File created: 19-Jan-93 10:43:28 **{DSK}<python>lde>lispcore>sources>LLARRAYELT.;2**

changes to: (RECORDS HARRAYP HASHSLOT SEQUENCEDESCRIPTOR ARRAYP ARRAYBLOCK SAFTABLE)

previous date: 4-Jan-93 23:51:47 {DSK}<python>lde>lispcore>sources>LLARRAYELT.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993 by Venue & Xerox Corporation.  All rights reserved.

(RPAQQ LLARRAYELTCOMS
       [(PROPS (LLARRAYELT FILETYPE))
        (COMS                                                   ; ARRAY entries
              (FNS AIN AOUT ARRAY ARRAYSIZE ARRAYTYP ARRAYORIG COPYARRAY)
              (DECLARE%: DONTCOPY (MACROS ARRAYSIZE))
              (FNS ELT ELTD SETA SETD SUBARRAY))
        [COMS                                                   ; HASHARRAY entries
              (FNS HARRAY HASHARRAY HARRAYP HARRAYPROP HARRAYSIZE CLRHASH MAPHASH GETHASH PUTHASH CL::PUTHASH
                   REMHASH \HASHRECLAIM \HASHACCESS REHASH \COPYHARRAYP \HASHTABLE.DEFPRINT)
              (FNS STRINGHASHBITS STRING-EQUAL-HASHBITS)
              (FNS \STRINGHASHBITS-UFN \STRING-EQUAL-HASHBITS-UFN)
              (DECLARE%: DONTCOPY (EXPORT (RECORDS HARRAYP)
                                          (MACROS \EQHASHINGBITS))
                     (RECORDS HASHSLOT)
                     (MACROS \FIRSTINDEX \HASHSLOT \REPROBE)
                     (CONSTANTS (CELLSPERSLOT 2))
                     (GLOBALVARS \HASH.NULL.VALUE SYSHASHARRAY))
              [DECLARE%: DONTEVAL@LOAD DOCOPY (P (DEFPRINT 'HARRAYP '\HASHTABLE.DEFPRINT]
              (INITRECORDS HARRAYP)
              (SYSRECORDS HARRAYP)
              (VARS (\HASH.NULL.VALUE '\Hash\Null\Value\]
        (COMS                                                   ; System entries for CODE
              (FNS \CODEARRAY \FIXCODENUM \FIXCODEPTR \FIXCODESYM))
        (COMS                                                   ; Internal
              (DECLARE%: DONTCOPY (MACROS EQPTR BUCKETINDEX FREEBLOCKCHAIN.N)
                     (CONSTANTS \MAXBUCKETINDEX)
                                                                ; \ADDBASE2 and \ADDBASE4 do \ADDBASE of 2*N and 4*N
                                                                ; without boxing
                     (EXPORT (MACROS \ADDBASE2 \ADDBASE4 HUNKSIZEFROMNUMBER \BYTELT \BYTESETA \WORDELT)
                            (CONSTANTS * BLOCKGCTYPECONSTANTS)
                            (CONSTANTS * ARRAYCONSTANTS)
                            (CONSTANTS * ARRAYTYPES)
                            (CONSTANTS \MAX.CELLSPERHUNK)
                            (CONSTANTS (\IN.MAKEINIT))
                            (RECORDS SEQUENCEDESCRIPTOR ARRAYP ARRAYBLOCK)
                            (GLOBALVARS \NxtArrayPage \FREEBLOCKBUCKETS \HUNKING?))
                     (GLOBALVARS \ArrayFrLst \ArrayFrLst2 \RECLAIM.COUNTDOWN))
              (FNS \ALLOCBLOCK \MAIKO.ALLOCBLOCK \ALLOCBLOCK.OLD \ALLOCBLOCK.NEW \PREFIXALIGNMENT?
                   \MAKEFREEARRAYBLOCK \DELETEBLOCK? \LINKBLOCK \MERGEBACKWARD \MERGEFORWARD \ARRAYBLOCKMERGER
                   \#BLOCKDATACELLS \COPYARRAYBLOCK \RECLAIMARRAYBLOCK \ADVANCE.ARRAY.SEGMENTS)
              (ADDVARS (\MAIKO.MOVDS (\MAIKO.ALLOCBLOCK \ALLOCBLOCK)))
              (FNS \BYTELT \BYTESETA \WORDELT)
              (FNS \ARRAYTYPENAME)
              (VARS (\ARRAYMERGING T))
              (GLOBALVARS \ARRAYMERGING)
              (COMS                                             ; for STORAGE
                    (FNS \SHOW.ARRAY.FREELISTS)
                    (INITVARS (\ABSTORAGETABLE NIL))
                    (GLOBALVARS \ABSTORAGETABLE)
                    (DECLARE%: DONTCOPY (RECORDS SAFTABLE)))
              (COMS                                             ; Debugging and RDSYS
                    (FNS \CHECKARRAYBLOCK \PARSEARRAYSPACE \PARSEARRAYSPACE1)
                    (INITVARS (ARRAYBLOCKCHECKING))
                    (GLOBALVARS ARRAYBLOCKCHECKING)))
        (COMS                                                   ; Basic hunking
              (FNS \ALLOCHUNK)
              (VARS \HUNK.PTRSIZES)
                                                                ; Compiler needs \HUNK.PTRSIZES for creating closure
                                                                ; environments
              (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (MACROS HUNKSIZEFROMNUMBER))
                     (CONSTANTS \HUNK.UNBOXEDSIZES \HUNK.CODESIZES \HUNK.PTRSIZES)
                     (GLOBALVARS \HUNKING? \UNBOXEDHUNK.TYPENUM.TABLE \CODEHUNK.TYPENUM.TABLE
                            \PTRHUNK.TYPENUM.TABLE))
              (COMS ;; Keep a list of all the hunks rejected due to poor page-straddling alignment, or to code falling off the end of a doublepage
                    (VARS (\HUNKREJECTS))
                    (GLOBALVARS \HUNKREJECTS)))
        [COMS                                                   ; for MAKEINIT
              (FNS PREINITARRAYS POSTINITARRAYS FILEARRAYBASE FILEBLOCKTRAILER FILECODEBLOCK FILEPATCHBLOCK)
              (COMS                                             ; Hunk Initialization
                    (FNS \SETUP.HUNK.TYPENUMBERS \COMPUTE.HUNK.TYPEDECLS \TURN.ON.HUNKING \SETUP.TYPENUM.TABLE))
```

```
            (DECLARE%: DONTCOPY (ADDVARS (INITVALUES (\NxtArrayPage)
                                                     (\HUNKING?))
                                         (INITPTRS (\FREEBLOCKBUCKETS)
                                                (\ArrayFrLst)
                                                (\ArrayFrLst2)
                                                (\UNBOXEDHUNK.TYPENUM.TABLE)
                                                (\CODEHUNK.TYPENUM.TABLE)
                                                (\PTRHUNK.TYPENUM.TABLE))
                                         (INEWCOMS (FNS \#BLOCKDATACELLS \PREFIXALIGNMENT? \ALLOCBLOCK
                                                        \MAIKO.ALLOCBLOCK \ALLOCBLOCK.NEW \MAKEFREEARRAYBLOCK
                                                        \MERGEBACKWARD \LINKBLOCK \ALLOCHUNK)
                                                (FNS PREINITARRAYS POSTINITARRAYS FILEARRAYBASE FILEBLOCKTRAILER
                                                     FILECODEBLOCK FILEPATCHBLOCK)
                                                (FNS \SETUP.HUNK.TYPENUMBERS \COMPUTE.HUNK.TYPEDECLS
                                                     \TURN.ON.HUNKING \SETUP.TYPENUM.TABLE))
                                         (MKI.SUBFNS (\IN.MAKEINIT . T)
                                                (\ALLOCBLOCK.OLD . NILL)
                                                (\MERGEFORWARD . NILL)
                                                (\FIXCODENUM . I.FIXUPNUM)
                                                (\FIXCODESYM . I.FIXUPSYM)
                                                (\FIXCODEPTR . I.FIXUPPTR)
                                                (\CHECKARRAYBLOCK . NILL)
                                                (\ARRAYMERGING PROGN NIL))
                                         (EXPANDMACROFNS \ADDBASE2 \ADDBASE4 HUNKSIZEFROMNUMBER BUCKETINDEX
                                                 FREEBLOCKCHAIN.N)
                                         (RDCOMS (FNS \CHECKARRAYBLOCK \PARSEARRAYSPACE \PARSEARRAYSPACE1))
                                         (RD.SUBFNS (EQPTR . EQUAL)
                                                (ARRAYBLOCKCHECKING . T))
                                         (RDPTRS (\FREEBLOCKBUCKETS))
                                         (RDVALS (\ArrayFrLst)
                                                (\ArrayFrLst2)))
                        EVAL@COMPILE
                        (ADDVARS (DONTCOMPILEFNS PREINITARRAYS POSTINITARRAYS FILEARRAYBASE FILEBLOCKTRAILER
                                         FILECODEBLOCK FILEPATCHBLOCK)
                               (DONTCOMPILEFNS \SETUP.HUNK.TYPENUMBERS \COMPUTE.HUNK.TYPEDECLS \TURN.ON.HUNKING
                                      \SETUP.TYPENUM.TABLE]
      (COMS                                                    ; Debugging aids
            (DECLARE%: EVAL@COMPILE DONTCOPY (GLOBALVARS \ArrayFrLst)
                  (CONSTANTS \ArrayBlockPassword)
                  (ADDVARS (DONTCOMPILEFNS \HUNKFIT? \AB.NEXT \AB.BACK)))
            (FNS \HUNKFIT? \AB.NEXT \AB.BACK))
      (LOCALVARS . T)
      (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA CL::PUTHASH HARRAYPROP])

(PUTPROPS LLARRAYELT FILETYPE :BCOMPL)

;; ARRAY entries

(DEFINEQ

(AIN
  [LAMBDA (APTR INDEX N FILE)                                  ; Edited 23-Nov-86 16:44 by jop:
    ;; Reads N elements into APTR starting at INDEX.  INDEX and N are in terms of the array's indexing units
    (COND
       ((NOT (OR (STRINGP APTR)
                 (ARRAYP APTR)))
        (LISPERROR "ILLEGAL ARG" APTR))
       ((IGREATERP 0 INDEX)
        (LISPERROR "ILLEGAL ARG" INDEX)))
    (LET (BASE LENGTH OFFST TYP ORIG STBYTE)
         (if (STRINGP APTR)
             then (SETQ BASE (ffetch (STRINGP BASE) of APTR))
                  (SETQ LENGTH (ffetch (STRINGP LENGTH) of APTR))
                  (SETQ OFFST (ffetch (STRINGP OFFST) of APTR))
                  (SETQ TYP (ffetch (STRINGP TYP) of APTR))
                  (SETQ ORIG 1)
           else (SETQ BASE (ffetch (ARRAYP BASE) of APTR))
                (SETQ LENGTH (ffetch (ARRAYP LENGTH) of APTR))
                (SETQ OFFST (ffetch (ARRAYP OFFST) of APTR))
                (SETQ TYP (ffetch (ARRAYP TYP) of APTR))
                (SETQ ORIG (ffetch (ARRAYP ORIG) of APTR)))
         (SETQ STBYTE (IDIFFERENCE INDEX ORIG))
         (COND
            ((ILESSP (SELECTC TYP
                             ((LIST \ST.BYTE \ST.CODE)
                                 LENGTH)
                             (\ST.POS16 (SETQ OFFST (UNFOLD OFFST BYTESPERWORD))
                                        (SETQ N (UNFOLD N BYTESPERWORD))
                                        (SETQ STBYTE (UNFOLD STBYTE BYTESPERWORD))
                                        (UNFOLD LENGTH BYTESPERWORD))
                             ((LIST \ST.INT32 \ST.FLOAT)
                                  (SETQ OFFST (UNFOLD OFFST BYTESPERCELL))
                                  (SETQ N (UNFOLD N BYTESPERCELL))
                                  (SETQ STBYTE (UNFOLD STBYTE BYTESPERCELL))
```

```
                              (UNFOLD LENGTH BYTESPERCELL))
                         (\ST.BIT)
                         (LISPERROR "ILLEGAL ARG" APTR))
                    (IPLUS STBYTE N))
               (LISPERROR "ILLEGAL ARG" APTR)))
          (\BINS (\GETOFD FILE 'INPUT)
                 BASE
                 (IPLUS STBYTE OFFST)
                 N)
          APTR])
```

(AOUT

```
[LAMBDA (APTR INDEX N FILE)                                   ; Edited 23-Nov-86 16:49 by jop:
                                                              ; INDEX and N are in terms of the array's indexing unit
   (COND
      ((NOT (OR (STRINGP APTR)
                (ARRAYP APTR)))
       (LISPERROR "ILLEGAL ARG" APTR))
      ((IGREATERP 0 INDEX)
       (LISPERROR "ILLEGAL ARG" INDEX)))
   ;; Used to be in terms of the block record SEQUENCEDESCRIPTOR, but changed to refer explicitly to arrayp's and stringp's since stringp's no
   ;; longer look like arrayp's
   (LET (BASE LENGTH OFFST TYP ORIG STBYTE)
        (if (STRINGP APTR)
            then (SETQ BASE (ffetch (STRINGP BASE) of APTR))
                 (SETQ LENGTH (ffetch (STRINGP LENGTH) of APTR))
                 (SETQ OFFST (ffetch (STRINGP OFFST) of APTR))
                 (SETQ TYP (ffetch (STRINGP TYP) of APTR))
                 (SETQ ORIG 1)
          else (SETQ BASE (ffetch (ARRAYP BASE) of APTR))
               (SETQ LENGTH (ffetch (ARRAYP LENGTH) of APTR))
               (SETQ OFFST (ffetch (ARRAYP OFFST) of APTR))
               (SETQ TYP (ffetch (ARRAYP TYP) of APTR))
               (SETQ ORIG (ffetch (ARRAYP ORIG) of APTR)))
        (SETQ STBYTE (IDIFFERENCE INDEX ORIG))                ; Standardize units before comparing
        (COND
           ((ILESSP (SELECTC TYP
                            ((LIST \ST.BYTE \ST.CODE)
                                LENGTH)
                            (\ST.POS16 (SETQ N (UNFOLD N BYTESPERWORD))
                                       (SETQ STBYTE (UNFOLD STBYTE BYTESPERWORD))
                                       (SETQ OFFST (UNFOLD OFFST BYTESPERWORD))
                                       (UNFOLD LENGTH BYTESPERWORD))
                            ((LIST \ST.INT32 \ST.FLOAT)
                                (SETQ N (UNFOLD N BYTESPERCELL))
                                (SETQ STBYTE (UNFOLD STBYTE BYTESPERCELL))
                                (SETQ OFFST (UNFOLD OFFST BYTESPERCELL))
                                (UNFOLD LENGTH BYTESPERCELL))
                            (LISPERROR "ILLEGAL ARG" APTR))
                    (IPLUS STBYTE N))
               (LISPERROR "ILLEGAL ARG" APTR)))
        (\BOUTS (\GETOFD FILE 'OUTPUT)
                BASE
                (IPLUS STBYTE OFFST)
                N)
        APTR])
```

(ARRAY

