```
File created:   11-Oct-93 14:34:11  {Pele:mv:envos}<LispCore>Sources>CLTL2>DTDECLARE.;1

  changes to:   (FUNCTIONS TYPE-VARIABLE-FROM-TYPE-NAME)

previous date:  15-Dec-92 14:08:39  {Pele:mv:envos}<LispCore>Sources>DTDECLARE.;6

  Read Table:   XCL

    Package:    INTERLISP

      Format:     XCCS
```

```
(RPAQQ DTDECLARECOMS
       (;; declaring DATATYPES --- part of ABC too
        (FNS /DECLAREDATATYPE DECLAREDATATYPE TRANSLATE.DATATYPE \\REUSETO \\TYPEGLOBALVARIABLE)
        (FUNCTIONS TYPE-VARIABLE-FROM-TYPE-NAME)
        (FNS |BitFieldMask| |BitFieldShift| |BitFieldShiftedMask| |MakeBitField| |BitFieldWidth| |BitFieldFirst|
             )
        (OPTIMIZERS FETCHFIELD FFETCHFIELD REPLACEFIELD FREPLACEFIELD REPLACEFIELDVAL FREPLACEFIELDVAL NCREATE
               \\DTEST)
        (PROP DMACRO \\TESTBITS)
        (FNS COMPILEDFETCHFIELD COMPILEDREPLACEFIELD COMPILEDNCREATE)
        (DECLARE\: DONTCOPY (EXPORT (RECORDS |FldDsc|)))
        (VARS DATATYPEFIELDTYPES)
        (COMS                                                    ; Macros which convert a record access form into an
                                                                 ; address-generating form
              (MACROS LOCF INDEXF)
              (FNS TRANSLATE.LOCF))
        (LOCALVARS . T)
        (PROP FILETYPE DTDECLARE)))

;; declaring DATATYPES --- part of ABC too

(DEFINEQ

(/DECLAREDATATYPE
  (LAMBDA (TYPENAME FIELDSPECS DLIST LEN SUPERTYPE)              ; Edited 18-May-87 17:09 by Snow
    (AND LISPXHIST TYPENAME (UNDOSAVE (LIST '/DECLAREDATATYPE TYPENAME (GETFIELDSPECS TYPENAME)
                                            NIL NIL (GETSUPERTYPE TYPENAME))))
    (CL:MULTIPLE-VALUE-BIND (DLIST REDECLARED)
        (DECLAREDATATYPE TYPENAME FIELDSPECS DLIST LEN SUPERTYPE)
      (COND
         (REDECLARED (LISPXPRINT (LIST '|datatype| TYPENAME '|redeclared|)
                            T T)))
      DLIST)))

(DECLAREDATATYPE
  (LAMBDA (TYPENAME FIELDSPECS DLIST LENGTH SUPERTYPE)           (* |Pavel| "16-Oct-86 14:52")

    ;; this is called twice when declaring records;  once where the DLIST and LENGTH hasn't been computed, and another time when it has.

    (LET ((SUPERSPECS (COND
                         (SUPERTYPE (GETFIELDSPECS SUPERTYPE)))))
                                                                 ; maybe an error if supertype doesn't exist?
         (SETQ FIELDSPECS (APPEND SUPERSPECS FIELDSPECS))
         (COND
            ((AND FIELDSPECS (OR (NOT DLIST)
                                 (NOT LENGTH)))                  ; the AND is an optimization -- do we really need to compute
                                                                 ; DLIST?
