

File created: 19-Jan-93 11:15:39 {DSK}<python>lde>lispcore>sources>RECORD.;3

previous date: 5-Jan-93 02:03:38 {DSK}<python>lde>lispcore>sources>RECORD.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;
;; Copyright (c) 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1990, 1993 by Venue & Xerox Corporation. All rights reserved.

(RPAQQ RECORDCOMS

```
[ (FNS RECORDTRAN RECREDECLARE RECREDECLARE1 RECREDECLARE2 RECORDECL RECORDFIELD? RECORDECL0 RECORDECL1
RECORDECLBLOCK RECORDECLTAIL CHECKRECORDNAME LISTRECORDEFS RECORD.REMOVE.COMMENTS DECLARERECORD
DECLSUBFIELD UNCLISPTRAN RECDEC? ALLOHASH GETSETQ RECORDACCESS RECORDFIELDNAMES RECEVAL FIELDLOOK
SIMPLEP RECORDBINDVAL RECORDPRIORITY RECORDACCESSFORM)
(FNS RECORDWORD MAKECREATE0 MAKECREATE1 CREATEFIELDS REBINDP CSUBST RECONS COPY1 CSUBSTLST
RECORD.FIELD.VALUE RECORD.FIELD.VALUE0 MAKECREATELST SMASHPATTERN SMASHPAT1 MAKECREATELST1
GETFIELDFORCREATE SUBFIELDCREATE MAKEHASHLINKS HASHLINKS RECLOOK ALLFIELDS SUBDECLARATIONS)
(FNS CLISPRECORD ACCESSDEF FIELDNAMESIN ACCESSDEF4 MAKEACCESS MAKEACCESS1 MKACCESSFN RECFIELDLOOK
RECORDCHAIN RECLOOK1 SYSRECLOOK1 TOPPATHS ALLPATHS CHECKDEFS JOINDEF)
(FNS NOTOKSWAP FIXFIELDORDER FINDFIELDUSAGE EMBEDPROG)
(FNS RECLISPLOOKUP CONSFN RECORDGENSYM RECORDBIND REORDER ERROR SETUPHASHARRAY DWIMIFYREC MKCONS MKPROGN)
(FNS RECORDINIT)
(VARS PATGENSYMVARS)
(INITVARS (RECORDINIT))
(INITVARS CLISPRECORDTYPES)
(INITVARS (RECORDTRANHASH (HASHARRAY 20)))
(FNS * (PROGN CLISPRECORDTYPES))
(FNS RECORDECLARATIONS RECORDALLOCATIONS SAVEONSYRECLST)
(ADDVARS (USERRECLST))
(VARS (DECLARATIONCHAIN)
MSBLIP NOSIDEFNS (RECORDSUBSTFLG)
(RECORDUSE)
DATATYPEFIELDCOERCIONS)
(INITVARS (RECORDCHANGEFN))
(VARS CLISPRECORDWORDS)
(PROP CLISPWORD /REPLACE COPYING FETCH FFETCH FREPLACE REPLACE REUSING SMASHING TYPE? USING /replace
copying fetch ffetch replace replace reusing smashing type? using OF of WITH with CREATE create
INITRECORD initrecord)
(DECLARE%: DONTCOPY (FILEPKGCOMS RECORDTYPES))
(RECORDTYPES RECORD TYPEPERCORD PROPERCORD HASHLINK ACCESSFN ACCESSFNS HASHRECORD ATOMRECORD ARRAYRECORD
DATATYPE BLOCKRECORD ASSOCRECORD CACCESSFNS ARRAYBLOCK SYNONYM)
(DECLARE%: DONTCOPY
(MACROS CREATE.RECORD ADD.RECORD.SUBDECS RECORD.ALLOCATIONS RECORD.CREATEINFO
RECORD.DEFAULTFIELDS RECORD.FIELDINFO RECORD.FIELDNAMES RECORD.NAME RECORD.SUBDECS
RECORD.TYPECHECK SET.RECORD.ALLOCATIONS SET.RECORD.CREATEINFO SET.RECORD.DEFAULTFIELDS
SET.RECORD.FIELDNAMES SET.RECORD.NAME SET.RECORD.TYPECHECK RECORD.DECL SET.RECORD.DECL
RECORD.PRIORITY SET.RECORD.PRIORITY))
(LOCALVARS . T)
(ADDVARS (SYSLOCALVARS $$1 $$2 $$3 $$4 $$5 $$6 $$7 $$8 $$9 $$10 $$11 $$12 $$13 $$14 $$15 $$16 $$17))
[COMS
; for handling datatype
(P (MOVD 'FETCHFIELD 'FFETCHFIELD)
(MOVD 'REPLACEFIELD 'FREPLACEFIELD))
(E (CLISPDEC 'STANDARD))
(IFPROP (LISPEN CLISPCLASS CLISPCLASSDEF)
FETCHFIELD FFETCHFIELD FREPLACEFIELD /REPLACEFIELD REPLACEFIELD)
(ADDVARS (DECLWORDS FFETCHFIELD FETCHFIELD REPLACEFIELD FREPLACEFIELD /REPLACEFIELD))
(P (NEW/FN 'REPLACEFIELD))
(VARS RECORDWORDS)
; for CHANGETRAN
[COMS
(PROP CLISPWORD ADD CHANGE POP PUSH PUSHNEW PUSHLIST add change pop push pushnew pushlist SWAP
swap /push /pushnew /PUSH /PUSHNEW)
(FNS CHANGETRAN CHANGETRAN1 FIXDATUM)
(PROP SETFN GETP GETPROP EVALV GETATOMVAL OPENR WORDCONTENTS \GETBASE \GETBASEBYTE \GETBASEBIT
FETCHFIELD))
(BLOCKS (RECORDBLOCK ACCESSDEF ACCESSDEF4 ALLFIELDS ALLOHASH ALLPATHS CHANGETRAN CHANGETRAN1 CHECKDEFS
CHECKRECORDNAME CLISPRECORD CONSFN COPY1 CREATEFIELDS CSUBST RECONS CSUBSTLST
DECLARERECORD DECLSUBFIELD DWIMIFYREC EMBEDPROG FIELDLOOK FIELDNAMESIN FINDFIELDUSAGE
FIXDATUM FIXFIELDORDER GETFIELDFORCREATE GETSETQ HASHLINKS JOINDEF LISTRECORDEFS
MAKEACCESS MAKEACCESS1 MAKECREATE0 MAKECREATE1 MAKECREATELST MAKECREATELST1 MAKEHASHLINKS
MKACCESSFN MKCONS MKPROGN NOTOKSWAP RECDEC? RECEVAL RECFIELDLOOK RECLISPLOOKUP
RECLOOK RECLOOK1 RECORD.FIELD.VALUE RECORD.FIELD.VALUE0 RECORDACCESS RECORDALLOCATIONS
RECORDBIND RECORDBINDVAL RECORDCHAIN RECORDECL RECORDECL0 RECORDECL1 RECORDECLBLOCK
RECORDECLTAIL RECORDECLARATIONS REORDER ERROR RECORDFIELD? RECORDFIELDNAMES RECORDGENSYM
RECORDTRAN RECORDWORD RECREDECLARE SETUPHASHARRAY SIMPLEP SUBDECLARATIONS SUBFIELDCREATE
TOPPATHS UNCLISPTRAN RECORDPRIORITY
(ENTRIES RECORDTRAN CHANGETRAN CLISPRECORD RECORDFIELD? RECORDECLARATIONS
RECORDALLOCATIONS RECORDACCESS RECORDFIELDNAMES RECLOOK SETUPHASHARRAY FIELDLOOK
RECORD.FIELD.VALUE DECLARERECORD RECORDPRIORITY)
(SPECVARS DWIMIFYFLG CLISPCHANGE NEWVALUE DECLARATIONCHAIN USINGTYPE USINGEXPR ARRAYDESC
EXPR FAULTFN VARS DECLST FIELDNAMES RECORDEXPRESSION RECORD.TRAN ALLOCATIONS
FIELDS.IN.CREATE PATGENSYMVARS NOSPELLFLG PATGENSYMVARS)
(LOCALFREEVARS FIELD.USAGE BINDINGS RNAME NAME TAIL SETQPART SETQTAIL DECL CREATEINFO
```

```

CLISPCHANGE FIELDINFO HASHLINKS ARGS AVOID BODY VAR1 NOTRANFLG SPECIALFIELDS
SUBSTYPE STRUCNAME)
(NOLINKFNS . T)
SMASHPATTERN SMASHPAT1))
(GLOBALVARS MSBLIP CLISPRECORDTYPES NOSIDEFNS CLISPWORDWORDS RECORDSTATS USERRECLST RECORDINIT
LAMBDA$PLST CLISPTRANFLG RECORDCHANGEFN COMMENTFLG CLISPCHARRAY LCASEFLG CLISPARRAY LISPXFNS
RECORDWORDS DATATYPEFIELD$COERCIONS DATATYPEFIELDTYPES RECORDTRANHASH RECORDINIT CLISPARRAY
CLISPRECORDTYPES RECORDTRANHASH)
(FNS EDITREC)
(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
(ADDVARS (NLAMA EDITREC SAVEON$SYSRECLST RECORDALLOCATIONS RECORDECLARATIONS SYNONYM ARRAYBLOCK
CACCESSFNS ASSOCRECORD BLOCKRECORD DATATYPE ARRAYRECORD ATOMRECORD HASHRECORD
ACCESSFNS ACCESSFN HASHLINK PROP$RECORD TYP$RECORD RECORD MESATYPE MESARECORD
MESAARRAY)
(NLAML)
(LAMA))

```

(DEFINEQ

(RECORDTRAN

[LAMBDA (RECORDEXPRESSION WORDTYPE) ; Edited 9-Jan-87 21:10 by Pavel

;; top level entry for translation of record expressions

```

(PROG ((PATGENSYMVARS PATGENSYMVARS)
(DECLST (GETLOCALDEC EXPR FAULTFN))
DEF NOTRANFLG (EXPRESSIONTYPE (RECORDWORD (CAR RECORDEXPRESSION)
RECORDEXPRESSION WORDTYPE))

BINDINGS TAIL)
(SETQ CLISPCHANGE T)
[COND
((SETQ DEF (ASSOC EXPRESSIONTYPE RECORDWORDS))
(SETQ DECLST (CONS (CADR DEF)
DECLST))
(SETQ EXPRESSIONTYPE (CADDR DEF)
(SETQ DEF
(SELECTQ EXPRESSIONTYPE
(fetch (OR (SETQ DEF (ACCESSDEF (CADR RECORDEXPRESSION)
(CADDDR RECORDEXPRESSION)
(CDR RECORDEXPRESSION)))
(RECORDERROR 7 RECORDEXPRESSION))
(SELECTQ (RECORDWORD (CAR (SETQ TAIL (CDDR RECORDEXPRESSION)))
TAIL)
((of OF)
(SETQ TAIL (CDR TAIL)))
NIL)
(DWIMIFYREC TAIL NIL RECORDEXPRESSION)
(MAKEACCESS DEF (MKPROGN TAIL)
NIL
'fetch))
(replace (COND
([NOT (SETQ DEF (ACCESSDEF (CADR RECORDEXPRESSION)
(CADDDR RECORDEXPRESSION)
(CDR RECORDEXPRESSION))
(RECORDERROR 7 RECORDEXPRESSION))
(SELECTQ (RECORDWORD (CAR (SETQ TAIL (CDDR RECORDEXPRESSION)))
TAIL)
((OF of)
(SETQ TAIL (CDR TAIL)))
NIL)
(DWIMIFYREC TAIL ' (with WITH)
RECORDEXPRESSION T)
(MAKEACCESS DEF (CAR TAIL)
(PROGN (DWIMIFYREC (CDR (SELECTQ (RECORDWORD (CADR TAIL)
(CDR TAIL))
(with WITH)
(SETQ TAIL (CDR TAIL)))
TAIL))
NIL RECORDEXPRESSION)
(CDR TAIL))
EXPRESSIONTYPE))
(create (PROG (DEC FIELDS.IN.CREATE TRAN SETQPART SETQTAIL TEM2 USING USINGTYPE USINGEXPR
(TL (CDDR RECORDEXPRESSION))
FIELDNAMES UNUSED)

```

;; BLIP is used throughout the computation to indicate a no-op -- i.e. a field which was not specified

```

[SETQ FIELDNAMES (ALLFIELDS (SETQ TRAN (RECORDECL (SETQ DEC (RELOOK
(CADR
RECORDEXPRESSION
)
(CDR
RECORDEXPRESSION
)
DECLST
RECORDEXPRESSION T])

```

;; RELOOK looks up the declaration for the record name given (CREATE A --) it returns the declaration for A ; Go through the create statement, picking up the field_'s and the ; USING or COPYING, etc

```

[while TL do (COND
  ((SETQ TEM2 (RECORDWORD (CAR TL)
                          TL))
   ; USING COPYING ETC
   (COND
    (USING (RECORDERROR [COND
      ((EQ (CAR TL)
            (CAR USING))
       (LIST (CAR TL)
              "occurs twice"))
      (T (LIST "both" (CAR TL)
                "and"
                (CAR USING]
           TL RECORDEXPRESSION))
      (T (SETQ USINGTYPE TEM2)
         (SETQ USING TL)))
    (DWIMIFYREC (CDR TL)
                 CLISPRECORDWORDS RECORDEXPRESSION)
    (SETQ TL (CDDR TL)))
  ((GETSETQ TL FIELDNAMES RECORDEXPRESSION CLISPRECORDWORDS NIL
            CLISPRECORDWORDS)
   ;; Adds the info to alist, or ERROR's --- if it returned NIL then a correction was made and
   ;; we should just retry the same TL
   (COND
    ((ASSOC (CAR SETQPART)
             FIELDS.IN.CREATE)
     (RECORDERROR 5 TL RECORDEXPRESSION))
    (T (SETQ FIELDS.IN.CREATE (CONS SETQPART FIELDS.IN.CREATE))
      (SETQ TL SETQTAIL]
  (COND
   (USINGTYPE (SETQ USINGEXPR (RECORDBINDVAL (COND
      ((FMEMB 'CHECK
              (CDR (RECORD.TYPECHECK
                    TRAN)))
       (LIST 'THE (RECORD.NAME TRAN)
              (CADR USING)))
      (T (CADR USING]
   (SETQ DEF (MAKECREATE0 TRAN (HASHLINKS TRAN)
                        T))
  (COND
   ((SETQ UNUSED (FIXFIELDORDER DEF))
    (PROG ((DECLST (CONS 'FAST DECLST))
           TEM)
          (SETQ DEF
                (CONS
                 'PROG1
                 (CONS (LIST 'SETQ (SETQ TEM (RECORDBIND))
                             DEF)
                       (for X in (DREVERSE UNUSED)
                            collect (MAKEACCESS (CAR (OR (ACCESSDEF4 (LIST (CAR X)
                                                                           TRAN)
                                                                           (RECORDERROR 'REPLACE
                                                                           (CAR X)
                                                                           RECORDEXPRESSION)))
                                     TEM
                                     (CDR X)
                                     'replace]
          (RETURN DEF)))
  (with ;; new feature: (with RECORDNAME of <expression> stuff) --- means execute <stuff> substituting the fieldnames
  (PROG ((SUBSTYPE 'WITH)
         [SPECIALFIELDS (LIST (LIST 'DATUM 'USING]
                               USINGEXPR RECORD.TRAN FIELDNAMES)
         [SETQ FIELDNAMES (ALLFIELDS (SETQ RECORD.TRAN (RECORDECL (RECLOOK (CADR
                                                                           RECORDEXPRESSION
                                                                           )
                                                                           (CDR
                                                                           RECORDEXPRESSION
                                                                           )
                                                                           )
                                                                           DECLST
                                                                           RECORDEXPRESSION T]
          (DWIMIFYREC (CDDR RECORDEXPRESSION)
                     (CONS 'DATUM FIELDNAMES)
                     RECORDEXPRESSION)
          (SETQ USINGEXPR (RECORDBINDVAL (CADDR RECORDEXPRESSION)))
          (RETURN (CSUBST (MKPROGN (CDDDR RECORDEXPRESSION))
                        (type? (OR [SETQ DEF (CAR (RECORD.TYPECHECK (RECORDECL (RECLOOK (CADR RECORDEXPRESSION)
                                                                           (CDR RECORDEXPRESSION)
                                                                           DECLST RECORDEXPRESSION T]
          (RECORDERROR 'TYPE? (CADR RECORDEXPRESSION)
                          RECORDEXPRESSION))
          (DWIMIFY0? (CDDR RECORDEXPRESSION)
                    RECORDEXPRESSION T T NIL FAULTFN 'VARSBOUND)
  (COND
   [(OR (NLISTP DEF)
        (FMEMB (CAR DEF)