```
[LAMBDA (SIZE TYPE INITVAL ORIG ALIGN)                        (* JonL "20-Sep-84 19:46")
   ;; extension of the normal VM definition of an array to allow many different TYPEs, and also allows ORIG of 0
   (SETQ SIZE (FIX SIZE))
   (COND
      ((OR (IGREATERP 0 SIZE)
           (IGREATERP SIZE \MaxArrayLen))
       (LISPERROR "ILLEGAL ARG" SIZE)))                       ; Coerce floats at outset; \ALLOCARRAY wants fixp
   (PROG (AP TYP GCTYPE (NCELLS SIZE))
         [SETQ TYP (SELECTQ TYPE
                           (BYTE (SETQ NCELLS (FOLDHI SIZE BYTESPERCELL))
                                 \ST.BYTE)
                           ((SMALLP SMALLPOSP WORD)
                               (SETQ NCELLS (FOLDHI SIZE WORDSPERCELL))
                               \ST.POS16)
                           ((NIL POINTER FLAG)
                               (SETQ GCTYPE PTRBLOCK.GCT)
                               \ST.PTR)
                           ((0 DOUBLEPOINTER)                ; INTERLISP-10 style arrays--each element is 2 cells
                               (SETQ NCELLS (UNFOLD SIZE 2))
                               (SETQ GCTYPE PTRBLOCK.GCT)
                               \ST.PTR2)
                           (FIXP \ST.INT32)
                           (FLOATP [COND
                                      (INITVAL (SETQ INITVAL (FLOAT INITVAL]
                                   \ST.FLOAT)
```

```
                        (BIT (SETQ NCELLS (FOLDHI SIZE BITSPERCELL))
                             \ST.BIT)
                        (SIGNEDWORD \ST.INT32)
                        (COND
                           ((EQ SIZE TYPE)                    ; = FIXP
                            \ST.INT32)
                           ((AND (LISTP TYPE)
                                 (EQ (CAR TYPE)
                                     'BITS))
                            (COND
                               ((IGREATERP (CADR TYPE)
                                           16)
                                \ST.INT32)
                               ((IGREATERP (CADR TYPE)
                                           8)
                                (SETQ NCELLS (FOLDHI SIZE WORDSPERCELL))
                                \ST.POS16)
                               ((IGREATERP (CADR TYPE)
                                           1)
                                (SETQ NCELLS (FOLDHI SIZE BYTESPERCELL))
                                \ST.BYTE)
                               (T (SETQ NCELLS (FOLDHI SIZE BITSPERCELL))
                                  \ST.BIT)))
                           (T (\ILLEGAL.ARG TYPE]
                (SETQ AP (create ARRAYP
                                TYP _ TYP
                                LENGTH _ SIZE
                                ORIG _ (SELECTQ ORIG
                                               ((0 1)
                                                   ORIG)
                                               (NIL 1)
                                               (LISPERROR "ILLEGAL ARG" ORIG))
                                OFFST _ 0
                                BASE _ (\ALLOCBLOCK NCELLS GCTYPE NIL ALIGN)))
                [AND INITVAL (PROG ((BASE (fetch (ARRAYP BASE) of AP))
                                    (NWORDS (SUB1 (UNFOLD NCELLS WORDSPERCELL)))
                                    LASTWORD2BASE)
                                   (SETQ LASTWORD2BASE (\ADDBASE BASE (SUB1 NWORDS)))
                                   (SELECTC TYP
                                        (\ST.BYTE (OR (EQ 0 INITVAL)
                                                      (PROGN (\PUTBASE LASTWORD2BASE 1
                                                                       (create WORD
                                                                               HIBYTE _ INITVAL
                                                                               LOBYTE _ INITVAL))
                                                             (\BLT BASE (\ADDBASE BASE 1)
                                                                   NWORDS))))
                                        (\ST.POS16 (OR (EQ 0 INITVAL)
                                                       (PROGN (\PUTBASE LASTWORD2BASE 1 INITVAL)
                                                              (\BLT BASE (\ADDBASE BASE 1)
                                                                    NWORDS))))
                                        (\ST.INT32 [OR (EQ 0 INITVAL)
                                                       (PROGN (\PUTBASEFIXP LASTWORD2BASE 0 INITVAL)
                                                              (\BLT BASE (\ADDBASE BASE WORDSPERCELL)
                                                                    (SUB1 NWORDS])
                                        ((LIST \ST.PTR \ST.PTR2)       ; Remove \ST.FLOAT when FLOATP is no longer stored in PTR
                                                                       ; mode.
                                            [PROG ((P BASE))
                                                  (FRPTQ NCELLS (\RPLPTR P 0 INITVAL)
                                                         (SETQ P (\ADDBASE P WORDSPERCELL])
                                        (\ST.FLOAT [OR (FEQP 0.0 INITVAL)
                                                       (PROGN (\PUTBASEFLOATP LASTWORD2BASE 0 INITVAL)
                                                              (\BLT BASE (\ADDBASE BASE WORDSPERCELL)
                                                                    (SUB1 NWORDS])
                                        (\ST.BIT (OR (EQ 0 INITVAL)
                                                     (PROGN (\PUTBASE LASTWORD2BASE 1 MASKWORD1'S)
                                                            (\BLT BASE (\ADDBASE BASE 1)
                                                                  NWORDS))))
                                        (SHOULDNT]
                (RETURN AP])
```

(ARRAYSIZE

```
  [LAMBDA (X)                                                  (* JonL " 4-NOV-83 12:44")
    (\MACRO.MX (ARRAYSIZE X])
```

(ARRAYTYP

```
  [LAMBDA (ARRAY)                                              (* rmk%: "30-Dec-83 13:12")
    ;; This is a VM function which returns valid 2nd argument to ARRAY
    (SELECTC (fetch (ARRAYP TYP) of (\DTEST ARRAY 'ARRAYP))
         (\ST.BYTE 'BYTE)
         (\ST.PTR2 'DOUBLEPOINTER)
         (\ST.PTR 'POINTER)
         (\ST.POS16 'SMALLPOSP)
         (\ST.CODE                                             ; not valid 2nd arg to ARRAY
                   'CODE)
```

```
        (\ST.INT32 'FIXP)
        (\ST.FLOAT 'FLOATP)
        (\ST.BIT 'BIT)
        (SHOULDNT])
```

(ARRAYORIG

```
  [LAMBDA (ARRAY)                                                    (* rmk%: "30-Dec-83 13:12")
    (fetch (ARRAYP ORIG) of (\DTEST ARRAY 'ARRAYP])
```

(COPYARRAY

```
  [LAMBDA (ARRAY)                                                    (* JonL "16-Oct-84 20:38")
    (COND
       [(HARRAYP ARRAY)
        (PROG [(NHARRAY (HASHARRAY (HARRAYSIZE ARRAY]
              (\COPYHARRAYP ARRAY NHARRAY)
              (RETURN (REHASH ARRAY NHARRAY]
       (T (PROG (NEWARRAY INDEX (ORIG (ARRAYORIG ARRAY))
                      (TYPE (ARRAYTYP ARRAY))
                      (SIZE (ARRAYSIZE ARRAY)))
                (SETQ NEWARRAY (ARRAY SIZE TYPE NIL ORIG))
                (SETQ INDEX ORIG)
                (FRPTQ SIZE (SETA NEWARRAY INDEX (ELT ARRAY INDEX))
                      (add INDEX 1))
                (SELECTQ TYPE
                     ((DOUBLEPOINTER)
                          (SETQ INDEX ORIG)
                          (FRPTQ SIZE (SETD NEWARRAY INDEX (ELTD ARRAY INDEX))
                                 (add INDEX 1)))
                     NIL)
                (RETURN NEWARRAY])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS ARRAYSIZE DMACRO [(A)
                            (ffetch (ARRAYP LENGTH) of (\DTEST A 'ARRAYP])
)
)

(DEFINEQ
```

(ELT

```
  [LAMBDA (A N)                                                      (* lmm " 7-Jun-84 17:53")
    (\DTEST A 'ARRAYP)
    (PROG [(BASE (fetch (ARRAYP BASE) of A))
           (N0 (IDIFFERENCE N (fetch (ARRAYP ORIG) of A]
          (COND
             ((OR (IGREATERP 0 N0)
                  (IGEQ N0 (fetch (ARRAYP LENGTH) of A)))
              (LISPERROR "ILLEGAL ARG" N)))
          (SETQ N0 (IPLUS N0 (fetch (ARRAYP OFFST) of A)))
          (RETURN (SELECTC (fetch (ARRAYP TYP) of A)
                       ((LIST \ST.PTR \ST.PTR2)
                            (\GETBASEPTR (\ADDBASE2 BASE N0)
                                   0))
                       (\ST.INT32 (SETQ BASE (\ADDBASE2 BASE N0))
                                  (\MAKENUMBER (\GETBASE BASE 0)
                                         (\GETBASE BASE 1)))
                       ((LIST \ST.BYTE \ST.CODE)
                            (\GETBASEBYTE BASE N0))
                       (\ST.POS16 (\GETBASE BASE N0))
                       (\ST.BIT (LOGAND (LRSH (\GETBASE BASE (FOLDLO N0 BITSPERWORD))
                                              (IDIFFERENCE (SUB1 BITSPERWORD)
                                                     (IMOD N0 BITSPERWORD)))
                                        1))
                       (\ST.FLOAT (\GETBASEFLOATP BASE (UNFOLD N0 WORDSPERCELL)))
                       (LISPERROR "ILLEGAL ARG" A])
```

(ELTD

```
  [LAMBDA (A N)                                                      (* rmk%: "30-Dec-83 13:13")
    (\DTEST A 'ARRAYP)
    (SELECTC (fetch (ARRAYP TYP) of A)
         (\ST.PTR2 (PROG [(BASE (fetch (ARRAYP BASE) of A))
                          (N0 (IDIFFERENCE N (fetch (ARRAYP ORIG) of A]
                         (COND
                            ((OR (IGREATERP 0 N0)
                                 (IGEQ N0 (fetch (ARRAYP LENGTH) of A)))
                             (LISPERROR "ILLEGAL ARG" N)))
                         (SETQ N0 (IPLUS N0 (fetch (ARRAYP OFFST) of A)))
                         (RETURN (\GETBASEPTR (\ADDBASE2 (\ADDBASE2 BASE (fetch (ARRAYP LENGTH) of A))
                                                N0)
```

```
                                    0))))
        (ELT A N])
```

(**SETA**

```
  [LAMBDA (A N V)                                                    (* bvm%: " 6-Feb-85 15:54")
    (COND
       ([fetch (ARRAYP READONLY) of (SETQ A (\DTEST A 'ARRAYP]
        (LISPERROR "ILLEGAL ARG" A)))
    (PROG [(BASE (fetch (ARRAYP BASE) of A))
           (N0 (IDIFFERENCE N (fetch (ARRAYP ORIG) of A]
          (COND
             ((OR (ILESSP N0 0)
                  (IGEQ N0 (fetch (ARRAYP LENGTH) of A)))
              (LISPERROR "ILLEGAL ARG" N)))
          (SETQ N0 (IPLUS N0 (fetch (ARRAYP OFFST) of A)))
          (RETURN (SELECTC (fetch (ARRAYP TYP) of A)
                       ((LIST \ST.PTR \ST.PTR2)
                           (\RPLPTR (\ADDBASE2 BASE N0)
                                  0 V))
                       (\ST.INT32                                    ; 32-bit 2's complement integers
                                 (\PUTBASEFIXP (\ADDBASE2 BASE N0)
                                        0 V))
                       ((LIST \ST.BYTE \ST.CODE)
                           (\PUTBASEBYTE BASE N0 V))
                       (\ST.POS16                                    ; Unsigned 16-bit numbers
                                 (\PUTBASE BASE N0 V))
                       (\ST.BIT [\PUTBASE BASE (FOLDLO N0 BITSPERWORD)
                                       (COND
                                          [(EQ 0 V)
                                           (LOGAND (\GETBASE BASE (FOLDLO N0 BITSPERWORD))
                                                   (LOGXOR (LLSH 1 (IDIFFERENCE (SUB1 BITSPERWORD)
                                                                                (IMOD N0 BITSPERWORD)))
                                                           (SUB1 (LLSH 1 BITSPERWORD]
                                          (T (LOGOR (\GETBASE BASE (FOLDLO N0 BITSPERWORD))
                                                    (LLSH 1 (IDIFFERENCE (SUB1 BITSPERWORD)
                                                                         (IMOD N0 BITSPERWORD]
                                V)
                       (\ST.FLOAT (\PUTBASEFLOATP BASE (UNFOLD N0 WORDSPERCELL)
                                         (FLOAT V)))
                       (LISPERROR "ILLEGAL ARG" A])
```

(**SETD**

```
  [LAMBDA (A N V)                                                    (* rmk%: "30-Dec-83 13:14")
    (\DTEST A 'ARRAYP)
    (SELECTC (fetch (ARRAYP TYP) of A)
        (\ST.PTR2 (COND
                     ((fetch (ARRAYP READONLY) of A)
                      (LISPERROR "ILLEGAL ARG" A)))
                  (PROG [(BASE (fetch (ARRAYP BASE) of A))
                         (N0 (IDIFFERENCE N (fetch (ARRAYP ORIG) of A]
                        (COND
                           ((OR (IGREATERP 0 N0)
                                (IGEQ N0 (fetch (ARRAYP LENGTH) of A)))
                            (LISPERROR "ILLEGAL ARG" N)))
                        (SETQ N0 (IPLUS N0 (fetch (ARRAYP OFFST) of A)))
                        (\RPLPTR (\ADDBASE2 (\ADDBASE2 BASE (fetch (ARRAYP LENGTH) of A))
                                            N0)
                                 0 V)
                        (RETURN V)))
        (SETA A N V])
```

(**SUBARRAY**

```
  [LAMBDA (X N M OLD NEWORIG)                                        (* rmk%: "30-Dec-83 13:15")
    (\DTEST X 'ARRAYP)
    (PROG ((LEN (fetch (ARRAYP LENGTH) of X))
           (ORIG (fetch (ARRAYP ORIG) of X))
           (N1 N)
           (M1 M))                                                   ; N1 and M1 so don't reset user arg
          [COND
             ((IGREATERP 0 N1)                                      ; Coerce the first index
              (SETQ N1 (IPLUS N1 LEN 1]
          [COND
             ((NULL M1)                                             ; Now coerce the second index
              (SETQ M1 LEN))
             ((IGREATERP 0 M1)
              (SETQ M1 (IPLUS M1 LEN 1]                             ; Go uninterruptable to protect the OLD~=NIL case.
          (RETURN (AND (IGEQ N1 ORIG)
                       (ILEQ N1 M1)
                       (ILEQ M1 LEN)
                       (UNINTERRUPTABLY
                           (create ARRAYP smashing (OR (ARRAYP OLD)
                                                       (create ARRAYP))
                                   BASE _ (fetch (ARRAYP BASE) of X)
                                   LENGTH _ (ADD1 (IDIFFERENCE M1 N1))
```

