
         (TYPERECORD 'LISTP)
         'ANY])

(REPROPTB
  [LAMBDA (TB PROPS INHERITING)                         (* bas%: " 7-NOV-79 15:46")
                                                        (* Propgates changes in properties)
    (PROG [(NEWP (for old PROPS by (CDDR PROPS) while PROPS
                    unless [if INHERITING
                               then (FINDPROP (fetch PROPS of TB)
                                              (CAR PROPS))
                             else (EQUAL (CADR PROPS)
                                         (LISTGET (fetch PROPS of TB)
                                                  (CAR PROPS]
                    join (SETTBPROP TB (CAR PROPS)
                                   (COPY (CADR PROPS))
                                   INHERITING)
                         (LIST (CAR PROPS)
                               (CADR PROPS]
          (DECLARE (SPECVARS NEWP))
          [if NEWP
              then (UNSAVETYPE TB)                      (* Probably not necessary, but we cant tell)
                   (MAPTYPEUSERS (fetch NAME of TB)
                          (FUNCTION (LAMBDA (X)
                                      (REPROPTB X NEWP T]

          (* Any recursions bottom out b/c the change will have been made the first time the type is reached)

          (RETURN NEWP)                                 (* Indicate if any changes)
    ])
```

```
(SETTBPROP
  [LAMBDA (TB P V BLKONLY)                                          (* bas%: " 7-NOV-79 16:55")
    (SELECTQ P
         (BINDFN (replace BINDFN of TB with (CREATEFNVAL V 'BINDFN)))
         (EVERYFN (CREATEFNVAL V 'EVERYFN))
         (SETFN (replace SETFN of TB with (CREATEFNVAL V 'SETFN)))
         (TESTFN [SELECTQ (fetch NAME of TB)
                      ((ANY NONE)
                           (DECLERROR "(Futile) attempt to change TESTFN of" (fetch NAME of TB)))
                      (replace TESTFN of TB with (CREATEFNVAL V 'TESTFN])
         NIL)                                                       (* Unless BLKONLY, must also put on property list so it is
                                                                    known)
    (if BLKONLY
      elseif (fetch PROPS of TB)
        then (LISTPUT (fetch PROPS of TB)
                    P V)
      else (replace PROPS of TB with (LIST P V])

(TBDEFPRINT
  [LAMBDA (TB)                                                      (* bas%: "22-NOV-78 14:32")
                                                                    (* DEFPRINTer for TYPEBLOCKs.
                                                                    Made a function to allow supression of constant cons)
    (CBOX (CONCAT "{DECLTYPE: " (fetch NAME of TB)
                 "}")
          (PACK])

(TETYPE
  [LAMBDA (TE)                                                      (* rmk%: "18-Feb-85 18:07")
                                                                    (* returns the type of a type expression)
    (if (LITATOM TE)
        then 'PRIMITIVE
      elseif (LISTP TE)
        then (SELECTQ (CAR TE)
                 ((ALLOF MEMQ ONEOF SHARED SUBTYPE SYNONYM TUPLE)
                      (CAR TE))
                 (AND (LISTP (CDR TE))
                      (SELECTQ (CADR TE)
                           (OF 'OF)
                           (SELECTQ (CAR (LISTP (CADR TE)))
                                ((SATISFIES WHOSE)
                                     (CAADR TE))
                                NIL])

(TYPEMSANAL
  [NLAMBDA (KIND)                                                   (* rmk%: " 9-NOV-83 09:24")
                                                                    (* Returns the information that the various templates expect.)
    (DECLARE (USEDFREE EXPR FNNAME))
    (SELECTQ KIND
         (COVERS (SCRATCHLIST (CBOX)
                        (TYPEMSANAL1 EXPR)))
         ((type? the)
              [LBOX KIND (SCRATCHLIST (CBOX)
                                (TYPEMSANAL1 (CADR EXPR)))
                    (OR (GETHASH EXPR CLISPARRAY)
                        (RESETVARS (FILEPKGFLG (NOSPELLFLG T)
                                               (DWIMESSGAG T))
                                   (PROG (LISPXHIST)
                                         (DECLARE (SPECVARS LISPXHIST))
                                         (DWIMIFY0? EXPR EXPR NIL NIL NIL FNNAME))
                                   (RETURN (GETHASH EXPR CLISPARRAY])
         (\*DECL

              (* We assume that the \*DECL came from a previous dwimification which also got the testfn.
              The typeblock should already exist, but sometimes it isn't found cause the clisparray gets cleared.
              The MAKEAPPLYFORM means that bogus VALUE's are most likely eliminated)

                (LBOX [SCRATCHLIST (CBOX)
                                (TYPEMSANAL1 (fetch DECL of (fetch VARDECL of EXPR]
                      (APPLYFORM [fetch TESTFN of (GETDECLTYPE (fetch DECL of (fetch VARDECL of EXPR]
                                (fetch VARNAME of EXPR))))
         (SHOULDNT])

(TYPEMSANAL1
  [LAMBDA (TYPEXP)                                                  (* bas%: "16-AUG-79 11:55")

              (* Collects from a type expression the names of all the named types that it uses)

    (if (LITATOM TYPEXP)
        then (ADDTOSCRATCHLIST TYPEXP)
      elseif (LISTP TYPEXP)
        then [SELECTQ (CAR TYPEXP)
                 ((ALLOF ONEOF SHARED SUBTYPE TUPLE)
                    (for X in (CDR TYPEXP) do (TYPEMSANAL1 X)))
                (MEMQ NIL)
                (PROGN                                              (* Infix operator so CAR is a type)
                      (TYPEMSANAL1 (CAR TYPEXP))                    (* CDR TYPEXP must also be a listp)
                      (if (EQ (CADR TYPEXP)
                              'OF)
                          then [TYPEMSANAL1 (CAR (LISTP (CDDR TYPEXP]
                        else (SELECTQ (CAR (LISTP (CADR TYPEXP)))
                                    (SATISFIES NIL)
                                    (WHOSE (for I in (CDADR TYPEXP) do (TYPEMSANAL1 (CADR I))))
                                    (SHOULDNT]
     else (SHOULDNT])

(UNCOMPLETE
  [LAMBDA (TB)                                                      (* bas%: " 7-NOV-79 16:08")
                                                                    (* Reinitializes the TYPEBLOCK for NAME, recursing if
                                                                    necessary)
    (replace BINDFN of TB with (CREATEFNPROP (fetch PROPS of TB)
                                       'BINDFN))
    (replace SETFN of TB with (CREATEFNPROP (fetch PROPS of TB)
                                       'SETFN))
    (replace TESTFN of TB with (CREATEFNPROP (fetch PROPS of TB)
                                       'TESTFN))
    (if (fetch CTYPE of (fetch DEF of TB))
        then (replace CTYPE of (fetch DEF of TB) with NIL)
             (MAPTYPEUSERS (fetch NAME of TB)
                    (FUNCTION UNCOMPLETE])

(UNSAVETYPE
  [LAMBDA (TYPE)                                                    (* rmk%: " 7-SEP-81 03:44")
    (DECLARE (SPECVARS TYPE))
    [MAPHASH CLISPARRAY (FUNCTION (LAMBDA (TRAN ORIG)
                                    (if (FORMUSESTB ORIG TRAN TYPE)
                                        then (PUTHASH ORIG NIL CLISPARRAY]
                                                                    (* Clear translations that depend on this type)
    (AND MSDATABASELST (MSNEEDUNSAVE (GETRELATION (fetch NAME of TYPE)
                                            '(USE TYPE)
                                            T)
                              "type declarations" T])

(USERDECLTYPE
  [LAMBDA (NAME DECL PROPS)                                         (* rmk%: " 2-AUG-81 08:42")
                                                                    (* User entry to MAKEDECLTYPE)
    (if (LITATOM NAME)
        then (WITH ((TB (GETHASH NAME DECLTYPESARRAY)))            (* We use GETHASH to avoid creating record based types)
                   (if [OR (EQ DECL NAME)
                           (AND TB (EQUAL DECL (fetch TYPEXP of TB]
                       then (AND (REPROPTB (GETDECLTYPE NAME)
                                        PROPS)
                                 (MARKASCHANGED NAME 'DECLTYPES))  (* Adding properties to existing type)

                     else (SELECTQ NAME
                                  ((ANY NONE)
                                       (DECLERROR "(Futile) attempt to redefine" NAME))
                                  NIL)
                          [MARKASCHANGED NAME 'DECLTYPES (COND
                                                             (TB 'CHANGED)
                                                             (T 'DEFINED]
                          (if TB
                              then (DELETETB TB))                  (* Forget it if it exists then remake it)
                          (MAKEDECLTYPE NAME (OR (LISTP DECL)
                                                 (LIST 'SYNONYM DECL))
                                 PROPS)))
             NAME
      else (DECLERROR "Non-atomic DECLTYPE name" NAME])

(USESTYPE
  [LAMBDA (NAME TE)                                                 (* rmk%: "18-Feb-85 18:14")
                                                                    (* Computes whether NAME appears in TE)
    (OR (EQ NAME TE)
        (SELECTQ (TETYPE TE)
            ((ALLOF ONEOF SHARED SUBTYPE SYNONYM)
                 (for I in (CDR TE) thereis (USESTYPE NAME I)))
            (MEMQ (for I in (CDR TE) thereis (EQ I (DTYPENAME I))))
            (OF [OR (USESTYPE NAME (CAR TE))
                    (AND [LISTP (CDR (LISTP (CDR TE]
                         (USESTYPE NAME (CADDR TE])
            (PRIMITIVE NIL)
            (SATISFIES (USESTYPE (CAR TE)))
            (TUPLE [OR (EQ NAME (if (CDR TE)
                                    then 'LISTP
                                  else NIL))
                   (USESTYPE NAME (CONS 'ALLOF (CDR TE])
            (WHOSE [OR (USESTYPE NAME (CAR TE))
                       (for I in (CADR TE) when (LISTP (CDR (LISTP I))) thereis (USESTYPE NAME (CADR I])
            (SHOULDNT])

)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: LCCTYPE LCCTYPE LCC2)

(BLOCK%: TYPEMSANAL TYPEMSANAL TYPEMSANAL1)
)


          (* * Test fn creation block)

(DEFINEQ

(MAKETESTFN
  [LAMBDA (TB)                                                (* rmk%: "29-NOV-81 22:41")

          (* Computes the test fn for a type block. Called from TESTFN fetch function.
          This is not a part of the MAKETESTFNBLOCK, so that the name MAKETESTFN is a reliable indicator for checking
          recursion (DECLRECURSING))

    (MAKETESTFNBLOCK TB])

(MAKETESTFNBLOCK
  [LAMBDA (TB)                                                (* rmk%: " 6-FEB-82 14:25")
                                                              (* Computes the test fn for a type block)
    (WITH [(TE (fetch TYPEXP of TB))
           (BINDINGNAME (OR (AND (LITATOM (fetch PROPS of TB))
                                 (fetch PROPS of TB))
                            'VALUE]
          (DECLARE (SPECVARS BINDINGNAME))
          (COND
             [(DECLRECURSING 'MAKETESTFNBLOCK TB)             (* Name will be returned)
              (replace TESTFN of TB with (PACK* (fetch NAME of TB)
                                                '.TestFn]
             (T (WITH [(DEF (SELECTQ (TETYPE TE)
                                (ALLOF (COMBINE.TESTS (SMASHCAR (COLLECTTYPES (CDR TE)
                                                                              'DOWN)
                                                                (FUNCTION TESTFORM))
                                              'AND))
                                (MEMQ [LAMBIND (COND
                                                  [(COVERSTB (GETDECLTYPE '(ONEOF LITATOM SMALLP))
                                                             TB)
                                                   (COND
                                                      ((CDDR TE)
                                                       (LIST 'SELECTQ BINDINGNAME (LIST (CDR TE)
                                                                                        T)
                                                             NIL))
                                                      (T (LIST 'EQ BINDINGNAME (KWOTE (CADR TE]
                                                  (T (COND
                                                        [(CDDR TE)
                                                         (LIST 'MEMBER BINDINGNAME (KWOTE (CDR TE]
                                                        (T (LIST 'EQUAL BINDINGNAME (KWOTE (CADR TE])
                                (OF [OF.TESTFN (GETDECLTYPE (CAR TE))
                                           (GETDECLTYPE (COND
                                                           ((CDDDR TE)
                                                            (CDDR TE))
                                                           (T (CADDR TE])
                                (ONEOF (COMBINE.TESTS (SMASHCAR (COLLECTTYPES (CDR TE)
                                                                              'UP)
                                                                (FUNCTION TESTFORM))
                                              'OR))
                                (SATISFIES (COMBINE.TESTS [LIST (TESTFORM (GETDECLTYPE (CAR TE)))
                                                                (COND
                                                                   ((CDDR (CADR TE))
                                                              (* There might be multiple forms or disconnected CLISP)
                                                                    (CONS 'AND (CDADR TE)))
                                                                   (T (CADR (CADR TE]
                                                  'AND))
                                ((SHARED SUBTYPE SYNONYM)
                                     (fetch TESTFN of (GETDECLTYPE (CADR TE))))
                                (TUPLE (TUPLE.TESTFN (CDR TE)))
                                (WHOSE (WHOSE.TESTFN TB (CAR TE)
                                              (CDADR TE)))
                                (SHOULDNT]
                       (WITH ((TF (fetch TF of TB)))
                             (replace TESTFN of TB with (COND
                                                           [TF