```

```

      LAMBDA(SPLST))
    (SETQ DEF (CONS DEF (CDDR RECORDEXPRESSION)
      (T (PROG [(SUBTYPE 'TYPE?)
        [SPECIALFIELDS (LIST (LIST 'DATUM 'USING)
          FIELDNAMES
          (USINGEXPR (MKPROGN (CDDR RECORDEXPRESSION)
            (RETURN (CSUBST DEF))
          (initrecord [SETQ DEF (MKPROGN (RECORD.ALLOCATIONS (RECORDECL (RECLOOK (CADR
            RECORDEXPRESSION
            )
            (CDR RECORDEXPRESSION)
            DECLST RECORDEXPRESSION
            T]))
          (CHANGETRAN1 EXPRESSIONTYPE RECORDEXPRESSION)))
    [COND
      (BINDINGS (SETQ DEF (EMBEDPROG DEF)
        (LET ((DWIMESSGAG T)
          (NOSPPELLFLG T)
          LISPXHIST)
          (DECLARE (SPECVARS LISPXHIST DWIMESSGAG NOSPPELLFLG)
            (DWIMIFY0? DEF DEF NIL NIL NIL FAULTFN 'VARSBOUND))
        [COND
          ((NLISTP DEF)
            (SETQ DEF (LIST 'PROGN DEF)
          (COND
            (NOTRANFLG (RETURN DEF)))
            (CLISPTRAN RECORDEXPRESSION DEF)
            (RETURN RECORDEXPRESSION])

```

(RECREDECLARE

```

[LAMBDA (RECNAME RECFIELDS OLDFLG) (* Imm "13-SEP-77 15:49")
  (DECLARE (SPECVARS RECNAME RECFIELDS))
  (AND RECORDCHANGEFN (APPLY* RECORDCHANGEFN RECNAME RECFIELDS OLDFLG))
  (AND CLISPARRAY (MAPHASH CLISPARRAY (FUNCTION RECREDECLARE1))

```

(RECREDECLARE1

```

[LAMBDA (TRAN ORIG) (* Imm "31-JUL-78 05:04")
  ;; Given an entry in CLISPARRAY, test if it is a record expression involving any of the fields that have changed, and remove the old translation
  (AND (RECREDECLARE2 ORIG)
    (/PUTHASH ORIG NIL CLISPARRAY])

```

(RECREDECLARE2

```

[LAMBDA (FORM) (* Imm "31-JUL-78 05:04")
  ; should this form be changed
  (SELECTQ (CAR (GETP (CAR FORM)
    'CLISPWORD))
    (RECORDTRAN (SELECTQ (CAR FORM)
      ((CREATE create TYPE? type?)
        (EQ (CADR FORM)
          RECNAME))
      (OR (LISTP (CADR FORM))
        (FMEMB (CADR FORM)
          RECFIELDS))))
    (CHANGETRAN (RECREDECLARE2 (CADR FORM)))
    NIL])

```

(RECORDECL

```

[LAMBDA (DEC) (* Imm%: "26-JUL-76 02:44:29")
  ;; Entry for lookup of record declarations --- retrieve the current translation of the declaration DECL, or create a new one and store it on DEC
  (PROG (ALLOCATIONS TEM)
    ;; Some declarations (specifically HASHLINKS and DATATYPES) require expressions to be evaluated at run-time. When these are encountered,
    ;; the run-times are added to ALLOCATIONS. The RECORDS prettydefmacro puts out the ALLOCATIONS within a DOCOPY so that they will be
    ;; inserted in the .COM file even if the declaration itself is dumped out DONTCOPY
    (AND (SETQ TEM (RECORDECL0 DEC))
      ALLOCATIONS
      (SET.RECORD.ALLOCATIONS TEM ALLOCATIONS))
    (RETURN TEM])

```

(RECORDFIELD?

```

[LAMBDA (FIELD DECLARATIONS) (* Imm "18-SEP-78 18:35")
  ;; Top level predicate if an atom is a field name. Used by DWIM to avoid ambiguity in X:FIELD9 -> X:FIELD
  (PROG (TEM)
    (RETURN (COND
      [(SETQ TEM (STRPOS '%. FIELD))
        (AND (RECLOOK (SUBATOM FIELD 1 (SUB1 TEM)))
          (RECORDFIELD? (SUBATOM FIELD (ADD1 TEM)
            -1))
        (T (for X in (OR DECLARATIONS USERRECLST) when [FMEMB FIELD (RECORD.FIELDNAMES

```

(SETQ X (RECORDECL X])

do (RETURN (OR (RECORD.NAME X) X])

(RECORDECL0

[LAMBDA (DEC PARENT)

(* Imm " 7-AUG-84 23:33")

:: Returns either NIL or the translation of a declaration expression

(if (NLISTP DEC) then NIL elseif (NOT (FMEMB (CAR DEC) CLISPRECORDTYPES)) then NIL elseif (GETHASH DEC RECORDTRANHASH) elseif (AND CLISPARRAY (GETHASH DEC CLISPARRAY)) else (PROG ((TRANSLATION (RECORDECL1 DEC PARENT))) (PUTHASH DEC TRANSLATION RECORDTRANHASH) (RETURN TRANSLATION]))

(RECORDECL1

[LAMBDA (DECL PARENT)

(* Imm " 7-Jul-86 10:32")

(if (NOT (FMEMB DECL DECLARATIONCHAIN)) then (LET ((DECLARATIONCHAIN (CONS DECL DECLARATIONCHAIN))) (SETQ DECL (RECORD.REMOVE.COMMENTS DECL)) (PROG (TEM1 TRANSLATION (NAME (CADR DECL)) (STRUCNAME (CADR DECL)) (TAIL (CDDDR DECL)) (CREATEINFO (CADDR DECL)) (CREATETYPE (CAR DECL)) FIELDINF TYPECHECK FIELDNAMES)

:: the vars CREATETYPE NAME CREATEINFO TAIL are bound to 'default' values. If declaration is in non-standard format (e.g. :: (RECORD (B . C))) these values are changed below.

RETRY

[SELECTQ (CAR DECL) (RECORD (CHECKRECORDNAME NIL T) (SETQ FIELDINF (LISTRECORDEFS CREATEINFO))) (TYPEARECORD

:: For RECORD and TYPEARECORD, the field info is a CROPS list, and the CREATEINFO is the original :: template (TYPEARECORD has NAME consed onto it)

(CHECKRECORDNAME T T T) (SETQ TYPECHECK (LIST 'EQ ' (CAR (LISTP DATUM)) (KWOTE STRUCNAME))) [SETQ FIELDINF (LISTRECORDEFS (SETQ CREATEINFO CREATEINFO) 'D) (SETQ CREATEINFO (CONS STRUCNAME CREATEINFO))] ((PROPRECORD ATOMRECORD ASSOCRECORD)

:: For these record types, the FIELDINF is the atom of the field name and the :: CREATEINFO is just the list of fields

(CHECKRECORDNAME)

[SETQ FIELDINF (for FIELD in CREATEINFO collect (CONS FIELD (CONS (CAR DECL) FIELD))

(ARRAYRECORD (CHECKRECORDNAME) (SETQ TYPECHECK (ARRAY DATUM))

:: for ARRAYRECORD, the fieldinfo is either n (index) or (D . n) (index for ELTD) and the :: CREATEINFO is just the total number of entries

; RECORDECLARRAY returns the FIELD information, but also ; smashes up CREATEINFO

(PROG ((CNT 0) X (CL CREATEINFO)) LP (COND (CL [COND [(SMALLP (CAR CL)) (SETQ CNT (IPLUS CNT (CAR CL)) (T (SETQ CNT (ADD1 CNT))) (COND ((CAR CL) [COND ((OR (NLISTP (SETQ X (CAR CL))) (SETQ X (CAR X))) (SETQ FIELDINF (CONS (CONS X (CONS 'ARRAYRECORD CNT)) FIELDINF] (COND ((CDR (LISTP (CAR CL))) (SETQ FIELDINF (CONS (CONS (CDR (CAR CL)) (CONS 'ARRAYRECORD (CONS 'D CNT))) FIELDINF)) (FRPLNODE CL (CAAR CL) (FRPLNODE (CAR CL)

```

(CDAR CL)
(CDR CL))
      (SETQ CL (CDR CL))
      (SETQ CL (CDR CL))
      (GO LP)))
      (SETQ CREATEINFO (CONS CNT CREATEINFO)))
(HASHRECORD [SETQ TEM1 (COND
  ((RECDEC? (CADR DECL))
    ;(hashlink (record --) --)
    (SETQ NAME NIL)
    (SETQ TAIL (CDR DECL))
    (LIST (GENSYM)))
  ((LISTP (CADR DECL))
    ;(hashlink (foo) --)
    (SETQ NAME NIL)
    (SETQ TAIL (CDDR DECL))
    (CADR DECL))
  ((NULL (CDDR DECL))
    ;(hashlink foo)
    (SETQ NAME NIL)
    (SETQ TAIL (CDDR DECL))
    (LIST (CADR DECL)))
  ((RECDEC? (CADDR DECL))
    ;(hashlink foo (record ---) --)
    (SETQ TAIL (CDDR DECL))
    (LIST (GENSYM)))
  ((NLISTP (CADDR DECL))
    ;(hashlink fie fum --)
    (LIST (CADDR DECL)))
  (T
    ; Finally, the 'right' way --- (hashlink name (field) --)
    (CADDR DECL)
  [SETQ CREATEINFO (LIST (CAR TEM1)
    (COND
      ((NUMBERP (CADR TEM1))
        ;(HASHLINK (FOO 100)) --- initial size
        (ALLOHASH (OR (CADDR TEM1)
          (CAR TEM1))
          (CADR TEM1)
          T))
      (T (ALLOHASH (CADR TEM1)
        (CADDR TEM1)
        T)
      ]
    [SETQ FIELDINF (LIST (CONS (CAR CREATEINFO)
      (CONS 'HASHRECORD (CDR CREATEINFO))
    ((ACCESSFNS CACCESSFNS)
      (CHECKRECORDNAME NIL T)
      [SETQ FIELDINF
        (for X in (COND
          ((LITATOM (CAR CREATEINFO))
            (LIST CREATEINFO))
          (T CREATEINFO))
        join (PROGN (COND
          ((OR (NLISTP X)
            (CDDDR X))
            (RECORDERROR 1 X DECL)))
          (COND
            [(LISTP (CAR X))
              (for Y in (CAR X) collect (CONS Y (CONS (CAR DECL)
                (CONS Y (CDR X))
              (T (LIST (CONS (CAR X)
                (CONS (CAR DECL)
                  X]
          (SETQ CREATEINFO)
          (SETQ CREATETYPE))
      ((BLOCKRECORD DATATYPE ARRAYBLOCK)
        (CHECKRECORDNAME (NEQ (CAR DECL)
          'DATATYPE)
          NIL T)
      [PROG ((ARRAYDESC)
        DEFL)
        [SETQ FIELDINF (CAR (SETQ DEFL (RECORDECLBLOCK DECL)
          (SETQ CREATEINFO (CONS (SELECTQ (CAR DECL)
            (DATATYPE (SETQ TYPECHECK (LIST 'TYPENAMEP
              'DATUM
                (KWOTE STRUCNAME))
            )
            STRUCNAME)
            (ARRAYBLOCK ARRAYDESC)
            (RETURN (SETQ CREATEINFO)))
          (CONS (MAPCAR FIELDINF (FUNCTION CAR))
            (CONS (CDR DEFL)
              FIELDINF])
      (COND
        ((SETQ TEM1 (GETPROP (CAR DECL)
          'USERRECORDTYPE))
          (RETURN (RECORDECL1 (APPLY* TEM1 DECL)
            PARENT)))

```



```

        SPECS)))
    (SELECTQ (CAR DEC)
      (DATATYPE (SETQ ALLOCATIONS (CONS `[/DECLAREDATATYPE ',STRUCNAME ',SPECS
                                         ',(CDR DLIST)
                                         ',(CAR DLIST]
                                         ALLOCATIONS)))
      NIL)
    (SETQ FI (for X in (CDR DLIST) as Y in FIELDNAMES collect (CONS Y (CONS 'DATATYPE X]
    (RETURN (CONS FI DEFAULTS]))

```

(RECORDECLTAIL

```

[LAMBDA (NAME FIELDNAMES TL DEC TRANSLATION) (* gbn "9-Jun-86 21:36")
  (PROG [SETQTAIL SETQPART (TYPES (APPEND ' (CCREATE CREATE TYPE? SUBRECORD INIT DECL SYSTEM)
                                         CLISPRECORDTYPES))
        (LOCALVARS (COND
                    (NAME (CONS NAME FIELDNAMES))
                    (T FIELDNAMES]
    LP (COND
      ((NULL TL)
       (RETURN)))
    (COND
      ((LISTP (CAR TL))
       [SELECTQ (CAAR TL)
        (INCLUDES
          ; change the translation to have includes on the end if the
          ; declaration is a datatype
          (LET ((RUNTIMEDECL (ASSOC '/DECLAREDATATYPE ALLOCATIONS)))
            (if RUNTIMEDECL
              then [RPLACD (NTH RUNTIMEDECL 5)
                          `(', (CADAR TL) ; as a signal that the super's fields have not been included, set
                          ; the length to NIL
                          (RPLACA (NTH RUNTIMEDECL 5)
                                NIL))))
            (SUBRECORD (DECLSUBFIELD (CAR TL)
                                   TRANSLATION DEC))
            (INIT (APPLY 'PROGN (CDAR TL))
              ;; We'd like the builtin INIT's to be done before the user's, so that, e.g., a datatype has been declared before the user
              ;; does a DEFPRINT in the INIT.
              (SETQ ALLOCATIONS (APPEND ALLOCATIONS (CDAR TL))))
            ((CREATE CCREATE)
             [SET.RECORD.CREATEINFO TRANSLATION (CONS (CAAR TL)
                                                       (CONS (CADAR TL)
                                                             (RECORD.CREATEINFO TRANSLATION]))
             (TYPE? (SET.RECORD.TYPECHECK TRANSLATION (CONS (OR (CADAR TL)
                                                                (CAR (RECORD.TYPECHECK TRANSLATION)))
                                                            (CDDAR TL))))
             (DECL (SET.RECORD.DECL TRANSLATION (CAR TL)))
             (SYSTEM (SET.RECORD.PRIORITY TRANSLATION 'SYSTEM))
             (COND
              ((EQ (CAAR TL)
                   COMMENTFLG))
              ((RECDEC? (CAR TL))
               (DECLSUBFIELD (UNCLISPTRAN (CAR TL))
                             TRANSLATION DEC))
              (T (GO TRYASSIGN]
            (GO NXT))
            ((EQ (CADR TL)
                 'e)
             (COND
              [(EQ (CAR TL)
                   NAME)
               (SETQ TL (CONS (LIST 'TYPE? (CADDR TL))
                             (CDDDR TL]
               (T (RECORDERROR 1 TL DEC)))
              (GO LP)))
            TRYASSIGN
            (COND
              ((GETSETQ TL LOCALVARS DEC NIL TYPES NIL T)
               [COND
                [(EQ (CAR SETQPART)
                     NAME)
                 (SET.RECORD.CREATEINFO TRANSLATION (CONS 'CREATE (CONS (CADR SETQPART)
                                                                           (RECORD.CREATEINFO TRANSLATION]
                 (T (SET.RECORD.DEFAULTFIELDS TRANSLATION (CONS (LIST (CAR SETQPART)
                                                                       (CADR SETQPART))
                                                                           (RECORD.DEFAULTFIELDS TRANSLATION]
                 ; Add the 'default' value to the default-value-association-list
                 (SETQ TL SETQTAIL)
                 (GO LP))
                (T (GO LP)))
            NXT (SETQ TL (CDR TL))
            (GO LP])

```

(CHECKRECORDNAME

```

[LAMBDA (NEEDSNAME 3MUSTLISTP OKSTRUCDIFF) (* Imm "29-AUG-78 23:57")

```


:: DECL is the declaration; NEEDSNAME is on if it's ok for record to have no record-name; OKSTRUCDIFF is ok if it is OK for STRUCNAME to be different from NAME

```
[COND
  ((NOT (AND NAME (LITATOM NAME))))
  (COND
    ((AND OKSTRUCDIFF (LISTP NAME)
      (LITATOM (CAR NAME))
      (LITATOM (CADR NAME))
      (NULL (CDDR NAME))))
      (SETQ STRUCNAME (CADR NAME))
      (SETQ NAME (CAR NAME)))
    (T (COND
      (NEEDSNAME (RECORDERROR 0 DECL)))
      (SETQ NAME NIL)
      (SETQ TAIL (CDDR DECL))
      (SETQ CREATEINFO (CADR DECL]
    (COND
      ((AND (NOT 3MUSTLISTP)
        (NLISTP CREATEINFO))
        (RECORDERROR 1 (CADDR DECL)
          DECL]))
```

(LISTRECORDEFS

(* lmm " 8-AUG-83 23:19")

```
[LAMBDA (FORMAT CROPS TL)
  (COND
    ((NULL FORMAT)
      TL)
    ((NLISTP FORMAT)
      (CONS (CONS FORMAT (CONS 'RECORD CROPS))
        TL))
    ((SMALLP (CAR FORMAT))
      (LISTRECORDEFS (CDR FORMAT)
        (to (CAR FORMAT) do (SETQ CROPS (CONS 'D CROPS)) finally (RETURN CROPS))
        TL))
    (T (AND (CAR FORMAT)
      (SETQ TL (LISTRECORDEFS (CAR FORMAT)
        (CONS 'A CROPS)
        TL))))
    (COND
      ((CDR FORMAT)
        (LISTRECORDEFS (CDR FORMAT)
          (CONS 'D CROPS)
          TL))
      (T TL]))
```