```
                                        TYP _ (fetch (ARRAYP TYP) of X)
                                        OFFST _ (IDIFFERENCE (IPLUS (fetch (ARRAYP OFFST) of X)
                                                                    N1)
                                                             ORIG)
                                        ORIG _ ORIG))])

)

;; HASHARRAY entries

(DEFINEQ

(HARRAY
  [LAMBDA (MINKEYS)                                                    (* rmk%: " 3-Jan-84 13:09")
    ;; For backward compatibility--produces a non-growing hasharray
    (HASHARRAY MINKEYS 'ERROR])

(HASHARRAY
  [LAMBDA (MINKEYS OVERFLOW HASHBITSFN EQUIVFN RECLAIMABLE REHASH-THRESHOLD)
                                                                       ; Edited 3-Oct-91 13:35 by jds
    ;; MINKEYS is the number of required slots;  actual number of slots is greater by the vacancy factor REHASH-THRESHOLD default 0.75 ---
    ;; MINKEYS is first adjusted by the vacancy factor, then bumped up to the next highest power of 2, so that hashkey can be computed with
    ;; LOGAND instead of IREMAINDER.
    [COND
       ((FIXP REHASH-THRESHOLD)                                        ; Scale it
        (SETQ REHASH-THRESHOLD (AND (FIXP OVERFLOW)
                                    (ILESSP REHASH-THRESHOLD OVERFLOW)
                                    (FQUOTIENT REHASH-THRESHOLD OVERFLOW]
    (LET ((PHYSLOTS (OR (bind [IDEALSIZE _ (IMAX MINKEYS (IMIN (- (FOLDLO \MaxArrayNCells CELLSPERSLOT)
                                                                  2)
                                                               (COND
                                                                  (REHASH-THRESHOLD (FIXR (FQUOTIENT
                                                                                           (SUB1 MINKEYS)
                                                                                           REHASH-THRESHOLD)))
                                                                  (T (LLSH (IQUOTIENT (SUB1 MINKEYS)
                                                                                      3)
                                                                           2]
                              find I from 8 to 16384 by I suchthat (IGREATERP I IDEALSIZE))
                        [for I from [IMAX MINKEYS (IMIN 32749 (- (FOLDLO \MaxArrayNCells CELLSPERSLOT)
                                                                 2)
                                                          (COND
                                                             (REHASH-THRESHOLD (FIXR (FQUOTIENT (SUB1 MINKEYS)
                                                                                                REHASH-THRESHOLD)))
                                                             (T (LLSH (IQUOTIENT (SUB1 MINKEYS)
                                                                                 3)
                                                                      2]
                           to 32749 suchthat
                                                                       ;; Find a prime table-size between our ideal and the maximum, which is 32749 (largest prime <
                                                                       ;; array limit)
                                                                       (for J from 2 to (FIXR (SQRT I)) never (ZEROP (IREMAINDER I J]
                                                                 32768))
          LOGSLOTS NCELLS)
         (SETQ NCELLS (UNFOLD PHYSLOTS CELLSPERSLOT))
         (COND
            ((IGREATERP NCELLS \MaxArrayNCells)
             (ERROR "HARRAY TOO LARGE" MINKEYS))
            (T [SETQ LOGSLOTS (COND
                                 (REHASH-THRESHOLD (FIXR (FTIMES REHASH-THRESHOLD PHYSLOTS)))
                                 (T (IPLUS (LRSH PHYSLOTS 1)
                                           (LRSH PHYSLOTS 2]     ; Number of logical slots is REHASH-THRESHOLD * number of
                                                                 ; physical slots
               (create HARRAYP
                       HARRAYPBASE _ (\ALLOCBLOCK NCELLS PTRBLOCK.GCT)
                       LASTINDEX _ (SUB1 PHYSLOTS)
                       RECLAIMABLE _ RECLAIMABLE
                       OVERFLOWACTION _ OVERFLOW
                       NUMSLOTS _ LOGSLOTS
                       NULLSLOTS _ LOGSLOTS
                       NUMKEYS _ 0
                       HASHBITSFN _ HASHBITSFN
                       EQUIVFN _ EQUIVFN])

(HARRAYP
  [LAMBDA (X)                                                          (* rmk%: "21-Dec-83 22:20")
    (AND (type? HARRAYP X)
         X])

(HARRAYPROP
  [LAMBDA NARGS                                                        (* bvm%: "21-Jan-86 11:02")
                                                                       ; Nospread so we can tell whether a new value was specified
    (PROG ((HARRAY (AND (IGREATERP NARGS 0)
```

```
                        (ARG NARGS 1)))
           (PROP (AND (IGREATERP NARGS 1)
                      (ARG NARGS 2)))
           (NEWVALP (IGREATERP NARGS 2))
           HA NEWVALUE)
          (SETQ HA (\DTEST HARRAY 'HARRAYP))                    ; Keep HARRAY explicitly so can tell LISTP case
          (AND NEWVALP (SETQ NEWVALUE (ARG NARGS 3)))
          [RETURN (SELECTQ PROP
                      (SIZE (AND NEWVALP (GO CANTUPDATE))
                            (HARRAYSIZE HA))
                      (OVERFLOW [COND
                                    [(LISTP HARRAY)             ; For compatibility with old code that would enlist the hasharray
                                     (PROG1 (CDR HARRAY)
                                         (AND NEWVALP (RPLACD HARRAY NEWVALUE)))]
                                    (T (PROG1 (fetch (HARRAYP OVERFLOWACTION) of HA)
                                              (AND NEWVALP (replace (HARRAYP OVERFLOWACTION) of HA with NEWVALUE)))])
                      (NUMKEYS (AND NEWVALP (GO CANTUPDATE))
                               (fetch (HARRAYP NUMKEYS) of HA))
                      (EQUIVFN (PROG1 (fetch (HARRAYP EQUIVFN) of HA)
                                      [AND NEWVALP (COND
                                                       ((NEQ (fetch (HARRAYP NUMKEYS) of HA)
                                                             0)     ; Absurd to change equivalence relation in midstream
                                                        (GO CANTUPDATE))
                                                       (T (replace (HARRAYP EQUIVFN) of HA with NEWVALUE]))
                      (RECLAIMABLE (PROG1 (fetch (HARRAYP RECLAIMABLE) of HA)
                                          (AND NEWVALP (replace (HARRAYP RECLAIMABLE) of HA with NEWVALUE))))
                      (HASHBITSFN (PROG1 (fetch (HARRAYP HASHBITSFN) of HA)
                                         [AND NEWVALP (COND
                                                          ((NEQ (fetch (HARRAYP NUMKEYS) of HA)
                                                                0)
                                                           (GO CANTUPDATE))
                                                          (T (replace (HARRAYP HASHBITSFN) of HA with NEWVALUE]))
                      (PROG1 (LISTGET (SETQ HARRAY (fetch (HARRAYP HASHUSERDATA) of HA))
                                      PROP)
                          [AND NEWVALP (COND
                                           ((NULL HARRAY)
                                            (replace (HARRAYP HASHUSERDATA) of HA with (LIST PROP NEWVALUE)))
                                           (T (LISTPUT HARRAY PROP NEWVALUE])]
      CANTUPDATE
          (ERROR "Can't update this hash array property" PROP])
```

(HARRAYSIZE

```
  [LAMBDA (HARRAY)                                              (* rmk%: "21-Dec-83 23:33")
    (fetch NUMSLOTS of (\DTEST HARRAY 'HARRAYP])
```

(CLRHASH

```
  [LAMBDA (HARRAY)                                              (* bvm%: "21-Jan-86 11:32")
    (PROG ((HA (\DTEST HARRAY 'HARRAYP))
           SLOT)
          (SETQ SLOT (fetch HARRAYPBASE of HA))
          (UNINTERRUPTABLY
              (bind [LASTSLOT _ (fetch (HASHSLOT NEXTSLOT) of (\HASHSLOT SLOT (fetch (HARRAYP LASTINDEX)
                                                                                      of HA]
                 do (replace (HASHSLOT KEY) of SLOT with NIL)
                    (replace (HASHSLOT VALUE) of SLOT with NIL)
                 repeatuntil (EQ (SETQ SLOT (fetch (HASHSLOT NEXTSLOT) of SLOT))
                                 LASTSLOT))
              (replace (HARRAYP NULLSLOTS) of HA with (fetch (HARRAYP NUMSLOTS) of HA))
              (replace (HARRAYP NUMKEYS) of HA with 0))
          (RETURN HARRAY])
```

(MAPHASH

```
  [LAMBDA (HARRAY MAPHFN)                                       (* bvm%: "21-Jan-86 11:28")
    (DECLARE (LOCALVARS . T))
    (LET ((HA (\DTEST HARRAY 'HARRAYP))
          SLOT)
         (SETQ SLOT (fetch HARRAYPBASE of HA))
         (bind V [LASTSLOT _ (fetch (HASHSLOT NEXTSLOT) of (\HASHSLOT SLOT (fetch (HARRAYP LASTINDEX) of HA]
               (NULLVALUE _ \HASH.NULL.VALUE) when (SETQ V (fetch (HASHSLOT VALUE) of SLOT))
           do (APPLY* MAPHFN (AND (NEQ V NULLVALUE)
                                  V)
                          (fetch (HASHSLOT KEY) of SLOT))
           repeatuntil (EQ (SETQ SLOT (fetch (HASHSLOT NEXTSLOT) of SLOT))
                           LASTSLOT)
           finally (RETURN HARRAY])
```

(GETHASH

```
  [LAMBDA (ITEM HARRAY DEFAULT RETURNMVS)                       ; Edited 26-Feb-91 13:07 by jds
```

;;; RETURNMVS, if true return multiple values, else don't.

```
    (PROG ((HA (\DTEST HARRAY 'HARRAYP))
           INDEX SLOT SKEY FIRSTINDEX REPROBE LIMIT BITS EQFN ABASE VALUE)
```

```
          [SETQ BITS (COND
                        ((SETQ BITS (fetch (HARRAYP HASHBITSFN) of HA))
                         (APPLY* BITS ITEM))
                        (T (\EQHASHINGBITS ITEM]
          (SETQ INDEX (\FIRSTINDEX BITS HA))                    ; Do first index outside of loop, so don't have to do setup on fast
                                                                ; case
          (SETQ ABASE (fetch HARRAYPBASE of HA))
          (SETQ SLOT (\HASHSLOT ABASE INDEX))
          [COND
             ((SETQ VALUE (fetch (HASHSLOT VALUE) of SLOT))     ; Slot is occupied
              (COND
                 ((OR (EQ ITEM (SETQ SKEY (fetch (HASHSLOT KEY) of SLOT)))
                      (AND (SETQ EQFN (fetch (HARRAYP EQUIVFN) of HA))
                           (APPLY* EQFN ITEM SKEY)))
                  (GO FOUND)))                                  ; else try again

              )
             [(NULL (fetch (HASHSLOT KEY) of SLOT))             ; Null slot
              (RETURN (COND
                         (RETURNMVS (CL:VALUES DEFAULT NIL))
                         (T DEFAULT]
             (T                                                 ; Deleted slot: null value, non-nil key
                (SETQ EQFN (fetch (HARRAYP EQUIVFN) of HA]      ; Perhaps we hit right on
          (SETQ FIRSTINDEX INDEX)
          (SETQ REPROBE (\REPROBE BITS HA))                     ; Compute reprobe interval
          (SETQ LIMIT (ADD1 (fetch (HARRAYP LASTINDEX) of HA)))
     LP   (SETQ INDEX (IREMAINDER (IPLUS INDEX REPROBE)
                                  LIMIT))
```

;; Since table size is a power of two, any wraparound in the IPLUS16 will be consistent with the LOGAND

```
          (COND
             ((EQ INDEX FIRSTINDEX)                             ; Should never happen, since we don't allow full occupancy
              (SHOULDNT "Hashing in full hash table")))
          (SETQ SLOT (\HASHSLOT ABASE INDEX))
          (SETQ SKEY (fetch (HASHSLOT KEY) of SLOT))
          [COND
             [(SETQ VALUE (fetch (HASHSLOT VALUE) of SLOT))     ; Slot is occupied
              (COND
                 ((OR (EQ (SETQ SKEY (fetch (HASHSLOT KEY) of SLOT))
                          ITEM)
                      (AND EQFN (APPLY* EQFN ITEM SKEY)))       ; Found it
                  (GO FOUND]
             ((NULL (fetch (HASHSLOT KEY) of SLOT))             ; Empty slot
              (RETURN (COND
                         (RETURNMVS (CL:VALUES DEFAULT NIL))
                         (T DEFAULT]
          (GO LP)
      FOUND
          (RETURN (COND
                     (RETURNMVS (CL:VALUES (AND (NEQ VALUE \HASH.NULL.VALUE)
                                                VALUE)
                                           T))
                     (T (AND (NEQ VALUE \HASH.NULL.VALUE)
                             VALUE])
```

(**PUTHASH**

```
  [LAMBDA (KEY VAL HARRAY)                                      (* raf "22-Aug-86 16:55")
```

;;; Store new value VAL, or remove old value if VAL = NIL

```
    (\HASHACCESS KEY VAL HARRAY (NULL VAL))
    VAL])
```

(**CL::PUTHASH**

```
  (CL:LAMBDA (KEY CL:HASH-TABLE VALUE &OPTIONAL (EXTRA NIL EXTRA-P))
                                                                ; Edited 23-Mar-87 12:00 by bvm:
```

;; SETF inverse for LISP:GETHASH. Subtlety is that LISP:GETHASH has an optional arg DEFAULT, so if you passed one of those
;; 3-argument forms to SETF, you'd get 4 arguments in this call. In this case, the fourth argument is the new value and you should ignore
;; the third.