             (SETQ DLIST (TRANSLATE.DATATYPE TYPENAME FIELDSPECS))
             (SETQ LENGTH (|pop| DLIST)))))
    (OR (AND TYPENAME (LITATOM TYPENAME))
        (LISPERROR "ILLEGAL ARG" TYPENAME))
    (LET ((PTRS (|for| P |in| DLIST |when| (SELECTQ (|fetch| |fdType| |of| P)
                                                 ((POINTER FULLPOINTER)
                                                      T)
                                                 NIL)
                   |collect| (|fetch| |fdOffset| |of| P))))
         (CL:MULTIPLE-VALUE-BIND (TYPENUM REDECLARED)
             (\\ASSIGNDATATYPE1 TYPENAME DLIST LENGTH FIELDSPECS PTRS SUPERTYPE)
           (SETTOPVAL (\\TYPEGLOBALVARIABLE TYPENAME T)
                  TYPENUM)
           (CL:VALUES DLIST REDECLARED)))))

(TRANSLATE.DATATYPE
  (LAMBDA (TYPENAME FIELDSPECS)                                  (* DECLARATIONS\: (RECORD SPEC
                                                                 (N LEN . FD)))
    (DECLARE (SPECVARS TYPENAME UNUSED BIT OFFSET FD))           ; Edited 15-Dec-92 13:57 by jds
    (COND
       ((NULL TYPENAME))
       ((OR (NOT (LITATOM TYPENAME))
            (EQ TYPENAME '**DEALLOC**))
        (ERROR "Invalid type name" TYPENAME)))
```

```
(PROG ((N 0)
       UNUSED
       (OFFSET 0)
       (BIT 0)
       DLIST REUSE LEN FD)
      (SETQ DLIST (|for| S |in| FIELDSPECS
                     |collect| (|create| SPEC
                                   N _ (|add| N 1)
                                   LEN _ (SELECTQ S
                                              ((POINTER XPOINTER)
                                                  28)
                                              ((FIXP FLOATP SWAPPEDFIXP FULLPOINTER SWAPPEDXPOINTER
                                                     FULLXPOINTER)
                                                  BITSPERCELL)
                                              (FLAG (SETQQ S FLAGBITS)
                                                    1)
                                              (BYTE (SETQQ S BITS)
                                                    BITSPERBYTE)
                                              (WORD (SETQQ S BITS)
                                                    BITSPERWORD)
                                              (SIGNEDWORD (SETQQ S SIGNEDBITS)
                                                          BITSPERWORD)
                                              (SELECTQ (CAR (LISTP S))
                                                  ((BITS FLAGBITS SIGNEDBITS)
                                                       (PROG1 (CADR S)
                                                           (SETQ S (CAR S))))
                                                  (ERROR "invalid field spec: " S)))
                                   FD _ (|create| |FldDsc|
                                               |fdTypeName| _ TYPENAME
                                               |fdType| _ S
                                               |fdOffset| _ NIL))))
      (|for| S |in| DLIST
         |do| (|replace| |fdOffset| |of| (SETQ FD (|fetch| FD |of| S))
                |with| (SELECTQ (|fetch| |fdType| |of| FD)
                            ((POINTER XPOINTER)
                                 (COND
                                    ((AND TYPENAME
                                          (|find| X |in| UNUSED
                                             |suchthat| (AND (EQ 0 (LOGAND (CAR X)
                                                                            1))
                                                              (IGEQ (CADDR X)
                                                                    (- 28 BITSPERWORD))
                                                              (EQ (IPLUS (CADR X)
                                                                         (CADDR X))
                                                                  BITSPERWORD)
                                                              (|find| Y |in| UNUSED
                                                                 |suchthat| (AND (EQ (CAR Y)
                                                                                     (ADD1 (CAR X)))
                                                                                 (EQ (CADDR Y)
                                                                                     BITSPERWORD))))))
                                                                   ; unused 24 bit quantity
                                                                   ; this case not implemented yet

                                     ))
                                 (COND
                                    ((IGREATERP BIT 4)                 ; Less than 8 bits left in this word
                                     (\\REUSETO BITSPERWORD)))
                                 (COND
                                    ((ODDP OFFSET WORDSPERCELL)        ; not on double word boundary
                                     (\\REUSETO BITSPERWORD)))
                                 (COND
                                    ((NEQ BIT 4)
                                     (\\REUSETO 4 (EQ BIT 0))))
                                 (SETQ BIT 0)                          ;
                                 (PROG1 OFFSET (|add| OFFSET WORDSPERCELL)))
                            ((FIXP SWAPPEDFIXP FLOATP SWAPPEDXPOINTER)