(RECORD.REMOVE.COMMENTS

(* lmm " 8-AUG-83 23:26")

```
[LAMBDA (X)
  (COND
    ((NLISTP X)
      X)
    ((EQ (CAR (LISTP (CAR X)))
      COMMENTFLG)
      (RECORD.REMOVE.COMMENTS (CDR X)))
    (T (PROG [A (RECORD.REMOVE.COMMENTS (CAR X))]
      [D (RECORD.REMOVE.COMMENTS (CDR X)]
      (RETURN (COND
        ((AND (EQ A (CAR X))
          (EQ D (CDR X)))
          X)
        (T (CONS A D]))
```

(DECLARERECORD

; Edited 12-Jan-88 14:44 by drc:

```
[LAMBDA (DEC)
  (PROG (TRANSLATION TEM RECNAME OLDTRAN OLDFLG)
    [COND
      ((SETQ TEM (CL:MEMBER DEC USERRECLST :TEST 'CL:EQUAL))
        ; There is already an EQUAL declaration (this can often happen
        ; with DOEVAL@COMPILE declarations)
      (RETURN (OR (RECORD.NAME (RECORDECL (CAR TEM)))
        TEM])
```

:: the COPY is so that later when the the whole declaration is stored on USERRECLST that a subsequent edit to the same structure won't get confused.

```
(OR [SETQ TRANSLATION (RECORDECL (SETQ DEC (COPY DEC)
  (RECORDERROR 1 DEC))
  [if (SETQ RECNAME (RECORD.NAME TRANSLATION))
    then
      ; If the declaration has a name, check if some previous
      ; declaration exists with same name
      [if [SETQ TEM (SOME USERRECLST (FUNCTION (LAMBDA (X)
        (EQ (RECORD.NAME (SETQ OLDTRAN (RECORDECL X)))
          RECNAME]
      then (SETQ OLDFLG T)
        (PUTHASH TEM NIL RECORDTRANHASH)
```

```

      (OR (EQ DFNFLG T)
          (LISPXPRT (LIST 'record RECNAME 'redeclared)
                   T T))
    else (SETQ OLDTRAN) ; OLDTRAN is used below to get the names of the fields which
                                ; USE TO BE in this record
      (SETQ TEM (SETQ USERRECLST (CONS NIL USERRECLST))
            ; TEM is the location in USERRECLST where the declaration will
            ; go

    else (SETQ TEM NIL)
      (for X in USERRECLST do (for Y in (RECORD.FIELDNAMES (RECORDECL X)) unless (FMEMB Y TEM)
                                when (FMEMB Y (RECORD.FIELDNAMES TRANSLATION))
                                do (LISPXPRT (LIST 'record 'field Y 'redeclared)
                                             T T)
                                  (SETQ TEM (CONS Y TEM))

                                ;; TEM is the list of field names which appear in other declarations --- normally, field names that
                                ;; appear in multiple declarations are ok, since they can be qualified with the record name. If there is
                                ;; no name, however, the old declarations are ignored... e.g. if you define (RECORD A (B . C)) and
                                ;; then define (RECORD (D C)) you will get the latter interpretation if you just say C, and the former if
                                ;; you say A.C
                                ))
      (SETQ TEM (SETQ USERRECLST (CONS NIL USERRECLST)
                                (MAPC (RECORD.ALLOCATIONS TRANSLATION)
                                      (FUNCTION EVAL))
                                ; At this point, TEM points to the tail of USERRECLST where this
                                ; declaration should be smashed

      (/RPLACA TEM DEC)
      (AND FILEPKGFLG (MARKASCHANGED (OR RECNAME DEC)
                                      'RECORDS))
      (RECREDECLARE RECNAME (UNION (RECORD.FIELDNAMES OLDTRAN)
                                   (RECORD.FIELDNAMES TRANSLATION))
                   OLDFLG)

      ;; RECREDECLARE takes care of removing current CLISP translations involving the old or new declaration and (possibly) unsavedef'ing
      ;; compiled code that involves those declarations
      (RETURN RECNAME]))

```

(DECLSUBFIELD

```

[LAMBDA (SUBDECL TRANSLATION DEC) ; (* Imm "13-Mar-85 16:12")
                                ; Translate SUBDECL and insert it into the 'meaning' of the
                                ; superior

(PROG (SUBTRAN SUBNAME)
      (COND
        ((EQ (CAR SUBDECL)
              'SUBRECORD)
         (OR (ASSOC (CADR SUBDECL)
                    (RECORD.FIELDINFO TRANSLATION))
             (GO ERR)))
        (T (OR (SETQ SUBTRAN (RECORDECL0 SUBDECL TRANSLATION))
                (RECORDEERROR 1 SUBDECL DEC))
           [COND
             ((NULL (SETQ SUBNAME (RECORD.NAME SUBTRAN)))
              (SET.RECORD.NAME SUBTRAN (SETQ SUBNAME (COND
                                                         ((EQ (CAR (RECORD.CREATEINFO TRANSLATION))
                                                                'HASHRECORD)
                                                         (CAAR (RECORD.FIELDINFO TRANSLATION)))
                                                         (T (RECORD.NAME TRANSLATION]
              (OR (EQ (RECORD.NAME TRANSLATION)
                      SUBNAME)
                  (ASSOC SUBNAME (RECORD.FIELDINFO TRANSLATION))
                  (GO ERR))
              (SET.RECORD.FIELDNAMES TRANSLATION (APPEND (RECORD.FIELDNAMES SUBTRAN)
                                                         (RECORD.FIELDNAMES TRANSLATION)))
              ; Add the sub-declaration to the list of sub-declarations in the
              ; parent's translation
            ))
            (RETURN (ADD.RECORD.SUBDECS TRANSLATION SUBDECL))
            ERR (RECORDEERROR -1 SUBDECL DEC]))

```

(UNCLISPTRAN

```

[LAMBDA (EXPRESSION) ; (* Imm%: 28 JUL 75 437)
  [COND
    ((EQ (CAR EXPRESSION)
          CLISPTRANFLG)
     (/RPLNODE2 EXPRESSION (CDDR EXPRESSION))
     (AND CLISPARRAY (/PUTHASH EXPRESSION NIL CLISPARRAY))
     EXPRESSION])

```

(RECDEC?

```

[LAMBDA (X) ; (* Simple test if X is a record declaration)
  (COND
    ((NLISTP X)
     NIL)
    ((EQ (CAR X)

```

```

      CLISPTRANFLG)
    (RECDEC? (CDDR X)))
  (T (FMEMB (CAR X)
    CLISPRECORDTYPES])

```

(ALLOCHASH

(* lmm " 7-MAY-82 16:43")

```

[LAMBDA (HASHTABLENAME SIZE FLAG)
  (COND
    ((OR (AND SIZE (NOT (NUMBERP SIZE)))
      (NOT (LITATOM HASHTABLENAME)))
      (ERROR SIZE "bad hash array size")))
    [AND FLAG HASHTABLENAME (SETQ ALLOCATIONS (CONS (LIST 'DECLARE%: 'EVAL@COMPILE (LIST 'GLOBALVARS
      HASHTABLENAME))
      (CONS (LIST 'SETUPHASHARRAY (KWOTE HASHTABLENAME)
        SIZE)
        ALLOCATIONS])
      (SETUPHASHARRAY HASHTABLENAME SIZE)
      HASHTABLENAME])

```

(GETSETQ

(* lmm " 7-AUG-84 23:48")

;; Sets the free variables SETQTAIL and SETQPART --- SETQTAIL is the tail of TL after a SETQ type expression; SETQPART is (var value);
 ;; does spelling correction and/or dwimifying if necessary --- returns T if a setq was found, and NIL if an OKVAR is found (or corrected) or if a form
 ;; starting with an OKFN is found (or corrected) and prints an error message otherwise

```

(PROG NIL
  RETRY
    (COND
      ((NULL TL)
        (RETURN))
      ((FMEMB (CAR TL)
        OKVARS)
        (RETURN))
      ((LISTP (CAR TL))
        [SELECTQ (CAAR TL)
          (* (SETQ TL (CDR TL))
            (GO RETRY))
          ((SETQ SAVESETQ))
          ((SETQ SAVESETQ)
            [/RPLNODE (CAR TL)
              'SETQ
              (LIST (CADAR TL)
                (KWOTE (CADDR (CAR TL)))
              (COND
                ((FMEMB (CAAR TL)
                  OKFNS)
                  (RETURN))
                (T (GO DWIM])
              (OR (FMEMB (CADAR TL)
                NVAR)
                (FIXSPELL (CADAR TL)
                  70 NVAR)
                (NIL NIL NIL T)
                (RECORDERROR 7 TL PARENT))
                (SETQ SETQTAIL (CDR TL))
                (SETQ SETQPART (APPEND (CDAR TL)))
                [/RPLNODE TL (CADAR TL)
                  (CONS '_ (CONS (CADDR (CAR TL))
                    (CDR TL))
                  (RETURN T))
                ([AND (FMEMB (CAR TL)
                  NVAR)
                  (EQ (CADR TL)
                    '_)
                  (PROGN (COND
                    ((COND
                      [(NLISTP (CADDR TL))
                        (AND (LITATOM (CADDR TL))
                          (STRPOS CLISPCHARARRAY (CADDR TL)
                            (T (NOT VARSPLST)))
                        (DWIMIFYREC (CDDR TL)
                          NIL PARENT T INDECL))
                      (OR (NULL (CDDR TL))
                        (LISTP (CADDR TL))
                        (FMEMB (CADDR TL)
                          NVAR)
                        (FMEMB (CADDR TL)
                          OKVARS])
                    ))
                    ;; Kludge: Don't call DWIMIFY0? in previous conditional if called from RECORDSTATEMENT but do if in a declaration
                    (SETQ SETQTAIL (CDDR TL))
                    (SETQ SETQPART (LIST (CAR TL)
                      (CADDR TL)))
                    (RETURN T))
                  ))

```

DWIM

```
(COND
  ((AND OKFNS (LISTP (CAR TL))
        (FIXSPELL (CAAR TL)
                  70
                  (CONS 'SETQ OKFNS)
                  NIL
                  (CAR TL)
                  NIL NIL NIL T))
   (GO RETRY))
  ((DWIMIFYREC TL (APPEND NVAR (OR VARSPLST OKVARS))
                PARENT NIL INDECL)
   (GO RETRY))
  (T (RECORDERROR 'P (CAR TL)
                  PARENT]))
```

(RECORDACCESS

(* Imm "21-MAR-82 18:19")

```
[LAMBDA (FIELD DATUM DEC TYPE NEWVALUE)
  (DECLARE (SPECVARS DATUM))
  (PROG (RECS DECLST TEM DEF EXPR (FAULTFN 'TYPE-IN)
        (DWIMIFYFLG 'EVAL)
        VARS RECORDEXPRESSION BINDINGS)
    RETRY
      (COND
        ((LISTP FIELD)
         (COND
          ((NULL (CDR FIELD))
           (SETQ FIELD (CAR FIELD))
           (GO RETRY)))
          (UNCLISPTRAN FIELD)
          (SETQ DEF (RECORDCHAIN FIELD)))
         [[SETQ RECS (COND
                   [DEC (COND
                        ((RECDEC? DEC)
                         (RECFIELDLOOK (LIST DEC)
                                         FIELD))
                        (T (RECORDERROR 1 DEC]
                           (T (RECFIELDLOOK USERRECLST FIELD] ; RECFIELDLOOK returns a list of of declarations
                              (SETQ DEF (CHECKDEFS (for X in RECS join (ACCESSDEF4 (LIST FIELD)
                                                                              (RECORDECL X]
                              ((SETQ TEM (FIXSPELL FIELD NIL (FIELDNAMESIN USERRECLST)
                                             NIL NIL NIL NIL NIL T)) ; Finally, attempt spelling correction
                              (SETQ FIELD TEM)
                              (GO RETRY))
                              (T (SETQ DEF)))
          (COND
           ((NOT DEF)
            (RECORDERROR 7 FIELD)))
          (RETURN (EVAL (EMBEDPROG (MAKEACCESS DEF 'DATUM (SELECTQ TYPE
                                             ((NIL ffetch fetch FETCH FFETCH)
                                              (SETQ TYPE 'fetch)
                                              NIL)
                                              ((replace freplace /replace REPLACE FREPLACE
                                                       /REPLACE)
                                              (SETQ TYPE 'replace)
                                              (LIST (KWOTE NEWVALUE)))
                                              (ERROR TYPE "not FETCH or REPLACE"))
                                             TYPE]))
```

(RECORDFIELDNAMES

(* Imm "24-FEB-79 12:10")

```
[LAMBDA (RECORDNAME FLG)
  (PROG ([DECL (RECORDECL (OR (LISTP RECORDNAME)
                              (RECLOOK RECORDNAME)
                              VAL)
        [COND
          ((NULL FLG)
           (RETURN (RECORD.FIELDNAMES DECL)))
          (EQ FLG 'DECL)
           (RETURN (RECORD.DECL DECL]
          (for S in (RECORD.SUBDECS DECL) do (SETQ VAL (CONS (RECORDFIELDNAMES S T)
                                                             VAL)))
          (for X in (RECORD.FIELDINFO DECL) collect (SETQ VAL (CONS (CAR X)
                                                                    VAL)))
          (RETURN (CONS (RECORD.NAME DECL)
                       VAL]))
```

(RECEVAL

(* Imm "31-JUL-78 07:15")

(* ASSERT%: ((REMOTE EVAL) DATUM NEWVALUE FIELDNAME))

```
[LAMBDA (FORM DATUM NEWVALUE FIELDNAME)
  (DECLARE (SPECVARS NEWVALUE DATUM FIELDNAME))
  (AND FORM (COND
    ((AND (LISTP FORM)
          (NEQ (CAR FORM)
                'LAMBDA))
```

(EVAL FORM))
(T (APPLY* FORM DATUM NEWVALUE FIELDNAME])

(FIELDLOOK

[LAMBDA (FIELDNAME)
(**RECFIELDLOOK** USERRECLST FIELDNAME])

(SIMPLEP

[LAMBDA (X N) (* lmm "3-Jul-85 12:24")

;; is it worth it to bind a variable if this is being computed twice? --- returns N-{complexity} or NIL

(OR N (SETQ N 3))
(COND
 ((OR (NLISTP X)
 (CONSTANTEXPRESSIONP X))
 N)
 ((GETP (CAR X)
 'CROPS)
 (AND [NOT (MINUSP (SETQ N (IDIFFERENCE N (LENGTH (GETP (CAR X)
 'CROPS])
 (SIMPLEP (CADR X)
 N)))
 (T (SELECTQ (CAR X)
 (PROGN (AND [EVERY (CDR X)
 (FUNCTION (LAMBDA (Z)
 (SETQ N (SIMPLEP Z N]
 N))
 ((fetch FFETCH)
 (AND CLISPARRAY (SETQ X (GETHASH X CLISPARRAY))
 (SIMPLEP X N)))
 NIL]))

(RECORDBINDVAL

[LAMBDA (VAL)
(COND
 ((SIMPLEP VAL 3)
 VAL)
 (T (**RECORDBIND** VAL])

(RECORDPRIORITY

[LAMBDA (RECNAME PRIORITY) (* rmk%: "30-JUN-82 23:21")

;; This is hackish--shouldn't really smash the user's declaration, cause it might be of a different form given by his own translation function.

(PROG (TRAN PREV (DECL (**RECLOOK** RECNAME)))
 (SETQ TRAN (**RECORDECL** DECL))
 (SETQ PREV (SELECTQ (RECORD.PRIORITY TRAN)
 (NIL 'USER)
 'SYSTEM))
 (SELECTQ PRIORITY
 (USER (COND
 ((NEQ PREV 'USER)
 (/DREMOVE (ASSOC 'SYSTEM DECL)
 DECL)
 (SET.RECORD.PRIORITY TRAN NIL))))
 (SYSTEM [COND
 ((NEQ PREV 'SYSTEM)
 (/NCONC1 DECL (CONS 'SYSTEM))
 (SET.RECORD.PRIORITY TRAN 'SYSTEM])
 NIL)
 (RETURN PREV])

(RECORDACCESSFORM

[LAMBDA (FIELD DATUM TYPE NEWVALUE) (* rrb "28-OCT-83 16:30")

;; returns the form that results from a record access.