```
        (CL:CHECK-TYPE CL:HASH-TABLE CL:HASH-TABLE)
        (\HASHACCESS KEY (CL:IF EXTRA-P
                                EXTRA
                                VALUE)
               CL:HASH-TABLE NIL)
      VALUE))
```

(**REMHASH**

```
  [LAMBDA (KEY HARRAY)                                          (* bvm%: "20-Jan-86 18:54")
    (\HASHACCESS KEY NIL HARRAY T])
```

(**HASHRECLAIM**

```
  [LAMBDA (HARRAY)                                              (* bvm%: "21-Jan-86 11:36")
```

;;; Remove from HARRAY any keys whose ref cnt is 1

```
    (PROG ((HA (\DTEST HARRAY 'HARRAYP))
           SLOT)
          (SETQ SLOT (fetch (HARRAYP HARRAYPBASE) of HA))
          (UNINTERRUPTABLY
              (bind KEY [LASTSLOT _ (fetch (HASHSLOT NEXTSLOT) of (\HASHSLOT SLOT (fetch (HARRAYP LASTINDEX)
                                                                                    of HA]
                    (NUMDELETED _ 0) when (AND (SETQ KEY (fetch (HASHSLOT KEY) of SLOT))
                                               (NEQ KEY T)
                                               (\EQREFCNT1 KEY))
                    do                                          ; Slot is occupied with key with ref cnt 1, so delete it
                       (replace (HASHSLOT KEY) of SLOT with T)
                       (replace (HASHSLOT VALUE) of SLOT with NIL)
                       (add NUMDELETED 1)
                    repeatuntil (EQ LASTSLOT (SETQ SLOT (fetch (HASHSLOT NEXTSLOT) of SLOT)))
                    finally (replace (HARRAYP NUMKEYS) of HA with (IDIFFERENCE (fetch (HARRAYP NUMKEYS) of HA)
                                                                               NUMDELETED))))
          (RETURN HARRAY])
```

(**HASHACCESS**

```
  [LAMBDA (ITEM VAL HARRAY REMOVE)                              ; Edited 26-Feb-91 13:16 by jds
```

;;; Add or remove something from hash array HARRAY -- REMOVE = T means remove the item, which is necessarily distinct from adding a VAL = NIL

```
    (PROG ((HA (\DTEST HARRAY 'HARRAYP))
           DELSLOT INDEX SLOT SKEY FIRSTINDEX REPROBE LIMIT BITS HASHBITSFN EQFN ABASE)
          [SETQ BITS (COND
                        ((SETQ HASHBITSFN (fetch (HARRAYP HASHBITSFN) of HA))
                         (APPLY* HASHBITSFN ITEM))
                        (T (\EQHASHINGBITS ITEM]
      PHTOP
          (SETQ INDEX (\FIRSTINDEX BITS HA))                    ; Handle first probe outside loop in case it wins
          (SETQ ABASE (fetch HARRAYPBASE of HA))
          (SETQ SLOT (\HASHSLOT ABASE INDEX))
          [COND
             ((fetch (HASHSLOT VALUE) of SLOT)                  ; Slot is occupied
              (COND
                 ((OR (EQ ITEM (SETQ SKEY (fetch (HASHSLOT KEY) of SLOT)))
                      (AND (SETQ EQFN (fetch (HARRAYP EQUIVFN) of HA))
                           (APPLY* EQFN ITEM SKEY)))
                  (GO FOUND)))                                  ; else try again

              )
             ((NULL (fetch (HASHSLOT KEY) of SLOT))             ; Null slot
              (GO ADDNEWENTRY))
             (T                                                 ; Deleted slot: null value, non-nil key
                (SETQ DELSLOT SLOT)
                (SETQ EQFN (fetch (HARRAYP EQUIVFN) of HA]
          (SETQ FIRSTINDEX INDEX)
          (SETQ REPROBE (\REPROBE BITS HA))
          (SETQ LIMIT (ADD1 (fetch (HARRAYP LASTINDEX) of HA)))
      LP  (SETQ INDEX (IREMAINDER (IPLUS INDEX REPROBE)
                                  LIMIT))
          (COND
             ((EQ INDEX FIRSTINDEX)
              ;; We don't allow full occupancy, so if we get to the beginning without finding an empty slot, we must have found a deleted one
              (SETQ SLOT (OR DELSLOT (ERROR "No vacant slot in hasharray")))
              (GO ADDNEWENTRY)))
          (SETQ SLOT (\HASHSLOT ABASE INDEX))
          [COND
             [(fetch (HASHSLOT VALUE) of SLOT)                  ; Slot is occupied
              (COND
                 ((OR (EQ (SETQ SKEY (fetch (HASHSLOT KEY) of SLOT))
                          ITEM)
                      (AND EQFN (APPLY* EQFN ITEM SKEY)))       ; Found it
                  (GO FOUND]
             (T (COND
                   ((NULL (fetch (HASHSLOT KEY) of SLOT))
                    ;; NIL as both key and value means empty slot.  New entry goes here, unless there was an earlier deleted slot
                    (AND DELSLOT (SETQ SLOT DELSLOT))
                    (GO ADDNEWENTRY))
                   ((NULL DELSLOT)                              ; Key non-NIL but value NIL means deleted.
                    (SETQ DELSLOT SLOT]
          (GO LP)
      FOUND
          (UNINTERRUPTABLY
              [COND
                 (REMOVE                                        ; Deleted slots are noted by value = NIL and key non-NIL
                        (replace (HASHSLOT KEY) of SLOT with T)
                        (replace (HASHSLOT VALUE) of SLOT with NIL)
                        (add (fetch (HARRAYP NUMKEYS) of HA)
                             -1))
```

```
                (T ;; If writing value NIL must write distinguished non-NIL value.  Ultimately, this should be a non-interned symbol, so that
                   ;; nobody could mistakenly type it (!) but it still wouldn't be ref counted (in present world)
                   (replace (HASHSLOT VALUE) of SLOT with (OR VAL \HASH.NULL.VALUE])
          (RETURN T)
      ADDNEWENTRY                                                    ; Didn't find this item in table.  If REMOVE is T, nothing to do.
          (COND
             (REMOVE (RETURN NIL)))
          (COND
             ((EQ 0 (fetch (HARRAYP NULLSLOTS) of HA))
              (COND
                 ((fetch (HARRAYP RECLAIMABLE) of HA)                ; Before rehashing, remove anything with ref cnt 1
                  (\HASHRECLAIM HA)))
              (SETQ HARRAY (HASHOVERFLOW (OR HARRAY SYSHASHARRAY)))
              (SETQ HA (\DTEST HARRAY 'HARRAYP))
              ;; ERRORX2 doesn't handle SYSHASHARRAY specially;  on 10, SYSHASHARRAY is rehashed directly in PUTHASH, without going
              ;; through ERRORX2 and independent of the normal LISTP conventions.
              (SETQ DELSLOT NIL)                                     ; Non-NIL DELSLOT is a pointer into the old array
              (GO PHTOP)))
          (UNINTERRUPTABLY
              (OR (EQ SLOT DELSLOT)
                  (add (fetch (HARRAYP NULLSLOTS) of HA)
                       -1))
              (add (fetch (HARRAYP NUMKEYS) of HA)
                   1)
              (replace (HASHSLOT KEY) of SLOT with ITEM)
              (replace (HASHSLOT VALUE) of SLOT with (OR VAL \HASH.NULL.VALUE)))
          (RETURN VAL])
```

(REHASH

```
  [LAMBDA (OLDAR NEWAR)                                              (* rmk%: "26-Dec-83 11:50")
    (CLRHASH NEWAR)
    (PROG [SLOT LASTSLOT V (APTR1 (\DTEST OLDAR 'HARRAYP]            ; This is maphash expanded out
          (SETQ SLOT (fetch HARRAYPBASE of APTR1))
          (SETQ LASTSLOT (\ADDBASE4 SLOT (fetch (HARRAYP LASTINDEX) of APTR1)))
      LP  (COND
             ((SETQ V (fetch (HASHSLOT VALUE) of SLOT))
              (PUTHASH (fetch (HASHSLOT KEY) of SLOT)
                       V NEWAR)))
          (COND
             ((EQ SLOT LASTSLOT)
              (RETURN NEWAR)))
          (SETQ SLOT (fetch (HASHSLOT NEXTSLOT) of SLOT))
          (GO LP])
```

(\COPYHARRAYP

```
  [LAMBDA (SOURCE TARGET)                                            (* rmk%: "31-Dec-83 13:58")
    ;; Copies all properties of SOURCE into TARGET;  called from HASHOVERFLOW
    (replace NULLSLOTS of TARGET with (fetch NULLSLOTS of SOURCE))
    (replace LASTINDEX of TARGET with (fetch LASTINDEX of SOURCE))
    (replace HARRAYPBASE of TARGET with (fetch HARRAYPBASE of SOURCE))
    (replace OVERFLOWACTION of TARGET with (fetch OVERFLOWACTION of SOURCE))
    (replace NUMSLOTS of TARGET with (fetch NUMSLOTS of SOURCE))
    (replace NUMKEYS of TARGET with (fetch NUMKEYS of SOURCE])
```

(\HASHTABLE.DEFPRINT

```
  [LAMBDA (CL:HASH-TABLE STREAM)                                     ; Edited 23-Mar-87 11:38 by bvm:
    ;; For benefit of common lisp, print harrayp by name "hash table", for example, #<Hash-Table @ 76,5432>
    [.SPACECHECK. STREAM (CONSTANT (+ (NCHARS "<Hash-Table @ >")
                                      (PROGN                         ; Longest address is '177,177777'
                                             10]
    (\OUTCHAR STREAM (fetch (READTABLEP HASHMACROCHAR) of *READTABLE*))
    (\SOUT "<Hash-Table @ " STREAM)
    (\PRINTADDR CL:HASH-TABLE STREAM)
    (\OUTCHAR STREAM (CHARCODE >))                                   ; Return T to say we printed it ourselves
    T])

)

(DEFINEQ
```

(STRINGHASHBITS

```
  [LAMBDA (STRING)                                                   ; Edited  2-Mar-89 14:11 by jds
    (MISCN STRINGHASHBITS STRING])
```

(STRING-EQUAL-HASHBITS

```
  [LAMBDA (STRING)                                                   ; Edited  2-Mar-89 14:14 by jds
```

;;; A hashbits function for the hash equivalence STRING-EQUAL.

```
;;; This is similar to the atom hash algorithm, but we OR in 40Q to cause uppercase and lowercase chars to have the same codes.

    (MISCN STRING-EQUAL-HASHBITS STRING])

)

(DEFINEQ

(\STRINGHASHBITS-UFN
  [LAMBDA (INDEX ARGCOUNT ARG-PTR)                          ; Edited  2-Mar-89 14:06 by jds
    ;; UFN for the STRINGHASHBITS MISCN opcode.  Computes a hash index for strings and symbols, so identical string CONTENTS hash to the
    ;; same place.
    (LET ((STRING (\GETBASEPTR ARG-PTR 0)))
         (for C inpname STRING bind (HASHBITS _ 0) do           ; This is similar to the atom hash algorithm
                                                    [SETQ HASHBITS
                                                     (IPLUS16 C (IPLUS16 (SETQ HASHBITS
                                                                          (IPLUS16 HASHBITS
                                                                                 (LLSH (LOGAND HASHBITS 4095)
                                                                                       2)))
                                                                        (LLSH (LOGAND HASHBITS 255)
                                                                              8]
           finally (RETURN HASHBITS])

(\STRING-EQUAL-HASHBITS-UFN
  [LAMBDA (INDEX ARGCOUNT ARG-PTR)                          ; Edited  2-Mar-89 14:09 by jds

;;; A hashbits function for the hash equivalence STRING-EQUAL.

;;; This is similar to the atom hash algorithm, but we OR in 40Q to cause uppercase and lowercase chars to have the same codes.

    (LET ((STRING (\GETBASEPTR ARG-PTR 0)))
         (for C inpname STRING bind (HASHBITS _ 0) do [SETQ HASHBITS
                                                       (IPLUS16 (LOGOR C 32)
                                                              (IPLUS16 (SETQ HASHBITS
                                                                        (IPLUS16 HASHBITS (LLSH (LOGAND HASHBITS
                                                                                                        4095)
                                                                                                2)))
                                                                       (LLSH (LOGAND HASHBITS 255)
                                                                             8]
           finally (RETURN HASHBITS])

)

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(DATATYPE HARRAYP ((NULLSLOTS WORD)                         ; Number of NIL-NIL slots, which break chains
                   (LASTINDEX WORD)                         ; Slot offset of last slot.  Used in probe computations
                                                            ; computations.  Microcode support for \ADDBASE4 would help
                   (HARRAYPBASE POINTER)
                   (RECLAIMABLE FLAG)                       ; True if keys can go away when no other refs
                   (OVERFLOWACTION POINTER)
                   (NUMSLOTS WORD)                          ; The maximum number of logical slots--returned by
                                                            ; HARRAYSIZE
                   (NUMKEYS WORD)                           ; The number of distinct keys in the array
                   (HASHBITSFN POINTER)
                   (EQUIVFN POINTER)
                   (HASHUSERDATA POINTER)))
)

(/DECLAREDATATYPE 'HARRAYP '(WORD WORD POINTER FLAG POINTER WORD WORD POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '14)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \EQHASHINGBITS MACRO [OPENLAMBDA (X)              ; Spread out objects whose low bits are in small arithmetic
                                                            ; progression, esp atoms
                                (LOGXOR (\HILOC X)
                                       (LOGXOR (LLSH (LOGAND (\LOLOC X)
                                                             8191)
                                                     3)
                                              (LRSH (\LOLOC X)
                                                    9])
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE
```

```
[BLOCKRECORD HASHSLOT ((KEY POINTER)
                       (VALUE POINTER))
       (ACCESSFNS ((NEXTSLOT (\ADDBASE DATUM (UNFOLD WORDSPERCELL CELLSPERSLOT]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FIRSTINDEX MACRO [(BITS APTR1)
                             (IREMAINDER BITS (ADD1 (fetch (HARRAYP LASTINDEX) of APTR1])

(PUTPROPS \HASHSLOT MACRO (= . \ADDBASE4))

(PUTPROPS \REPROBE MACRO ((BITS HA)
                          (LOGOR [IREMAINDER (LOGXOR BITS (LRSH BITS 8))
                                             (IMIN 64 (ADD1 (fetch (HARRAYP LASTINDEX) of HA]
                                 1)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ CELLSPERSLOT 2)

(CONSTANTS (CELLSPERSLOT 2))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \HASH.NULL.VALUE SYSHASHARRAY)
)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(DEFPRINT 'HARRAYP '\HASHTABLE.DEFPRINT)
)

(/DECLAREDATATYPE 'HARRAYP '(WORD WORD POINTER FLAG POINTER WORD WORD POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '14)

(ADDTOVAR SYSTEMRECLST
          (DATATYPE HARRAYP ((NULLSLOTS WORD)
                             (LASTINDEX WORD)
                             (HARRAYPBASE POINTER)
                             (RECLAIMABLE FLAG)
                             (OVERFLOWACTION POINTER)
                             (NUMSLOTS WORD)
                             (NUMKEYS WORD)
                             (HASHBITSFN POINTER)
                             (EQUIVFN POINTER)
                             (HASHUSERDATA POINTER))))

(RPAQQ \HASH.NULL.VALUE \Hash\Null\Value\)