                                                                   ; 32 bit quantities
                                 (COND
                                    ((NEQ BIT 0)
                                     (\\REUSETO BITSPERWORD)))
                                 (PROG1 OFFSET (|add| OFFSET WORDSPERCELL)))
                            ((FULLPOINTER FULLXPOINTER)            ; 32 bit doubleword-aligned quantities
                                 (COND
                                    ((NEQ BIT 0)
                                     (\\REUSETO BITSPERWORD)))
                                 (COND
                                    ((ODDP OFFSET WORDSPERCELL)
                                     (\\REUSETO BITSPERWORD)))
                                 (PROG1 OFFSET (|add| OFFSET WORDSPERCELL)))
                            ((BITS FLAGBITS SIGNEDBITS)
                                 (SETQ LEN (|fetch| LEN |of| S))
                                 (COND
                                    ((AND TYPENAME (SETQ REUSE (|find| X |in| UNUSED
                                                                  |suchthat| (ILEQ LEN (CADDR X)))))
                                     (RPLACA (CDDR REUSE)
                                            (IDIFFERENCE (CAR (CDDR REUSE))
                                                   LEN))
```

```
                                                 (|replace| |fdType| |of| FD |with| (CONS (|fetch| |fdType| |of| FD)
                                                                                        (|MakeBitField| (CADR REUSE)
                                                                                               LEN)))
                                                 (|add| (CADR REUSE)
                                                        LEN)
                                                 (CAR REUSE))
                                                ((IGREATERP LEN BITSPERWORD)
                                                                                ; more than 1 word --- Must right justify first word
                                                 (SETQ LEN (IDIFFERENCE LEN BITSPERWORD))
                                                 (COND
                                                    ((IGREATERP LEN (IDIFFERENCE BITSPERWORD BIT))
                                                     (\\REUSETO BITSPERWORD)))
                                                 (COND
                                                    ((NEQ (IDIFFERENCE BITSPERWORD BIT)
                                                          LEN)
                                                     (\\REUSETO (IDIFFERENCE BITSPERWORD LEN))))
                                                 (|replace| |fdType| |of| FD |with| (CONS 'LONGBITS (|MakeBitField| BIT LEN)))
                                                 (SETQ BIT 0)
                                                 (PROG1 OFFSET (|add| OFFSET 2)))
                                                (T (COND
                                                      ((IGREATERP LEN (IDIFFERENCE BITSPERWORD BIT))
                                                       (\\REUSETO BITSPERWORD)))
                                                   (|replace| |fdType| |of| FD |with| (CONS (|fetch| |fdType| |of| FD)
                                                                                          (|MakeBitField| BIT LEN)))
                                                   (|add| BIT LEN)
                                                   (PROG1 OFFSET
                                                       (COND
                                                          ((EQ BIT BITSPERWORD)
                                                           (SETQ BIT 0)
                                                           (|add| OFFSET 1)))))))
                                        (SHOULDNT))))
           (COND
              (TYPENAME (COND
                           ((NEQ BIT 0)
                            (\\REUSETO BITSPERWORD)))
                     (|while| (ODDP OFFSET WORDSPERCELL) |do| (|add| OFFSET 1))
                     (COND
                        ((IGREATERP OFFSET |\\MDSIncrement|)
                         (ERROR TYPENAME "DATATYPE TOO BIG")))))
           (RETURN (CONS OFFSET (MAPCAR DLIST (FUNCTION (LAMBDA (X)
                                                          (|fetch| FD |of| X)))))))))
```

```
(\\REUSETO
  (LAMBDA (N FLG)                                                     ; Edited 15-Dec-92 13:46 by jds

    ;; Skip over unused bits in a datatype or blockrecord declaration.  Advance the bin-int-word and word-offset pointers accordingly.  Complain if this
    ;; isn't supposed to be allowed.

    (SETQ N (IDIFFERENCE N BIT))
    (COND
       ((NEQ N 0)
        (COND
           ((AND (NULL TYPENAME)
                 (NOT FLG))
            (ERROR "Block/datatype field not aligned properly" FD)))
        (|push| UNUSED (LIST OFFSET BIT N))))
    (|add| BIT N)
    (COND
       ((EQ BIT 16)
        (SETQ BIT 0)
        (|add| OFFSET 1)))))
```

```
(\\TYPEGLOBALVARIABLE
  (LAMBDA (TYPENAME VARFLG)                                           ; Edited 18-May-87 17:14 by Snow

;;; Returns a constant or a variable that contains the datatype number of TYPENAME.  It is used when compiling type tests and assigning datatypes.  If
;;; TYPENAME is a system type, it returns the number.  Otherwise it creates a variable name and puts it on GLOBALVARS.