(PROG [EXP (TYPE (COND
 (TYPE (L-CASE TYPE))
 (T 'fetch]
 (SETQ EXP (SELECTQ TYPE
 ((fetch ffetch)
 (LIST TYPE FIELD 'OF DATUM))
 (LIST TYPE FIELD 'OF DATUM 'WITH NEWVALUE)))
 (RETURN (COMPILEUSERFN (CDR EXP)
 EXP])

)

(DEFINEQ

(RECORDWORD

[LAMBDA (WORD TL WORDTYPE) (* lmm "29-SEP-78 16:51")

(PROG (NEWWORD)

```
(RETURN (COND
  ([AND (SETQ NEWWORD (GETPROP WORD 'CLISPPWORD))
    (EQ (CAR NEWWORD)
      (OR WORDTYPE 'RECORDTRAN]
    [COND
      [(LISTP (CDR NEWWORD))
        (SETQ NEWWORD (CADR NEWWORD))
        (SETQ WORD (RECORDWORD (CADDR NEWWORD)
          (T (SETQ WORD (SETQ NEWWORD (CDR NEWWORD)
            (AND LCASEFLG TL NEWWORD (NEQ (CAR TL)
              NEWWORD)
            (/RPLACA TL NEWWORD))
          WORD])
```

(MAKECREATEO

```
[LAMBDA (RECORD.TRAN HASHLINKS NEEDACELL) (* lmm "23-SEP-78 02:08")
  (PROG ((FIELDINFO (RECORD.FIELDINFO RECORD.TRAN)))
    (RETURN (MAKECREATE1 (CAR (RECORD.CREATEINFO RECORD.TRAN))
      (CDR (RECORD.CREATEINFO RECORD.TRAN))
      NEEDACELL])
```

(MAKECREATE1

```
[LAMBDA (TYPE CREATEINFO NEEDACELL) ; Edited 21-Jul-88 18:14 by jrb:
  (PROG (DEF TEM TEM3 VAL SMASHFIELDS (USINGTYPE USINGTYPE)
    BINDINGS
    (CKVALFLG T))
    (AND HASHLINKS (SETQ NEEDACELL T))
    [if (EQ USINGTYPE 'smashing)
      then
        (SETQ DEF
          (SELECTQ TYPE
            (RECORD (if (LISTP CREATEINFO)
              then (SMASHPATTERN USINGEXPR CREATEINFO)
              else (MAKECREATLST CREATEINFO USINGEXPR NEEDACELL)))
            (TYPERECORD (SMASHPATTERN USINGEXPR CREATEINFO (LIST 'QUOTE (CAR CREATEINFO))))
            (ARRAYRECORD [SETQ SMASHFIELDS
              (DREVERSE (for FIELD in (CREATEFIELDS (CDR CREATEINFO))
                when (NEQ (SETQ VAL (GETFIELDFORCREATE FIELD USINGEXPR T T
                  USINGTYPE NIL T))
                  MSBLIP)
                collect (LIST FIELD VAL]
              USINGEXPR)
            ((ARRAYBLOCK DATATYPE)
              (SETQ DEF USINGEXPR)
              (for FIELD in (DREVERSE (CREATEFIELDS (CADR CREATEINFO)))
                when (NEQ (SETQ VAL (GETFIELDFORCREATE FIELD USINGEXPR 0 T USINGTYPE (CADDR
                  CREATEINFO
                  )))
                  MSBLIP)
              do (SETQ DEF (LIST (COND
                ((NULL CKVALFLG)
                  'FREPLACEFIELDVAL)
                (T (SETQ CKVALFLG)
                  'REPLACEFIELDVAL))
                [KWOTE (CDDR (ASSOC FIELD (CADDR CREATEINFO)
                  DEF VAL)))
                DEF)
              (CCREATE
                (PROG (FIELD.USAGE [SPECIALFIELDS (COPY ' ((DATUM CREATE)
                  (OLDDATUM USING]
                  (DECLST ' (FAST))
                  VAR1
                  (SUBTYPE 'CREATE))
                [SETQ DEF (CSUBST (COND
                  ((EQ TYPE 'CCREATE)
                    (EVAL (CAR CREATEINFO)))
                  (T (CAR CREATEINFO)
                    (COND
                      ((EQ (CADAR SPECIALFIELDS)
                        'CREATE)
                        ;; if this wasn't an 'advice' -- i.e. if didn't do the regular create when we saw DATUM , then need to
                        ;; make sure that the using/copying/default fields are incorporated
                        (SETQ SMASHFIELDS
                          (for x in FIELDINFO
                            when (NOT (OR (NULL (CAR X))
                              (ASSOC (CAR X)
                                FIELD.USAGE)
                              (ASSOC (CAR X)
                                FIELDS.IN.CREATE)
                              (EQ (SETQ TEM (GETFIELDFORCREATE
                                (CAR X)
                                USINGEXPR NIL T
                                (SELECTQ USINGTYPE
                                  (reusing 'using)
```



```

collect (LIST X TEM3)))
NIL)
;; GETFIELDFORCREATE returns MSBLIP if USINGTYPE = (QUOTE reusing) and the field does not
;; occur. All other reusing types are handled later, thus USINGTYPE is re-bound
;; TEM is the list of VALUES specified, where FIELD_VAL is included; plain USING expressions are not,
;; and only non-nil universal defaults are handled, but explicit defaults are there
(SETQ DEF ' (GENSYM))
(SELECTQ USINGTYPE
  (NIL (SETQ SMASHFIELDS TEM)
    DEF)
  (LIST 'PROGN [LIST 'SETPROPLIST (SETQ DEF (RECORDBIND DEF))
    (SELECTQ USINGTYPE
      (copying (CONS (FUNCTION COPYALL)
        (LIST (LIST 'GETPROPLIST USINGEXPR)
          )))
      (CONS (FUNCTION APPEND)
        (LIST (LIST 'GETPROPLIST USINGEXPR]
        DEF)))
    (ARRAYRECORD [SETQ SMASHFIELDS
      (DREVERSE (for FIELD in (CREATEFIELDS (CDR CREATEINFO))
        when (NEQ (SETQ VAL (GETFIELDFORCREATE FIELD USINGEXPR T T
          USINGTYPE))
            MSBLIP)
          collect (LIST FIELD VAL]
      (SELECTQ USINGTYPE
        ((using reusing)
          (COND
            ((OR SMASHFIELDS NEEDACELL)
              (SETQ SMASHFIELDS)
              (SETQ CKVALFLG)
              (LIST 'COPYARRAY USINGEXPR))
            (T (RETURN MSBLIP))))
          (copying (SETQ SMASHFIELDS)
            (LIST 'COPYALL USINGEXPR))
          (NIL (SETQ SMASHFIELDS (SUBSET SMASHFIELDS (FUNCTION CADR)))
            (SETQ CKVALFLG)
            (LIST 'ARRAY (CAR CREATEINFO)))
          (SHOULDNT)))
      ((ARRAYBLOCK DATATYPE)
        [SETQ DEF (SELECTQ USINGTYPE
          (copying (LIST 'COPYALL USINGEXPR))
          (COND
            [(EQ TYPE 'ARRAYBLOCK)
              (SETQ CKVALFLG)
              (COND
                (USINGTYPE (LIST 'COPYARRAY USINGEXPR))
                (T (LIST 'ARRAY (CAAR CREATEINFO)
                  (CDAR CREATEINFO]
                  (T (SETQ CKVALFLG)
                    (CONS 'NCREATE (CONS (KWOTE (CAR CREATEINFO))
                      (AND USINGTYPE (LIST USINGEXPR]
                    (for FIELD in (DREVERSE (CREATEFIELDS (CADR CREATEINFO)))
                      when (NEQ (SETQ VAL (GETFIELDFORCREATE FIELD USINGEXPR 0 T (SELECTQ USINGTYPE
                        (NIL USINGTYPE)
                        'reusing)
                          (CADDR CREATEINFO)))
                        MSBLIP)
                      do (SETQ DEF (LIST (COND
                        ((NULL CKVALFLG)
                          'FREPLACEFIELDVAL)
                        (T (SETQ CKVALFLG)
                          'REPLACEFIELDVAL))
                        [KWOTE (CDDR (ASSOC FIELD (CDDDR CREATEINFO)
                          DEF VAL)))
                        (COND
                          ((AND (NOT NEEDACELL)
                            (EQ USINGTYPE 'reusing)
                            (NEQ (CAR DEF)
                              'FREPLACEFIELD))
                            (RETURN MSBLIP)))
                        DEF)
                      ((CREATE CCREATE)
                        (PROG (FIELD.USAGE [SPECIALFIELDS (COPY ' (DATUM CREATE)
                          (OLDDATUM USING]
                            (DECLST ' (FAST))
                            VAR1
                            (SUBTYPE 'CREATE))
                        [SETQ DEF (CSUBST (COND
                          ((EQ TYPE 'CCREATE)
                            (EVAL (CAR CREATEINFO)))
                          (T (CAR CREATEINFO]
                          [COND
                            ((EQ (CADAR SPECIALFIELDS)
                              'CREATE)
                              ;; if this wasn't an 'advice' -- i.e. if didn't do the regular create when we saw DATUM , then need to
                              ;; make sure that the using/copying/default fields are incorporated

```



```

                                (SHOULDNT)))
                                (FRPLNODE (CDR TEM)
                                (COND
                                  ((SIMPLEP TEM2)
                                   2)
                                  (T (SETQ TEM2 (LIST 'PROGN TEM2))
                                   1))
                                TEM2)
                                TEM2)))
(FMEMB X FIELDNAMES)
(SELECTQ SUBSTYPE
 (CREATE (RECORD.FIELD.VALUE0 X))
 (WITH (MAKEACCESS (CAR (ACCESSDEF4 (LIST X)
                                     RECORD.TRAN))
                  USINGEXPR NIL 'fetch))
 (SHOULDNT)))
(T X))
[[LISTP (SETQ TEM (GETP (CAR X)
                        'CLISPWORD)]
 (SELECTQ (CDR TEM)
 (type? the)
 (RECONS (CAR X)
          (RECONS (CADR X)
                  (CSUBSTLST (CDDR X))
                  (CDR X))
          X))
(create
 (PROG ((VAL (LIST (CAR X)
                  (CADR X)))
        (X (CDDR X)))
 LP [COND
      ((NLISTP X)
       (RETURN VAL))
      ((EQ (CADR X)
           '_)
       [NCONC VAL (LIST (CAR X)
                       (CADR X)
                       (CSUBST (CADDR X]
                       (SETQ X (CDDDR X)))]
      ((RECORDWORD (CAR X))
       [NCONC VAL (LIST (CAR X)
                       (CSUBST (CADR X]
                       (SETQ X (CDDR X)))]
      (T (NCONC1 VAL (CSUBST (CAR X)))
         (SETQ X (CDR X]
         (GO LP)))
 (SELECTQ (CAR TEM)
 ((RECORDTRAN RECORDWORD)
 (RECONS (CAR X)
          (RECONS (CADR X)
                  (CSUBSTLST (CDDR X))
                  (CDR X))
          X))
 (MATCHWORD [PROG NIL
              (DWIMIFYREC (LIST X)
                          NIL RECORDEXPRESSION)
              (RETURN (CSUBST (OR (GETHASH X CLISPARRAY)
                                (RETURN (RECONS (CAR X)
                                                (RECONS (CSUBST
                                                         (CADR X))
                                                         (CDDR X)
                                                         (CDR X))
                                                         X]))
              ; some other clisp word
              (PROGN
                (RECONS (CAR X)
                        (CSUBSTLST (CDR X))
                        X]
                ((EQ (CAR X)
                     'QUOTE)
                 X)
                ((AND (LISTP (CAR X))
                      (EQ (CAAR X)
                          'LAMBDA))
                 (SETQ TEM (CSUBSTLST (CDR X)))
                 (RECONS (RECONS (CAAR X)
                                (RECONS (CADAR X)
                                        (CSUBSTLST (CDDAR X))
                                        (CDAR X))
                                (CAR X))
                 TEM X))
 ((SELECTQ SUBSTYPE
 (WITH (AND (EQ (CAR X)
                'SETQ)
            (FMEMB (CADR X)
                  FIELDNAMES)
 (MAKEACCESS (CAR (ACCESSDEF4 (LIST (CADR X))
                                RECORD.TRAN))

```

```

USINGEXPR
(CSUBSTLST (CDDR X))
'replace))
(REPLACE (RECONS (RECLISPLOOKUP (CSUBST (CAR X))
DECLST
(CAR ARGS))
(CSUBSTLST (CDR X))
X))
(CHANGE [COND
([OR (EQ (CAR (SETQ TEM X))
'DATUM_)
(AND (EQ (CAR X)
'SETQ)
(EQ (CAR (SETQ TEM (CDR X)))
'DATUM])
(COPY1 (SUBPAIR 'NEWVALUE (MKPROGN (CSUBSTLST (CDR TEM)))
(CADDR ARGS])
NIL))
(T (RECONS (CSUBST (CAR X))
(CSUBSTLST (CDR X))
X])

```

```

(RECONS
[LAMBDA (X Y C)
(COND
((AND (EQ X (CAR C))
(EQ Y (CDR C)))
C)
(T (CONS X Y]))
(* Imm "11-AUG-78 10:20")

```

```

(COPY1
[LAMBDA (X)
(COND
((LISTP X)
(CONS (CAR X)
(CDR X)))
(T (LIST 'PROGN X]))
(* Imm "31-JUL-78 04:11")

```

```

(CSUBSTLST
[LAMBDA (X)
(COND
((NLISTP X)
(AND X (CSUBST X)))
(T (RECONS (CSUBST (CAR X))
(CSUBSTLST (CDR X))
X]))
(* Imm "11-AUG-78 10:26")

```

```

(RECORD.FIELD.VALUE
[LAMBDA (FIELDNAME)
(PROG (TMP)
(RETURN (COND
((SETQ TMP (ASSOC FIELDNAME FIELDS.IN.CREATE))
(CADR TMP))
(T (GETFIELDFORCREATE FIELDNAME USINGEXPR T T USINGTYPE]))
(* Imm "13-Mar-85 16:12")

```

```

(RECORD.FIELD.VALUE0
[LAMBDA (FIELDNAME)
(CDAR (SETQ FIELD.USAGE (CONS (CONS FIELDNAME (GETFIELDFORCREATE FIELDNAME USINGEXPR T T USINGTYPE))
FIELD.USAGE]))
(* Imm "31-JUL-78 03:00")

```

```

(MAKECREATELST
[LAMBDA (TEMPLATE USING NEEDACELL)
;; Make the create expression for regular RECORD declaration (i.e. LISTRECORDS)
(MAKECREATELST1 TEMPLATE T USING NEEDACELL])
(* Imm "22-AUG-84 23:15")

```

```

(SMASHPATTERN
[LAMBDA (X PATTERN CARVAL EFF)
(if (LITATOM X)
then (CONS 'PROGN (SMASHPAT1 PATTERN X CARVAL EFF))
else ([LAMBDA (XV)
'([LAMBDA (% , XV)
(SMASHPAT1 PATTERN XV CARVAL EFF)
% , X]
(RECORDGENSYM]))
(* Imm "23-AUG-84 00:27")

```

```

(SMASHPAT1
[LAMBDA (PATTERN XV CARVAL EFF)
(* Imm "23-AUG-84 00:26")

```



```

[ (AND USETYPE (NEQ USETYPE 'smashing))
  (SETQ VALUE (OR (SUBFIELDCREATE MSBLIP)
    (SELECTQ USETYPE
      (reusing MSBLIP)
      (copying (LIST 'COPYALL USINGEXPR)
        USINGEXPR)
    ))
  ((SETQ TEM (ASSOC RNAME DEFAULTS)) ; Is there a specific default for this field?
    (SETQ DEFFLG T)
    (SETQ VALUE (CADR TEM)))
  (T (RETURN (OR (SUBFIELDCREATE MSBLIP)
    (PROGN (SETQ TEM (ASSOC 'DEFAULT DEFAULTS))
      (SELECTQ USEUNIVDEFAULT
        (0 (COND
          ((EQ USINGTYPE 'smashing)
            (CDR (ASSOC RNAME TOPDEFAULTS)))
          (T MSBLIP)))
        (NOTNIL (OR (CADR TEM)
          MSBLIP))
        (NIL MSBLIP)
        (CADR TEM)
      ))
    ))
  (RETURN (OR (SUBFIELDCREATE VALUE DEFFLG)
    VALUE]))

```

(SUBFIELDCREATE

(* lmm "13-Mar-85 16:12")

```

[LAMBDA (VAL DFLT)
  (PROG (TEM SUBDECL SUBTRAN HL)
    (SETQ HL (for DEC in (SUBDECLARATIONS RECORD.TRAN)
      when [AND (EQ (RECORD.NAME (SETQ TEM (RECORDECLO DEC)))
        RNAME)
        (OR (EQ (CAR (RECORD.CREATEINFO TEM))
          'HASHRECORD)
          (COND
            ((NULL SUBDECL) ; set SUBDECL and SUBTRAN to FIRST sub-declaration for this
              ; field, collecting HL separately
              (SETQ SUBDECL DEC)
              (SETQ SUBTRAN TEM)
              NIL]
          ))
      collect TEM) )

```

;; Then create the sub-record, putting on both the embedded hashlinks and the one from this record: e.g. (create (RECORD A (B . C)
 ;; (HASHRECORD B (RECORD (E . F))) (RECORD B (D . G) (HASHRECORD (FOO) DEFAULT _ (CONS))))
 ;; the VAL arg is what was given for the field in the create .. e.g. (RECORD A (B . C) (HASHLINK B FOO)) need both the value given for B and
 ;; the value given for FOO

```

[COND
  ([OR (EQ VAL MSBLIP)
    (AND DFLT (SOME (RECORD.FIELDNAMES SUBTRAN)
      (FUNCTION (LAMBDA (X)
        (ASSOC X FIELDS.IN.CREATE)
        ; if this field was not specified, then we do an implicit CREATE
        ; on the subdeclaration, if any
      ))
    ))
  (OR (NULL SUBTRAN)
    (EQ (SETQ TEM (MAKECREATEO SUBTRAN))
      MSBLIP)
    (SETQ VAL TEM)
  (RETURN (COND
    ((NULL HL)
      (AND (NEQ VAL MSBLIP)
        VAL))
    ((EQ VAL MSBLIP) ; Since the field has no content, the hashlink cannot either
      NIL)
    (T (MAKEHASHLINKS VAL HL])