;; System entries for CODE

(DEFINEQ

(\CODEARRAY
  [LAMBDA (NBYTES INITONPAGE)                                   (* lmm "15-Aug-84 11:51")
    (PROG NIL
      ;; NBYTES is the number of bytes required, INITONPAGE is the number of CELLS which must reside on same page
          (COND
             ((OR (IGREATERP 0 NBYTES)
                  (IGREATERP NBYTES 65535))
              (LISPERROR "ILLEGAL ARG" NBYTES)))                ; dolphin requires code blocks aligned quadword
          (RETURN (create ARRAYP
                          TYP _ \ST.CODE
                          BASE _ (\ALLOCBLOCK (FOLDHI NBYTES BYTESPERCELL)
                                              CODEBLOCK.GCT INITONPAGE CELLSPERQUAD)
                          LENGTH _ NBYTES
                          ORIG _ 0])

(\FIXCODENUM
  [LAMBDA (CA BN NUM MASK)                                      ; Edited  7-Jan-91 13:29 by jds
    (DECLARE (IGNORE MASK))                                     ; MASK is used by the renamed version of this function.
    ;; Do fixup for a 2-byte number in the code stream.  Used for type numbers only, for now.
    (PROG ((BASE (fetch (ARRAYP BASE) of CA)))
          (\PUTBASEBYTE BASE BN (LOGAND 255 NUM))
          (\PUTBASEBYTE BASE (SETQ BN (SUB1 BN))
                 (LOGOR (\GETBASEBYTE BASE BN)
                        (LRSH NUM 8)))
```

```
          (RETURN NUM])
```

(\FIXCODEPTR

```
  [LAMBDA (CA BN PTR MASK)                                ; Edited 12-Nov-92 17:03 by sybalsky:mv:envos
    (DECLARE (IGNORE MASK))                               ; MASK is used by the renamed version of this function.
    (PROG ((BASE (fetch (ARRAYP BASE) of CA))
           (LO (\LOLOC PTR)))
          (UNINTERRUPTABLY
              (\ADDREF PTR)
              (\PUTBASEBYTE BASE BN (LOGAND LO 255))
              (\PUTBASEBYTE BASE (SUB1 BN)
                     (LRSH LO 8))
              (\PUTBASEBYTE BASE (IDIFFERENCE BN 2)
                     (LOGOR (\GETBASEBYTE BASE (IDIFFERENCE BN 2))
                            (LOGAND (\HILOC PTR)
                                    255)))
              (\PUTBASEBYTE BASE (IDIFFERENCE BN 3)
                     (LOGOR (\GETBASEBYTE BASE (IDIFFERENCE BN 3))
                            (LRSH (\HILOC PTR)
                                  8))))
          (RETURN PTR])
```

(\FIXCODESYM

```
  [LAMBDA (CA BN SYM MASK)                                ; Edited 13-Nov-92 04:56 by sybalsky:mv:envos
    (DECLARE (IGNORE MASK))                               ; MASK is used by the renamed version of this function.
    ;; Perform fix-up for a symbol in an IL-Compiled function -- either 2 or 4 bytes, depending on the architecture.
    ;; CA -- the code array
    ;; BN -- byte number of the low-order byte to be fixed up
    ;; SYM -- the symbol, expressed as a FIXP or a NEW-ATOM.
    (NEW-SYMBOL-CODE (PROG (HIBYTE NUM (BASE (fetch (ARRAYP BASE) of CA)))
                           ;; For 3-byte-symbol machines, handle 3 bytes worth of atom number.
                           [COND
                              ((SMALLP SYM)
                               (SETQ NUM SYM)
                               (SETQ HIBYTE 0))
                              ((FIXP SYM)
                               (SETQ NUM (LOGAND SYM 65535))
                               (SETQ HIBYTE (LRSH SYM 16)))
                              (T (SETQ NUM (\LOLOC SYM))
                                 (SETQ HIBYTE (\HILOC SYM]
                           (UNINTERRUPTABLY
                               (\PUTBASEBYTE BASE BN (LOGAND NUM 255))
                               (\PUTBASEBYTE BASE (SUB1 BN)
                                      (LOGAND (LRSH NUM 8)
                                              255))
                               (\PUTBASEBYTE BASE (IDIFFERENCE BN 2)
                                      (LOGOR (\GETBASEBYTE BASE (IDIFFERENCE BN 2))
                                             (LOGAND HIBYTE 255)))
                               (\PUTBASEBYTE BASE (IDIFFERENCE BN 3)
                                      (LOGOR (\GETBASEBYTE BASE (IDIFFERENCE BN 3))
                                             (LRSH HIBYTE 8))))
                           (RETURN (+ (LLSH HIBYTE 16)
                                      NUM)))
            (PROG ((NUM (\LOLOC SYM))
                   (BASE (fetch (ARRAYP BASE) of CA)))
                  ;; 2-BYTE case:  Just fill it in.
                  (\PUTBASEBYTE BASE BN (LOGAND 255 NUM))
                  (\PUTBASEBYTE BASE (SETQ BN (SUB1 BN))
                         (LOGOR (\GETBASEBYTE BASE BN)
                                (LRSH NUM 8)))
                  (RETURN NUM])

)

;; Internal

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS EQPTR DMACRO (= . EQ))

(PUTPROPS BUCKETINDEX MACRO ((N)
                             (IMIN (INTEGERLENGTH N)
                                   \MAXBUCKETINDEX)))

(PUTPROPS FREEBLOCKCHAIN.N MACRO ((N)
                                  (\ADDBASE2 \FREEBLOCKBUCKETS (BUCKETINDEX N))))
)
```

```
(DECLARE%: EVAL@COMPILE

(RPAQQ \MAXBUCKETINDEX 30)

(CONSTANTS \MAXBUCKETINDEX)
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \ADDBASE2 MACRO (OPENLAMBDA (BASE N)
                            (\ADDBASE (\ADDBASE BASE N)
                                   N)))

(PUTPROPS \ADDBASE4 MACRO (OPENLAMBDA (BASE N)
                            (\ADDBASE2 (\ADDBASE2 BASE N)
                                   N)))

(PUTPROPS HUNKSIZEFROMNUMBER MACRO ((NTYPX)
                                    (FOLDLO (fetch DTDSIZE of (\GETDTD NTYPX))
                                           WORDSPERCELL)))

(PUTPROPS \BYTELT DMACRO (OPENLAMBDA (A J)
                           (\GETBASEBYTE (fetch (ARRAYP BASE) of A)
                                  (IPLUS (fetch (ARRAYP OFFST) of A)
                                         J))))

(PUTPROPS \BYTESETA DMACRO (OPENLAMBDA (A J V)
                            (\PUTBASEBYTE (fetch (ARRAYP BASE) of A)
                                   (IPLUS (fetch (ARRAYP OFFST) of A)
                                          J)
                                   V)))

(PUTPROPS \WORDELT DMACRO (OPENLAMBDA (A J)
                           [CHECK (AND (ARRAYP A)
                                       (EQ 0 (fetch (ARRAYP ORIG) of A))
                                       (EQ \ST.POS16 (fetch (ARRAYP TYP) of A]
                           (CHECK (IGREATERP (fetch (ARRAYP LENGTH) of A)
                                             J))
                           (\GETBASE (fetch (ARRAYP BASE) of A)
                                  (IPLUS (fetch (ARRAYP OFFST) of A)
                                         J))))
)

(RPAQQ BLOCKGCTYPECONSTANTS ((CODEBLOCK.GCT 2)
                             (PTRBLOCK.GCT 1)
                             (UNBOXEDBLOCK.GCT 0)))

(DECLARE%: EVAL@COMPILE

(RPAQQ CODEBLOCK.GCT 2)

(RPAQQ PTRBLOCK.GCT 1)

(RPAQQ UNBOXEDBLOCK.GCT 0)

(CONSTANTS (CODEBLOCK.GCT 2)
       (PTRBLOCK.GCT 1)
       (UNBOXEDBLOCK.GCT 0))
)

(RPAQQ ARRAYCONSTANTS
       (\ArrayBlockHeaderCells \ArrayBlockHeaderWords \ArrayBlockTrailerCells \ArrayBlockTrailerWords
              (\ArrayBlockOverheadCells (IPLUS \ArrayBlockHeaderCells \ArrayBlockTrailerCells))
              (\ArrayBlockOverheadWords (IPLUS \ArrayBlockHeaderWords \ArrayBlockTrailerWords))
              \ArrayBlockLinkingCells
              (\MinArrayBlockSize (IPLUS \ArrayBlockOverheadCells \ArrayBlockLinkingCells))
              (\MaxArrayBlockSize 65535)
              (\MaxArrayNCells (IDIFFERENCE \MaxArrayBlockSize \ArrayBlockOverheadCells))
              \MaxArrayLen
              (\ABPASSWORDSHIFT 3)
              (\ArrayBlockPassword (LRSH 43690 \ABPASSWORDSHIFT))
              (\FreeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                         (LLSH UNBOXEDBLOCK.GCT 1)))
              (\UsedArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                         1))
              (\CodeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                         (LLSH CODEBLOCK.GCT 1)
                                         1))))

(DECLARE%: EVAL@COMPILE

(RPAQQ \ArrayBlockHeaderCells 1)

(RPAQQ \ArrayBlockHeaderWords 2)
```

```
(RPAQQ \ArrayBlockTrailerCells 1)

(RPAQQ \ArrayBlockTrailerWords 2)

(RPAQ \ArrayBlockOverheadCells (IPLUS \ArrayBlockHeaderCells \ArrayBlockTrailerCells))

(RPAQ \ArrayBlockOverheadWords (IPLUS \ArrayBlockHeaderWords \ArrayBlockTrailerWords))

(RPAQQ \ArrayBlockLinkingCells 2)

(RPAQ \MinArrayBlockSize (IPLUS \ArrayBlockOverheadCells \ArrayBlockLinkingCells))

(RPAQQ \MaxArrayBlockSize 65535)

(RPAQ \MaxArrayNCells (IDIFFERENCE \MaxArrayBlockSize \ArrayBlockOverheadCells))

(RPAQQ \MaxArrayLen 65535)

(RPAQQ \ABPASSWORDSHIFT 3)

(RPAQ \ArrayBlockPassword (LRSH 43690 \ABPASSWORDSHIFT))

(RPAQ \FreeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                (LLSH UNBOXEDBLOCK.GCT 1)))

(RPAQ \UsedArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                1))

(RPAQ \CodeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                (LLSH CODEBLOCK.GCT 1)
                                1))

(CONSTANTS \ArrayBlockHeaderCells \ArrayBlockHeaderWords \ArrayBlockTrailerCells \ArrayBlockTrailerWords
       (\ArrayBlockOverheadCells (IPLUS \ArrayBlockHeaderCells \ArrayBlockTrailerCells))
       (\ArrayBlockOverheadWords (IPLUS \ArrayBlockHeaderWords \ArrayBlockTrailerWords))
       \ArrayBlockLinkingCells
       (\MinArrayBlockSize (IPLUS \ArrayBlockOverheadCells \ArrayBlockLinkingCells))
       (\MaxArrayBlockSize 65535)
       (\MaxArrayNCells (IDIFFERENCE \MaxArrayBlockSize \ArrayBlockOverheadCells))
       \MaxArrayLen
       (\ABPASSWORDSHIFT 3)
       (\ArrayBlockPassword (LRSH 43690 \ABPASSWORDSHIFT))
       (\FreeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                  (LLSH UNBOXEDBLOCK.GCT 1)))
       (\UsedArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                  1))
       (\CodeArrayFlagWord (LOGOR (LLSH \ArrayBlockPassword \ABPASSWORDSHIFT)
                                  (LLSH CODEBLOCK.GCT 1)
                                  1)))
)

(RPAQQ ARRAYTYPES ((\ST.BYTE 0)
                   (\ST.POS16 1)
                   (\ST.INT32 2)
                   (\ST.CODE 4)
                   (\ST.PTR 6)
                   (\ST.FLOAT 7)
                   (\ST.BIT 8)
                   (\ST.PTR2 11)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \ST.BYTE 0)

(RPAQQ \ST.POS16 1)

(RPAQQ \ST.INT32 2)

(RPAQQ \ST.CODE 4)

(RPAQQ \ST.PTR 6)

(RPAQQ \ST.FLOAT 7)

(RPAQQ \ST.BIT 8)

(RPAQQ \ST.PTR2 11)

(CONSTANTS (\ST.BYTE 0)
       (\ST.POS16 1)
       (\ST.INT32 2)
       (\ST.CODE 4)
       (\ST.PTR 6)
       (\ST.FLOAT 7)
       (\ST.BIT 8)
       (\ST.PTR2 11))
)
```

```
(DECLARE%: EVAL@COMPILE

(RPAQQ \MAX.CELLSPERHUNK 64)

(CONSTANTS \MAX.CELLSPERHUNK)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \IN.MAKEINIT NIL)

(CONSTANTS (\IN.MAKEINIT))
)

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD SEQUENCEDESCRIPTOR ((ORIG BITS 1)
                                 (NIL BITS 1)
                                 (READONLY FLAG)
                                 (NIL BITS 1)
                                 (BASE POINTER)
                                 (TYP BITS 4)
                                 (NIL BITS 4)
                                 (LENGTH BITS 24)
                                 (OFFST FIXP)))

(DATATYPE ARRAYP ((ORIG BITS 1)
                  (NIL BITS 1)
                  (READONLY FLAG)                          ; probably no READONLY arrays now
                  (NIL BITS 1)
                  (BASE POINTER)
                  (TYP BITS 4)
                  (NIL BITS 4)
                  (LENGTH BITS 24)
                  (OFFST FIXP))
       ;; note that while ARRAYP is a DATATYPE, the allocation of it actually happens at MAKEINIT time under INITDATATYPE{NAMES}

       )

[BLOCKRECORD ARRAYBLOCK ((PASSWORD BITS 13)
                         (GCTYPE BITS 2)                   ; Unboxed, Pointers, or Code
                         (INUSE FLAG)
                         (ARLEN WORD)
                         (FWD FULLXPOINTER)                ; Only when on free list
                         (BKWD FULLXPOINTER))
       (BLOCKRECORD ARRAYBLOCK ((ABFLAGS WORD)             ; Used for header and trailer

                                ))
       [ACCESSFNS ARRAYBLOCK ((DAT (\ADDBASE DATUM \ArrayBlockHeaderWords))
                              (TRAILER (\ADDBASE2 DATUM (IDIFFERENCE (fetch (ARRAYBLOCK ARLEN) of DATUM)
                                                                     \ArrayBlockTrailerCells]
       (TYPE? (AND (EQ 0 (NTYPX DATUM))
                   (IGEQ (\HILOC DATUM)
                         \FirstArraySegment]
)

(/DECLAREDATATYPE 'ARRAYP '((BITS 1)
                            (BITS 1)
                            FLAG
                            (BITS 1)
                            POINTER
                            (BITS 4)
                            (BITS 4)
                            (BITS 24)
                            FIXP)
       ;; ---field descriptor list elided by lister---
       '6)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \NxtArrayPage \FREEBLOCKBUCKETS \HUNKING?)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \ArrayFrLst \ArrayFrLst2 \RECLAIM.COUNTDOWN)
)
)