          (* Must be recursive with TF being the atom name.TestFn and DEF being a lambda expression.
          Convert to a LABEL expression, then translate it using DOLABEL from LABEL package.)
```

```
                                                          (DOLABEL (CONS 'LABEL (CONS TF (CDR DEF]
                                                      (T DEF])

(COMBINE.TESTS
  [LAMBDA (TESTS ANDOR)                                            (* bas%: "28-AUG-79 13:29")
                                                                   (* Composes TESTS under either AND or OR composition)
    (FUNIFY [for TST in TESTS join (COND
                                    ((EQ (CAR (LISTP TST))
                                         ANDOR)
                                     (APPEND (CDR TST)))
                                    ((EQ TST (EQ ANDOR 'AND))      (* AND T or OR NIL)
                                     NIL)
                                    ((EQ TST (EQ ANDOR 'OR))       (* AND NIL or OR T)
                                     (RETURN (LIST TST)))
                                    (T (LIST TST]
           ANDOR])

(FUNIFY
  [LAMBDA (TEST ANDOR)                                             (* bas%: "11-OCT-79 18:05")
                                                                   (* Provides LAMBDA abstraction for COMBINE.TESTS)
    (LAMBIND (COND
                ((NLISTP TEST)                                     (* No tests)
                 (EQ ANDOR 'AND))
                ((CDR TEST)                                        (* More than one clause)
                 (CONS ANDOR TEST))
                (T (CAR TEST])

(MKNTHCAR
  [LAMBDA (L N)                                                    (* bas%: " 8-MAR-79 17:55")
                                                                   (* Constructs an expression for getting the Nth car of L)
    (PROG [(F (MKNTHCDR L (SUB1 N]
          (RETURN (SELECTQ (CAR F)
                      (CDR (CONS 'CADR (CDR F)))
                      (CDDR (CONS 'CADDR (CDR F)))
                      (CDDDR (CONS 'CADDDR (CDR F)))
                      (LIST 'CAR F])

(MKNTHCDR
  [LAMBDA (L N)                                                    (* bas%: " 9-MAR-79 14:50")
                                                                   (* Constructs an expresssion for getting the Nth cdr of L)
    (if (ZEROP N)
        then L
      elseif (ILESSP N 5)
        then (LIST (SELECTQ N
                       (1 'CDR)
                       (2 'CDDR)
                       (3 'CDDDR)
                       (4 'CDDDDR)
                       (SHOULDNT))
                   L)
      elseif (ILESSP N 9)
        then (MKNTHCDR (LIST 'CDDDDR L)
                    (IDIFFERENCE N 4))
      else (LIST 'FNTH L (ADD1 N])

(OF.TESTFN
  [LAMBDA (AGG ELT)                                                (* rmk%: "19-AUG-81 00:08")
    (COMBINE.TESTS [LIST (TESTFORM AGG)
                         (LIST (OR (GETTBPROP AGG 'EVERYFN)
                                   (DECLERROR "OF construction used with non-aggregate type"))
                               BINDINGNAME
                               (LIST 'FUNCTION (fetch TESTFN of ELT]
           'AND])

(TUPLE.TESTFN
  [LAMBDA (TYPES)                                                  (* rmk%: "19-AUG-81 00:16")
                                                                   (* Constructs the test function for TUPLEs)
    (COND
       (TYPES (COMBINE.TESTS [CONS (LIST 'EQLENGTH BINDINGNAME (LENGTH TYPES))
                                   (for I in TYPES as J from 1 collect (APPLYFORM (fetch TESTFN of (GETDECLTYPE
                                                                                                     I))
                                                                                  (MKNTHCAR BINDINGNAME J]
                     'AND))
       (T 'NULL])

(WHOSE.TESTFN
  [LAMBDA (TB SNAM TAIL)                                           (* bas%: " 6-NOV-79 16:56")
                                                                   (* Constructs TESTFN for WHOSE expressions)
    (COMBINE.TESTS [CONS (TESTFORM (GETDECLTYPE SNAM))
                         (for I in TAIL collect (APPLYFORM [fetch TESTFN of (GETDECLTYPE (COND
                                                                                                                    ((EQLENGTH I 2)
                                                                                                                     (CADR I))
                                                                                                                    (T (CDR I]
                                                                         (COND
                                                                            [(EQ SNAM 'LISTP)
                                                                             (WITH ((V (CAR I)))
                                                                                   (SELECTQ V
                                                                                       ((CAR CDR CADR CDDR CAAR CDAR)
                                                                                            (LIST V BINDINGNAME))
                                                                                       (COND
                                                                                          ((AND (FIXP V)
                                                                                                (NOT (MINUSP V)))
                                                                                           (MKNTHCAR BINDINGNAME V]
                                                                            ((FMEMB (CAR I)
                                                                                    (RECORDFIELDNAMES SNAM))
                                                                             (LIST 'FETCH (LIST SNAM (CAR I))
                                                                                   'OF BINDINGNAME))
                                                                            (T (DECLERROR (CAR I)
                                                                                      " is not a valid fieldname"]
             'AND])

)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: MAKETESTFNBLOCK MAKETESTFNBLOCK COMBINE.TESTS FUNIFY MKNTHCAR MKNTHCDR OF.TESTFN TUPLE.TESTFN
       WHOSE.TESTFN)
)


            (* * Machinery to compile recursive testfns)

(FILESLOAD (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
       LABEL)


            (* * Idioms. Expressed as macros for now)

(DECLARE%: DONTCOPY EVAL@COMPILE

(RPAQQ DefaultBindFn PROGN)

(RPAQQ DefaultSetFn REALSETQ)

(ADDTOVAR NLAMA MAKEDECLTYPEQ)

(DECLARE%: EVAL@COMPILE

(PUTPROPS ANYC MACRO ((C)
                      (EQ C 'ANY)))

(PUTPROPS DECLVARERROR MACRO [ARGS (LSUBST ARGS 'ARGS '(DECLDWIMERROR ARGS T "   inside " VARD])

(PUTPROPS DTYPENAME MACRO [(X)
                           (COND
                              ((LARGEP X)
                               'LARGEP)
                              (T (TYPENAME X])

(PUTPROPS foreachTB MACRO [ARGS (LIST 'MAPHASH 'DECLTYPESARRAY (LIST 'FUNCTION
                                                                      (CONS 'LAMBDA (CONS (LIST (CAR ARGS))
                                                                                          (CDR ARGS])

(PUTPROPS GETCGETD MACRO ((X)
                          (GETCTYPE (GETDECLTYPE X))))

(PUTPROPS KWOTEBOX MACRO [(V)
                          ([LAMBDA ($$8)
                              (DECLARE (LOCALVARS $$8))
                              (FRPLACA (CDR $$8)
                                     V)
                              $$8]
                           ''Q])

(PUTPROPS LAMBIND MACRO (ARGS (APPEND '(LIST 'LAMBDA (LIST BINDINGNAME))
                                     ARGS)))

(PUTPROPS LAMVAL MACRO (ARGS (APPEND '(LIST 'LAMBDA '(VALUE))
                                    ARGS)))

(PUTPROPS MAKEDECLTYPEQ MACRO ((NAME DEF PROPS)
                               (MAKEDECLTYPE 'NAME 'DEF 'PROPS)))

(PUTPROPS NONEC MACRO ((C)
                       (EQ C 'NONE)))

(PUTPROPS TESTFORM MACRO ((TB)
                               (APPLYFORM (fetch TESTFN of TB)
                                      BINDINGNAME)))
)

(DEFINEQ

(TESTFORM
  [LAMBDA (TB)                                               (* rmk%: "24-NOV-81 22:17")

            (* Doesn't get compiled, cause it is macroed out. Symbolic definition exists because it get's APPLY*, not EVALed)

    (APPLYFORM (fetch TESTFN of TB)
           BINDINGNAME])

)

(ADDTOVAR DONTCOMPILEFNS TESTFORM)

(SETTEMPLATE 'foreachTB '(CALL BIND |..| EFFECT))

(SETTEMPLATE 'MAKEDECLTYPEQ '(CALL NIL NIL NIL . PPE))
)


            (* * Runtime utility functions)

(DEFINEQ

(EVERYCHAR
  [LAMBDA (STRNG FN)                                         (* bas%: " 6-MAR-79 17:58")
                                                             (* The EVERY function for strings)
    (for I to (NCHARS STRNG) always (APPLY* FN (NTHCHAR STRNG I])

(LARGEP
  [LAMBDA (X)                                                (* rmk%: "24-MAY-79 09:10")
                                                             (* For LARGEP type-tests)
    (AND (FIXP X)
         (NOT (SMALLP X])

(DECLRECURSING
  [LAMBDA (NAME ARG)                                         (* jtm%: "19-Feb-87 11:31")

            (* NAME is the name of a potentially looping function in our call chain.
            ARG is the first arg in that lowest call to NAME. Determines whether the function NAME exists higher on the stack with ARG
            as its first argument. Used to check for recursive loops.)

    (bind (S _ (STKPOS NAME -1)) while (STKPOS NAME -2 S S) when (EQ ARG (STKARG 1 S)) do

            (* At each step we back off one from the last frame we checked b/c it would otherwise be found by STKPOS, and search for
            the next one. S is reused by both stack fns and released if the loop terminates with it pointing to anything.)

                                                                                      (RELSTK S)
                                                                                      (RETURN T])

(SMASHCAR
  [LAMBDA (L FN)                                             (* bas%: "31-OCT-79 17:11")

            (* Maps over L smashing the result of applying FN to each car into that car)

    [MAP L (FUNCTION (LAMBDA (X)
                       (FRPLACA X (APPLY* FN (CAR X]
    L])

)

(DECLARE%: EVAL@COMPILE

(DECLARE%: EVAL@COMPILE

(PUTPROPS LARGEP MACRO [(X)
                              (AND (FIXP X)
                                   (NOT (SMALLP X])
)
)

(DECLARE%: DONTCOPY EVAL@COMPILE

(DECLARE%: EVAL@COMPILE

(PUTPROPS SMASHCAR MACRO [ARGS (SUBST [SELECTQ (CAADR ARGS)
                                          ((FUNCTION QUOTE)
                                               (APPLYFORM (CADADR ARGS)
                                                      '(CAR I)))
                                          (LIST 'APPLY* (CADR ARGS)
                                                      '(CAR I]
                                            'NEWVAL
                                            (LIST '[LAMBDA (L)
                                                      (DECLARE (LOCALVARS L))
                                                      [MAP L (FUNCTION (LAMBDA (I)
                                                                         (DECLARE (LOCALVARS I))
                                                                         (FRPLACA I NEWVAL]
                                                      L]
                                                 (CAR ARGS])
)
)
```

(* * translator of dprogs and dlambdas)

```
(DEFINEQ

(ASSERT
  [NLAMBDA ARGS                                                            (* rmk%: "11-NOV-83 08:00")

          (* ARGS is a mixed list of variable names and forms. Forms must be true, and the test function for variables must be true
          too.)