```

(MAKEHASHLINKS

(* lmm "5-OCT-78 05:41")

```

[LAMBDA (DEF TRANS)
  (PROG (TEM TEM2 BINDINGS)
    (COND
      ((NULL TRANS)
        (RETURN DEF))
    (SETQ TEM2 (for RECORD.TRAN in TRANS when (SETQ TEM (GETFIELDFORCREATE (CADR (RECORD.CREATEINFO
      RECORD.TRAN))
        USINGEXPR T T (SELECTQ USINGTYPE
          (reusing 'using)
          USINGTYPE)))
      collect (COND
        ((EQ USINGTYPE 'smashing)
          TEM)
        (T (CONS 'PUTHASH (CONS (SETQ DEF (RECORDBINDVAL DEF))
          (CONS TEM (CDDR (RECORD.CREATEINFO RECORD.TRAN]
      ))
    (RETURN (EMBEDPROG (MKPROGN (DREVERSE (CONS DEF TEM2]))

```

(HASHLINKS

(* lmm "7-OCT-77 15:50")

```

[LAMBDA (TRAN)
  (for DEC in (SUBDECLARATIONS TRAN) bind DEC1 when (SELECTQ [CAR (RECORD.CREATEINFO (SETQ DEC1 (RECORDECL
    DEC]

```

```
(HASHRECORD (OR (NULL (RECORD.NAME DEC1))
                 (EQ (RECORD.NAME TRAN)
                     (RECORD.NAME DEC1))))
NIL)
```

collect DEC1])

(RECLOOK

```
[LAMBDA (RECNAME TL LOCALDEC PARENT ERROR) (* Imm " 7-AUG-84 23:23")
```

;; Look for a declaration of a record named RECNAME

```
(OR (COND
     ((NULL RECNAME)
      NIL)
     [(NLISTP RECNAME)
      (CAR (OR (RECLOOK1 RECNAME LOCALDEC)
               (RECLOOK1 RECNAME USERRECLST)
               ((RECDEC? RECNAME)
                RECNAME))
            (AND ERROR (PROG (TEM)
                             (AND TL (SETQ TEM (FIXSPELL RECNAME 70 [NCONC [MAPCONC LOCALDEC
                                                                                   (FUNCTION (LAMBDA (X)
                                                                                     (AND
                                                                                       (SETQ X (RECORDECL
                                                                                         X))
                                                                                       (LIST (RECORD.NAME
                                                                                             X])
                                                                                   (MAPCAR USERRECLST
                                                                                   (FUNCTION (LAMBDA (DEC)
                                                                                     (RECORD.NAME
                                                                                       (RECORDECL DEC]
                                                                                   " -> " TL NIL NIL NIL T))
                (RETURN (RECLOOK TEM NIL LOCALDEC PARENT NIL)))
            (PROG ((FAULTFN))
                  (RECORDERROR 'NAME RECNAME PARENT])
```

(ALLFIELDS

```
[LAMBDA (TRAN) (* Imm " 5-SEP-83 13:09")
```

```
(NCONC [for Y in (RECORD.SUBDECS TRAN) when (EQ (CAR Y)
                                                  'SUBRECORD)
        join (APPEND (ALLFIELDS (RECORDECL (RECLOOK (CADR Y)
                                                    NIL DECLST Y T])
                    (RECORD.FIELDNAMES TRAN])
```

(SUBDECLARATIONS

```
[LAMBDA (TRAN) (* Imm " 7-OCT-77 16:46")
```

```
(for Y in (RECORD.SUBDECS TRAN) collect [COND
     ((EQ (CAR Y)
          'SUBRECORD)
      (PROG ((TEM (RECLOOK (CADR Y)
                          NIL DECLST Y T)))
            (SETQ Y (COND
                    [(CDDR Y)
                     (COND
                      ((EQ (CAR TEM)
                          CLISPTRANFLG)
                       (CDDR TEM))
                      (T (APPEND TEM (CDDR Y)
                                  (T TEM]
                    Y])
```

)

(DEFINEQ

(CLISPRECORD

```
[LAMBDA (E FIELD SETQFLG) (* Imm "13-OCT-78 01:57")
```

;; This is the entry to the record package for fetch and replace statements as well as for direct inputs like X:FIELD and X:FIELD_VALUE.

```
(PROG ((DECLST (GETLOCALDEC EXPR FAULTFN))
       (RETURN (COND
                [SETQFLG (COND
                        ((AND FIELD (NLISTP FIELD)) ; X : FIELD input
                          ; X:FIELD_expression is done in two passes; this is the first
                        (AND (OR (RECORDFIELD? FIELD DECLST)
                               (AND DECLST (RECORDFIELD? FIELD)))
                          (LIST 'REPLACE FIELD (COND
                                (LCASEFLG 'of)
                                (T 'OF))
                                E)))
                        ((NEQ (CAR E)
                          'REPLACE)
                         (SHOULDNT))
                        (T ; This is the second pass of the X:FIELD_expression input
                         (RECORDTRAN (NCONC [FRPLACA E (RECLISPLOOKUP (COND
```

(LCASEFLG 'replace)
(T 'REPLACE]

```

(CONS (COND
      (LCASEFLG 'with)
      (T 'WITH))
      FIELD]
(T (RECORDTRAN (CONSFN (COND
                      (LCASEFLG 'fetch)
                      (T 'FETCH))
      (LIST FIELD (COND
                  (LCASEFLG 'of)
                  (T 'OF))
      E]))

```

(ACCESSDEF

(* lmm "22-MAY-80 21:35")

```

[LAMBDA (FIELD V1 TL CFLG)
  (PROG (RECS CHRLST DOTTAIL TEM FIELDLST)
    RETRY

```

```

      (COND
        ([AND (LISTP FIELD)
              (FMEMB (RECORDWORD (CAR FIELD))
                    ' (fetch FETCH)
              (RETURN)))

```

```

      [COND
        ([AND [OR (NLISTP FIELD)
                  (AND (NULL (CDR FIELD))
                      (SETQ FIELD (CAR FIELD)
                      (SETQ RECS (OR (RECFIELDLOOK DECLST FIELD V1)
                                    (RECFIELDLOOK USERRECLST FIELD)

```

; RECFIELDLOOK returns a list of of declarations

```

      (RETURN (CHECKDEFS (for DEC in RECS join (ACCESSDEF4 (LIST FIELD)
                                                         (RECORDECL DEC)))
                    RECS FIELD T]

```

```

      [COND
        ((LISTP FIELD)
         (RETURN (RECORDCHAIN FIELD]

```

```

      (AND (NOT CFLG)
           (COND
            [(SETQ TEM (GETP FIELD 'ACCESSFN)) ; CFLG says it is from a CREATE
             (SETQ NOTRANFLG T)
             (RETURN (LIST (LIST 'ACCESSFNS FIELD TEM (GETP TEM 'SETFN]
            (AND [SETQ TEM (FMEMB '%: (SETQ CHRLST (UNPACK FIELD)
                    (NEQ TEM CHRLST))
            [/RPLNODE TL (SETQ FIELD (PACK (CDR TEM)))
                    (CONS 'OF (CONS (SETQ V1 (PACK (LDIFF CHRLST TEM)))
                                    (CDR TL]

```

```

      (GO RETRY))
      [(SETQ DOTTAIL (FMEMB '%. CHRLST))
       ;; check if FIELD contains a . within it, e.g. AB.CD. TL must be the tail of the input expression starting with FIELD

```

```

      (RETURN (PROG1 [RECORDCHAIN (SETQ FIELDLST (PROG ((TEM DOTTAIL)
                                                         R)
                                                         ; collect the atoms with .'s removed e.g. A.B.CD.E -> (A B CD E)
                                                         LP [COND

```

```

                                                         ((NULL TEM)
                                                          (RETURN (NCONC1 R
                                                         (COND
                                                          ((CDR CHRLST)
                                                           (PACK CHRLST))
                                                          (T (CAR CHRLST]
                                                         [SETQ R
                                                         (NCONC1 R
                                                         (COND
                                                         ((EQ (CDR CHRLST)
                                                         TEM)
                                                          (CAR CHRLST))
                                                         (T (PACK (LDIFF CHRLST TEM]
                                                         [SETQ TEM (FMEMB '%. (SETQ CHRLST
                                                         (CDR TEM]
                                                         (GO LP]

```

```

      (FRPLACA (OR TL (SHOULDNT))
               FIELDLST))
      ((SETQ TEM (FIXSPELL FIELD 70 (NCONC (FIELDNAMESIN DECLST)
                                           (FIELDNAMESIN USERRECLST))
               NIL TL NIL NIL NIL T)) ; Finally, attempt spelling correction
      (SETQ FIELD TEM)
      (GO RETRY))
      (T (RETURN])

```

(FIELDNAMESIN

(* lmm "12-SEP-77 02:19")

```

[LAMBDA (DECS)
  (MAPCONC DECS (FUNCTION (LAMBDA (X)

```

```

    (APPEND (RECORD.FIELDNAMES (RECORDECL X])

```

(ACCESSDEF4

(* Imm "13-Mar-85 16:12")

```

[LAMBDA (LST TRAN TL)
  (PROG (TEM SUBDECS AVOID)
    (RETURN (COND
      [[SETQ TEM (CDR (ASSOC (CAR LST)
        (RECORD.FIELDINFO TRAN)
        ;; The FIELDINFO part of the translation contains (fieldname type tokens) for TOP LEVEL fields --- this name (CAR LST) is
        ;; declared in this declaration
      [COND
        ([AND (NULL TL)
          (FMEMB 'CHECK (CDR (RECORD.TYPECHECK TRAN)
            (SETQ TL (CONS (CONS 'THE (RECORD.NAME TRAN))
              TL])
          (COND
            ((NULL (CDR LST))
              (LIST (JOINDEF TEM TL)))
            (T (OR (AND (SETQ SUBDECS (RECFIELDLOOK (RECORD.SUBDECS TRAN)
              (CADR LST)))
                (ALLPATHS (RECLOOK1 (CAR LST)
                  SUBDECS)
                  (CDR LST)
                  (JOINDEF TEM TL)))
                (TOPPATHS (CAR LST)
                  (CDR LST)
                  (JOINDEF TEM TL)
                ; Found (CAR LST) in a sub-declaration
            (T (for SUBDEC in (RECFIELDLOOK (RECORD.SUBDECS TRAN)
              (CAR LST))
              join (ALLPATHS (LIST SUBDEC)
                LST
                (JOINDEF [CDR (OR (ASSOC (SETQ TEM (RECORD.NAME (RECORDECL SUBDEC)))
                  (RECORD.FIELDINFO TRAN))
                (COND
                  ((OR (EQ TEM (RECORD.NAME TRAN))
                    (NULL TEM))
                    NIL)
                  (T (SHOULDNT]
                TL]))

```

(MAKEACCESS

(* Imm "1-AUG-78 00:58")

```

[LAMBDA (ACCESS BODY NEWVAL TYPE)
  (COND
    ((NULL ACCESS)
      (SELECTQ TYPE
        (fetch BODY)
        (SHOULDNT)))
    (T (MAKEACCESS1 (CAAR ACCESS)
      (CDAR ACCESS)
      (MAKEACCESS (CDR ACCESS)
        BODY NIL 'fetch)
      NEWVAL TYPE BODY]))

```

(MAKEACCESS1

(* Imm "23-SEP-78 01:17")

```

[LAMBDA (RECTYPE SPEC DAT NEWVAL TYPE BODY)
  (COND
    ((AND (NEQ TYPE 'fetch)
      (EQ RECTYPE 'RECORD)
      (CDR SPEC))
      (MAKEACCESS1 RECTYPE (LIST (CAR SPEC))
        (MAKEACCESS1 RECTYPE (CDR SPEC)
          DAT NIL 'fetch)
        NEWVAL TYPE BODY))
    ((EQ TYPE 'change)
      (LIST (MAKEACCESS1 RECTYPE SPEC (SETQ DAT (RECORDBINDVAL DAT))
        NIL
        'fetch)
        NIL
        (MAKEACCESS1 RECTYPE SPEC DAT NEWVAL 'replace BODY)))
    (T (SELECTQ RECTYPE
      (RECORD [SELECTQ TYPE
        (replace (COND
          ((CDR SPEC)
            (SHOULDNT)))
          (LIST (SELECTQ (CAR SPEC)
            (A 'CAR)
            (D 'CDR)
            (RECORDERROR 'REPLACE RECORDEXPRESSION))
            (CONSFN (SELECTQ (CAR SPEC)
              (A 'RPLACA)
              'RPLACD)
              (CONS DAT NEWVAL))))
          (COND
            [(CDDDDR SPEC)
              (LIST (PACK* 'C (CAR SPEC)

```



```

(CADR SPEC)
(CADDR SPEC)
(CADDDR SPEC)
'R)
(MAKEACCESS1 RECTYPE (CDDDDR SPEC)
DAT NIL 'fetch]
(NULL SPEC)
DAT)
(T (LIST [PACK (CONS 'C (APPEND SPEC (LIST 'R)
DAT]))
(HASHRECORD (SELECTQ TYPE
(replace (CONSFN 'PUTHASH (CONS DAT (CONS (CAR NEWVAL)
SPEC))))
(CONS 'GETHASH (CONS DAT SPEC))))
(ACCESSFNS (MKACCESSFN (SELECTQ TYPE
(replace (CADDR SPEC)
(CADR SPEC))
(CONS DAT NEWVAL)
TYPE
(CAR SPEC))))
(CACCESSFNS (MKACCESSFN (RECEVAL (SELECTQ TYPE
(replace (CADDR SPEC)
(CADR SPEC))
DAT
(MKPROGN (CAR NEWVALUE))
(CAR SPEC))
(CONS DAT NEWVAL)
TYPE
(CAR SPEC)))
(PROPRECORD (CONSFN (SELECTQ TYPE
(replace 'LISTPUT)
'LISTGET)
(CONS DAT (CONS (KWOTE SPEC)
NEWVAL))))
(ATOMRECORD (CONSFN (SELECTQ TYPE
(replace 'PUTPROP)
'GETPROP)
(CONS DAT (CONS (KWOTE SPEC)
NEWVAL))))
(ASSOCRECORD [SELECTQ TYPE
(replace (CONSFN 'PUTASSOC (CONS (KWOTE SPEC)
(LIST (CAR NEWVAL)
DAT))))
(LIST 'CDR (CONSFN 'ASSOC (LIST (KWOTE SPEC)
DAT))]
(ARRAYRECORD (CONSFN [SELECTQ TYPE
(replace (COND
((LISTP SPEC)
'SETD)
(T 'SETA)))
(COND
((LISTP SPEC)
'ELTD)
(T 'ELT]
(CONS DAT (CONS (COND
((LISTP SPEC)
(CDR SPEC))
(T SPEC))
NEWVAL))))
(DATATYPE (CONSFN (SELECTQ TYPE
(replace 'REPLACEFIELD)
'FETCHFIELD)
(CONS (KWOTE SPEC)
(CONS DAT NEWVAL))))
(THE (SELECTQ TYPE
(replace (SHOULDNT))
(LIST (COND
((FMEMB 'FAST DECLST)
'FTHE)
(T 'THE))
SPEC DAT)))
(SHOULDNT])

```

(MKACCESSFN

(* Imm "19-OCT-78 00:47")

```

[LAMBDA (FN ARGS TYPE FIELD)
(COND
((NULL FN)
(RECORDERROR (SELECTQ TYPE
(replace 'REPLACE)
'FETCH)
FIELD RECORDEXPRESSION)))
(COND
((EQ FN 'DATUM)
(CAR ARGS))
((OR (NLISTP FN)
(EQ (CAR FN)

```

```

      (LAMBDA))
    (CONSFN FN ARGS)
  [(FMEMB (CAR FN)
    (FAST STANDARD UNDOABLE))
   [SETQ FN (CLISPLOOKUP0 NIL (CAR ARGS)
     (CADR ARGS)
     (OR DECLST ' (DUMMY))
     (CADR FN)
     'DUMMY
     (LIST 'ACCESS (LISTGET FN 'STANDARD)
       (LISTGET FN 'UNDOABLE)
       (LISTGET FN 'FAST)
       (PROG ((DECLST (CONS 'STANDARD DECLST)))
         (RETURN (MKACCESSFN FN ARGS TYPE FIELD)
           (T (PROG (FIELDNAMES [SPECIALFIELDS (COPY ' (DATUM DATUM)
             (NEWVALUE NEWVALUE)
             (PARENT PARENT])
             (SUBSTTYPE 'REPLACE))
             (RETURN (CSUBST FN])