;;; This is a kludge that will go away when we have type resolution at load time.

;;; If VARFLG is true, always returns a var, rather than a system constant.  This is another kludge for backward compatibility.

    (OR (AND (NOT VARFLG)
             (|for| ENTRY |in| \\BUILT-IN-SYSTEM-TYPES |as| I |from| 1 |when| (EQ TYPENAME (CAR ENTRY))
                |do| (RETURN I)))
        (LET ((VAR (TYPE-VARIABLE-FROM-TYPE-NAME TYPENAME)))
             (COND
                ((NOT (OR (FMEMB VAR GLOBALVARS)
                          (GETPROP VAR 'GLOBALVAR)))
                 (PUTPROP VAR 'GLOBALVAR T)))
             VAR))))
)
```

```
(CL:DEFUN TYPE-VARIABLE-FROM-TYPE-NAME (TYPE-NAME)
;;; Convert a symbol naming a type into the unique global variable that holds the number for that type.  This can be tricky during that portion of the init
;;; before packages have been turned on.

    (IF (NULL *PACKAGE*)
        THEN ;; Packages are, indeed, not on yet.  We must check the type-name symbol to see if it begins with a known init-time package prefix.  If
             ;; so, we strip that off and put it back on the front.  The function NAMESTRING-CONVERSION-CLAUSE is from LLPACKAGE.

             (LET* ((BASE (|ffetch| (CL:SYMBOL PNAMEBASE) |of| TYPE-NAME))
                    (LEN (|ffetch| (CL:SYMBOL PNAMELENGTH) |of| TYPE-NAME))
                    (FATP (|ffetch| (CL:SYMBOL FATPNAMEP) |of| TYPE-NAME))
                    (CLAUSE (NAMESTRING-CONVERSION-CLAUSE BASE 1 LEN FATP)))
                   (COND
                      ((NULL CLAUSE)

                       ;; TYPE-NAME is homed in the Interlisp Package; we need to check for the types that are shared in IL and CL

                       (CL:DO ((S CMLSYMBOLS.SHARED (CDR S))
                               (HEADER "ÀÄ"))
                              ((COND
                                  ((NULL S))
                                  ((\\STRING-EQUALBASE (CAR S)
                                          BASE 1 LEN FATP)
                                   (SETQ HEADER "CL::ÀÄ")))
                               (PACK* HEADER TYPE-NAME "TYPE#"))))
                      (T                                         ; The symbol matched a clause.  We take the prefix off the front
                                                                 ; of the name and put it at the beginning.
                         (LET* ((PREFIX (CL:FIRST CLAUSE))
                                (PREFIX-LENGTH (FFETCH (STRINGP LENGTH) OF PREFIX))
                                (INTERNAL? (IF (EQ (CL:FOURTH CLAUSE)
                                                   :EXTERNAL)
                                               THEN ":"
                                             ELSE "")))

                               ;; The INTERNAL? jazz handles the bizaroid case of READTABLEP, and should handle future similar
                               ;; problems; I can't conceive of one of these kludgy type-variables EVER being an external symbol in any
                               ;; rational package

                               (PACK* PREFIX INTERNAL? "ÀÄ" (SUBSTRING TYPE-NAME (CL:1+ PREFIX-LENGTH))
                                      "TYPE#")))))
      ELSE ;; Packages are on; this is the normal case.

           (LET ((PACKAGE (CL:SYMBOL-PACKAGE TYPE-NAME)))
                (IF (OR (EQ PACKAGE *LISP-PACKAGE*)
                        (EQ PACKAGE *COMMON-LISP-PACKAGE*))
                    THEN ;; We have to be careful here; these two packages are intertwingled strangely

                         (LET* ((STRING (CONCAT "ÀÄ" (MKSTRING TYPE-NAME)
                                                "TYPE#"))
                                (LISPSYM (CL:FIND-SYMBOL STRING *LISP-PACKAGE*))
                                (CLSYM (CL:FIND-SYMBOL STRING *COMMON-LISP-PACKAGE*)))

                               ;; After this COND, CLSYM contains the type symbol; LISPSYM may also contain it.