(DEFINEQ

(\ALLOCBLOCK
```

```
[LAMBDA (NCELLS GCTYPE INITONPAGE ALIGN)                    (* bvm%: " 7-Feb-85 15:30")
```

;; NCELLS is number of cells wanted not counting overhead cell. For code arrays, INITONPAGE is number of cells to be kept on a single page. It ;; might be necessary to flag a block with an aligned indicator, to help a compacting garbage collector preserve the align proprty. --- Does not ;; assume that caller is uninterruptable --- Returns NIL if NCELLS = 0 --- GCTYPE is one of the constants PTRBLOCK.GCT, CODEBLOCK.GCT, ;; UNBOXEDBLOCK.GCT, indicating any special behavior to be performed when the block is reclaimed. NIL defaults to UNBOXEDBLOCK.GCT

```
(DECLARE (GLOBALVARS \ArrayFrLst))
(COND
   ((ILESSP NCELLS \ArrayBlockLinkingCells)
    (COND
       ((ILESSP NCELLS 0)
        (\ILLEGAL.ARG NCELLS)))
    (SETQ NCELLS \ArrayBlockLinkingCells))
   ((IGREATERP NCELLS \MaxArrayNCells)
    (\LISPERROR NCELLS "ARRAY STORAGE BLOCK TOO LARGE")))    ; NCELLS is number of data cells;  remember for allocation
                                                             ; counter below
(SELECTQ GCTYPE
     (NIL (SETQ GCTYPE UNBOXEDBLOCK.GCT))
     (T (SETQ GCTYPE PTRBLOCK.GCT))
     NIL)                                                    ; This SELECTQ can be removed when all callers are upgraded
                                                             ; to constants
(COND
   ((AND INITONPAGE (OR (ILESSP INITONPAGE 0)
                        (IGREATERP INITONPAGE CELLSPERPAGE)))
    (\ILLEGAL.ARG INITONPAGE)))
(COND
   ((NULL ALIGN))
   ((OR (ILESSP ALIGN 0)
        (IGREATERP ALIGN CELLSPERPAGE))
    (\ILLEGAL.ARG ALIGN))
   ((ILEQ ALIGN 1)
    (SETQ ALIGN))
   ((AND INITONPAGE (PROGN                                   ; Some check for consistency between ALIGN and INITONPAGE
                                                             ; is needed here
                            NIL))
    (ERROR "INITONPAGE and ALIGN too high")))
(OR (AND \HUNKING? (ILEQ NCELLS \MAX.CELLSPERHUNK)
         (\ALLOCHUNK NCELLS GCTYPE INITONPAGE ALIGN))
    (PROG ((ARLEN (IPLUS NCELLS \ArrayBlockOverheadCells))
           ABLOCK)
      RETRY
          (UNINTERRUPTABLY                                   ; Comment PPLossage
              (SETQ ABLOCK (OR (\ALLOCBLOCK.OLD ARLEN GCTYPE INITONPAGE ALIGN)
                               (\ALLOCBLOCK.NEW ARLEN GCTYPE INITONPAGE ALIGN)
                               (PROGN (FRPTQ 10 (RECLAIM))
                                      ;; We're probably out of array space;  our last chance is to collect and hope something shows
                                      ;; up on the free list.
                                      (\ALLOCBLOCK.OLD ARLEN GCTYPE INITONPAGE ALIGN))
                               (GO FULL)))                   ; ABLOCK now points to the beginning of the actual block of
                                                             ; storage to be used
              (replace (ARRAYBLOCK INUSE) of ABLOCK with T)
              (replace (ARRAYBLOCK INUSE) of (fetch (ARRAYBLOCK TRAILER) of ABLOCK) with T)
              (replace (ARRAYBLOCK GCTYPE) of ABLOCK with GCTYPE)
              (\CHECKARRAYBLOCK ABLOCK NIL)
              (.INCREMENT.ALLOCATION.COUNT. NCELLS)          ; NCELLS because CREATEREF accounts for overhead cell
              (SETQ ABLOCK (\ADDBASE ABLOCK \ArrayBlockHeaderWords))
              (\CREATEREF ABLOCK)
              (RETURN ABLOCK))
      FULL
          (LISPERROR "ARRAYS FULL" NIL T)                    ; User might release something, so retry.
          (GO RETRY])
```

(\MAIKO.ALLOCBLOCK

```
[LAMBDA (NCELLS GCTYPE INITONPAGE ALIGN)                    ; Edited 29-Jun-90 12:17 by ON
```

;; Maiko specific \ALLOCBLOCK. Does not decrement \RECLAIM.COUNTDOWN.

;; NCELLS is number of cells wanted not counting overhead cell. For code arrays, INITONPAGE is number of cells to be kept on a single page. It ;; might be necessary to flag a block with an aligned indicator, to help a compacting garbage collector preserve the align proprty. --- Does not ;; assume that caller is uninterruptable --- Returns NIL if NCELLS = 0 --- GCTYPE is one of the constants PTRBLOCK.GCT, CODEBLOCK.GCT, ;; UNBOXEDBLOCK.GCT, indicating any special behavior to be performed when the block is reclaimed. NIL defaults to UNBOXEDBLOCK.GCT

```
(DECLARE (GLOBALVARS \ArrayFrLst))
(COND
   ((ILESSP NCELLS \ArrayBlockLinkingCells)
    (COND
       ((ILESSP NCELLS 0)
        (\ILLEGAL.ARG NCELLS)))
    (SETQ NCELLS \ArrayBlockLinkingCells))
   ((IGREATERP NCELLS \MaxArrayNCells)
    (\LISPERROR NCELLS "ARRAY STORAGE BLOCK TOO LARGE")))    ; NCELLS is number of data cells;  remember for allocation
                                                             ; counter below
(SELECTQ GCTYPE
     (NIL (SETQ GCTYPE UNBOXEDBLOCK.GCT))
     (T (SETQ GCTYPE PTRBLOCK.GCT))
     NIL)                                                    ; This SELECTQ can be removed when all callers are upgraded
                                                             ; to constants
```

```
;; Maiko doesn't have to worry about INITONPAGE.  ----- '90/06/29 on.
;; (COND ((AND INITONPAGE (OR (ILESSP INITONPAGE 0) (IGREATERP INITONPAGE CELLSPERPAGE))) (\ILLEGAL.ARG INITONPAGE)))
(COND
   ((NULL ALIGN))
   ((OR (ILESSP ALIGN 0)
        (IGREATERP ALIGN CELLSPERPAGE))
    (\ILLEGAL.ARG ALIGN))
   ((ILEQ ALIGN 1)
    (SETQ ALIGN))
   ((AND INITONPAGE (PROGN                                  ; Some check for consistency between ALIGN and INITONPAGE
                                                            ; is needed here
                          NIL))
    (ERROR "INITONPAGE and ALIGN too high")))
(OR (AND \HUNKING? (ILEQ NCELLS \MAX.CELLSPERHUNK)          ; Maiko doesn't have to worry about INITONPAGE so call
                                                            ; \ALLOCHUNK with arg INITONPAGE as NIL.
         (\ALLOCHUNK NCELLS GCTYPE NIL ALIGN))
    (PROG ((ARLEN (IPLUS NCELLS \ArrayBlockOverheadCells))
           ABLOCK)
      RETRY
          (UNINTERRUPTABLY                                  ; Comment PPLossage
              (SETQ ABLOCK (OR (\ALLOCBLOCK.OLD ARLEN GCTYPE NIL ALIGN)
                               (\ALLOCBLOCK.NEW ARLEN GCTYPE NIL ALIGN)
                               (PROGN (FRPTQ 10 (RECLAIM))
                                      ;; We're probably out of array space;  our last chance is to collect and hope something shows
                                      ;; up on the free list.
                                      (\ALLOCBLOCK.OLD ARLEN GCTYPE INITONPAGE ALIGN))
                               (GO FULL)))                  ; ABLOCK now points to the beginning of the actual block of
                                                            ; storage to be used
              (replace (ARRAYBLOCK INUSE) of ABLOCK with T)
              (replace (ARRAYBLOCK INUSE) of (fetch (ARRAYBLOCK TRAILER) of ABLOCK) with T)
              (replace (ARRAYBLOCK GCTYPE) of ABLOCK with GCTYPE)
              (\CHECKARRAYBLOCK ABLOCK NIL)
              (.CHECK.ALLOCATION.COUNT. NCELLS)
              (SETQ ABLOCK (\ADDBASE ABLOCK \ArrayBlockHeaderWords))
              (PROG1 (\DELREF ABLOCK)
                     (.CHECK.ALLOCATION.COUNT. 1))
              (RETURN ABLOCK))
      FULL
          (LISPERROR "ARRAYS FULL" NIL T)                   ; User might release something, so retry.
          (GO RETRY])
```

(**ALLOCBLOCK.OLD**

```
[LAMBDA (ARLEN GCTYPE INITONPAGE ALIGN)                     (* bvm%: "15-Feb-85 11:01")
   ;; Returns a block of the right size and alignment, or NIL if one couldn't be found.
   (for BKTI from (BUCKETINDEX ARLEN) to \MAXBUCKETINDEX bind ABLOCK
      when (AND (SETQ ABLOCK (\GETBASEPTR (\ADDBASE2 \FREEBLOCKBUCKETS BKTI)
                                          0))
                (bind (1STBLOCK.IN.FREECHAIN _ ABLOCK)
                      USABLELEN REMAINDERLEN PREFIXLEN repeatuntil (EQ (SETQ ABLOCK (fetch (ARRAYBLOCK FWD)
                                                                                       of ABLOCK))
                                                                       1STBLOCK.IN.FREECHAIN)
                   when (PROGN [COND
                                  ((OR (NEQ (fetch (ARRAYBLOCK PASSWORD) of ABLOCK)
                                            \ArrayBlockPassword)
                                       (NEQ (fetch (ARRAYBLOCK PASSWORD) of (fetch (ARRAYBLOCK TRAILER)
                                                                                   of ABLOCK))
                                            \ArrayBlockPassword))
                                   (RETURN (\MP.ERROR \MP.BADARRAYBLOCK "Bad Array Block" ABLOCK]
                               (SETQ PREFIXLEN (COND
                                                  ((OR ALIGN INITONPAGE)
                                                   (\PREFIXALIGNMENT? ARLEN INITONPAGE ALIGN GCTYPE ABLOCK))
                                                  (T 0)))
                               (IGEQ (SETQ USABLELEN (IDIFFERENCE (fetch (ARRAYBLOCK ARLEN) of ABLOCK)
                                                                  PREFIXLEN))
                                     ARLEN))
                   do (\CHECKARRAYBLOCK ABLOCK T T)
                      (\DELETEBLOCK? ABLOCK)                ; take it off the free list
                      [COND
                         ((NEQ PREFIXLEN 0)                 ; We must split off a bit initially, in order to preserve the
                                                            ; INITONPAGE request
                          (\MERGEBACKWARD (\MAKEFREEARRAYBLOCK ABLOCK PREFIXLEN))
                          (SETQ ABLOCK (\ADDBASE2 ABLOCK PREFIXLEN]
                      (SETQ REMAINDERLEN (IDIFFERENCE USABLELEN ARLEN))
                      (COND
                         [(IGREATERP REMAINDERLEN (COND
                                                     (\HUNKING? (IPLUS \MAX.CELLSPERHUNK
                                                                       \ArrayBlockOverheadCells))
                                                     (T 0)))    ; Split off any extra space from the end of the block.
                          (\MERGEFORWARD (\LINKBLOCK (\MAKEFREEARRAYBLOCK (\ADDBASE2 ABLOCK ARLEN)
                                                                          REMAINDERLEN]
                         (\HUNKING?                         ; Coerce the length upwards so as not to have a runt block
                                (SETQ ARLEN USABLELEN)))
                      (COND
```

```
                        ((OR (NEQ PREFIXLEN 0)
                             (NEQ USABLELEN ARLEN))          ; If we changed the length of the block, store the new length now
                         (\MAKEFREEARRAYBLOCK ABLOCK ARLEN)))
                      (\CHECKARRAYBLOCK ABLOCK T)
                      (\CLEARCELLS (\ADDBASE ABLOCK \ArrayBlockHeaderWords)
                            (IDIFFERENCE ARLEN \ArrayBlockOverheadCells))
                                                           ; clear out old garbage
                                                           ; signal that we found one
                      (RETURN T)))
    do (RETURN ABLOCK])
```

(\ALLOCBLOCK.NEW

```
[LAMBDA (ARLEN GCTYPE INITONPAGE ALIGN)                    ; Edited 4-Jan-93 02:06 by jds
  (DECLARE (GLOBALVARS \ArrayFrLst \NxtArrayPage))
```

;; Patch up a new section of memory beginning at the end of current arrayspace, and make it a freeblock for subsequent usage. Also used to
;; increment to the next page/segment boundary when allocating code arrays

```
  (PROG (FINALWORD FINALPAGE NEXTFREEBLOCK PREFIXLEN)
    RETRY
        [COND
           ([AND (OR INITONPAGE ALIGN)
                 (NEQ 0 (SETQ PREFIXLEN (\PREFIXALIGNMENT? ARLEN INITONPAGE ALIGN GCTYPE \ArrayFrLst]
                                                           ; Gobble up a modest amount of space in order to insure correct
                                                           ; alignment.
            (COND
               ((SETQ PREFIXLEN (\ALLOCBLOCK.NEW PREFIXLEN))
                (\MERGEBACKWARD PREFIXLEN)                 ; Problem: what happens if array space switch happened inside
                                                           ; this \ALLOCBLOCK.NEW ?

               )
              (T (RETURN]
        (SETQ FINALWORD (\ADDBASE (\ADDBASE \ArrayFrLst ARLEN)
                                  (SUB1 ARLEN)))
```

;; FINALWORD is pointer to the last word of the new block. The new \ArrayFrLst will be one past that, i.e., at (\ADDBASE2 \ArrayFrLst ARLEN) --
;; The double \ADDBASE avoids large integer arithmetic and computing FINALWORD first avoids negative arguments to \ADDBASE

```
        (SETQ NEXTFREEBLOCK (\ADDBASE FINALWORD 1))
        [COND
           ((IGREATERP (SETQ FINALPAGE (fetch (POINTER PAGE#) of FINALWORD))
                   (IDIFFERENCE \NxtMDSPage \GUARDSTORAGEFULL))
                                                           ; Make sure that there are enough pages to satisfy this request
                                                           ; before we make any global changes.
            (SELECTQ (\CHECKFORSTORAGEFULL (ADD1 (IDIFFERENCE FINALPAGE \NxtArrayPage)))
                 (T                                        ; Is ok, go ahead
                   )
                 (0                                        ; Is ok, but \NxtArrayPage moved.
                    (GO RETRY))
                 (RETURN NIL]
```

;; \NxtArrayPage is the page after the page of FINALWORD, the next one that needs to be \NEWPAGEd. \ArrayFrLst's page will be (SUB1
;; \NxtArrayPage) except when it is allowed to be EQ to the first word on \NxtArrayPage

```
        (until (IGREATERP \NxtArrayPage FINALPAGE) do (\MAKEMDSENTRY \NxtArrayPage 0)
                                                      (\NEW2PAGE (create POINTER
                                                                         PAGE# _ \NxtArrayPage))
                                                      (\PUTBASEFIXP \NxtArrayPage 0 (IPLUS \NxtArrayPage 2)))
        (RETURN (PROG1 (\MAKEFREEARRAYBLOCK \ArrayFrLst ARLEN)
                       (SETQ.NOREF \ArrayFrLst NEXTFREEBLOCK])
```

(\PREFIXALIGNMENT?