```

(RECFIELDLOOK

```
[LAMBDA (RECLST FIELD VAR EDITRECFLG) (* Imm "18-SEP-78 19:03")
```

:: Looks up on either local or global declst for records relavant to field and var

```

(for Y in RECLST join (AND (LISTP Y)
  (COND
    ((EQ (CAR Y)
      'RECORDS)
     (RECFIELDLOOK [MAPCAR (CDR Y)
      (FUNCTION (LAMBDA (X)
        (RECLOOK X]
        FIELD VAR))
     ((EQ (CAR Y)
      'SUBRECORD)
     (RECFIELDLOOK (LIST (RECLOOK (CADR Y)))
      FIELD VAR))
     ((AND VAR (EQ (CAR Y)
      VAR))
     (RECFIELDLOOK (CDR Y)
      FIELD))
     ([OR (FMEMB FIELD (RECORD.FIELDNAMES (RECORDECL Y)))
      (AND EDITRECFLG (EQ FIELD (RECORD.NAME (RECORDECL Y]
      (LIST Y])

```

(RECORDCHAIN

```
[LAMBDA (LST) (* Imm "23-SEP-78 02:08")
```

:: Search for the sequence of record declarations which are for the sequence of field names given in LST. (e.g. if LST is (A B) will look for the declaration of A which contains B) Return the list of declarations. The name of each declaration (except the first) should be a field in the previous one

```

(CHECKDEFS (TOPPATHS (CAR LST)
  (CDR LST))
  NIL LST T])

```

(RECLOOK1

```
[LAMBDA (RECNAME DECS AVOIDDECS) (* Imm%: "27-JUL-76 04:13:50")
```

:: Search DECS for declaration with name RECNAME

```

(SUBSET DECS (FUNCTION (LAMBDA (DEC)
  (AND (NOT (FMEMB DEC AVOIDDECS))
    (EQ (RECORD.NAME (RECORDECL DEC))
      RECNAME]))

```

(SYSRECLOOK1

```
[LAMBDA (RECNAME) (* rmk%: " 4-JAN-82 17:12")
```

:: returns the declaration of a system record.

```

(DECLARE (GLOBALVARS SYSTEMRECLST))
(for D in SYSTEMRECLST when (EQ RECNAME (CADR D)) do (RETURN D])

```

(TOPPATHS

```
[LAMBDA (FIELD LST TL DECS AVOID) (* Imm "25-AUG-78 13:41")
```

```

(ALLPATHS (OR (RECLOOK1 FIELD DECS)
  (RECLOOK1 FIELD DECLST)
  (RECLOOK1 FIELD USERRECLST))
  LST TL])

```

(ALLPATHS

```
[LAMBDA (DECLS LST TL) (* Imm "24-FEB-79 12:08")
```

```

(PROG (TRAN ANY DEFS DEC)
  (COND

```

```

((NULL DECLS)
 (RETURN))
(SETQ DEFS (for DEC in DECLS when [AND (NOT (FMEMB DEC AVOID))
                                       (FMEMB (CAR LST)
                                               (RECORD.FIELDNAMES (SETQ TRAN (RECORDECL DEC]
                                                                     join (SETQ ANY T)
                                                                     (ACCESSDEF4 LST TRAN TL)))
(RETURN (COND
 (ANY DEFS)
 (T (SETQ DEFS (APPEND DECLS AVOID))
     (for DEC in DECLS when (NOT (FMEMB DEC AVOID))
       join (NCONC (ALLPATHS (RECLOOK1 (RECORD.NAME (SETQ TRAN (RECORDECL DEC)))
                                     (RECORD.SUBDECS TRAN)
                                     AVOID)
                 LST TL)
         (PROGN [COND
                ([AND (NULL TL)
                      (FMEMB 'CHECK (CDR (RECORD.TYPECHECK TRAN)
                                       (SETQ TL (CONS (CONS 'THE (RECORD.NAME TRAN))
                                                       TL]
                (for PR in (RECORD.FIELDINFO TRAN)
                  join (TOPPATHS (CAR PR)
                              LST
                              (JOINDEF (CDR PR)
                                       TL)
                              (RECORD.SUBDECS TRAN)
                              DEFS])

```

(CHECKDEFS

```

[LAMBDA (DEFS RECS FIELDS MUST)
 (COND
  ([AND [SOME (CDR DEFS)
            (FUNCTION (LAMBDA (X)
                      (NOT (EQUAL X (CAR DEFS)]
            (OR (NULL RECS)
                (bind FOUND for D on DEFS as R in RECS unless (EQ (RECORD.PRIORITY (RECORDECL R)
                                                                    'SYSTEM)
                    do (COND
                      ((NOT FOUND)
                       (SETQ FOUND D))
                      ((NOT (EQUAL (CAR D)
                                   (CAR FOUND)))
                       (RETURN T)))
                    finally (SETQ DEFS FOUND)
                           (RETURN NIL]
  (RECORDERROR [CONS "ambiguous" (CONS (COND
                                         ((LISTP FIELDS)
                                          "path")
                                         (T "field"))
                                         (CONS "appears in" (for X in RECS join (LIST ' "
                                                                                       " (RETDWIM2 X]
                                         FIELDS RECORDEXPRESSION))
  ((AND MUST (NULL DEFS))
   (RECORDERROR 2 FIELDS RECORDEXPRESSION)))
 (CAR DEFS])

```

(JOINDEF

```

[LAMBDA (DEF DEFLST)
 (COND
  ((NULL DEF)
   DEFLST)
  ([AND DEFLST (EQ (CAR DEF)
                  'RECORD)
   (OR (EQ (CAAR DEFLST)
           'RECORD)
       (NULL (CDR DEF]
  ;; If merging two RECORD expressions with CAR's and CDR's, do it here so that the ambiguous path checker can just use EQUAL
  ;; This also handles the case of 'synonym' records where there is just (RECORD A B)
  (CONS (CONS (CAAR DEFLST)
              (NCONC (APPEND (CDR DEF))
                    (CDAR DEFLST)))
        (CDR DEFLST)))
 (T (CONS DEF DEFLST])

```

)

(DEFINEQ

(NOTOKSWAP

```

[LAMBDA (EXPR1 EXPR2)
 (NOT (ARGS.COMMUTABLEP EXPR1 EXPR2])
 (* Imm " 3-Jul-85 12:36")

```

(FIXFIELDORDER

```

[LAMBDA (EXPRESSION)
  (* DECLARATIONS%: FAST
  (* lmm "3-Jul-85 12:36")
  (PROG (REVFIELDS LASTFIELDTAIL TEM FIELD.USAGE USE1 USE2 PLACE1 PLACE2 UNUSEDFIELDS)
    (FINDFIELDUSAGE EXPRESSION)
    ;; The elements of FIELDS.IN.CREATE are entries of the form (field.name value.given.in.create . seen) where seen is NIL initially, the last 'place'
    ;; field.name was
    [for X in (REVERSE FIELDS.IN.CREATE) do (COND
      ((ASSOC (CAR X)
        FIELD.USAGE))
      (T (SETQ UNUSEDFIELDS (CONS [CONS (CAR X)
        (SETQ TEM
          (LIST (CADR X)
            UNUSEDFIELDS))
          (SETQ FIELD.USAGE (CONS (CONS (CAR X)
            TEM)
          FIELD.USAGE])
        ]))
    ]))
  LP (COND
    ((NULL FIELD.USAGE) ; Done
    (RETURN UNUSEDFIELDS)))
  [COND
    ((NOT (OR (CONSTANTEXPRESSIONP (CADAR FIELD.USAGE))
      (ASSOC (CADAR FIELD.USAGE)
        BINDINGS)))
    (COND
      ([SETQ TEM (for X in (CDR FIELD.USAGE) when (EQ (CAR X)
        (CAAR FIELD.USAGE))
        do (SETQ $$VAL (CONS X $$VAL)
        (FRPLACA (CADR TEM)
          (LIST 'SETQ (RECORDBIND)
            (CADAR TEM)))
        [MAPC (CONS (CAR FIELD.USAGE)
          (CDR TEM))
          (FUNCTION (LAMBDA (X)
            (FRPLACA (CDR X)
              (CADR (CADAR TEM)))
            (FRPLACA X NIL)
            (FRPLACD (CAR TEM)
              (CDDR (CADAR TEM)))
            (SETQ FIELD.USAGE (CDR FIELD.USAGE))
            (GO LP]
        ]))
    ]))
    ((NULL (CAAR FIELD.USAGE))
    (SETQ FIELD.USAGE (CDR FIELD.USAGE)))
    (EQ (CAAR FIELD.USAGE)
      (CAAR FIELDS.IN.CREATE))
    ;; Both FIELD.USAGE and FIELDS.IN.CREATE are in reverse order of occurrence of expression in the translation and occurrence in
    ;; the original CREATE; if order of ends is the same, we can ignore those fields
    (SETQ FIELD.USAGE (CDR FIELD.USAGE))
    (SETQ FIELDS.IN.CREATE (CDR FIELDS.IN.CREATE)))
    ((OR (CONSTANTEXPRESSIONP (CADAR FIELD.USAGE))
      (ASSOC (CADAR FIELD.USAGE)
        BINDINGS)) ; The last field used is a constant
    (AND (SETQ TEM (ASSOC (CAAR FIELD.USAGE)
      FIELDS.IN.CREATE))
      (FRPLACD (CDR TEM)
        T))
    (SETQ FIELD.USAGE (CDR FIELD.USAGE)))
    ((OR (CDDAR FIELDS.IN.CREATE)
      (CONSTANTEXPRESSIONP (CADAR FIELDS.IN.CREATE))
      (ASSOC (CADAR FIELDS.IN.CREATE)
        BINDINGS)) ; This one has been seen before
    (SETQ FIELDS.IN.CREATE (CDR FIELDS.IN.CREATE)))
    (T (SETQ REVFIELDS)
      [for X in FIELDS.IN.CREATE do (COND
        ((EQ (CAR X)
          (CAAR FIELD.USAGE))
          (RETURN)))
        (COND
          ((NOTOKSWAP (CADR X)
            (CADAR FIELD.USAGE))
            (SETQ REVFIELDS (CONS (CAR X)
              REVFIELDS]
    ]))
  ;; REVFIELDS is the list of fields which are specified in the CREATE after the last field used and which must be referenced AFTER
  ;; what is now the last-field-used
  (COND
    (REVFIELDS
      ;; The last field referenced (CAR FIELDS.IN.CREATE) must actually be referenced before any of REVFIELDS
      (for TL on FIELD.USAGE when (MEMB (CAAR TL)
        REVFIELDS)
        do (SETQ LASTFIELDTAIL TL))
      (OR LASTFIELDTAIL (SHOULDNT)) ; In particular, it must be referenced before LASTFIELDTAIL
      (SETQ USE1 (CAR LASTFIELDTAIL))
      (SETQ USE2 (CAR FIELD.USAGE))
    ]))

```

```
(SETQ FIELD.USAGE (CDR FIELD.USAGE))
(FRPLACD LASTFIELDTAIL (CONS USE2 (CDR LASTFIELDTAIL)))
; Reorder FIELD.USAGE list
```

:: Now comes the incredible list structure patch: USE1= (NAME1 EXPR1 ...) USE2= (NAME2 EXPR2 ...) --- first
 :: change USE1 to (PROGN (SETQ TEM EXPR2) EXPR1) then change USE2 to TEM; then make USE pointers
 :: point back to the EXPRS

```
(FRPLACA (CDR USE1)
  (CONS 'PROGN (CONS [CONS 'SETQ (CONS (SETQ TEM (RECORDBIND))
    (SETQ PLACE2 (LIST (CADR USE2])
    (SETQ PLACE1 (LIST (CADR USE1])
    TEM)
  (FRPLACD USE1 PLACE1)
  (FRPLACD USE2 PLACE2))
```

(T ; It is ok that this field is used out of order

```
(AND (SETQ TEM (ASSOC (CAAR FIELD.USAGE)
  FIELDS.IN.CREATE))
  (FRPLACD (CDR TEM)
    T))
(SETQ FIELD.USAGE (CDR FIELD.USAGE]
```

(GO LP])

(FINDFIELDUSAGE

[LAMBDA (EXPRESSION)

(* Imm%: "22-AUG-76 23:01:55")

:: Sets the list FIELD.USAGE to the list (in reverse order) of the places where FIELDS.IN.CREATE are used --- originally, the FIELDS.IN.CREATE
 :: items are set up in the expression as the entire ALIST entry. FINDFIELDUSAGE also replaces them with the 'right' expression

```
(COND
  ((NLISTP EXPRESSION))
  ((NLISTP (CAR EXPRESSION))
    (FINDFIELDUSAGE (CDR EXPRESSION)))
  [(NLISTP (CAAR EXPRESSION))
    (COND
      ((FMEMB (CAR EXPRESSION)
        FIELDS.IN.CREATE)
        (SETQ FIELD.USAGE (CONS (CONS (CAAR EXPRESSION)
          EXPRESSION)
        FIELD.USAGE))
        (FRPLACA EXPRESSION (CADAR EXPRESSION))
        (FINDFIELDUSAGE (CDR EXPRESSION)))
      ((EQ (CAAR EXPRESSION)
        'LAMBDA)
        (FINDFIELDUSAGE (CDR EXPRESSION))
        (FINDFIELDUSAGE (CDDAR EXPRESSION)))
      (T (FINDFIELDUSAGE (CDAR EXPRESSION))
        (FINDFIELDUSAGE (CDR EXPRESSION))
      (T (FINDFIELDUSAGE (CAR EXPRESSION))
        (FINDFIELDUSAGE (CDR EXPRESSION))])
```

; Add (FIELDNAME . LOCATION) onto FIELD.USAGE

; The CDR is executed first

(EMBEDPROG

[LAMBDA (DEF)
 (COND

(* Imm "25-AUG-78 12:38")

```
[BINDINGS (PROG ((BINDVARS (MAPCAR (SETQ BINDINGS (DREVERSE BINDINGS))
  (FUNCTION CAR)))
  [BINDVALS (MAPCAR BINDINGS (FUNCTION (LAMBDA (X)
    (COND
      ((AND (EQ (CAR (SETQ X (CADR X)))
        'PROGN)
        (NULL (CDDR X)))
        (CADR X))
      (T X]
```

```
LE LL)
(SETQ BINDINGS)
(RETURN (COND
```

```
[[AND (LISTP (CAR DEF))
  (EQ (CAAR DEF)
    'LAMBDA)
  (NOT (REBINDP BINDVARS (CDR DEF))
  (CONS (CONS 'LAMBDA (CONS (NCONC BINDVARS (CADAR DEF))
    (CDDAR DEF))))
  (NCONC BINDVALS (CDR DEF))
  [[AND (NULL (CDR BINDVARS))
  (EQ [CAR (SETQ LE (LISTP (CAR (LISTP (CAR BINDVALS)
    'LAMBDA)
  (NULL (CDR (CADR LE)))
  (EQ (CAADR LE)
    (CAR (SETQ LL (LAST LE])
  (CONS [NCONC (LDIFF LE LL)
    (SUBPAIR BINDVARS (CADR LE)
    (COND
      ((EQ (CAR DEF)
        'PROGN)
        (CDR DEF))
      (T (LIST DEF]
```



```

      AT)
    else (ERROR MESSAGE (OR AT IN]
(FIXPRINTIN FAULTFN)
(LISPXSPACES 1)
(COND
  ((NLISTP MESSAGE)
   (LISPXPRIN1 MESSAGE T))
  (T (MAPRINT MESSAGE T NIL NIL NIL NIL T)))
(LISPXTERPRI T)
[COND
  (AT (LISPXPRIN1 " at " T)
   (COND
     ((NLISTP AT)
      (LISPXPRIN2 AT T T)
      (LISPXPRIN1 " " " T))
     ([AND IN (SETQ TEM (OR (MEMB AT IN)
                           (TAILP AT IN]
                          (MAPRINT (RETDWIM2 (COND
                                        (CDRFLG (NLEFT IN 1 TEM))
                                        (T TEM))
                                    (CDDR AT))
                                T "... " ")
                                " NIL NIL T))
      (T (LISPXPRINT (RETDWIM2 AT)
                    T T])
   (COND
     (IN (LISPXPRIN1 "in " T)
      (LISPXPRINT (RETDWIM2 IN)
                  T T)))
(DWIMERRORRETURN 'ALREADYPRINTED])

```

(SETUPHASHARRAY

```

[LAMBDA (ARRAYNAME SIZE)
  (PROG (TEM)
    [COND
      [(NULL (SETQ TEM (GETATOMVAL ARRAYNAME)
                    ((HASHARRAYP TEM)
                     (T (SET ARRAYNAME (HASHARRAY (OR SIZE 100]
                    (RETURN ARRAYNAME]))

```

(* Imm "12-Jul-84 22:40")

(DWIMIFYREC

```

[LAMBDA (DWIMTAIL NEWVARS PARENT ONEFLG INDECL)
  (AND DWIMTAIL (if INDECL
                    then [PROG ((EXPR DECL)
                                (VARS NEWVARS)
                                (FAULTFN (LIST (CADR DECL)
                                                'declaration))
                                (DWIMIFYFLG 'VARSBOUND))
                            (RETURN (DWIMIFY0? DWIMTAIL PARENT T T ONEFLG FAULTFN 'VARSBOUND])
                    else (PROG ((VARS (APPEND NEWVARS VARS)))
                              (RETURN (DWIMIFY0? DWIMTAIL PARENT T T ONEFLG FAULTFN 'VARSBOUND]))