                               (COND
                                  ((AND LISPSYM CLSYM)
                                   (IF (NOT (EQ LISPSYM CLSYM))
                                       THEN (ERROR "Somehow BOTH these type symbols exist" (LIST LISPSYM CLSYM))
                                     ))
                                  (LISPSYM (IMPORT (SETQ CLSYM LISPSYM)
                                                  *COMMON-LISP-PACKAGE*))
                                  (CLSYM (IMPORT CLSYM *LISP-PACKAGE*))
                                  (T (IMPORT (SETQ CLSYM (CL:INTERN STRING *COMMON-LISP-PACKAGE*))
                                            *LISP-PACKAGE*)))

                               ;; There's one last way to lose big-time here: we can have version skews such that the type# symbol already
                               ;; exists, and its home package is not the same as the home package of the type symbol.  Check that here and
                               ;; force the right thing (grrr)...

                               (CL:UNLESS (EQ PACKAGE (CL:SYMBOL-PACKAGE CLSYM))
                                   (CL:SETF (CL:SYMBOL-PACKAGE CLSYM)
                                          PACKAGE))
                               CLSYM)
                  ELSE (CL:INTERN (CONCAT "ÀÄ" (MKSTRING TYPE-NAME)
                                          "TYPE#")
                              PACKAGE)))))

(DEFINEQ

(|BitFieldMask|
  (LAMBDA (FD)                                               ; Edited 18-May-87 17:14 by Snow
    (SUB1 (LLSH 1 (|BitFieldWidth| FD)))))

(|BitFieldShift|
  (LAMBDA (FD)                                               ; Edited 18-May-87 17:14 by Snow
    (IDIFFERENCE 16 (IPLUS (|BitFieldFirst| FD)
                           (|BitFieldWidth| FD)))))

(|BitFieldShiftedMask|
  (LAMBDA (FD)                                                          ; Edited 18-May-87 17:15 by Snow
    (IDIFFERENCE (LLSH 1 (IDIFFERENCE 16 (|BitFieldFirst| FD)))
           (LLSH 1 (IDIFFERENCE 16 (IPLUS (|BitFieldFirst| FD)
                                          (|BitFieldWidth| FD)))))))

(|MakeBitField|
  (LAMBDA (FIRST WIDTH)                                                 ; Edited 18-May-87 17:15 by Snow
    (LOGOR (LLSH FIRST 4)
           (SUB1 WIDTH))))

(|BitFieldWidth|
  (LAMBDA (FD)                                                          ; Edited 18-May-87 17:16 by Snow
    (ADD1 (LOGAND FD 15))))

(|BitFieldFirst|
  (LAMBDA (FD)                                                          ; Edited 18-May-87 17:16 by Snow
    (LRSH FD 4)))
)

(DEFOPTIMIZER FETCHFIELD (&REST X)
                         (COMPILEDFETCHFIELD X))

(DEFOPTIMIZER FFETCHFIELD (&REST X)
                          (COMPILEDFETCHFIELD X T))

(DEFOPTIMIZER REPLACEFIELD (&REST X)
                           (COMPILEDREPLACEFIELD X))

(DEFOPTIMIZER FREPLACEFIELD (&REST X)
                            (COMPILEDREPLACEFIELD X T))

(DEFOPTIMIZER REPLACEFIELDVAL (&REST ARGS)
                              (CONS '(OPENLAMBDA (DESCRIPTOR DATUM NEWVALUE)
                                            (PROG1 DATUM (REPLACEFIELD DESCRIPTOR DATUM NEWVALUE)))
                                    ARGS))

(DEFOPTIMIZER FREPLACEFIELDVAL (&REST ARGS)
                               (CONS '(OPENLAMBDA (DESCRIPTOR DATUM NEWVALUE)
                                             (PROG1 DATUM (FREPLACEFIELD DESCRIPTOR DATUM NEWVALUE)))
                                     ARGS))

(DEFOPTIMIZER NCREATE (&REST X)
                      (COMPILEDNCREATE X))

(DEFOPTIMIZER \\DTEST (VALUE TYPE &ENVIRONMENT ENV)
                      (COND
                         ((AND (EQ (CAR TYPE)
                                   'QUOTE)
                               (LITATOM (CADR TYPE)))