    (DECLARE (LOCALVARS . T))
    (for V in ARGS do (if (LITATOM V)
                          then (\VARASRT V)
                        elseif (LISTP V)
                          then (OR (EVAL V 'INTERNAL)
                                   (ASSERTFAULT V NIL))
                        else (ERRORX (LIST 27 V])

(ASSERTFAULT
  [LAMBDA (DECL VARNAME)                                                   ; Edited 23-Feb-87 08:35 by jtm:
                                                                           (* Prints out the assertion error messages.)
    (LET [(FN (STKNAME (REALSTKNTH -1 'ASSERTFAULT]
         [COND
            ((LISTP FN)
             (SETQ FN (STKNAME (REALSTKNTH -4 'ASSERTFAULT]
         (LISPXPRIN1 (if VARNAME
                         then "DECLARATION"
                       else "ASSERTION")
                T)
         (LISPXPRIN1 " NOT SATISFIED IN " T)
         (LISPXPRIN2 FN T)
         (LISPXPRIN1 (CONCAT ": " (if VARNAME
                                      then (LIST VARNAME DECL)
                                    else DECL))
                T)
         (APPLY* (FUNCTION BREAK1)
                NIL T FN])

(ASSERTMAC
  [LAMBDA (ARGS)                                                           (* rmk%: " 2-AUG-79 23:21")
                                                                           (* Compiler for ASSERT forms.)
    (if (IGNOREDECL)
        then (CBOX COMMENTFLG (LBOX (CBOX 'ASSERT ARGS)))
      else (for V in ARGS collect (if (LISTP V)
                                      then (LIST 'OR V (LIST 'ASSERTFAULT (KWOTE V)))
                                    elseif (LITATOM V)
                                      then (MAKETESTFORM V (TYPEBLOCKOF V))
                                    else (ERRORX (LIST 27 V)))
              finally (RETURN (if (CDR $$VAL)
                                  then (CONS 'PROGN $$VAL)
                                else (CAR $$VAL])

(\*DECL
  [NLAMBDA ARGS                                                            (* DECLARATIONS%: (RECORD ARGRECORD
                                                                           (SL . FORMS)))
                                                                           (* rmk%: "11-NOV-83 07:59")

          (* This maintains the proper bindings of SATISFIESLIST. It is wrapped around function bodies by dprog's and dlambda's.
          Compiles open, depending on COMPILEIGNOREDECL.)

    (PROG [(SATISFIESLIST (if [OR (NULL (fetch SL of ARGS))
                                  (LISTP (CAAR (fetch SL of ARGS]
                              then

          (* If NIL, then this is the top binding for a declarative without any bindings.
          If LISTP, then this is the first binding in this lexical scope.)

                                   (fetch SL of ARGS)
                            else (CONS (fetch SL of ARGS)
                                       SATISFIESLIST]
          (DECLARE (SPECVARS SATISFIESLIST))                               (* Use \DECLPROGN instead of PROGN so BAKTRACELST
                                                                           can recognize us)
          (RETURN (APPLY (FUNCTION \DECLPROGN)
                         (fetch FORMS of ARGS)
                         'INTERNAL])

(*DECLMAC
  [LAMBDA (ARGS)                                                (* DECLARATIONS%: (RECORD ARGRECORD
                                                                (SL . FORMS)))
                                                                (* rmk%: "15-Sep-87 10:16")
    (DECLARE (USEDFREE CSATISFIESLIST FREEVARS))
    (PROG [FIRSTSL (FV FREEVARS)
                   (SL (fetch SL of ARGS))
                   (FORM (if (CDR (fetch FORMS of ARGS))
                             then (CONS 'PROGN (fetch FORMS of ARGS))
                           else (CAR (fetch FORMS of ARGS]
          [SETQ FIRSTSL (OR (NULL SL)
                            (LISTP (CAAR SL]                    (* The first declaration in this function)

          (* Should maintain list of free variables for the benefit of clisp words arising in macros, but FREEVARS in compiling context
          has different meaning from FREEVARS in dwimify context. Interlisp-D byte compiler gets confused.)

          (for V in (if FIRSTSL
                        then (CAR SL)
                      else SL)
             when (EQ (fetch PROGNFLAG of (fetch VARDECL of V))
                      'FREE)
             do (push FV (fetch VARNAME of V)))
          [SETQ SL (LIST 'CSATISFIESLIST (KWOTE (if FIRSTSL
                                                    then SL
                                                  else (CONS SL CSATISFIESLIST]

          (* Assumes that *DECLMAC is executed in the same compiletime context as CBIND.
          If this is the first declaration, we bind COMPILEIGNOREDECL for the benefit of compiler-generated sub-functions)

          (RETURN (LIST '.CBIND. [CONS SL (NCONC (PROG1 NIL     (* (if (NEQ FV FREEVARS) then (LIST
                                                                (LIST (QUOTE FREEVARS) (KWOTE FV)))))
                                                        )
                                                 (if FIRSTSL
                                                     then (LIST (LIST 'COMPILEIGNOREDECL (IGNOREDECL]
                      FORM])

(\CHKINIT
  [NLAMBDA ARGS                                                 (* bas%: " 9-OCT-79 23:15")

          (* ARGS is a list of variable names whose nearest assertions are to be checked.
          Calls to \CHKINIT are generated by DLAMTRAN just for variables that have inital values and test-forms in the nearest
          SATISFIESLIST entry.)

    (DECLARE (LOCALVARS . T))
    (for V D in ARGS unless (APPLY* (fetch TESTFN of (GETDECLTYPE [SETQ D (fetch DECL
                                                                               of (fetch VARDECL
                                                                                      of (ASSOC V (CAR SATISFIESLIST]
                                                                  V))
                                    (EVALV V))
       do (ASSERTFAULT D V])

(CHKINITMAC
  [LAMBDA (ARGS)                                                (* rmk%: " 9-NOV-83 09:20")
                                                                (* Compiler for \CHKINIT forms.)
    (DECLARE (USEDFREE COMMENTFLG CSATISFIESLIST))
    (if (IGNOREDECL)
        then (CBOX COMMENTFLG (CBOX '\CHKINIT ARGS))
      else                                                      (* The \CHKINIT only includes variables whose testform is not T)
           (for V D TEMP in ARGS collect (LIST 'OR (APPLYFORM (fetch TESTFN
                                                                  of (GETDECLTYPE
                                                                      [SETQ D (fetch DECL
                                                                                 of (fetch VARDECL
                                                                                       of (ASSOC V (CAR CSATISFIESLIST
                                                                                                        ]
                                                                      V))
                                                              V)
                                                   (LIST 'ASSERTFAULT (if (LISTP D)
                                                                          then (CONS 'DECLMSGMAC D)
                                                                        else (KWOTE D))
                                                         (KWOTE V)))
              finally (RETURN (if (CDR $$VAL)
                                  then (CONS 'PROGN $$VAL)
                                else (CAR $$VAL])

(DECLCONSTANTP
  [LAMBDA (X)                                                   (* bas%: " 9-OCT-79 21:32")
    (OR (NULL X)
        (EQ X T)
        (NUMBERP X)
        (STRINGP X)
        (AND (LISTP X)
             (SELECTQ (CAR X)
                 ('CONSTANT
                      T)
                 (WITH ((TEMP (if (GETP (CAR X)
                                        'MACRO)
                                  then (EXPANDMACRO X T)
                                else X)))
```

(* If we did a DECLOF and got a MEMQ, we'd have a constant.
Thus, this code wouldn't have to duplicate what goes on in DECLOF, and we would get the funny PROG and SELECTQ cases for free.)

```
                       (if (AND (NEQ TEMP X)
                                (NEQ TEMP 'IGNOREMACRO))
                           then (DECLCONSTANTP TEMP)
                         elseif (SELECTQ (CAR X)
                                    ((SELECTQ CLOSER GO PROG COND)
```

(* CLOSER has side-effects. The others have CTYPE properties but their arguments can't be simply checked)

```
                                         NIL)
                                    (GETP (CAR X)
                                          'CTYPE))
                           then
```

(* The test we really want is that the function doesn't reference freevariables or cause side-effects.)

```
                                (EVERY (CDR X)
                                       (FUNCTION DECLCONSTANTP])
```

(**DD**

```
  [NLAMBDA X                                     (* DECLARATIONS%: (RECORD ARGRECORD
                                                 (NAME . DEF) (RECORD DEF (ARGS . BODY))))
                                                 (* rmk%: "24-JUL-78 08:13")
```

(* For Defining DLambda functions. NAME is the function name and DEF the rest of its definition.)

```
    (DEFINE [LIST (LIST (fetch NAME of X)
                        (CONS 'DLAMBDA (fetch DEF of X]
           T])
```

(**DECLCLISPTRAN**

```
  [LAMBDA (X TRAN)                                (* rmk%: " 2-Nov-84 15:30")
    [PROG [DECL DFORM DPROGFLAG RETURNS (DECLARETAGS '(LOCALVARS SPECVARS ADDTOVAR DEFLIST PUTPROPS CONSTANTS
                                                                  SETQQ USEDFREE TYPE]
          (if [AND (LISTP TRAN)
                   [EQ 'FORWORD (CAR (GETPROP (CAR (LISTP X))
                                              'CLISPWORD]
                   [if (EQ 'PROG (CAR TRAN))
                       then (SETQ DPROGFLAG T)
                            (SETQ DFORM TRAN)
                     elseif (AND [EQ 'FUNCTION (CAR (SETQ DFORM (CAR (LISTP (CDR (LISTP (CDR TRAN]
                                 (EQ 'LAMBDA (CAR (SETQ DFORM (CAR (LISTP (CDR DFORM]
                   (OR (NULL NEWSATLIST)
                       (for D in (CDR (ASSOC 'DECLARE DFORM)) thereis (AND (LISTP D)
                                                                           (NOT (FMEMB (CAR D)
                                                                                       DECLARETAGS]
              then
              (FRPLACA DFORM (if DPROGFLAG
                                 then 'DPROG
                               else 'DLAMBDA))
              [for F (PROGVARS _ (AND DPROGFLAG (CADR DFORM))) in (CDDR DFORM) when (EQ (CAR F)
                                                                                      'DECLARE)
                 do (for D V DCLARE in (CDR F)
                       do (if (OR (NLISTP D)
                                  (FMEMB (CAR D)
                                         DECLARETAGS))
                              then (push DCLARE D)
                            elseif DPROGFLAG
                              then
```

(* Distribute declarations of local variables in the DPROG bindings.
This means that initial values will be taken into account)

```
                                   (for L on PROGVARS
                                      do (if (EQ (CAR D)
                                                 (CAR L))
                                             then [FRPLACA L (CONS (CAR L)
                                                                   (CONS NIL (CDR D]
                                                  (RETURN)
                                           elseif (AND (LISTP (CAR L))
                                                       (EQ (CAR D)
                                                           (CAAR L)))
                                                  then [if (NLISTP (CDAR L))
                                                          then (FRPLACD (CAR L)
                                                                         (LIST (CDAR L]
                                                                        (* In case it's a list with no CADR)
                                                       (NCONC (CAR L)
                                                              (CDR D))
                                                       (RETURN))
                                          finally (push DECL D))
                               elseif (EQ (CAR D)
                                          'RETURNS)
                                 then (push RETURNS D)
                               else (push DECL D))
                          finally (FRPLACD F (DREVERSE DCLARE]
               [if DECL
                   then (push (CDDR DFORM)
                              (CONS 'DECL (DREVERSE DECL]
               (RESETVARS (CLISPRETRANFLG)                              (* Resetting this flag avoids redundancies in a !DW)
                          (if RETURNS
                              then (SETQ TRAN (LIST 'DPROGN (DREVERSE RETURNS)
                                                     TRAN))
                                   (DWIMIFY0? TRAN TRAN NIL NIL NIL FAULTFN)
                                                                        (* Only happens for a MAPC/AR etc that has a RETURNS)
                                   (SETQ TRAN (PROG1 (CAR (CDDDDR (GETHASH TRAN CLISPARRAY)))
                                                      (PUTHASH TRAN NIL CLISPARRAY)))
                                                                        (* Skip down to the CHECKVALUE)

                            else (DWIMIFY0? DFORM DFORM NIL NIL NIL FAULTFN)))
               (if (PROG1 (SETQ DECL (GETHASH DFORM CLISPARRAY))
                          (PUTHASH DFORM NIL CLISPARRAY))
                   then                                                 (* Don't clobber DFORM with an empty translation, which
                                                                        probably comes from a lower-level error)
                        (FRPLNODE2 DFORM DECL)
                 else (SETQ TRAN NIL]
        (REALCLISPTRAN X TRAN])
```

(DECLMSG

```
  [NLAMBDA DECLMSG                                                      (* rmk%: "16-AUG-81 23:17")

            (* Purely for saving storage. For list declarations, the DECL argument of ASSERTFAULT and VALUEERROR is compiled
            as a call to DECLMSGMAC which has a macro that calls DECLMSG as a compiletime or loadtime constant.
            We attempt to find an already existing copy of that list-structure, and if we do, we return a pointer to that instead)

    (DECLARE (SPECVARS DECLMSG)
           (GLOBALVARS DECLTYPESARRAY DECLMESSAGES))
    (if (GETHASH DECLMSG DECLTYPESARRAY)
        then

            (* This never works when we're loading from a file, but always works when we are storing the compiled code directly into
            core.)