```

(* Imm " 7-AUG-84 23:32")

(MKCONS

```

[LAMBDA (CARPART CDRPART)
  (COND
    [(OR (EQ (CAR (LISTP CDRPART))
            'LIST)
         (NULL CDRPART))
     (CONS 'LIST (CONS CARPART (CDR CDRPART))
           (T (LIST 'CONS CARPART CDRPART]))

```

(* Imm%: 15-APR-76 15 30)

(MKPROGN

```

[LAMBDA (X)
  (COND
    ((NULL (CDR X))
     (CAR X))
    (T (CONS 'PROGN X])
)

```

(DEFINEQ

(RECORDINIT

```

[LAMBDA NIL
  [MAPC RECORDINIT (FUNCTION (LAMBDA (X)
                              (APPLY (CAR X)
                                      (CDR X]
    (/SET 'RECORDINIT])
)

```

(* Imm%: " 3-FEB-77 18:51:20")

(RPAQQ PATGENSYMVARS (GENSYMVARS%: \$\$1 \$\$2 \$\$3 \$\$4 \$\$5 \$\$6 \$\$7 \$\$8 \$\$9 \$\$10 \$\$11 \$\$12 \$\$13 \$\$14 \$\$15 \$\$16 \$\$17
)

(RPAQ? RECORDINIT)

(RPAQ? CLISPRECORDTYPES NIL)

(RPAQ? RECORDTRANHASH (HASHARRAY 20))

(DEFINEQ

(RECORD

```

[NLAMBDA NAME&FIELDS (* Imm " 3-MAR-82 11:20")
  (PROG ((N -1)
        (NAM)
        (LP (COND
              [(FMEMB (SETQ NAM (STKNTHNAME N))
                      CLISPRECORDTYPES)
               (RETURN (DECLARERECORD (CONS NAM NAME&FIELDS)
                                         (NAM (SETQ N (SUB1 N))
                                               (GO LP)))
                        (HELP "Record definition called, but no framename matches CLISPRECCORDTYPES"])]))

```

(TYPERECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'TYPERECORD NAME&FIELDS])

```

(PROPRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'PROPRECORD NAME&FIELDS])

```

(HASHLINK

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'HASHLINK NAME&FIELDS])

```

(ACCESSFN

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ACCESSFN NAME&FIELDS])

```

(ACCESSFNS

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ACCESSFNS NAME&FIELDS])

```

(HASHRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'HASHRECORD NAME&FIELDS])

```

(ATOMRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ATOMRECORD NAME&FIELDS])

```

(ARRAYRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ARRAYRECORD NAME&FIELDS])

```

(DATATYPE

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'DATATYPE NAME&FIELDS])

```

(BLOCKRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'BLOCKRECORD NAME&FIELDS])

```

(ASSOCRECORD

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ASSOCRECORD NAME&FIELDS])

```

(CACCESSFNS

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'CACCESSFNS NAME&FIELDS])

```

(ARRAYBLOCK

```

[NLAMBDA NAME&FIELDS (* edited%: "13-OCT-81 14:39")
  (DECLARERECORD (CONS 'ARRAYBLOCK NAME&FIELDS])

```


(SYNONYM

```
[NLAMBDA NAME&FIELDS
  (DECLARERECORD (CONS 'SYNONYM NAME&FIELDS])
```

(* edited%: "13-OCT-81 14:39")

)

(DEFINEQ

(RECORDECLARATIONS

```
[NLAMBDA DECS
```

(* bvm%: "10-Oct-86 18:18")

:: Entry from the RECORDS prettymacro. Given a list of record names {DECS} prints the record declarations

```
(PROG (TEM)
  (PRIN1 " (")
  (MAPRINT ' (DECLARE%: EVAL@COMPILE)
    NIL NIL NIL NIL (FUNCTION PRIN2))
  (TERPRI)
  [MAPC DECS (FUNCTION (LAMBDA (NAM DEC)
    [SETQ TEM (COND
      ([AND (LITATOM NAM)
        (SETQ DEC (CAR (RECLOOK1 NAM USERRECLST]
      (COND
        ((AND (LISTP DEC)
          (EQ (CAR DEC)
            CLISPTRANFLG))
        (CDDR DEC))
        (T DEC)))
      ((AND (LISTP NAM)
        (PROGN [COND
          ((EQ (CAR NAM)
            CLISPTRANFLG)
          (SETQ NAM (CDDR NAM]
          (FMEMB (CAR NAM)
            CLISPRECORDTYPES)))
        (SETQ DEC NAM))
      (T (LIST 'QUOTE (LISPXPRINT (APPEND ' (no RECORD declaration for)
        (LIST NAM))
          T T]
      (COND
        ((EQ (CADR TEM)
          NAM)
        (PRETTYVAR1 (CAR TEM)
          (CADR TEM)
          (CDDR TEM)
          T T))
        (T (PRINTDEF TEM 0 T)
          (TERPRI]
  (PRIN1 " )
    "])
```

(RECORDALLOCATIONS

```
[NLAMBDA DECS
```

(* lmm "27-OCT-77 15:20")

```
(for X in DECS join (APPEND (RECORD.ALLOCATIONS (RECORDECL (CAR (RECLOOK1 X USERRECLST])
```

(SAVEONSYSRECLST

```
[NLAMBDA NAMES
```

(* bvm%: "16-Nov-86 17:20")

:: Entry from SYSRECORDS prettymacro. Given a list of record names {DECS} prints an expression that saves their record declarations on the variable SYSTEMRECLST

```
(printout NIL " (")
(MAPRINT ' (ADDTOVAR SYSTEMRECLST )
  NIL NIL NIL NIL (FUNCTION PRIN2))
(TERPRI)
[for N DECL in NAMES do (COND
  ((NULL (SETQ DECL (RECLOOK N)))
  (CL:FORMAT T " (no RECORD declaration for ~S)~%" N))
  ((EQ N (CADR DECL))
  (PRETTYVAR1 (CAR DECL)
    (CADR DECL)
    (COND
      [(EQ (CAR DECL)
        'DATATYPE)
      ;; The usual case. Save only the fields declaration, sans comments, since that is all the inspector
      ;; needs, and it reduces the cruft in a loaded system
      (LIST (for FIELD in (CADDR DECL) collect FIELD
        unless (EQ (CAR (LISTP FIELD))
          COMMENTFLG]
          (T (CDDR DECL)))
      T T))
  (T (PRINTDEF DECL 0 T)
    (TERPRI]
(printout NIL " )" T])
```

)

```

(ADDTOVAR USERRECLST )
(RPAQQ DECLARATIONCHAIN NIL)
(RPAQQ MSBLIP "sysout and inform Masinter@PARC")
(RPAQQ NOSIDEFNS (fetch CONS NLISTP PROG APPEND LIST NEQ MEMB MEMBER FMEMB ASSOC TAILP COPY create ELT ELTD
AND OR ADD1 SUB1 IPLUS IDIFFERENCE EQ EQUAL NOT NULL))
(RPAQQ RECORDSUBSTFLG NIL)
(RPAQQ RECORDUSE NIL)
(RPAQQ DATATYPEFIELDCOERCIONS ((INTEGER . FIXP)
(REAL . FLOATP)
(FLOATING . FLOATP)))
(RPAQ? RECORDCHANGEFN )
(RPAQQ CLISPRECORDWORDS (smashing using copying reusing SMASHING USING COPYING REUSING))
(PUTPROPS /REPLACE CLISPCWORD (RECORDTRAN . /replace))
(PUTPROPS COPYING CLISPCWORD (RECORDTRAN . copying))
(PUTPROPS FETCH CLISPCWORD (RECORDTRAN . fetch))
(PUTPROPS FFETCH CLISPCWORD (RECORDTRAN . ffetch))
(PUTPROPS FREPLACE CLISPCWORD (RECORDTRAN . freplace))
(PUTPROPS REPLACE CLISPCWORD (RECORDTRAN . replace))
(PUTPROPS REUSING CLISPCWORD (RECORDTRAN . reusing))
(PUTPROPS SMASHING CLISPCWORD (RECORDTRAN . smashing))
(PUTPROPS TYPE? CLISPCWORD (RECORDTRAN . type?))
(PUTPROPS USING CLISPCWORD (RECORDTRAN . using))
(PUTPROPS /replace CLISPCWORD (RECORDTRAN . /replace))
(PUTPROPS copying CLISPCWORD (RECORDTRAN . copying))
(PUTPROPS fetch CLISPCWORD (RECORDTRAN . fetch))
(PUTPROPS ffetch CLISPCWORD (RECORDTRAN . ffetch))
(PUTPROPS freplace CLISPCWORD (RECORDTRAN . freplace))
(PUTPROPS replace CLISPCWORD (RECORDTRAN . replace))
(PUTPROPS reusing CLISPCWORD (RECORDTRAN . reusing))
(PUTPROPS smashing CLISPCWORD (RECORDTRAN . smashing))
(PUTPROPS type? CLISPCWORD (RECORDTRAN . type?))
(PUTPROPS using CLISPCWORD (RECORDTRAN . using))
(PUTPROPS OF CLISPCWORD (RECORDTRAN . of))
(PUTPROPS of CLISPCWORD (RECORDTRAN . of))
(PUTPROPS WITH CLISPCWORD (RECORDTRAN . with))
(PUTPROPS with CLISPCWORD (RECORDTRAN . with))
(PUTPROPS CREATE CLISPCWORD (RECORDTRAN . create))
(PUTPROPS create CLISPCWORD (RECORDTRAN . create))
(PUTPROPS INITRECORD CLISPCWORD (RECORDTRAN . initrecord))
(PUTPROPS initrecord CLISPCWORD (RECORDTRAN . initrecord))
(DECLARE%: DONTCOPY
[PUTDEF 'RECORDTYPES 'FILEPKGCOMS '(COM MACRO (X (IFPROP USERRECORDTYPE . X)
(ADDVARS (CLISPRECORDTYPES . X))
(P (MAPC 'X (FUNCTION (LAMBDA (FN)
(MOVD? 'RECORD FN)
)
)
(PUTPROPS HASHLINK USERRECORDTYPE [LAMBDA (DEC)

```

(CONS 'HASHRECORD (CDR DEC))

(PUTPROPS ACCESSFN USERRECORDTYPE [LAMBDA (DEC)
(CONS 'ACCESSFNS (CDR DEC))

(PUTPROPS SYNONYM USERRECORDTYPE [LAMBDA
(DEC)
(CONS 'RECORD
(CONS (CADR DEC)
(CONS [CAR (OR (LISTP (CADDR DEC))
(CAR (/RPLACA (CDDR DEC)
(LIST (CADDR DEC)
(NCONC [MAPCAR (CDR (CADDR DEC))
(FUNCTION (LAMBDA (X)
(LIST 'RECORD
(CADR DEC)
X])
(CDDDR DEC])

(ADDTOVAR CLISPRECORDTYPES RECORD TYPERECORD PROPRECORD HASHLINK ACCESSFN ACCESSFNS HASHRECORD ATOMRECORD
ARRAYRECORD DATATYPE BLOCKRECORD ASSOCRECORD CACCESSFNS ARRAYBLOCK SYNONYM)

[MAPC '(RECORD TYPERECORD PROPRECORD HASHLINK ACCESSFN ACCESSFNS HASHRECORD ATOMRECORD ARRAYRECORD DATATYPE
BLOCKRECORD ASSOCRECORD CACCESSFNS ARRAYBLOCK SYNONYM)
(FUNCTION (LAMBDA (FN)
(MOVD? 'RECORD FN]

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS CREATE.RECORD MACRO ((FIELDNAMES NAME FIELDINFO CREATEINFO TYPECHECK SUBDECS ALLOCATIONS
DEFAULTFIELDS DECL PRIORITY)
(LIST FIELDNAMES NAME FIELDINFO CREATEINFO TYPECHECK SUBDECS ALLOCATIONS
DEFAULTFIELDS DECL PRIORITY)))

(PUTPROPS ADD.RECORD.SUBDECS MACRO ((TRAN NEWVALUE)
(FRPLACA (CDR (CDDDDR TRAN))
(NCONC1 (CADR (CDDDDR TRAN))
NEWVALUE))))

(PUTPROPS RECORD.ALLOCATIONS MACRO ((TRAN)
(CADDR (CDDDDR TRAN))))

(PUTPROPS RECORD.CREATEINFO MACRO ((TRAN)
(CADDDR TRAN)))

(PUTPROPS RECORD.DEFAULTFIELDS MACRO ((TRAN)
(CADDDR (CDDDDR TRAN))))

(PUTPROPS RECORD.FIELDINFO MACRO ((TRAN)
(CADDR TRAN)))

(PUTPROPS RECORD.FIELDNAMES MACRO ((TRAN)
(CAR TRAN)))

(PUTPROPS RECORD.NAME MACRO ((TRAN)
(CADR TRAN)))

(PUTPROPS RECORD.SUBDECS MACRO [LAMBDA (TRAN)
(CADR (CDDDDR TRAN])

(PUTPROPS RECORD.TYPECHECK MACRO ((TRAN)
(CAR (CDDDDR TRAN))))

(PUTPROPS SET.RECORD.ALLOCATIONS MACRO ((TRAN NEWVALUE)
(FRPLACA (CDDR (CDDDDR TRAN))
NEWVALUE)))

(PUTPROPS SET.RECORD.CREATEINFO MACRO ((TRAN NEWVALUE)
(FRPLACA (CDDDR TRAN)
NEWVALUE)))

(PUTPROPS SET.RECORD.DEFAULTFIELDS MACRO ((TRAN NEWVALUE)
(FRPLACA (CDDDR (CDDDDR TRAN))
NEWVALUE)))

(PUTPROPS SET.RECORD.FIELDNAMES MACRO ((TRAN NEWVALUE)
(FRPLACA TRAN NEWVALUE)))

(PUTPROPS SET.RECORD.NAME MACRO ((TRAN NEWVALUE)
(FRPLACA (CDR TRAN)
NEWVALUE)))

(PUTPROPS SET.RECORD.TYPECHECK MACRO ((TRAN NEWVALUE)
(FRPLACA (CDDDDR TRAN)
NEWVALUE)))

```

(PUTPROPS RECORD.DECL MACRO ((X)
                               (CAR (FNTH X 9))))

(PUTPROPS SET.RECORD.DECL MACRO ((X Y)
                                   (FRPLACA (FNTH X 9)
                                             Y)))

(PUTPROPS RECORD.PRIORITY MACRO ((X)
                                   (CAR (FNTH X 10))))

(PUTPROPS SET.RECORD.PRIORITY MACRO ((X Y)
                                       (/RPLACA (FNTH X 10)
                                                Y)))

)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(ADDTOVAR SYSLOCALVARS $$1 $$2 $$3 $$4 $$5 $$6 $$7 $$8 $$9 $$10 $$11 $$12 $$13 $$14 $$15 $$16 $$17)

;; for handling datatype

(MOVD 'FETCHFIELD 'FFETCHFIELD)

(MOVD 'REPLACEFIELD 'FREPLACEFIELD)

(PUTPROPS FETCHFIELD LISPFN FETCHFIELD)

(PUTPROPS FREPLACEFIELD LISPFN FREPLACEFIELD)

(PUTPROPS REPLACEFIELD LISPFN REPLACEFIELD)

(PUTPROPS FETCHFIELD CLISPCLASS FETCHFIELD)

(PUTPROPS FFETCHFIELD CLISPCLASS FFETCHFIELD)

(PUTPROPS FREPLACEFIELD CLISPCLASS REPLACEFIELD)

(PUTPROPS /REPLACEFIELD CLISPCLASS REPLACEFIELD)

(PUTPROPS REPLACEFIELD CLISPCLASS REPLACEFIELD)

(PUTPROPS FETCHFIELD CLISPCLASSDEF (ACCESS FETCHFIELD NIL FFETCHFIELD))

(PUTPROPS REPLACEFIELD CLISPCLASSDEF (ACCESS REPLACEFIELD /REPLACEFIELD FREPLACEFIELD))

(ADDTOVAR DECLWORDS FFETCHFIELD FETCHFIELD REPLACEFIELD FREPLACEFIELD /REPLACEFIELD)

(NEW/FN 'REPLACEFIELD)

(RPAQQ RECORDWORDS ((/replace UNDOABLE replace)
                    (/push UNDOABLE push)
                    (/pushnew UNDOABLE pushnew)
                    (freplace FAST replace)
                    (ffetch FAST fetch)))

;; for CHANGETRAN

(PUTPROPS ADD CLISPWORD (CHANGETRAN . add))

(PUTPROPS CHANGE CLISPWORD (CHANGETRAN . change))

(PUTPROPS POP CLISPWORD (CHANGETRAN . pop))

(PUTPROPS PUSH CLISPWORD (CHANGETRAN . push))

(PUTPROPS PUSHNEW CLISPWORD (CHANGETRAN . pushnew))

(PUTPROPS PUSHLIST CLISPWORD (CHANGETRAN . pushlist))

(PUTPROPS add CLISPWORD (CHANGETRAN . add))

(PUTPROPS change CLISPWORD (CHANGETRAN . change))

(PUTPROPS pop CLISPWORD (CHANGETRAN . pop))

(PUTPROPS push CLISPWORD (CHANGETRAN . push))

(PUTPROPS pushnew CLISPWORD (CHANGETRAN . pushnew))

(PUTPROPS pushlist CLISPWORD (CHANGETRAN . pushlist))