                          (COND
                             ((FMEMB :4-BYTE (COMPILER::ENV-TARGET-ARCHITECTURE ENV))
                              `((OPCODES DTEST 0 0 0 (ATOM \\\, (CADR TYPE)))
                                ,VALUE))
                             ((FMEMB :3-BYTE (COMPILER::ENV-TARGET-ARCHITECTURE ENV))
                              `((OPCODES DTEST 0 0 (ATOM \\\, (CADR TYPE)))
                                ,VALUE))
                             (T `((OPCODES DTEST 0 (ATOM \\\, (CADR TYPE)))
                                  ,VALUE))))
                         (T 'IGNOREMACRO)))

(PUTPROPS \\TESTBITS DMACRO ((X N FD)
                             (NEQ 0 (\\GETBITS X N FD))))

(DEFINEQ

(COMPILEDFETCHFIELD
  (LAMBDA (X FASTFLG)                                                   ; Edited 18-May-87 17:32 by Snow
    (COND
       ((EQ (CAR (LISTP (CAR X)))
            'QUOTE)
        ((LAMBDA (DESCRIPTOR DATUM)
           (PROG (TYPENAME)
                 (COND
                    ((AND (NOT FASTFLG)
                          (SETQ TYPENAME (|fetch| |fdTypeName| |of| DESCRIPTOR)))
```

```
                     (SETQ DATUM (LIST (FUNCTION \\DTEST)
                                       DATUM
                                       (KWOTE TYPENAME)))))
                 (RETURN (SELECTQ (|fetch| |fdType| |of| DESCRIPTOR)
                              ((POINTER XPOINTER FULLPOINTER FULLXPOINTER)
                                   (LIST '\\GETBASEPTR DATUM (|fetch| |fdOffset| |of| DESCRIPTOR)))
                              (SWAPPEDXPOINTER
                                   `((OPENLAMBDA (D)
                                         (\\VAG2 (\\GETBASE D ,(ADD1 (|fetch| |fdOffset| |of| DESCRIPTOR)))
                                                (\\GETBASE D ,(|fetch| |fdOffset| |of| DESCRIPTOR))))
                                       ,DATUM))
                              (FLOATP `(\\GETBASEFLOATP ,DATUM ,(|fetch| |fdOffset| |of| DESCRIPTOR)))
                              (FIXP `(\\GETBASEFIXP ,DATUM ,(|fetch| |fdOffset| |of| DESCRIPTOR)))
                              (SWAPPEDFIXP `((OPENLAMBDA (D)
                                                 (\\MAKENUMBER (\\GETBASE D ,(ADD1 (|fetch| |fdOffset| |of| DESCRIPTOR
                                                                                            )))
                                                        (\\GETBASE D ,(|fetch| |fdOffset| |of| DESCRIPTOR))))
                                             ,DATUM))
                              (PROG ((FT (|fetch| |fdType| |of| DESCRIPTOR))
                                     (OFF (|fetch| |fdOffset| |of| DESCRIPTOR)))
                                    (RETURN (SELECTQ (CAR FT)
                                                (BITS (LIST '\\GETBITS DATUM OFF (CDR FT)))
                                                (SIGNEDBITS `(SIGNED (\\GETBITS ,DATUM ,OFF ,(CDR FT))
                                                                    ,(|BitFieldWidth| (CDR FT))))
                                                (FLAGBITS (LIST '\\TESTBITS DATUM OFF (CDR FT)))
                                                (LONGBITS `((OPENLAMBDA (D)
                                                                (\\MAKENUMBER (\\GETBITS D ,OFF
                                                                                         ,(CDR FT))
                                                                       (\\GETBASE D ,(ADD1 OFF))))
                                                            ,DATUM))
                                                (SHOULDNT)))))))
         (CADAR X)
         (CADR X)))
       (T 'IGNOREMACRO))))
```

(**COMPILEDREPLACEFIELD**

```
  (LAMBDA (X FASTFLG RPLVALFLG)                                         ; Edited 18-May-87 17:29 by Snow