             DECLMSG
      else [foreachTB TB (if (EQUAL DECLMSG (fetch NAME of TB))
                              then (RETFROM 'DECLMSG (fetch NAME of TB]

            (* Fall through if didn't locate it. Look it up on our special message database list.)

            (CAR (OR (MEMBER DECLMSG DECLMESSAGES)
                     (push DECLMESSAGES DECLMSG])
```

(DECLDWIMERROR

```
  [LAMBDA ARGS                                                          (* bas%: "10-OCT-79 18:24")
    (DECLARE (USEDFREE FAULTFN))
    (LISPXTERPRI T)
    (LISPXPRIN1 "{in " T)
    (LISPXPRIN1 FAULTFN T)
    (LISPXPRIN1 "} " T)
    (for I to ARGS do (if (EQ T (ARG ARGS I))
                          then (LISPXTERPRI T)
                        else (LISPXPRIN1 (ARG ARGS I)
                                     T)))
    (LISPXTERPRI T)
    (ERROR!])
```

(DECLDWIMTESTFN

```
  [LAMBDA (TB)                                                          (* rmk%: " 6-FEB-82 14:26")
                                                                        (* Returns the dwimified TESTFN of TB)
    (DECLARE (USEDFREE FAULTFN))
    (PROG ((FN (fetch TESTFN of TB)))
          (if [AND (LISTP FN)
                   (OR CLISPRETRANFLG (NOT (GETHASH FN CLISPARRAY]
              then (DWIMIFY0? FN FN NIL NIL NIL FAULTFN)

            (* We hash the FN to itself to avoid repetitive dwimification unless CLISPRETRANFLG is on.
            But we're careful to avoid circularity if FN begins with a CLISPWORD.)
```

```
                  (OR (GETHASH FN CLISPARRAY)
                      (PUTHASH FN FN CLISPARRAY)))
         (RETURN FN])
```

(**DECLSET**

```
  [LAMBDA (VAR VAL)                                           (* rmk%: "11-NOV-83 08:00")

             (* Version of SET that does ASSERT checks. This is moved to SET when DECLTRAN is loaded.
             The old definition of SET is available through the name REALSET.
             Uses VARSETFN to find the run-time type-dependent SETFN, which will be that for the lowest declaration on the
             satisfieslist for a DPROGN)

    (DECLARE (LOCALVARS . T))
    (PROG1 (APPLY (VARSETFN VAR)
                  (LBOX VAR (KWOTEBOX VAL)))
           (\VARASRT VAR])
```

(**DECLSETQ**

```
  [NLAMBDA U                                                  (* rmk%: "11-NOV-83 08:00")

             (* Version of SETQ that does ASSERT checks. The old definition of SETQ is available through the name REALSETQ.
             The contortions are so DWIM gets to see the value forms in the environment of the running function.)

    (DECLARE (LOCALVARS . T))
    (WITH [(V (APPLY (FUNCTION PROG1)
                     (CDR U)
                     'INTERNAL]                               (* Bind the value so no recursion thru the LBOX)
          (PROG1 (APPLY (VARSETFN (CAR U))
                        (LBOX (CAR U)
                              (KWOTEBOX V))
                        'INTERNAL)
                 (\VARASRT (CAR U)))])
```

(**DECLSETQQ**

```
  [NLAMBDA (XSET YSET)                                        (* bas%: " 1-NOV-79 17:54")
    (APPLY* (FUNCTION DECLSETQ)
           XSET
           (KWOTEBOX YSET])
```

(**DECLTRAN**

```
  [LAMBDA (FORM)                                              (* DECLARATIONS%: FAST (RECORD FORM
                                                              (ATOM DCLS . FORMS)))
                                                              (* rmk%: " 2-Nov-84 15:24")
                                                              (* Translator for declarative statements)
    (DECLARE (USEDFREE VARS CLISPCHANGE))                     (* Used for DPROGN variable names)
    (SETQ CLISPCHANGE T)
    (PROG (TEMP CLISP%: DECLARE TOP BS PROGDCLS SPECVARS SAT INITVARS VARBINDFORMS RETURNS LOCALVARS
                (ATOM (fetch ATOM of FORM))
                (FORMS (fetch FORMS of FORM))
                (VARS VARS))
          (DECLARE (SPECVARS VARS DECLARE PROGDCLS SPECVARS SAT INITVARS RETURNS LOCALVARS VARBINDFORMS))
          (if (LISTP (SETQ TEMP (fetch DCLS of FORM)))
              then [for V in old TEMP do (if (AND (EQ ATOM 'DPROG)
                                                  (EQ V 'THEN))
                                             then [SETQ FORMS (LIST (LIST 'RETURN (CONS 'DPROG (CONS (CDR TEMP)
                                                                                                     FORMS]
                                                  (RETURN))
                                         (DECLVAR V (EQ ATOM 'DPROG)
                                                (NEQ ATOM 'DPROGN]
                   (SETQ PROGDCLS (DREVERSE PROGDCLS))
            else (if (AND TEMP (LITATOM TEMP)
                          (EQ ATOM 'DLAMBDA))
                     then (DECLVAR (LIST TEMP 'CARDINAL)
                                 NIL T)                       (* Handles no-spread case; not necessary to do \CHKINIT)
                          (SETQ INITVARS NIL))
                 (SETQ PROGDCLS TEMP))
          (if [AND (EQ ATOM 'DLAMBDA)
                   (OR (EQ [CAR (SETQ TEMP (LISTP (CAR FORMS]
                           'CLISP%:)
                       (AND (EQ (CAR TEMP)
                                COMMENTFLG)
                            (EQ (CADR TEMP)
                                'DECLARATIONS%:]
              then (SETQ CLISP%: TEMP)
                   (SETQ FORMS (CDR FORMS)))
          [if (NEQ ATOM 'DPROGN)
              then (for F DECL in old FORMS do (if (NLISTP F)
                                                   then (GO $$OUT)
                                                 elseif (EQ (CAR F)
                                                            COMMENTFLG)
                                                 elseif (EQ (CAR F)
                                                            'DECLARE)
                                                   then                   (* APPEND combines multiple declares)
                                                        (SETQ DECLARE (APPEND DECLARE (CDR F)))
                                                 elseif (EQ (CAR F)
                                                            'DECL)
                                                   then (SETQ DECL (APPEND DECL (CDR F)))
                                                 else (GO $$OUT))
                         finally [if (EQ ATOM 'DPROG)
                                     then

               (* This PROG represents the user's PROG, to which his RETURN and GO statements are referred.
               The PROG introduced below is for the actual bindings, and allows intervening checks for variables and RETURNS to be
               inserted.)

                                     (SETQ FORMS (LIST (CONS 'PROG (CONS NIL FORMS]
                             (if DECL
                                 then (SETQ FORMS (LIST (CONS 'DPROGN (CONS DECL FORMS]

               (* The test-functions don't appear in the code, so they have to be dwimified separately.
               This can't be done in MAKEDECLTYPE, because the variables in the testfn aren't known until this whole binding set has
               been processed to add them to VARS. -
               We don't have to worry about set and bind functions, cause they are attached only to named types and thus are dwimified
               when the type is defined.)

               [for V in SAT when (SETQ V (fetch VARDECL of V)) do (DECLDWIMTESTFN (OR (FINDDECLTYPE (fetch DECL
                                                                                                         of V))
                                                                                       (SHOULDNT]
               (if SPECVARS
                   then (push DECLARE (CONS 'SPECVARS SPECVARS)))
               (if LOCALVARS
                   then (push DECLARE (CONS 'LOCALVARS LOCALVARS)))
               [if RETURNS
                   then (SETQ FORMS (LIST (CONS 'the (CONS RETURNS FORMS]
               (if DECLARE
                   then (push DECLARE 'DECLARE))
               (SETQ BS (CONS 'PROGN (NCONC [if INITVARS
                                                then (LIST (CONS '\CHKINIT (DREVERSE INITVARS]
                                            FORMS)))
               [if VARBINDFORMS
                   then (FRPLACD BS (NCONC (DREVERSE VARBINDFORMS)
                                           (CDR BS]                (* VARBINDFORMS is hook for type-dependent initializations)
               (SELECTQ ATOM
                    ((DLAMBDA)                                     (* In parens to suppress PPDECL here)
                         (SETQ FORMS (LIST BS))
                         (if DECLARE
                             then (push FORMS DECLARE))
                         [push FORMS (CONS COMMENTFLG '(ASSERT%: (CLISP DLAMBDA]
                         (if CLISP%:
                             then (push FORMS CLISP%:))
                         (SETQ TOP (CONS 'LAMBDA (CONS PROGDCLS FORMS))))
                    ((DPROG)
                         (SETQ TOP (LIST 'PROG PROGDCLS (LIST 'RETURN BS)))
                         (if DECLARE
                             then (push (CDDR TOP)
                                        DECLARE)))
                    (SETQ TOP BS))                                 (* DPROGN falls through)
               (PROG (NEWSATLIST)                                  (* Lower decl's are not new.)
                     (DECLARE (SPECVARS NEWSATLIST))
                     (DWIMIFY0? TOP TOP NIL NIL NIL FAULTFN))
               (if (OR SAT NEWSATLIST)
                   then (FRPLACA BS '\*DECL)

               (* If no variables were declared, leave the PROGN that was to make sure that the forms got dwimified correctly)

                         (SETQ SAT (DREVERSE SAT))                 (* So satlist is ordered like decls)
                         (push (CDR BS)
                               (if (AND NEWSATLIST SAT)
                                   then (LIST SAT)
                                 else SAT)))                       (* We can do the extra CONS statically when this is a newsatlist)
               (RETURN (if (EQ ATOM 'DLAMBDA)
                           then TOP
                         else (REALCLISPTRAN FORM TOP)
                              FORM])
```

```
(DECLVAR
  [LAMBDA (VARD DPROGFLAG BINDFLAG)                                (* DECLARATIONS%: FAST)
                                                                   (* rmk%: " 2-Nov-84 15:33")
    (DECLARE (USEDFREE FAULTFN DECLARE RETURNS SAT INITVARS PROGDCLS LOCALVARS SPECVARS VARBINDFORMS VARS)
           (GLOBALVARS GLOBALVARS))
    (PROG (TYPEBLOCK DECL TEMP TESTFORM NAME INITV TAIL SATFORM (PROGNFLAG (NOT BINDFLAG)))
          (if (LISTP VARD)
              then (SETQ NAME (CAR VARD))
                   (SELECTQ NAME
                        ((RETURNS VALUE)
                             (if RETURNS
                                 then (DECLVARERROR "Multiple RETURNS/VALUE declaration"))
                             (SETQ DPROGFLAG (SETQ BINDFLAG NIL))
```

```
                    (SETQQ NAME VALUE))
               NIL)
          (SETQ TAIL (CDR VARD))
          (if DPROGFLAG
              then (RESETVARS ((NOSPELLFLG T)
                               (DWIMESSGAG T))
                              (DWIMIFY0? TAIL VARD TAIL NIL NIL FAULTFN))

(* This will glue all the components of the initial value together.
It will also walk through the declarations, but no spelling corrections will be done.
Corrections in the SATISFIES will happen when the whole translation is dwimified in DECLTRAN.)