```
[LAMBDA (ARLEN INITONPAGE ALIGN GCTYPE BASE)               (* Pavel "16-Oct-86 14:15")
```

;; how many cells must be added to to the base address of BASE to get a block whose first data word is aligned according to ALIGN and which has
;; its first INITONPAGE cells all on one page

```
  (PROG ((DAT (fetch (POINTER CELLINSEGMENT) of (\ADDBASE BASE \ArrayBlockHeaderWords)))
         (ADJUSTMENT 0)
         FUDGE)
```

;; DAT will hold the cell-in-segment offset of the first dataword of the arrayblock; it is this first dataword which must be aligned etc rather than the
;; true beginning of the block.

```
    LP  (COND
           ((AND ALIGN (NEQ (SETQ FUDGE (IREMAINDER DAT ALIGN))
                            0))                            ; Not aligned, so adjust first for that.
            (add ADJUSTMENT (SETQ FUDGE (IDIFFERENCE ALIGN FUDGE)))
            (add DAT FUDGE)))
        (COND
           ((AND INITONPAGE (NEQ (FLOOR DAT CELLSPERPAGE)
                                 (FLOOR (IPLUS DAT INITONPAGE -1)
                                        CELLSPERPAGE)))     ; There aren't INITONPAGE cells on the page, so go to next
                                                           ; page boundary
            [add ADJUSTMENT (SETQ FUDGE (IDIFFERENCE CELLSPERPAGE (IMOD DAT CELLSPERPAGE]
            (add DAT FUDGE)
```

;; No need to realign at this point. ALIGN must be a power of two, so it's either an alignment less than CELLSPERPAGE, in which
;; case this page boundary satisfies it, or it's a multiple of CELLSPERPAGE, in which case the first COND satisfied it and we didn't
;; have to touch it in this COND

```
            ))
         (COND
            ([AND (EQ GCTYPE CODEBLOCK.GCT)
                  (IGREATERP (IDIFFERENCE ARLEN \ArrayBlockOverheadCells)
                         (SETQ FUDGE (IDIFFERENCE CELLSPERSEGMENT (SETQ DAT (IMOD DAT CELLSPERSEGMENT]
               ;; Code arrays cannot cross segment boundaries.  Note that ARLEN includes the overhead cells, hence the extra subtraction.
               (add ADJUSTMENT FUDGE)
               (add DAT FUDGE)
               ;; No need to re-check the alignment since ALIGN and INITONPAGE are both guaranteed satisified by a block starting on a segment
               ;; boundary

               ))
      ;; The following code claims to prevent splitting off too small a block, but it's not clear this is intrinsically bad, and the code does not appear to do
      ;; anything rational.  -- bvm --- (COND ((AND (NEQ ADJUSTMENT 0) \HUNKING?  (IGREATERP (SETQ FUDGE (IDIFFERENCE (IPLUS
      ;; \MAX.CELLSPERHUNK \ArrayBlockOverheadCells) ADJUSTMENT)) 0) (PROGN (* * Account for potential merging backwards when this initial
      ;; piece is split off.) (AND (EQ (fetch (ARRAYBLOCK PASSWORD) of (SETQ PREVTRAILER (\ADDBASE BASE (IMINUS
      ;; \ArrayBlockTrailerCells)))) \ArrayBlockPassword) (NOT (fetch (ARRAYBLOCK INUSE) of PREVTRAILER)) (ILESSP (fetch (ARRAYBLOCK
      ;; ARLEN) of PREVTRAILER) FUDGE)))) (* Just to ensure that we don't break up a large arrayblocks into two pieces one of which is too small to
      ;; be usable.) (add ADJUSTMENT FUDGE) (SETQ DAT (IPLUS DAT FUDGE)) (* Go around again, since this function wouldn't have been called
      ;; unless one of INITONPAGE or ALIGN were non-null.) (GO LP)))
      (RETURN ADJUSTMENT])
```

(\MAKEFREEARRAYBLOCK

```
[LAMBDA (BLOCK LENGTH)                                          (* lmm "25-Jul-84 13:07")
   (replace (ARRAYBLOCK ABFLAGS) of BLOCK with \FreeArrayFlagWord)
   (replace (ARRAYBLOCK ARLEN) of BLOCK with LENGTH)
   (replace (ARRAYBLOCK ABFLAGS) of (fetch (ARRAYBLOCK TRAILER) of BLOCK) with \FreeArrayFlagWord)
   (replace (ARRAYBLOCK ARLEN) of (fetch (ARRAYBLOCK TRAILER) of BLOCK) with LENGTH)
   BLOCK])
```

(\DELETEBLOCK?

```
[LAMBDA (BASE)                                                  (* bvm%: "15-Feb-85 11:04")
   (COND
      ((AND (IGEQ (fetch (ARRAYBLOCK ARLEN) of BASE)
                  \MinArrayBlockSize)
            (fetch (ARRAYBLOCK FWD) of BASE))                   ; Allegedly, BASE has been 'checked' before coming here.
       (PROG [(F (fetch (ARRAYBLOCK FWD) of BASE))
              (B (fetch (ARRAYBLOCK BKWD) of BASE))
              (FBL (FREEBLOCKCHAIN.N (fetch ARLEN of BASE]
             (COND
                ((EQ BASE F)
                 (COND
                    ((EQ BASE (\GETBASEPTR FBL 0))
                     (\PUTBASEPTR FBL 0 NIL))
                    (T (\MP.ERROR \MP.BADDELETEBLOCK "deleting last block # FREEBLOCKLIST")))
                 (RETURN))
                ((EQ BASE (\GETBASEPTR FBL 0))
                 (\PUTBASEPTR FBL 0 F)))
             (replace (ARRAYBLOCK BKWD) of F with B)
             (replace (ARRAYBLOCK FWD) of B with F])
```

(\LINKBLOCK

```
[LAMBDA (BASE)                                                  (* JonL "16-Jan-85 02:46")
   ;; Add BASE to the free list.  Assumes that BASE is a well-formed free block.
   [COND
      (\FREEBLOCKBUCKETS (COND
                            ((ILESSP (fetch (ARRAYBLOCK ARLEN) of BASE)
                                     \MinArrayBlockSize)
                             (\CHECKARRAYBLOCK BASE T))
                            (T (PROG ((FBL (FREEBLOCKCHAIN.N (fetch ARLEN of BASE)))
                                      FREEBLOCK)
                                     (SETQ FREEBLOCK (\GETBASEPTR FBL 0))
                                     (COND
                                        ((NULL FREEBLOCK)
                                         (replace (ARRAYBLOCK FWD) of BASE with BASE)
                                         (replace (ARRAYBLOCK BKWD) of BASE with BASE))
                                        (T (replace (ARRAYBLOCK FWD) of BASE with FREEBLOCK)
                                           (replace (ARRAYBLOCK BKWD) of BASE with (fetch (ARRAYBLOCK BKWD)
                                                                                          of FREEBLOCK))
                                           (replace (ARRAYBLOCK FWD) of (fetch (ARRAYBLOCK BKWD) of FREEBLOCK)
                                              with BASE)
                                           (replace (ARRAYBLOCK BKWD) of FREEBLOCK with BASE)))
                                     (\PUTBASEPTR FBL 0 BASE)
                                     (\CHECKARRAYBLOCK BASE T T]
   BASE])
```

(\MERGEBACKWARD

```
[LAMBDA (BASE)                                                  (* bvm%: " 6-Feb-85 16:53")
   ;; Caller is uninterruptable and asserts that a non-NIL BASE is a free but unlinked arrayblock.  We return a linked (if possible) block, either BASE
   ;; itself or an enlarged previous free block that is linked (if possible) and includes the BASE storage.
```

```
    (PROG (ARLEN PARLEN PBASE PTRAILER SPLIT)
          [COND
             ((NULL BASE)
              (RETURN NIL))
             ([OR (NOT \ARRAYMERGING)
                  (EQ BASE \ARRAYSPACE)
                  (EQ BASE \ARRAYSPACE2)
                  (fetch (ARRAYBLOCK INUSE) of (SETQ PTRAILER (\ADDBASE BASE (IMINUS \ArrayBlockTrailerWords]
              ;; If this is the absolute 'first' block of array space, then there is nothing behind it to merge;  similarly, if the block behind it is in use,
              ;; then don't merge.
              (RETURN (\LINKBLOCK BASE]
          [SETQ PBASE (\ADDBASE2 BASE (IMINUS (fetch (ARRAYBLOCK ARLEN) of PTRAILER]
          (\CHECKARRAYBLOCK PBASE T)
          (\DELETEBLOCK? PBASE)
          (RETURN (\ARRAYBLOCKMERGER PBASE BASE])
```

(\MERGEFORWARD

```
  [LAMBDA (BASE)                                                    (* bvm%: "15-Feb-85 11:18")
```

;; BASE is a free and linked (if possible) block. Merge with the next block if it is free and not too big. Caller must be uninterruptable.

```
    (PROG (NBASE NBINUSE)
          (COND
             ((OR (NOT \ARRAYMERGING)
                  (NULL BASE)
                  (\CHECKARRAYBLOCK BASE T T)
                  (EQ (SETQ NBASE (\ADDBASE2 BASE (fetch (ARRAYBLOCK ARLEN) of BASE)))
                      \ArrayFrLst)
                  (EQ NBASE \ArrayFrLst2)
                  [\CHECKARRAYBLOCK NBASE (NOT (SETQ NBINUSE (fetch (ARRAYBLOCK INUSE) of NBASE]
                  NBINUSE)
              (RETURN NIL)))                                        ; Note that if we ever get to here, both blocks have been
                                                                    ; 'checked'
          (\DELETEBLOCK? NBASE)
          (\DELETEBLOCK? BASE)
          (\ARRAYBLOCKMERGER BASE NBASE])
```

(\ARRAYBLOCKMERGER

```
  [LAMBDA (BASE NBASE)                                              (* bvm%: "13-Feb-85 14:57")
```

;;; BASE and NBASE are two consecutive unlinked freeblocks. (Called only after the two blocks have been 'checked')

```
    (PROG ((ARLEN (fetch (ARRAYBLOCK ARLEN) of BASE))
           (NARLEN (fetch (ARRAYBLOCK ARLEN) of NBASE))
           SECONDBITE MINBLOCKSIZE SHAVEBACK)
          (SETQ SECONDBITE (IDIFFERENCE \MaxArrayBlockSize ARLEN))
          (COND
             ((IGREATERP NARLEN SECONDBITE)
              ;; check if sum of NARLEN+ARLEN is leq maximum.  (Written this way to stay within small number range.) If not, then break up into
              ;; two freeblocks since one can't hold all the cells.
              (SETQ ARLEN \MaxArrayBlockSize)
              (SETQ NARLEN (IDIFFERENCE NARLEN SECONDBITE))
              ;; Normal overflow case is just to make the first block as big as possible, then leave the rest in the second block.  So the code above
              ;; adds to ARLEN and subtracts from NARLEN an equal amount to achieve the desired split.  However, check that the remaining
              ;; NBASE block is not too small
              (COND
                 ([ILESSP NARLEN (SETQ MINBLOCKSIZE (COND
                                                       (\HUNKING? (IPLUS \ArrayBlockOverheadCells
                                                                         \MAX.CELLSPERHUNK))
                                                       (T \MinArrayBlockSize]
                  ;; Decrease ARLEN and SECONDBITE by the amount it will take to get NARLEN up to MINBLOCKSIZE -- SHAVEBACK is
                  ;; negative
                  (SETQ SHAVEBACK (IDIFFERENCE NARLEN (SETQ NARLEN MINBLOCKSIZE)))
                  (add ARLEN SHAVEBACK)
                  (add SECONDBITE SHAVEBACK)))
              ;; Okay, make a tail of the second block into a free block of its own
              (\LINKBLOCK (\MAKEFREEARRAYBLOCK (\ADDBASE2 NBASE SECONDBITE)
                                               NARLEN))
              (SETQ NARLEN 0)))
          (RETURN (\LINKBLOCK (\MAKEFREEARRAYBLOCK BASE (IPLUS ARLEN NARLEN])
```

(\#BLOCKDATACELLS

```
  [LAMBDA (DATAWORD)                                                (* JonL "20-Sep-84 19:07")
```

;; DATAWORD is a pointer as would be returned by \ALLOCBLOCK Returns the number of cells available to the caller. Compiled closed so that
;; we can change internal representations without clients needing to be recompiled.

```
    (PROG ((TYPENO (NTYPX DATAWORD)))
          (RETURN (COND
                     [(EQ 0 TYPENO)
```

```
                    (COND
                       ((type? ARRAYBLOCK DATAWORD)
                        (IDIFFERENCE (fetch (ARRAYBLOCK ARLEN) of (\ADDBASE DATAWORD (IMINUS
                                                                                        \ArrayBlockHeaderWords
                                                                                          )))
                              \ArrayBlockOverheadCells))
                       (T (\ILLEGAL.ARG DATAWORD]
                    (T (OR (AND (OR \HUNKING? (fetch DTDHUNKP of (\GETDTD TYPENO)))
                                (HUNKSIZEFROMNUMBER TYPENO))
                           (\ILLEGAL.ARG DATAWORD])
```

(\COPYARRAYBLOCK

```
[LAMBDA (OLD)                                          ; Edited 3-Mar-87 22:28 by bvm:
  (LET [(HEADER (\ADDBASE OLD (IMINUS \ArrayBlockHeaderWords]
       (COND
          [(AND (IEQ \ArrayBlockPassword (fetch PASSWORD of HEADER))
                (fetch (ARRAYBLOCK INUSE) of HEADER))
           (LET* ((LEN (- (fetch (ARRAYBLOCK ARLEN) of HEADER)
                          \ArrayBlockOverheadCells))
                  (TYP (fetch (ARRAYBLOCK GCTYPE) of HEADER))
                  (NEW (\ALLOCBLOCK LEN TYP)))
                 (PROG1 NEW
                     (SELECTC TYP
                          (PTRBLOCK.GCT                ; Have to reference count the pointers as we copy
                                       (FRPTQ LEN (\RPLPTR NEW 0 (COPYALL (\GETBASEPTR OLD 0)))
                                              (SETQ NEW (\ADDBASE NEW WORDSPERCELL))
                                              (SETQ OLD (\ADDBASE OLD WORDSPERCELL))))
                          (CODEBLOCK.GCT               ; should increment references from code
                              (\COPYCODEBLOCK NEW OLD (UNFOLD LEN WORDSPERCELL)))
                          (\BLT NEW OLD (UNFOLD LEN WORDSPERCELL))))]
          (T                                           ; Not an array block
             OLD])
```

(\RECLAIMARRAYBLOCK

```
[LAMBDA (P)                                            ; Edited 8-Jan-88 18:31 by jop
```

;; Called to reclaim objects of type 0. This is called with interrupts turned off. Returns T to tell GC that we reclaimed it.