```

;; compile code for replacing field values. Goes to great length to ensure that the coerced value is returned

```
    (COND
       ((EQ (CAR (LISTP (CAR X)))
            'QUOTE)
        ((LAMBDA (DESCRIPTOR DATUM NEWVALUE)
           (PROG ((TYPENAME (|fetch| |fdTypeName| |of| DESCRIPTOR))
                  (FT (|fetch| |fdType| |of| DESCRIPTOR))
                  (OFFSET (|fetch| |fdOffset| |of| DESCRIPTOR)))
                 (COND
                    ((AND (NOT FASTFLG)
                          TYPENAME)
                     (SETQ DATUM (LIST (FUNCTION \\DTEST)
                                       DATUM
                                       (KWOTE TYPENAME)))))
                 (RETURN (SELECTQ FT
                              ((POINTER FULLPOINTER)
                                   (LIST (FUNCTION \\RPLPTR)
                                         DATUM OFFSET NEWVALUE))
                              (XPOINTER (LIST (FUNCTION PUTBASEPTRX)
                                              DATUM OFFSET NEWVALUE))
                              (FULLXPOINTER (LIST '\\PUTBASEPTR DATUM OFFSET NEWVALUE))
                              (SWAPPEDXPOINTER
                                   `((OPENLAMBDA (D R)
                                         (\\PUTBASE D ,OFFSET (\\LOLOC R))
                                         (\\PUTBASE D ,(ADD1 OFFSET)
                                                (\\HILOC R))
                                         R)
                                       ,DATUM
                                       ,NEWVALUE))
                              (FIXP `(\\PUTBASEFIXP ,DATUM ,OFFSET ,NEWVALUE))
                              (SWAPPEDFIXP `(\\PUTSWAPPEDFIXP (\\ADDBASE ,DATUM ,OFFSET)
                                                   ,NEWVALUE))
                              (FLOATP `(\\PUTBASEFLOATP ,DATUM ,OFFSET ,NEWVALUE))
                              (SELECTQ (CAR FT)
                                   (BITS (LIST '\\PUTBITS DATUM OFFSET (CDR FT)
                                               NEWVALUE))
                                   (LONGBITS (LIST (SUBPAIR '(OFFSET FT)
                                                          (LIST OFFSET (CDR FT))
                                                          '(OPENLAMBDA (D V)
                                                              (\\PUTBITS D OFFSET FT (\\HINUM V))
                                                              (\\PUTBASE D (ADD1 OFFSET)
                                                                     (\\LONUM V))
                                                              V))
                                                   DATUM NEWVALUE))
                                   (SIGNEDBITS `(SIGNED (\\PUTBITS ,DATUM ,OFFSET ,(CDR FT)
                                                               (UNSIGNED ,NEWVALUE ,(|BitFieldWidth| (CDR FT))))
                                                       ,(|BitFieldWidth| (CDR FT))))
                                   (FLAGBITS `(NEQ (\\PUTBITS ,DATUM ,OFFSET ,(CDR FT)
                                                                      (COND
                                                                         (,NEWVALUE ,(|BitFieldMask| (CDR FT)))
                                                                         (T 0)))
                                                              0))
                                           (RETURN 'IGNOREMACRO))))))
          (CADAR X)
          (CADR X)
          (CADDR X)))
        (T 'IGNOREMACRO))))
```

(**COMPILEDNCREATE**

```
  (LAMBDA (X)                                                       ; Edited 18-May-87 17:34 by Snow
```

;;; compiles code for NCREATEs. Exists to eliminate the call to \TYPENUMBERFROMNAME.