                    (SETQ INITV (pop TAIL)))
  else (SETQ NAME VARD))
(if (NOT (AND NAME (LITATOM NAME)))
    then (DECLVARERROR "Illegal variable name"))
(for V in TAIL do                                    (* RETRY is a label)
                 RETRY
                 (if (if BINDFLAG
                         then [SELECTQ V
                                   (SPECIAL (if (FMEMB NAME LOCALVARS)
                                                then (DECLVARERROR "Variable can't be both LOCAL and
                                                           SPECIAL:  " NAME)
                                              else (push SPECVARS NAME)))
                                   (LOCAL (if (FMEMB NAME SPECVARS)
                                              then (DECLVARERROR "Variable can't be both LOCAL and
                                                         SPECIAL:  " NAME)
                                            else (push LOCALVARS NAME)))
                                   (if (EQ (CAR (LISTP V))
                                           'USEDIN)
                                       then (if (FMEMB NAME LOCALVARS)
                                                then (DECLVARERROR "Variable can't be both LOCAL and
                                                           USEDIN:  " NAME)
                                              else (push SPECVARS NAME]
                       elseif (EQ V 'GLOBAL)
                         then (pushnew GLOBALVARS NAME)
                       elseif (OR (EQ V 'FREE)
                                  (EQ (CAR (LISTP V))
                                      'BOUNDIN))
                         then (SETQ PROGNFLAG 'FREE))
                   elseif (EQ (CAR (LISTP V))
                              'SATISFIES)
                     then (if SATFORM
                              then (DECLVARERROR "Multiple SATISFIES"))
                          (SETQ SATFORM V)
                   elseif (EQ (CAR (LISTP V))
                              COMMENTFLG)
                   elseif (SETQ TEMP (GETDECLTYPE.NOERROR V NAME))
                     then (if TYPEBLOCK
                              then (DECLVARERROR "more than one type declaration:  " V))
                          (SETQ TYPEBLOCK TEMP)
                          (SETQ DECL V)
                   elseif (AND (LISTP V)
                               (FIXSPELL (CAR V)
                                         80
                                         '(SATISFIES BOUNDIN USEDIN)
                                         T V))
                     then (AND FAULTFN (NEQ FAULTFN 'TYPE-IN)
                               (MARKASCHANGED FAULTFN 'FNS))
                          (GO RETRY)
                   else (DECLVARERROR "invalid declaration: " V)))
(if (NULL TYPEBLOCK)
    then (SETQQ DECL ANY))
(if SATFORM
    then (SETQ DECL (LIST DECL SATFORM))
         (if (NULL (SETQ TYPEBLOCK (GETDECLTYPE.NOERROR DECL NAME)))
             then (DECLVARERROR "invalid declaration: " DECL)))
(if (EQ NAME 'VALUE)
    then (SETQ RETURNS DECL)                         (* This gets reprocessed by THETRAN)
         (RETURN))
(if BINDFLAG
    then (for D in PROGDCLS when [OR (EQ NAME D)
                                     (EQ NAME (CAR (LISTP D]
            do (DECLVARERROR "more than one binding for " NAME)))
                                                     (* TYPEBLOCK=NIL if default ANY with no SATISFIES)
[if TYPEBLOCK
    then (if (SETQ TEMP (GETTBPROP TYPEBLOCK 'DECLARESPEC))
             then (push DECLARE (SUBST NAME 'VAR TEMP)))
         (if (EQ (SETQ TEMP (fetch BINDFN of TYPEBLOCK))
                 (CONSTANT DefaultBindFn))
           elseif DPROGFLAG
             then [SETQ INITV (CONS TEMP (if INITV
                                             then (LIST INITV)
                                           else      (* Indicate that the initialization is not to be checked)
                                                (SETQ DPROGFLAG NIL]
           else (push VARBINDFORMS (LIST 'REALSETQ NAME (LIST TEMP NAME]
(if (NEQ DECL 'ANY)
              then                                                       (* A missing VARDECL is interpreted as ANY, so don't bother to
                   stick one in.)
                   (push SAT (create SLISTENTRY
                                    VARNAME _ NAME
                                    VARDECL _ (create VARDECL
                                                      DECL _ DECL
                                                      PROGNFLAG _ PROGNFLAG)))
                                                                         (* PROGNFLAG means that inherited declarations will be
                                                                         checked)
                   (if (if INITV
                           then DPROGFLAG
                         elseif (NULL DPROGFLAG))
                       then (push INITVARS NAME))
            elseif BINDFLAG
              then                                                       (* The empty VARDECL conceals type information for higher
                                                                         declarations)
                   (push SAT (create SLISTENTRY
                                    VARNAME _ NAME
                                    VARDECL _ NIL)))
          (if BINDFLAG
              then (push PROGDCLS (if INITV
                                      then (LIST NAME INITV)
                                    else NAME)))
          (push VARS NAME])
```

(**DLAMARGLIST**

```
  [LAMBDA (DEF)                                                          (* rmk%: " 6-APR-78 10:13")
    (if (LISTP (CADR DEF))
        then (for A in (CADR DEF) unless (EQ (CAR (LISTP A))
                                             'RETURNS)
                collect (if (LISTP A)
                            then (CAR A)
                          else A))
      else (CADR DEF])
```

(**DTYPE?TRAN**

```
  [LAMBDA (FORM)                                                         (* bas%: " 6-NOV-79 16:58")
    (SETQ CLISPCHANGE T)
    (if LCASEFLG
        then (/RPLACA FORM 'type?))
    [PROG (TESTFORM (TYPEBLOCK (GETDECLTYPE.NOERROR (CADR FORM)))
                (FORMS (CDDR FORM)))
          (if (NULL TYPEBLOCK)
              then (DECLDWIMERROR "invalid type declaration: " (CADR FORM)))
          (DWIMIFY0? FORMS FORM NIL NIL NIL FAULTFN)

          (* The forms are dwimified first so that we can decide whether the testform should be set-up for a bound VALUE.)

          (SETQ FORMS (if (CDR FORMS)
                          then (CONS 'PROGN FORMS)
                        else (CAR FORMS)))
          (SETQ TESTFORM (APPLYFORM (DECLDWIMTESTFN TYPEBLOCK)
                                FORMS))
          (REALCLISPTRAN FORM (if (NEQ TESTFORM T)
                                  then TESTFORM
                                elseif (LISTP FORMS)
                                  then (LIST 'PROGN FORMS T)
                                else                                     (* Cause PPT prints a non-list translation funny)
                                     '(PROGN T]
    FORM])
```

(**EDITNEWSATLIST**

```
  [LAMBDA NIL                                                            (* rmk%: " 7-SEP-81 03:31")

          (* Called from DW edit macro. True if there is no higher declarative on the current edit chain.)

    (DECLARE (USEDFREE L))
    (NOTANY (CDR L)
           (FUNCTION (LAMBDA (X)
                       (AND (LISTP X)
                            [OR (LITATOM (SETQ X (CAR X)))
                                (LITATOM (SETQ X (CAR X]
                            (OR (FMEMB X DECLATOMS)
                                (EQ (CAR (GETPROP X 'CLISPWORD))
                                    'FORWORD])
```

(**FORMUSESTB**

```
  [LAMBDA (FORM TRANS TB)                                                (* rmk%: " 9-NOV-83 09:24")

          (* Decides if FORM or its TRANSlation made use of the definition of the typeblock TB
          (Currently, T for any decl expression regardless of typeblock))

    (OR [AND (LISTP FORM)
             (FMEMB (CAR FORM)
                    '(type? TYPE? the THE DLAMBDA DPROG DPROGN]
        (AND (LISTP TRAN)
             (OR (EQ (CAR TRAN)
                     '\*DECL)
                 (AND (EQ [CAR (LISTP (GETP (CAR (LISTP FORM))
                                            'CLISPWORD]
                          'FORWORD)
                      (EQ [CAR (LISTP (SETQ TRAN (CAR (LAST TRAN]
                          'RETURN)
                      (EQ [CAR (LISTP (CAR (LISTP (CDR TRAN]
                          '\*DECL])
```

(IGNOREDECL

```
  [LAMBDA NIL                                            (* rmk%: " 4-APR-79 00:04")

            (* Should be called only in macros; T if the function currently being compiled should have debug information suppressed)

            (* FN is bound by COMPILE1 during ordinary compile, XXX during block compile.
            The LISTP check inhibits the EVALV, and is necessary when called from CHECKVALUEMAC inside masterscope.)

    (DECLARE (USEDFREE COMPILEIGNOREDECL))
    (OR (EQ COMPILEIGNOREDECL T)
        (AND (LISTP COMPILEIGNOREDECL)
             (MEMB (EVALV 'FN 'COMPILE1)
                   COMPILEIGNOREDECL)
             T])
```

(MAKETESTFORM

```
  [LAMBDA (VAR TYPE)                                     (* rmk%: "16-AUG-81 23:12")

            (* Makes a form that tests VAR to be of type TYPE and reports errors if test fails)

    (WITH ((TEST (APPLYFORM (fetch TESTFN of TYPE)
                     VAR)))
          (if (EQ TEST T)
              then (CBOX COMMENTFLG (LBOX (LBOX 'ASSERT VAR)))
            else (LIST 'OR TEST (LIST 'ASSERTFAULT (WITH ((TN (fetch NAME of TYPE)))
                                                       (if (LISTP TN)
                                                           then (CONS 'DECLMSGMAC TN)
                                                         else (KWOTE TN)))
                                   (KWOTE VAR])
```

(PPDECL

```
  [LAMBDA (FORM)                                         (* rmk%: "28-JUN-82 12:44" posted%: "17-MAY-77 22:06")
                                                         (* Special prettyprinter for DLAMBDA's and DPROG's.
                                                         Called from PRETTYPRINTMACROS)
    (DECLARE (GLOBALVARS %#RPARS CLISPARRAY PRETTYTRANFLG COMMENTFLG))
    (COND
       ((OR (NLISTP (CDR FORM))
            (AND PRETTYTRANFLG (GETHASH FORM CLISPARRAY)))
        FORM)
       (T (SELECTQ (CAR FORM)
              (DLAMBDA [PROG [(FORMPOS (IPLUS 2 (POSITION]
                             (PRIN1 (COND
                                       (%#RPARS "[")
                                       (T "(")))
                             (PRIN1 "DLAMBDA ")
                             (PPVARLIST (CADR FORM))
                             (COND
                                ((AND (LISTP (SETQ FORM (CDDR FORM)))
                                      (NEQ (CAR FORM)
                                           COMMENTFLG))
                                 (printout NIL .TAB0 FORMPOS)))
                             (PRINTDEF FORM FORMPOS T T FNSLST)
                             (PRIN1 (COND
                                       (%#RPARS "]")
                                       (T ")"])
              (DPROG (PROG [FORMPOS (LABELPOS (ADD1 (POSITION]   (* For DPROG's. Highlights the THEN's in the argument list and
                     formats initial values)
                         (SETQ FORMPOS (IPLUS LABELPOS 4))
                         (PRIN1 "(DPROG ")
                         [COND
                            ((LISTP (CADR FORM))
                             (PRIN1 "(")
                             [for V VTAIL (LASTLIST _ T)
                                 (VARPOS _ (IPLUS LABELPOS 7)) in (CADR FORM)
                                do (COND
                                      ((LISTP V)
                                       (printout NIL .TAB0 VARPOS "(" .P2 (CAR V))
                                       [COND
                                          ((SETQ VTAIL (CDR V))
                                           (SPACES 1)
                                           (for X in old VTAIL do (PRINTDEF X (POSITION)
                                                                              T NIL FNSLST)
                                                    repeatwhile (FMEMB (COND
                                                                        ((AND (LISTP X)
                                                                              (NLISTP (CADR VTAIL)))
                                                                         (NTHCHAR (CADR VTAIL)
                                                                                1))
                                                                        ((AND (NLISTP X)
                                                                              (LISTP (CADR VTAIL)))
                                                                         (NTHCHAR X -1)))
                                                                     CLISPCHARS)
                                                    finally (SETQ VTAIL (CDR VTAIL)))
                                                                   (* Supress spaces in clisp initial values)
                                                 (for X in VTAIL do (SPACES 1)
                                                                    (PRINTDEF X (POSITION)
                                                                           T NIL FNSLST]
                                             (COND
                                                ((ILESSP (POSITION)
                                                        VARPOS)
                                                 (TAB VARPOS)
                                                 (PRIN1 ")"))
                                                (T (PRIN3 ")")))
                                             (SETQ LASTLIST T))
                                            ((EQ V 'THEN)
                                             (printout NIL .TAB0 (IPLUS LABELPOS 2)
                                                    'THEN))
                                            (T (COND
                                                  (LASTLIST (TAB VARPOS 0))
                                                  (T (SPACES 1)))
                                               (SETQ LASTLIST NIL)
                                               (PRIN2 V]
                                   (PRIN3 ")"))
                                  (T (PRIN2 (CADR FORM]
                              [for F in (CDDR FORM) do (COND
                                                          ((LITATOM F)
                                                           (printout NIL .TAB LABELPOS .P2 F))
                                                          (T (COND
                                                                ((NEQ (CAR (LISTP F))
                                                                      COMMENTFLG)
                                                                 (printout NIL .TAB0 FORMPOS)))
                                                             (PRINTDEF F (POSITION)
                                                                    T NIL FNSLST]
                              (PRIN1 ")")))
                  (DECL (PRIN1 "(DECL ")
                        (PPVARLIST (CDR FORM)
                               T)
                        (PRIN3 ")"))
                  (DPROGN (PROG [(FORMPOS (IPLUS 3 (POSITION]
                                (PRIN1 "(DPROGN ")
                                (PPVARLIST (CADR FORM))
                                (COND
                                   ((AND (LISTP (SETQ FORM (CDDR FORM)))
                                         (NEQ (CAR FORM)
                                              COMMENTFLG))
                                    (printout NIL .TAB0 FORMPOS)))
                                (PRINTDEF FORM FORMPOS T T FNSLST)
                                (PRIN1 ")")))
                  NIL)
              NIL])
```

(PPVARLIST