```

(PUTPROPS **SWAP CLISPPWORD** (CHANGETRAN . swap))

(PUTPROPS **swap CLISPPWORD** (CHANGETRAN . swap))

(PUTPROPS **/push CLISPPWORD** (CHANGETRAN . /push))

(PUTPROPS **/pushnew CLISPPWORD** (CHANGETRAN . /pushnew))

(PUTPROPS **/PUSH CLISPPWORD** (CHANGETRAN . /push))

(PUTPROPS **/PUSHNEW CLISPPWORD** (CHANGETRAN . /pushnew))

(DEFINEQ

(CHANGETRAN

[LAMBDA (X) (* Imm "29-SEP-78 16:51")
(**RECORDTRAN** X 'CHANGETRAN)]

(CHANGETRAN1

[LAMBDA (CHANGEWORD RECORDEXPRESSION) (* rmk%:" 6-JUN-79 16:56")
(PROG (TEM FORM VAR1 NOTRANFLG ARGS [SPECIALFIELDS (COPY ' ((DATUM DATUM)
FIELDNAMES
(SUBSTYPE 'CHANGE))
(**DWIMIFYREC** (CDR RECORDEXPRESSION)
' (DATUM)
RECORDEXPRESSION)
(SETQ ARGS (**FIXDATUM** (SETQ VAR1 (CADR RECORDEXPRESSION))
DECLST))
[SETQ FORM (COND
((SETQ TEM (GETPROP CHANGEWORD 'CHANGEWORD))
(APPLY* TEM RECORDEXPRESSION))
(T (SELECTQ CHANGEWORD
(add [LIST 'DATUM_ (CONS (**RECLISLOOKUP** '+ DECLST VAR1 (CADDR RECORDEXPRESSION)
)
(CONS 'DATUM (CDDR RECORDEXPRESSION))
(change (LIST 'DATUM_ (CADDR RECORDEXPRESSION))
(pop ' (PROG1 (CAR DATUM)
(DATUM_ (CDR DATUM))))
(push (LIST 'DATUM_ (**for** ELT (EXP _ 'DATUM) **in** (REVERSE (CDDR RECORDEXPRESSION))
do (SETQ EXP (LIST 'CONS ELT EXP))
finally (RETURN EXP))))
(pushnew [SUBST (**RECORDBINDVAL** (CADDR RECORDEXPRESSION))
' NEWELT
' (COND
((FMEMB NEWELT DATUM)
DATUM)
(T (DATUM_ (CONS NEWELT DATUM))
(pushlist [LIST 'DATUM_ (CONS 'APPEND (APPEND (CDDR RECORDEXPRESSION)
(LIST 'DATUM])
(swap (SETQ TEM (**FIXDATUM** (CADDR RECORDEXPRESSION)
DECLST))
[LIST 'DATUM_ (LIST 'PROG1 (CAR TEM)
(SUBST 'DATUM 'NEWVALUE (CADDR TEM))
(**RECORDERROR** "Undefined CHANGEWORD" RECORDEXPRESSION)]
(RETURN (PROG (BINDINGS)
(RETURN (**EMBEDPROG** (CSUBST FORM))

(FIXDATUM

[LAMBDA (FORM DECLST) (* Imm " 3-Jul-85 12:37")

;; turn a form into one which can be smashed more easily

(PROG (TEM (X FORM))
LP [COND
[(LITATOM X)
(COND
((AND (STRPOS CLISPCARRAY X)
(CLISPNOTVARP X))
(**RECORDERROR** "unable to DWIMify" X RECORDEXPRESSION)))
(RETURN (LIST X NIL (LIST (**RECLISLOOKUP** 'SETQ DECLST)
X
'NEWVALUE]
((LISTP X)
(SELECTQ (CAR X)
((fetch FETCH ffetch FFETCH)
(RETURN (**MAKEACCESS** (OR (**ACCESSDEF** (CADR X)
(CADDR X))
(**RECORDERROR** "unable to DWIMify" (CADR X)
RECORDEXPRESSION))
(SELECTQ (CADDR X)
((of OF)
(**MKPROGN** (CADDR X)))
(**MKPROGN** (CDDR X)))
' (NEWVALUE)
' change)))
(AND [SETQ X (SELECTQ (CAR X)

```

((CAR CDR GETHASH)
 X)
((NTH FNTH NLEFT)
 [LIST 'CDR (LIST (CAR X)
 (CADR X)
 ([LAMBDA (N X)
 (COND
 ((FIXP X)
 (APPLY* N X))
 (T (LIST N X])
 (COND
 ((EQ (CAR X)
 'NLEFT)
 'ADD1)
 (T 'SUB1))
 (CADDR X])
 (LAST FLAST)
 (LIST 'CDR (LIST 'NLEFT (CADR X)
 2)))
(COND
 ((AND (SETQ TEM (GETPROP (CAR X)
 'SETFN))
 (LITATOM TEM))
 X)
 [(SETQ TEM (GETP (CAR X)
 'CROPS))
 (LIST (SELECTQ (CAR (SETQ TEM (REVERSE TEM)))
 (A 'CAR)
 (D 'CDR)
 (SHOULDNT))
 (CONS [PACK (CONS 'C (NCONC1 (CDR TEM)
 'R]
 (CDR X]
 ([AND (SETQ TEM (GETMACROPROP (CAR X)
 COMPILERMACROPROPS))
 (NOT (EQUAL X (SETQ TEM (MACROEXPANSION X TEM))
 (SETQ X TEM)
 (GO LP]
(RETURN (LIST [SETQ X
 (CONS (CAR X)
 (PROG ((TEM T)
 VAL)
 (for Y in (REVERSE (CDR X))
 do (SETQ VAL (CONS (COND
 ((OR (AND TEM (SETQ TEM (SIMPLEP
 Y)))
 (CONSTANTEXPRESSIONP Y))
 Y)
 (T (RECORDBIND Y)))
 VAL)))
 (RETURN VAL]
NIL
([LAMBDA (Y)
 (SELECTQ (CAR X)
 ((CAR CDR)
 (LIST (CAR X)
 Y))
 Y]
(CONS (RECLISPLOOKUP (SELECTQ (CAR X)
 (CAR 'RPLACA)
 (CDR 'RPLACD)
 (GETHASH 'PUTHASH)
 (GETP (CAR X)
 'SETFN))
 DECLST)
(COND
 [(EQ (CAR X)
 'GETHASH)
 (CONS (CADR X)
 (CONS 'NEWVALUE (CDDR X)
 (T (APPEND (CDR X)
 ' (NEWVALUE]
(RECORDERROR 'CHANGE FORM RECORDEXPRESSION])
)
(PUTPROPS GETP SETFN PUT)
(PUTPROPS GETPROP SETFN PUTPROP)
(PUTPROPS EVALV SETFN SET)
(PUTPROPS GETATOMVAL SETFN SETATOMVAL)
(PUTPROPS OPENR SETFN CLOSER)
(PUTPROPS WORDCONTENTS SETFN SETWORDCONTENTS)

```

(PUTPROPS **GETBASE SETFN** \PUTBASE)

(PUTPROPS **GETBASEBYTE SETFN** \PUTBASEBYTE)

(PUTPROPS **GETBASEBIT SETFN** \PUTBASEBIT)

(PUTPROPS **FETCHFIELD SETFN** REPLACEFIELD)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: RECORDBLOCK ACCESSDEF ACCESSDEF4 ALLFIELDS ALLOHASH ALLPATHS CHANGETRAN CHANGETRAN1 CHECKDEFS CHECKRECORDNAME CLISPRECORD CONSFN COPY1 CREATEFIELDS CSUBST RECONS CSUBSTLST DECLARERECORD DECLSUBFIELD DWIMIFYREC EMBEDPROG FIELDLOOK FIELDNAMESIN FINDFIELDUSAGE FIXDATUM FIXFIELDORDER GETFIELDFORCREATE GETSETQ HASHLINKS JOINDEF LISTRECORDEFS MAKEACCESS MAKEACCESS1 MAKECREATE0 MAKECREATE1 MAKECREATELST MAKECREATELST1 MAKEHASHLINKS MKACCESSFN MKCONS MKPROGN NOTOKSWAP REBINDP RECDEC? RECEVAL RECFIELDLOOK RECLISPLOOKUP RECLOOK RECLOOK1 RECORD.FIELD.VALUE RECORD.FIELD.VALUE0 RECORDACCESS RECORDALLOCATIONS RECORDBIND RECORDBINDVAL RECORDCHAIN RECORDECL RECORDECL0 RECORDECL1 RECORDECLBLOCK RECORDECLTAIL RECORDECLARATIONS RECORDERROR RECORDFIELD? RECORDFIELDNAMES RECORDGENSYM RECORDTRAN RECORDWORD RECREDECLARE SETUPHASHARRAY SIMPLEP SUBDECLARATIONS SUBFIELDCREATE TOPATHS UNCLISPTRAN RECORDPRIORITY (ENTRIES RECORDTRAN CHANGETRAN CLISPRECORD RECORDFIELD? RECORDECLARATIONS RECORDALLOCATIONS RECORDACCESS RECORDFIELDNAMES RECLOOK SETUPHASHARRAY FIELDLOOK RECORD.FIELD.VALUE DECLARERECORD RECORDPRIORITY) (SPECVARS DWIMIFYFLG CLISPCHANGE NEWVALUE DECLARATIONCHAIN USINGTYPE USINGEXPR ARRAYDESC EXPR FAULTFN VARS DECLST FIELDNAMES RECORDEXPRESSION RECORD.TRAN ALLOCATIONS FIELDS.IN.CREATE PATGENSYM VARS NOSPELLFLG PATGENSYM VARS) (LOCALFREEVARS FIELD.USAGE BINDINGS RNAME NAME TAIL SETQPART SETQTAIL DECL CREATEINFO CLISPCHANGE FIELDINFO HASHLINKS ARGS AVOID BODY VAR1 NOTRANFLG SPECIALFIELDS SUBTYPE STRUCNAME) (NOLINKFNS . T) SMASHPATTERN SMASHPAT1)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS MSBLIP CLISPRECORDTYPES NOSIDEFNS CLISPRECORDWORDS RECORDSTATS USERRECLST RECORDINIT LAMBDA SPLST CLISPTRANFLG RECORDCHANGEFN COMMENTFLG CLISPCHARRAY LCASEFLG CLISPARRAY LISPFNS RECORDWORDS DATATYPEFIELD COERCIONS DATATYPEFIELDTYPES RECORDTRANHASH RECORDINIT CLISPARRAY CLISPRECORDTYPES RECORDTRANHASH)

(DEFINEQ

(EDITREC

[NLAMBDA L
(EDITDEF (IF (NLISTP L)
 THEN L
 ELSE (CAR L))
 'RECORDS])

(* Imm "15-Nov-86 00:41")

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

(ADDTOVAR **NLAMA** EDITREC SAVEONSYRECLST RECORDALLOCATIONS RECORDECLARATIONS SYNONYM ARRAYBLOCK ACCESSFNS ASSOCRECORD BLOCKRECORD DATATYPE ARRAYRECORD ATOMRECORD HASHRECORD ACCESSFNS ACCESSFN HASHLINK PROPRECORD TYPRECORD RECORD MESATYPE MESARECORD MESAARRAY)

(ADDTOVAR **NLAML**)

(ADDTOVAR **LAMA**)

(PUTPROPS **RECORD COPYRIGHT** ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1993))

FUNCTION INDEX

ACCESSDEF	23	DECLSUBFIELD	10	MKPROGN	31	RECORDECLBLOCK	7
ACCESSDEF4	24	DWIMIFYREC	31	NOTOKSWAP	27	RECORDECLTAIL	8
ACCESSFN	32	EDITREC	39	PROPRECORD	32	RECORDEROR	30
ACCESSFNS	32	EMBEDPROG	29	REBINDP	17	RECORDFIELD?	4
ALLFIELDS	22	FIELDLOOK	13	RECDEC?	10	RECORDFIELDNAMES	12
ALLOHASH	11	FIELDNAMESIN	23	RECEVAL	12	RECORDGENSYM	30
ALLPATHS	26	FINDFIELDUSAGE	29	RECFIELDLOOK	26	RECORDINIT	31
ARRAYBLOCK	32	FIXDATUM	37	RECLISPLLOOKUP	30	RECORDPRIORITY	13
ARRAYRECORD	32	FIXFIELDORDER	27	RECLOSE	22	RECORDTRAN	2
ASSOCRECORD	32	GETFIELDFORCREATE	20	RECLOSE1	26	RECORDWORD	13
ATOMRECORD	32	GETSETQ	11	RECONS	19	RECREDECLARE	4
BLOCKRECORD	32	HASHLINK	32	RECORD	32	RECREDECLARE1	4
CACCESSFNS	32	HASHLINKS	21	RECORD.FIELD.VALUE	19	RECREDECLARE2	4
CHANGETRAN	37	HASHRECORD	32	RECORD.FIELD.VALUE0	19	SAVEONSYSRECLST	33
CHANGETRAN1	37	JOINDEF	27	RECORD.REMOVE.COMMENTS	9	SETUPHASHARRAY	31
CHECKDEFS	27	LISTRECORDEFS	9	RECORDACCESS	12	SIMPLE	13
CHECKRECORDNAME	8	MAKEACCESS	24	RECORDACCESSFORM	13	SMASHPAT1	19
CLISPRECORD	22	MAKEACCESS1	24	RECORDALLOCATIONS	33	SMASHPATTERN	19
CONSFN	30	MAKECREATE0	14	RECORDBIND	30	SUBDECLARATIONS	22
COPY1	19	MAKECREATE1	14	RECORDBINDVAL	13	SUBFIELDCREATE	21
CREATEFIELDS	17	MAKECREATELST	19	RECORDCHAIN	26	SYNONYM	33
Csubst	17	MAKECREATELST1	20	RECORDECL	4	SYSRECLOOK1	26
Csubstlst	19	MAKEHASHLINKS	21	RECORDECL0	5	TOPPATHS	26
DATATYPE	32	MKACCESSFN	25	RECORDECL1	5	TYPERECORD	32
DECLARERECORD	9	MKCONS	31	RECORDECLARATIONS	33	UNCLISPTRAN	10

PROPERTY INDEX

/push	37	change	36	FFETCHFIELD	36	of	34	replace	34	USING	34
/PUSH	37	COPYING	34	FREPLACE	34	OPENR	38	REPLACEFIELD	36	using	34
/pushnew	37	copying	34	freplace	34	POP	36	REUSING	34	WITH	34
/PUSHNEW	37	CREATE	34	FREPLACEFIELD	36	pop	36	reusing	34	with	34
/REPLACE	34	create	34	GETATOMVAL	38	PUSH	36	SMASHING	34	WORDCONTENTS	38
/replace	34	EVALV	38	GETP	38	push	36	smashing	34	\GETBASE	39
/REPLACEFIELD	36	FETCH	34	GETPROP	38	PUSHLIST	36	SWAP	37	\GETBASEBIT	39
ACCESSFN	35	fetch	34	HASHLINK	34	pushlist	36	swap	37	\GETBASEBYTE	39
ADD	36	FFETCHFIELD	36,39	INITRECORD	34	PUSHNEW	36	SYNONYM	35		
add	36	FFETCH	34	initrecord	34	pushnew	36	TYPE?	34		
CHANGE	36	ffetch	34	OF	34	REPLACE	34	type?	34		

MACRO INDEX

ADD.RECORD.SUBDECS	35	RECORD.FIELDNAMES	35	SET.RECORD.DECL	36
CREATE.RECORD	35	RECORD.NAME	35	SET.RECORD.DEFAULTFIELDS	35
RECORD.ALLOCATIONS	35	RECORD.PRIORITY	36	SET.RECORD.FIELDNAMES	35
RECORD.CREATEINFO	35	RECORD.SUBDECS	35	SET.RECORD.NAME	35
RECORD.DECL	36	RECORD.TYPECHECK	35	SET.RECORD.PRIORITY	36
RECORD.DEFAULTFIELDS	35	SET.RECORD.ALLOCATIONS	35	SET.RECORD.TYPECHECK	35
RECORD.FIELDINFO	35	SET.RECORD.CREATEINFO	35		

VARIABLE INDEX

CLISPRECORDTYPES	32,35	DECLWORDS	36	RECORDCHANGEFN	34	RECORDUSE	34
CLISPRECORDWORDS	34	MSBLIP	34	RECORDINIT	32	RECORDWORDS	36
DATATYPEFIELDCOERCIONS	34	NOSIDEFNS	34	RECORDSUBSTFLG	34	SYSLOCALVARS	36
DECLARATIONCHAIN	34	PATGENSYMVAR	31	RECORDTRANHASH	32	USERRECLST	34