```
    (COND
       ((EQ (CAR (LISTP (CAR X)))
            'QUOTE)
        (COND
           ((NULL (CADR X))
            (LIST 'CREATECELL (\\TYPEGLOBALVARIABLE (CADAR X))))
           (T (LIST 'NCREATE2 (\\TYPEGLOBALVARIABLE (CADAR X))
                     (CADR X)))))
       (T 'IGNOREMACRO))))

)

(DECLARE\: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE\: EVAL@COMPILE

(RECORD |FldDsc| (|fdTypeName| |fdOffset| |fdType|))
)
)
```

;; END EXPORTED DEFINITIONS

```
(RPAQQ DATATYPEFIELDTYPES
       ((FLOATP 0.0)
        (FIXP 0)
        (SWAPPEDFIXP 0)
        (POINTER NIL)
        (XPOINTER NIL)
        (FULLPOINTER NIL)
        (FULLXPOINTER NIL)
        (SWAPPEDXPOINTER NIL)
        (FLAG NIL)
        (BYTE 0)
        (WORD 0)
        (SIGNEDWORD 0)))
```

;; Macros which convert a record access form into an address-generating form

```
(DECLARE\: EVAL@COMPILE

(PUTPROPS LOCF DMACRO (X (TRANSLATE.LOCF X)))

(PUTPROPS INDEXF DMACRO (X (TRANSLATE.LOCF X T)))
)

(DEFINEQ
```

(**TRANSLATE.LOCF**

```
  (LAMBDA (ARGS INDEXONLY)                                          ; Edited 18-May-87 17:35 by Snow
    (DECLARE (GLOBALVARS CLISPARRAY))
    (PROG ((FORM (MKPROGN ARGS))
           NEWFORM OFFSET SPEC)
      RETRY
          (SELECTQ (CAR (LISTP FORM))
               (PROGN (COND
                         ((NOT (CDDR FORM))                         ; get rid of extra PROGN's inserted by record package
                          (SETQ FORM (CADR FORM))
                          (GO RETRY))))
               ((FETCHFIELD FFETCHFIELD)
                    (COND
                       ((AND (SETQ OFFSET (LISTP (CADR FORM)))
                             (EQ (CAR OFFSET)
                                 'QUOTE)
                             (SETQ OFFSET (CADR (SETQ SPEC (CADR OFFSET))))
                             (FIXP OFFSET))
                        (RETURN (COND
                                   (INDEXONLY OFFSET)
                                   ((EQ OFFSET 0)
                                          (CADDR FORM))
                                         (T (SETQ FORM (CADDR FORM))

                                            ;; loop in order to merge \ADDBASEs.  Should actually be done by compiler optimization, but
                                            ;; apparently that is currently broken

                                            (|repeatwhile| (SELECTQ (CAR (LISTP FORM))
                                                                (PROGN (COND
                                                                          ((NULL (CDDR FORM))
                                                                           (SETQ FORM (CADR FORM))
                                                                           T)))
                                                                ((ADDBASE \\ADDBASE)
                                                                     (COND
                                                                        ((FIXP (CADDR FORM))
                                                                         (|add| OFFSET (CADDR FORM))
                                                                         (SETQ FORM (CADR FORM))
                                                                         T)))
                                                                (COND
                                                                   ((NEQ (SETQ NEWFORM (CL:MACROEXPAND-1 FORM))
                                                                         FORM)
                                                                    (SETQ FORM NEWFORM)
                                                                    T))))
                                            (LIST '\\ADDBASE FORM OFFSET)))))))
                  (COND
                     ((NEQ FORM (SETQ FORM (CL:MACROEXPAND-1 FORM)))
                      (GO RETRY))))
             (ERROR "LOCF Can't figure out this argument" ARGS)
             (RETURN 'IGNOREMACRO))))
)

(DECLARE\: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(PUTPROPS DTDECLARE FILETYPE CL:COMPILE-FILE)

(PUTPROPS DTDECLARE COPYRIGHT ("Venue & Xerox Corporation" 1981 1982 1983 1984 1985 1986 1987 1990 1992 1993))
```

FUNCTION INDEX

OPTIMIZER INDEX

MACRO INDEX

PROPERTY INDEX

VARIABLE INDEX

RECORD INDEX