```
  [LAMBDA (VLIST TAILFLG)                                              (* rmk%: "12-JUN-78 16:07")

             (* Pretty-prints the variable declarations for DLAMBDA, DPROGN, DECL.
             The list begins at the current line position; unless TAILFLG, enclosing parens are printed)

    (if (LISTP VLIST)
        then (OR TAILFLG (PRIN1 "("))
             (for V (VARPOS _ (POSITION))
                 (LASTLIST _ T) in VLIST do (if (LISTP V)
                                                then (printout NIL .TAB0 VARPOS "(" .P2 (CAR V))
                                                     (for X in (CDR V) do (SPACES 1)
                                                                          (PRINTDEF X (POSITION)
                                                                                 T NIL FNSLST))
                                                     (if (ILESSP (POSITION)
                                                                VARPOS)
                                                         then (TAB VARPOS)
                                                              (PRIN1 ")")
                                                       else (PRIN3 ")"))
                                                     (SETQ LASTLIST T)
                                              else (if LASTLIST
                                                       then (TAB VARPOS 0)
                                                     else (SPACES 1))
                                                   (SETQ LASTLIST NIL)
                                                   (PRIN2 V))
                 finally (if $$LST1
                             then (PRIN1 " . ")
                             (PRIN2 $$LST1)))
              (OR TAILFLG (PRIN3 ")"))
        else (PRIN2 VLIST])
```

```
(SETQMAC
  [LAMBDA (ARGS)                                              (* bas%: "18-OCT-79 18:22")
                                                              (* Compiler macro for SETQ. Enforces declarations.)

    (PROG [SETFORM (TB (TYPEBLOCKOF (CAR ARGS]
          (SETQ SETFORM (CONS (fetch SETFN of TB)
                              ARGS))

          (* We can suppress the run time test if either IGNOREDECLS, type is ANY, the value is a constant which passes the test fn
          now, or TB covers the possible set of values. Can't do constant evaluation if there's a setfn, cause a setfn clearly must have
          side-effects, and it may be doing coercions.)

          (RETURN (if [OR (IGNOREDECL)
                          (EQ (fetch TYPEXP of TB)
                              'ANY)
                          [AND (EQ (fetch SETFN of TB)
                                   (CONSTANT DefaultSetFn))
                               (DECLCONSTANTP (CADR ARGS))
                               (PROG (TEMP HELPFLAG (TST (fetch TESTFN of TB)))
                                     (DECLARE (SPECVARS HELPFLAG))
                                     (RETURN (AND (OR (SUBRP TST)
                                                      (NOT (FREEVARS TST)))
                                                  [NLSETQ (OR [SETQ TEMP (APPLY* TST (EVAL (CADR ARGS]
                                                              (COMPEM " Warning: Probable type fault in"
                                                                     (CONS 'SETQ ARGS]
                                                  TEMP]
                          (COVERSTB TB (TYPEBLOCKOF (if (EQ (fetch SETFN of TB)
                                                            (CONSTANT DefaultSetFn))
                                                        then SETFORM
                                                      else (CADR ARGS]
                      then                                    (* The variable's type includes the value's, so we're OK.)
                           SETFORM
                    else

          (* PROG1 is used rather than embedding the SETFORM in the test to give MAKEAPPLYFORM a better chance of
          simplifying)

                         (LIST 'PROG1 SETFORM (MAKETESTFORM (CAR ARGS)
                                                     TB])
```

```
(THETRAN
  [LAMBDA (FORM)                                              (* rmk%: " 9-NOV-83 09:17")
    (DECLARE (USEDFREE LCASEFLG CLISPCHANGE))
    (SETQ CLISPCHANGE T)
    (if LCASEFLG
        then (/RPLACA FORM 'the))
    [WITH [(TYPEBLOCK (GETDECLTYPE.NOERROR (CADR FORM]
          (if (NULL TYPEBLOCK)
              then (DECLDWIMERROR "invalid type declaration: " (CADR FORM)))
          (DWIMIFY0? (CDDR FORM)
                 FORM
                 (CDDR FORM)
                 NIL NIL FAULTFN)
          (WITH [(TESTFORM (APPLYFORM (DECLDWIMTESTFN TYPEBLOCK)
                                  'VALUE))
                 (VALFORM (if (CDDDR FORM)
                              then (CONS 'PROGN (CDDR FORM))
                            else (CADDR FORM]
                (REALCLISPTRAN
                 FORM
                 (if (EQ TESTFORM T)
                     then VALFORM
                   else (LIST '\CHKVAL (APPLYFORM [LAMVAL (LIST 'COND (LIST TESTFORM 'VALUE)
                                                              (LIST T (LIST 'VALUEERROR 'VALUE
                                                                            (if (LISTP (CADR FORM))
                                                                                then (CONS 'DECLMSGMAC
                                                                                           (CADR FORM))
                                                                              else (KWOTE (CADR FORM]
                                                  VALFORM]
    FORM])
```

```
(VALUEERROR
  [LAMBDA (VALUE DECL)                                        (* rmk%: "16-AUG-81 15:48")
    (DECLARE (SPECVARS VALUE))
    (LISPXPRIN1 "
        VALUE ASSERTION NOT SATISFIED IN " T)
    (bind POS when [LITATOM (STKNAME (SETQ POS (REALSTKNTH -1 (OR POS 'VALUEERROR)
                                                     NIL POS]
       do (LISPXPRIN2 (STKNAME POS)
                 T)
          (RELSTK POS)
          (RETURN))

          (* VALUE is the break expression so that an OK will simply return it.
          Also, typing the command VALUE in the break will cause VALUE to be printed out, given the EVAL command that sets it up.
          There are some paradoxes though%: If the user sets VALUE, he will not see the change in the break unless he does
          another EVAL. Instead, he must work with !VALUE.)

    (APPLY* (FUNCTION BREAK1)
           'VALUE T (LIST 'VALUE DECL)
           '(EVAL])

(\VARASRT
  [LAMBDA (VARNAME)                                          (* rmk%: " 2-DEC-78 14:47")
                                                             (* Checks all the declaration predicates for VARNAME in the
                                                             run-time context.)
    (DECLARE (LOCALVARS . T)
           (USEDFREE SATISFIESLIST))
    (VARASRT1 VARNAME SATISFIESLIST])

(VARASRT1
  [LAMBDA (VARNAME SLIST)                                    (* bas%: " 9-OCT-79 23:24")

          (* Checks all run-time assertions for VARNAME. Evaluates the highest predicate in the current scope first for DPROGN
          variables.)

    (DECLARE (LOCALVARS . T))
    (for S D in old SLIST when (SETQ D (ASSOC VARNAME S))
       do (if (NULL (SETQ D (fetch VARDECL of D)))
              then (RETURN))
          (if (fetch PROGNFLAG of D)
              then (VARASRT1 VARNAME (CDR SLIST)))
          (if (APPLY* (fetch TESTFN of (GETDECLTYPE (fetch DECL of D)
                                                VARNAME))
                     (EVALV VARNAME))
              then (RETURN))
          (ASSERTFAULT (fetch DECL of D)
                 VARNAME])

(VARSETFN
  [LAMBDA (VARNAME)                                          (* rmk%: " 2-Nov-84 15:05")

          (* Called by DECLSET and returns the setfn for VARNAME, or NIL if there isn't one.
          The setfn is the lowest one found on a DPROGN chain. Should be equivalent to
          (fetch SETFN of (VARDECL VARNAME T))%, but is opencoded to avoid consing up the type each time.)

    (DECLARE (USEDFREE SATISFIESLIST))
    (for S TEMP D in SATISFIESLIST when (SETQ D (ASSOC VARNAME S))
       do (RETURN (fetch SETFN of (GETDECLTYPE (if (NULL (SETQ D (fetch VARDECL of D)))
                                                    then 'ANY
                                                  else (fetch DECL of D))
                                             VARNAME)))
       finally (RETURN (CONSTANT DefaultSetFn])
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: DECLTRAN DECLTRAN DECLVAR)

(BLOCK%: PPDECL PPDECL PPVARLIST)

(BLOCK%: \VARASRT \VARASRT VARASRT1)
)


          (* * Declaration database fns)

(DEFINEQ

(DECLOF
  [LAMBDA (FORM DECLCONTEXT)                                 (* bas%: "31-JUL-79 13:40")

          (* Returns a declaration for FORM in the context maintained by the code reading system DECLCONTEXT)

    (DECLARE (USEDFREE CSATISFIESLIST SATISFIESLIST DECLVARSLST))
    (fetch NAME of (TBOF FORM (SELECTQ DECLCONTEXT
                                  (COMPILER CSATISFIESLIST)
                                  (INTERPRETER SATISFIESLIST)
                                  (NIL (if (BOUNDP DECLVARSLST)
                                           then DECLVARSLST
                                         else CSATISFIESLIST))
                                  (ERRORX (LIST 27 DECLCONTEXT])
(DECLOF1
  [LAMBDA (FORM)                                                  (* rmk%: " 9-NOV-83 09:24")

                (* Computes a declaration form for FORM. May be redundant as it will be checked in DECLOF.)

    (if (LITATOM FORM)
        then (SELECTQ FORM
                 (NIL NIL)
                 (T '(MEMQ T))
                 (VARDECL FORM))
      elseif (LISTP FORM)
        then
        [PROG (TEMP)
              (RETURN
               (if (LITATOM (CAR FORM))
                   then
                   [OR
                    (if (AND (EQ [CAR (LISTP (SETQ TEMP (GETP (CAR FORM)
                                                              'DECLOF]
                                 'FUNCTION)
                             (NEQ (CAR (LISTP (CDR TEMP)))
                                  'SATISFIES))
                        then (APPLY* (CADR TEMP)
                                     FORM)
                      else TEMP)
                    (SELECTQ (CAR FORM)
                        ((SETQ SETQQ)
                             [PROG [VSF (VD (VARDECL (CADR FORM]
                                   (SETQ VSF (fetch SETFN of (GETDECLTYPE VD)))
                                   (RETURN (if (EQ VSF (CONSTANT DefaultSetFn))
                                               then [LIST 'ALLOF VD (if (EQ (CAR FORM)
                                                                            'SETQQ)
                                                                        then (LIST 'MEMQ (CADDR FORM))
                                                                      else (DECLOF1 (CADDR FORM]
                                             else (DECLOF1 (CONS VSF (CDR FORM])
                        (PROG

                (* Declaration is known only if the first and only executable statement in the prog is a RETURN)

                              [for TAIL TEMP on (CDDR FORM) suchthat (SELECTQ (SETQ TEMP (CAAR TAIL))
                                                                         ((ASSERT DECLARE)
                                                                              NIL)
                                                                         (NEQ TEMP COMMENTFLG))
                                 finally (RETURN (if (AND (EQ TEMP 'RETURN)
                                                          (NULL (CDR TAIL)))
                                                     then (DECLOF1 (CADAR TAIL))
                                                   else 'ANY])
                        (PROGN (DECLOF1 (CAR (LAST FORM))))
                        (COND [for CL D TFLAG in (CDR FORM)
                                 unless (if (EQ [SETQ D (DECLOF1 (CAR (LAST CL]
                                                'ANY)
                                            then (RETURN 'ANY)
                                          else (if (EQ (CAR CL)
                                                       T)
                                                   then (SETQ TFLAG T))
                                               (MEMBER D $$VAL))
                                 collect D finally (if (NOT (OR TFLAG (FMEMB NIL $$VAL)))
                                                       then (SETQ $$VAL (NCONC1 $$VAL NIL)))
                                                   (RETURN (if (CDR $$VAL)
                                                               then (CONS 'ONEOF $$VAL)
                                                             else (CAR $$VAL])
                        (SELECTQ [for TAIL D on (CDDR FORM)
                                    unless (if (EQ [SETQ D (DECLOF1 (if (CDR TAIL)
                                                                        then (CAR (LAST (CDAR TAIL)))
                                                                      else (CAR TAIL]
                                                   'ANY)
                                               then (RETURN 'ANY)
                                             else (MEMBER D $$VAL))
                                    collect D finally (RETURN (if (CDR $$VAL)
                                                                  then (CONS 'ONEOF $$VAL)
                                                                else (CAR $$VAL])
                        ((REPLACEFIELD FREPLACEFIELD /REPLACEFIELD)
                             (DECLOF1 (CADDDR FORM)))
                        (REALSETQ (DECLOF1 (CADDR FORM)))
                        ((FETCHFIELD FFETCHFIELD)
                             (if (FIXP (CADR FORM))
                                 then (SELECTQ (LRSH (LOGAND (CADR FORM)
                                                             12582912)
                                                     22)
                                          (1 'FIXP)
                                          (2 'FLOATP)
                                          (3                           (* FLAG)
                                             '(MEMQ NIL T))
                                          (PROGN                       (* 0=pointer)
                                                 'ANY))
                               else 'ANY))
                        (REPLACEFIELDVAL
                                (DECLOF1 (CADDR FORM)))
                            (PROG1 (DECLOF1 (CADR FORM)))
                            (\*DECL [PROG [(DECLVARSLST (if [OR (NULL (CADR FORM))
                                                                (LISTP (CAR (CAADR FORM]
                                                            then (CADR FORM)
                                                          else (CONS (CADR FORM)
                                                                     DECLVARSLST]
                                                                     (* Maintain proper DECLVARSLST for recursion)
                                        (DECLARE (SPECVARS DECLVARSLST))
                                        (RETURN (DECLOF1 (CAR (LAST (CDDR FORM])
                            ((the THE)
                                (CADR FORM))
                            ((create CREATE)
                                (CADR FORM))
                            (QUOTE                                   (* Could be done in the constant eval, but here for efficiency
                                                                     cause very common)
                                  (LIST 'MEMQ (CADR FORM)))
                            (if (AND (NEQ FORM (SETQ TEMP (EXPANDMACRO FORM T)))
                                     (NEQ TEMP 'IGNOREMACRO))
                                then (DECLOF1 TEMP)
                              else (if [SETQ TEMP (OR (GETHASH FORM CLISPARRAY)
                                                      (AND (GETP (CAR FORM)
                                                                 'CLISPWORD)
                                                           (RESETVARS (FILEPKGFLG (NOSPELLFLG T)
                                                                                  (DWIMESSGAG T))
                                                                      (DWIMIFY0? FORM FORM)
                                                                      (RETURN (GETHASH FORM CLISPARRAY]
                                       then (DECLOF1 TEMP)
                                     elseif (DECLCONSTANTP FORM)
                                       then (LIST 'MEMQ (EVAL FORM))
                                     else 'ANY]
                  elseif [AND (LISTP (CAR FORM))
                              (SETQ TEMP (SELECTQ (CAAR FORM)
                                                  ([LAMBDA NLAMBDA]
                                                       (CAR (LAST (CDDAR FORM))))
                                                  (PROGN             (* Hope it's a translated LAMBDAWORD)
                                                         (GETHASH (CAR FORM)
                                                                  CLISPARRAY]
                    then (DECLOF1 TEMP)
                  else 'ANY]
      else (LIST 'MEMQ FORM])
```

(TBOF

```
  [LAMBDA (FORM DECLVARSLST)                                         (* bas%: " 9-OCT-79 23:27")
                                                                     (* Returns a type block for the value of form)
    (DECLARE (SPECVARS DECLVARSLST))                                 (* DECLVARSLST is SPECIAL for an eventual call on
                                                                     VARDECL on an atom.)
    (GETDECLTYPE (DECLOF1 FORM])
```

(TYPEBLOCKOF

```
  [LAMBDA (FORM)                                                     (* bas%: "31-JUL-79 13:40")
                                                                     (* Gets type block for compiler declaration of FORM)
    (DECLARE (USEDFREE CSATISFIESLIST))
    (TBOF FORM CSATISFIESLIST])
```

(VARDECL

```
  [LAMBDA (VARNAME)                                                  (* bas%: "30-JUL-79 18:07")

          (* Returns the declaration for VARNAME. The declaration will include all inherited attributes for DPROGN variables.)

    (DECLARE (USEDFREE DECLVARSLST))
    (for S ONE DECLS D in DECLVARSLST when (SETQ D (fetch VARDECL of (ASSOC VARNAME S)))
       do (if ONE
              then (SETQ DECLS (LIST (fetch DECL of D)
                                     ONE))                           (* ONE is to avoid the cons in the common single-test case)
                   (SETQ ONE NIL)
            elseif DECLS
              then (push DECLS (fetch DECL of D))
            else (SETQ ONE (fetch DECL of D)))
          (if (NOT (fetch PROGNFLAG of D))
              then (GO $$OUT))
       finally (RETURN (if DECLS
                           then (CONS 'ALLOF (DREVERSE DECLS))
                         elseif ONE
                         else 'ANY])
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: DECLOFBLK DECLOF DECLOF1 TBOF TYPEBLOCKOF VARDECL (ENTRIES DECLOF TYPEBLOCKOF))
)
```

```
          (* * Enabling and disabling fns)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

[ACCESSFNS FNEQUIVS ((DECLFN (PACK* 'DECL DATUM))
                     (REALFN (PACK* 'REAL DATUM]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS MOVEPROP MACRO ((PROP FROM TO)
                          (PUTIFPROP TO PROP (GETPROP FROM PROP))))

(PUTPROPS PUTIFPROP MACRO [(ATM PROP VAL)
                           (WITH ((V VAL))
                                 (COND
                                    (V (PUTPROP ATM PROP V))
                                    (T (REMPROP ATM PROP)
                                       NIL])
)
)

(DEFINEQ

(STARTDECLS
  [LAMBDA NIL                                            (* rmk%: "12-Mar-85 09:15")

          (* Repository of various code that sets up dummy function defns and other things that would require P commands in the file
          coms. Distinct from DODECLS which actually activates DECLs.)

    (for I in '(DECL WHOSE) do (MOVD? 'QUOTE I))
    (for I in '(\CHKVAL \DECLPROGN) do (MOVD? 'PROGN I))
    [for I in '(CHANGERECORD CLISPTRAN SETQ SET SETQQ) do (AND (MOVD? I (fetch REALFN of I))
                                                               (BOUNDP 'SYSLINKEDFNS)
                                                               (BOUNDP 'LINKEDFNS)
                                                               (FMEMB I SYSLINKEDFNS)
                                                               (push LINKEDFNS (fetch REALFN of I]
    (for P in MACROPROPS do (MOVEPROP P 'SETQ 'REALSETQ))
    (for P in MACROPROPS do (MOVEPROP P 'SET 'REALSET))
    (if (AND (BOUNDP 'DECLTYPESARRAY)
             (EQ (ASKUSER DWIMWAIT 'N "Reinitialize DECLTYPE lattice?  ")
                 'N))
      else (INITDECLTYPES))
    (for I in '((COVERS CALL (IF (EQ (CAR EXPR)
                                     'QUOTE)
                                 [NIL (@ (TYPEMSANAL COVERS)
                                         '((|..| TYPE]
                                 EVAL)
                         (IF (EQ (CAR EXPR)
                                 'QUOTE)
                             [NIL (@ (TYPEMSANAL COVERS)
                                     '((|..| TYPE]
                             EVAL) . PPE)
                (SELCOVERSQ . MACRO)
                (SELTYPEQ . MACRO)
                (\*DECL NIL [IF NULL NIL (IF (LISTP (CAAR EXPR))
                                             [(|..| (@ (TYPEMSANAL \*DECL)
                                                       '((|..| TYPE)
                                                         TEST]
                                             (|..| (@ (TYPEMSANAL \*DECL)
                                                      '((|..| TYPE)
                                                        TEST]
                        |..| EFFECT RETURN)
                (\CHKINIT NIL)
                (\CHKVAL NIL EVAL)
                (THE @ (TYPEMSANAL the)
                     '(CLISP (|..| TYPE)
                             RETURN))
                (TYPE? @ (TYPEMSANAL type?)
                       '(CLISP (|..| TYPE)
                               RETURN))
                (the @ (TYPEMSANAL the)
                     '(CLISP (|..| TYPE)
                             RETURN))
                (type? @ (TYPEMSANAL type?)
                       '(CLISP (|..| TYPE)
                               RETURN))
                (VALUEERROR NIL))
       do (PUTHASH (CAR I)
                   (CDR I)
                   MSTEMPLATES))
    (DODECLS T])

(DODECLS
  [LAMBDA (FLG)                                                      (* DECLARATIONS%: (RECORD DSF
                                                                     (ATM FN . PRPLST)))
    (DECLARE (USEDFREE COMPILEIGNOREDECL))                           (* rmk%: "12-Mar-85 09:07")

            (* Turns decls on if FLG; off if not. If turning on when they are currently off, then the old values are saved in a private cons
            so they can be restored if DECLS are turned off.)

    (SETQ COMPILEIGNOREDECL (NOT FLG))                               (* Reset the compile switch)
    (WITH [[DECLSETFROM '((CHANGERECORD T)
                          (CLISPTRAN T)
                          (SET T)
                          (SETQ T BYTEMACRO NIL MACRO (ARGS (SETQMAC ARGS)))
                          (SETQQ T)
                          (TYPE? NIL CLISPWORD (DTYPE?TRAN . type?))
                          (type? NIL CLISPWORD (DTYPE?TRAN . type?]
            (DECLUNSAVELST (CONSTANT (LIST NIL]
          [if (AND FLG (NOT (CAR DECLUNSAVELST)))
              then                                                   (* Collect the values to be restored)
                   (RPLACA DECLUNSAVELST (for F in DECLSETFROM
                                            collect (create DSF
                                                           ATM _ (fetch ATM of F)
                                                           FN _ (GETD (fetch ATM of F))
                                                           PRPLST _
                                                           (for J on (fetch PRPLST of F) by (CDDR J)
                                                              join (LIST (CAR J)
                                                                         (GETPROP (fetch ATM of F)
                                                                                (CAR J]
          [for F in (if FLG
                        then DECLSETFROM
                      else (CAR DECLUNSAVELST))
             do [WITH ((DEF (fetch FN of F)))
                      (AND DEF (PUTD (fetch ATM of F)
                                     (if (AND FLG (EQ DEF T))
                                         then (GETD (fetch DECLFN of (fetch ATM of F)))
                                       else DEF]
                (for J on (fetch PRPLST of F) by (CDDR J) do (PUTIFPROP (fetch ATM of F)
                                                                        (CAR J)
                                                                        (CADR J]
          (if FLG
            else (RPLACA DECLUNSAVELST NIL)                          (* Nothing saved anymore)))
    FLG])
)

(FILESLOAD (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
       LAMBDATRAN)

(DECLARE%: EVAL@COMPILE

(FILESLOAD (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
       SIMPLIFY)
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DEFINEQ

(IBOX
  [LAMBDA (IVAL)                                                     (* edited%: "29-Jan-85 17:28")
                                                                     (* If needed, give field the initial value defined in the record)
    (create IBOX
           I _ (OR IVAL 0])

(FBOX
  [LAMBDA (FVAL)                                                     (* rmk%: "23-SEP-77 09:39")
    (create FBOX
           F _ (OR FVAL 0.0])

(NBOX
  [LAMBDA (NVAL)                                                     (* rmk%: "10-OCT-77 10:17")

            (* A boxing function for numbers of unknown type. Since most functions that produce unknown-typed numbers compile
            closed and box internally, this is really useful only to copy boxes produced by those functions into new boxes at setq's.
            E.g. (SETQ X (NBOX Y))%, where previously there was (SETQ Y
            (DIFFERENCE A B)))

    (if (FLOATP NVAL)
        then (create FBOX
                    F _ NVAL)
      else (create IBOX
                  I _ NVAL])
)

(MOVD? 'LIST 'LBOX)
```

```
(MOVD? 'CONS 'CBOX)

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD FBOX ((F FLOATING))
       (CREATE (SELECTQ (SYSTEMTYPE)
                   ((TENEX TOPS-20)
                        (FPLUS 0.0))
                   (D (\CREATECELL (CONSTANT \FLOATP)))
                   (HELP "FBOX CREATE NOT DEFINED FOR SYSTEMTYPE " (SYSTEMTYPE]

[BLOCKRECORD IBOX ((I INTEGER))
       (CREATE (SELECTQ (SYSTEMTYPE)
                   ((TENEX TOPS-20)
                        (IPLUS 100000))
                   (D (\CREATECELL (CONSTANT \FIXP)))
                   (HELP "IBOX CREATE NOT DEFINED FOR SYSTEMTYPE " (SYSTEMTYPE]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS IBOX DMACRO [ARGS (COND
                                (ARGS (APPEND '(create IBOX smashing (LOADTIMECONSTANT (\CREATECELL (CONSTANT
                                                                                                   \FIXP)))
                                                                          I _)
                                             ARGS))
                                (T '(LOADTIMECONSTANT (\CREATECELL (CONSTANT \FIXP])

(PUTPROPS FBOX DMACRO [ARGS (COND
                                (ARGS (APPEND '(create FBOX smashing (LOADTIMECONSTANT (\CREATECELL (CONSTANT
                                                                                                   \FLOATP)))
                                                                          F _)
                                             ARGS))
                                (T '(LOADTIMECONSTANT (\CREATECELL (CONSTANT \FLOATP])

(PUTPROPS NBOX DMACRO [OPENLAMBDA (NVAL)
                           (COND
                              ((FLOATP NVAL)
                               (FBOX NVAL))
                              (T (IBOX NVAL])
)

(DECLARE%: EVAL@COMPILE

(PROGN (PUTPROPS CBOX MACRO ((X Y)
                              (FRPLNODE (CONSTANT (CONS))
                                     X Y)))
       (PUTPROPS CBOX DMACRO (= . CONS)))

(PROGN (PUTPROPS LBOX MACRO [ARGLIST (PROG (NILIST (FORM '$X$))
                                           [MAP ARGLIST (FUNCTION (LAMBDA (ARG)
                                                                     (SETQ NILIST (CONS NIL NILIST))
                                                                     (SETQ FORM (LIST 'FRPLACA FORM
                                                                                      (CAR ARG)))
                                                                     (AND (CDR ARG)
                                                                          (SETQ FORM (LIST 'CDR FORM]
                                           (RETURN (LIST (LIST 'LAMBDA '($X$)
                                                               '(DECLARE (LOCALVARS $X$))
                                                               FORM
                                                               '$X$)
                                                         (KWOTE NILIST])
       (PUTPROPS LBOX DMACRO (= . LIST)))
)

(DECLARE%: EVAL@COMPILE

[I.S.OPR 'scratchcollect '(SETQ $$SCPTR (FRPLACA [OR (CDR $$SCPTR)
                                                  (CDR (FRPLACD $$SCPTR (CAR (FRPLACA $$SCCONS (CONS]
                                            BODY))
       '(BIND $$SCPTR $$SCCONS _ (CONSTANT (CONS)) FIRST (SETQ $$SCPTR $$SCCONS)
           FINALLY (SETQ $$VAL (AND (NEQ $$SCPTR $$SCCONS)
                                    (PROG1 (CDR $$SCCONS)
                                        [COND
                                           ((CDR $$SCPTR)
                                            (FRPLACD $$SCCONS (PROG1 (CDR $$SCPTR)
                                                                  (FRPLACD $$SCPTR NIL)
                                                                  (FRPLACD (PROG1 (CAR $$SCCONS)
                                                                                  (FRPLACA $$SCCONS $$SCPTR))
                                                                         (CDR $$SCCONS)))])]
)

(ADDTOVAR SYSLOCALVARS $$SCCONS $$SCPTR)

(ADDTOVAR INVISIBLEVARS $$SCCONS $$SCPTR)

(DECLARE%: EVAL@COMPILE
(PUTPROPS WITH MACRO [ARGS (CONS (CONS 'LAMBDA (CONS [for I in (CAR ARGS) collect (COND
                                                                                        ((LITATOM I)
                                                                                         I)
                                                                                        ((LISTP I)
                                                                                         (CAR I))
                                                                                        (T (ERROR "Invalid WITH form
                                                                                                  binding" I]
                                                               (CDR ARGS)))
                                  (for I in (CAR ARGS) collect (CADR (LISTP I])
)

(SETTEMPLATE 'WITH '((BOTH (|..| (IF LISTP (NIL EVAL |..| EFFECT)
                                     NIL))
                           (|..| (IF LISTP (BIND EVAL |..| EFFECT) BIND)))
                     |..| EFFECT RETURN))

(REMPROP 'WITH 'CLISPWORD)

(ADDTOVAR DWIMEQUIVLST (WITH . PROG))

(ADDTOVAR PRETTYEQUIVLST (WITH . PROG))
)

(DECLARE%: DOCOPY

(DECLARE%: EVAL@LOADWHEN

(NEQ (SYSTEMTYPE)
     'D)

[OR (GETPROP 'LOADTIMECONSTANT 'FILEDATES)
    (PROG ((X (FINDFILE (PACKFILENAME 'NAME 'LOADTIMECONSTANT 'EXTENSION COMPILE.EXT)
                     T LISPUSERSDIRECTORIES)))
          (COND
             (X (LOAD X 'SYSLOAD))
             ((NOT (GETPROP 'LOADTIMECONSTANT 'MACRO))
              (PUTPROP 'LOADTIMECONSTANT 'MACRO '((FORM)
                                                  (CONSTANT FORM]
)
)

(ADDTOVAR OPENFNS \DECLPROGN \CHKVAL \CHKINIT ASSERT \*DECL \VARASRT)

(PUTPROPS DPROG CLISPWORD (DECLTRAN . DPROG))

(PUTPROPS DPROGN CLISPWORD (DECLTRAN . DPROGN))

(PUTPROPS THE CLISPWORD (THETRAN . the))

(PUTPROPS the CLISPWORD (THETRAN . the))

(PUTPROPS DLAMBDA INFO BINDS)

(PUTPROPS DPROG INFO (BINDS LABELS))

(PUTPROPS DPROGN INFO EVAL)

(RPAQQ SATISFIESLIST NIL)

(RPAQQ CSATISFIESLIST NIL)

(RPAQQ NEWSATLIST T)

(RPAQ? DECLMESSAGES )

(RPAQ? COMPILEIGNOREDECL )

(ADDTOVAR DECLATOMS DLAMBDA DPROG DPROGN)

(ADDTOVAR LAMBDASPLST DLAMBDA)

(ADDTOVAR SYSLOCALVARS VALUE)

(ADDTOVAR DESCRIBELST ["types:    " (GETRELATION FN '(USE TYPE])

(ADDTOVAR BAKTRACELST (\DECLPROGN (DPROGN APPLY *PROG*LAM \*DECL *ENV*)
                             (NIL APPLY *PROG*LAM \*DECL))
                      (PROG (DPROG \DECLPROGN APPLY *PROG*LAM \*DECL)))

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SLISTENTRY (VARNAME . VARDECL))

(RECORD VARDECL (DECL . PROGNFLAG))
```

```
)
)

(ADDTOVAR LAMBDATRANFNS (DLAMBDA DECLTRAN EXPR DLAMARGLIST))

(DECLARE%: DONTEVAL@LOAD

(ADDTOVAR PRETTYPRINTMACROS (DPROGN . PPDECL)
                            (DECL . PPDECL)
                            (DLAMBDA . PPDECL)
                            (DPROG . PPDECL))
)

(PUTPROPS ASSERT INFO EVAL)

(DECLARE%: EVAL@COMPILE

(PUTPROPS ASSERT MACRO (ARGS (ASSERTMAC ARGS)))

[PROGN (PUTPROPS .CBIND. BYTEMACRO [APPLY (LAMBDA (PV BODY)
                                            (APPLY* 'PROG PV '(RETURN (COMP.EXP1 BODY])
       (PUTPROPS .CBIND. MACRO (X (HELP "Compiler dependent macro must be supplied for .CBIND.")))]

(PUTPROPS \CHKINIT MACRO (ARGS (CHKINITMAC ARGS)))

(PUTPROPS \CHKVAL MACRO [ARGS (COND
                                [(IGNOREDECL)
                                 (COND
                                    ((EQ (CAAR ARGS)
                                         'COND)
                                     (CADADR (CAR ARGS)))
                                    (T (CADAR ARGS]
                                (T (CAR ARGS])

(PUTPROPS \*DECL MACRO (ARGS (*DECLMAC ARGS)))

(PUTPROPS DECL MACRO (X (COMPEM "DECL in illegal location" (CONS 'DECL X))))

[PROGN (PUTPROPS DECLMSGMAC DMACRO ((X . Y)
                                     (CONSTANT (DECLMSG X . Y))))
       (PUTPROPS DECLMSGMAC MACRO ((X . Y)
                                    (LOADTIMECONSTANT (DECLMSG X . Y))))]

(PROGN (DEFMACRO REALSETQ (X &REST CL:REST)
           (CONS 'CL:SETQ (CONS X CL:REST)))
       (PUTPROPS REALSETQ BYTEMACRO COMP.SETQ))
)


          (* * MACROS REALSET)

(AND (GETD 'STARTDECLS)
     (STARTDECLS))

[PROG [(COM (CDR (ASSOC 'DW EDITMACROS]
      (AND COM (RPLACD COM (CONS (APPEND '(RESETVAR NEWSATLIST (EDITNEWSATLIST))
                                         (CDR COM]


          (* * Builtin DECLOF properties)

(PUTPROPS APPEND DECLOF LST)

(PUTPROPS CONS DECLOF LISTP)

(PUTPROPS EQ DECLOF (MEMQ T NIL))

(PUTPROPS LIST DECLOF [FUNCTION (LAMBDA (FORM)
                                  (AND (CDR FORM)
                                       'LISTP])

(PUTPROPS LISTP DECLOF LST)

(PUTPROPS NCONC DECLOF LST)

(DECLARE%: EVAL@COMPILE DONTCOPY

(RESETSAVE DWIMIFYCOMPFLG NIL)

[AND (GETD 'DODECLS)
     (RESETSAVE (DODECLS)
            '(DODECLS T]
)

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

(ADDTOVAR NLAMA DECLSETQ DECLMSG DD \CHKINIT \*DECL ASSERT DECLTYPES DECLTYPE)
(ADDTOVAR NLAML DECLSETQQ TYPEMSANAL)

(ADDTOVAR LAMA DECLDWIMERROR)
)

(PUTPROPS DECL COPYRIGHT ("Xerox Corporation" 1983 1984 1985 1987))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

PROPERTY INDEX

RECORD INDEX

TEMPLATE INDEX

I.S.OPR INDEX