```
(PROG ((B (\ADDBASE P (IMINUS \ArrayBlockHeaderWords)))
       (RECLAIM-P T))
```

;; B points to arrayblock header, P to first and subsequent data words

```
      (IF (OR (< (\HILOC P)
                 \FirstArraySegment)
              (NOT (IEQ \ArrayBlockPassword (fetch PASSWORD of B)))
              (NOT (fetch (ARRAYBLOCK INUSE) of B)))
          THEN
```

;; RAID instead of \GCERROR because this error is continuable with ^N.

```
               (\MP.ERROR \MP.BADARRAYRECLAIM "Bad array block reclaimed--continue with ^N but save state
                      ASAP")
               (RETURN T))
      (SELECTC (fetch (ARRAYBLOCK GCTYPE) of B)
          (PTRBLOCK.GCT                                ; Release all pointers
                        (for old P (TRAILER _ (fetch (ARRAYBLOCK TRAILER) of B)) by (\ADDBASE P WORDSPERCELL)
                           until (EQ P TRAILER) do (\RPLPTR P 0 NIL)))
          (CODEBLOCK.GCT                               ; Release literals
```

;; Since \reclaimcodeblock is a finalization function -- returns nil if do reclaim and t if don't reclaim

```
               (SETQ RECLAIM-P (NOT (\RECLAIMCODEBLOCK P))))
          NIL)
      [IF RECLAIM-P
          THEN (\MERGEFORWARD (\MERGEBACKWARD (\MAKEFREEARRAYBLOCK B (fetch ARLEN of B]
```

;; Always tell GC that we have reclaimed it

```
      (RETURN T])
```

(\ADVANCE.ARRAY.SEGMENTS

```
[LAMBDA (NXTPAGE)                                      ; Edited 4-Jan-93 02:08 by jds
```

;;; Called when the first 8mb are exhausted, and we want to switch array space into the next area, starting with page NXTPAGE -- have to first clean up
;;; what's left in the old area

```
    (PROG (NCELLSLEFT)
          (SETQ.NOREF \ArrayFrLst2 (COND
                                      ((IGEQ [SETQ NCELLSLEFT (IPLUS (UNFOLD (SUB1 (IDIFFERENCE
                                                                                     \NxtArrayPage
                                                                                     (fetch (POINTER PAGE#)
                                                                                        of \ArrayFrLst)))
                                                                             CELLSPERPAGE)
                                                                     (IDIFFERENCE CELLSPERPAGE
                                                                            (fetch (POINTER CELLINPAGE)
                                                                               of \ArrayFrLst]
                                             \MinArrayBlockSize)   ; Make the rest of the already allocated array space into a small
                                                                   ; block
                                       (\MERGEBACKWARD (\MAKEFREEARRAYBLOCK \ArrayFrLst NCELLSLEFT))
```

```
                                        (create POINTER
                                               PAGE# _ \LeastMDSPage))
                                      (T \ArrayFrLst)))
          [SETQ.NOREF \ARRAYSPACE2 (SETQ.NOREF \ArrayFrLst (create POINTER
                                                                 PAGE# _ (\PUTBASEFIXP \NxtArrayPage 0 NXTPAGE]
                                                                ; Return code to tell \ALLOCBLOCK.NEW to notice the new
                                                                ; arrangement
          (RETURN 0])

)

(ADDTOVAR \MAIKO.MOVDS (\MAIKO.ALLOCBLOCK \ALLOCBLOCK))

(DEFINEQ
```

(\BYTELT

```
  [LAMBDA (A J)                                                    (* JonL "20-Sep-84 20:01")
```

;; A special function for system accesses to 0-origin byte arrays, of which syntax-tables are the primary example. This compiles open into a
;; GETBASEBYTE, with no checking for argument validity!

```
    (OR [AND [EQ 0 (fetch (ARRAYP ORIG) of (SETQ A (\DTEST A 'ARRAYP]
             (OR (EQ \ST.BYTE (fetch (ARRAYP TYP) of A))
                 (EQ \ST.CODE (fetch (ARRAYP TYP) of A]
        (LISPERROR "ILLEGAL ARG" A))
    (OR (IGREATERP (fetch (ARRAYP LENGTH) of A)
                J)
        (LISPERROR "ILLEGAL ARG" J))
    (\GETBASEBYTE (fetch (ARRAYP BASE) of A)
           (IPLUS (fetch (ARRAYP OFFST) of A)
                  J])
```

(\BYTESETA

```
  [LAMBDA (A J V)                                                  (* JonL "20-Sep-84 20:01")
```

;; A special function for system setting of 0-origin byte arrays, of which syntax-tables are the primary example. This compiles open into a
;; GETBASEBYTE, with no checking for argument validity! --- NOTE: The value is undefined, not V!

```
    (OR [AND [EQ 0 (fetch (ARRAYP ORIG) of (SETQ A (\DTEST A 'ARRAYP]
             (OR (EQ \ST.BYTE (fetch (ARRAYP TYP) of A))
                 (EQ \ST.CODE (fetch (ARRAYP TYP) of A]
        (LISPERROR "ILLEGAL ARG" A))
    (OR (IGREATERP (fetch (ARRAYP LENGTH) of A)
                J)
        (LISPERROR "ILLEGAL ARG" J))
    (AND (fetch (ARRAYP READONLY) of A)
         (LISPERROR "ILLEGAL ARG" A))
    (\PUTBASEBYTE (fetch (ARRAYP BASE) of A)
           (IPLUS (fetch (ARRAYP OFFST) of A)
                  J)
           V])
```

(\WORDELT

```
  [LAMBDA (A J)                                                    (* JonL "20-Sep-84 20:02")
```

;; A special function for system accesses to 0-origin word arrays, This compiles open into a GETBASE, with no checking for argument validity!

```
    (OR (AND [EQ 0 (fetch (ARRAYP ORIG) of (SETQ A (\DTEST A 'ARRAYP]
             (EQ \ST.POS16 (fetch (ARRAYP TYP) of A)))
        (LISPERROR "ILLEGAL ARG" A))
    (OR (IGREATERP (fetch (ARRAYP LENGTH) of A)
                J)
        (LISPERROR "ILLEGAL ARG" J))
    (\GETBASE (fetch (ARRAYP BASE) of A)
           (IPLUS (fetch (ARRAYP OFFST) of A)
                  J])

)

(DEFINEQ
```

(\ARRAYTYPENAME

```
  [LAMBDA (X)                                                      (* rmk%: "21-Dec-83 14:55")
```

;; This is called from the VM function TYPENAME to determine the 'logical' type of the array X

```
    (SELECTC (fetch (ARRAYP TYP) of X)
         (\ST.CODE 'CCODEP)
         'ARRAYP])

)

(RPAQQ \ARRAYMERGING T)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \ARRAYMERGING)
)
```

;; for STORAGE

```
(DEFINEQ

(\SHOW.ARRAY.FREELISTS
  [LAMBDA (SIZESLST)                                          (* bvm%: "12-Feb-85 15:25")
    (COND
       ((OR SIZESLST (SETQ SIZESLST STORAGE.ARRAYSIZES))
        (RESETFORM (RECLAIMMIN MAX.SMALLP)
               (PROG ((TABLE \ABSTORAGETABLE)
                      (N (LENGTH SIZESLST))
                      (TOTAL 0)
                      FBL ABLOCK ARLEN)
                     [COND
                        ((OR (NOT (\BLOCKDATAP TABLE))
                             (IGEQ N (FOLDLO (\#BLOCKDATACELLS TABLE)
                                             2)))
                         ;; Need bigger table if someone has enlarged SIZESLST since last time.  There are 2 cells per table entry
                         (SETQ \ABSTORAGETABLE (SETQ TABLE (\ALLOCBLOCK (UNFOLD (IPLUS N 4)
                                                                                2)
                                                                        UNBOXEDBLOCK.GCT]
                     (\CLEARCELLS TABLE (\#BLOCKDATACELLS TABLE))
                     [for BKTI from 0 to \MAXBUCKETINDEX
                        do (COND
                              ((SETQ FBL (\GETBASEPTR (\ADDBASE2 \FREEBLOCKBUCKETS BKTI)
                                                      0))
                               (SETQ ABLOCK FBL)
                               (repeatuntil (EQ FBL (SETQ ABLOCK (fetch (ARRAYBLOCK FWD) of ABLOCK)))
                                  do (add TOTAL (SETQ ARLEN (fetch (ARRAYBLOCK ARLEN) of ABLOCK)))
                                      (for (SAFENTRY _ TABLE) by (\ADDBASE SAFENTRY (TIMES 2 WORDSPERCELL))
                                         as X in SIZESLST when (OR (NULL X)
                                                                   (ILEQ ARLEN X))
                                         do (add (fetch SAFITEMS of SAFENTRY)
                                                 1)
                                            (add (fetch SAFCELLS of SAFENTRY)
                                                 ARLEN)
                                            (RETURN]
                     (printout NIL T " variable-datum free list: " T)
                     (for (SAFENTRY _ TABLE) by (\ADDBASE SAFENTRY (TIMES 2 WORDSPERCELL)) as X in SIZESLST
                        do (COND
                              (X (printout NIL "le " X))
                              (T (printout NIL "others ")))
                           (printout NIL 10 .I8 (fetch SAFITEMS of SAFENTRY)
                                  " items; " .I8 (fetch SAFCELLS of SAFENTRY)
                                  " cells." T))
                     (printout NIL T "Total cells free: " .I8 TOTAL "  total pages: " .I4 (FOLDHI TOTAL
                                                                                                CELLSPERPAGE)
                           T T])

)

(RPAQ? \ABSTORAGETABLE NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \ABSTORAGETABLE)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD SAFTABLE ((SAFITEMS WORD)
                       (NIL WORD)
                       (SAFCELLS FIXP)))
)
)
```

;; Debugging and RDSYS

```
(DEFINEQ

(\CHECKARRAYBLOCK
  [LAMBDA (BASE FREE ONFREELIST)                              (* bvm%: "13-Feb-85 14:50")
    (COND
       (ARRAYBLOCKCHECKING (PROG (ERROR TRAILER)
                                 (COND
                                    ((NEQ (fetch (ARRAYBLOCK PASSWORD) of BASE)
                                          \ArrayBlockPassword)
                                     (SETQ ERROR "ARRAYBLOCK Password wrong"))
                                    ((NEQ (fetch (ARRAYBLOCK INUSE) of BASE)
                                          (NOT FREE))
                                     (SETQ ERROR "ARRAYBLOCK INUSE bit set wrong"))
                                    ((UNLESSRDSYS (AND FREE (NEQ (\REFCNT BASE)
                                                                 1))
```

```
                    NIL)
             (SETQ ERROR "Free ARRAYBLOCK with RefCnt not 1"))
            ((NEQ (fetch (ARRAYBLOCK PASSWORD) of (SETQ TRAILER (fetch (ARRAYBLOCK TRAILER
                                                                       )
                                                                  of BASE)))
                  \ArrayBlockPassword)
             (SETQ ERROR "ARRAYBLOCK Trailer password wrong"))
            ((NEQ (fetch (ARRAYBLOCK ARLEN) of BASE)
                  (fetch (ARRAYBLOCK ARLEN) of TRAILER))
             (SETQ ERROR "ARRAYBLOCK Header and Trailer length don't match"))
            ((NEQ (fetch (ARRAYBLOCK INUSE) of BASE)
                  (NOT FREE))
             (SETQ ERROR "ARRAYBLOCK Trailer INUSE bit set wrong"))
            ((OR (NOT ONFREELIST)
                 (ILESSP (fetch (ARRAYBLOCK ARLEN) of BASE)
                        \MinArrayBlockSize))
                                        ; Remaining tests only for blocks on free list
             (RETURN))
            ((OR (NOT (EQPTR (fetch (ARRAYBLOCK FWD) of (fetch (ARRAYBLOCK BKWD)
                                                               of BASE))
                             BASE))
                 (NOT (EQPTR (fetch (ARRAYBLOCK BKWD) of (fetch (ARRAYBLOCK FWD)
                                                                of BASE))
                             BASE)))
             (SETQ ERROR "ARRAYBLOCK links fouled"))
            [(bind (FBL _ (FREEBLOCKCHAIN.N (fetch (ARRAYBLOCK ARLEN) of BASE)))
                   ROVER first (OR (SETQ ROVER (\GETBASEPTR FBL 0))
                                   (RETURN (SETQ ERROR "Free block's bucket empty")))
                do (AND (EQPTR ROVER BASE)
                        (RETURN))
                   (\CHECKARRAYBLOCK ROVER T)
                repeatuntil (EQ (SETQ ROVER (fetch (ARRAYBLOCK FWD) of ROVER))
                                (\GETBASEPTR FBL 0]
             (T                         ; Everything ok
                (RETURN)))
           (UNLESSRDSYS (\MP.ERROR \MP.BADARRAYBLOCK ERROR BASE T)
                  (ERROR BASE ERROR))
           (RETURN ERROR])
```

(\PARSEARRAYSPACE

```
  [LAMBDA (FN)                                              (* bvm%: "16-Apr-86 17:05")
    (COND
       ((NEQ \ArrayFrLst2 \ARRAYSPACE2)                     ; Array space is in two chunks
        (\PARSEARRAYSPACE1 FN \ARRAYSPACE \ArrayFrLst2)
        (\PARSEARRAYSPACE1 FN \ARRAYSPACE2 \ArrayFrLst))
       (T (\PARSEARRAYSPACE1 FN \ARRAYSPACE \ArrayFrLst])
```

(\PARSEARRAYSPACE1

```
  [LAMBDA (FN START END)                                    (* bvm%: " 9-Jan-85 17:10")
    (for (ROVER _ START) repeatuntil [EQPTR END (SETQ ROVER (\ADDBASE2 ROVER (fetch (ARRAYBLOCK ARLEN) of ROVER]
       do (\CHECKARRAYBLOCK ROVER (NOT (fetch (ARRAYBLOCK INUSE) of ROVER))
                 (AND (NOT (fetch (ARRAYBLOCK INUSE) of ROVER))
                      (fetch (ARRAYBLOCK FWD) of ROVER)))
          (AND FN (APPLY* FN ROVER (fetch (ARRAYBLOCK ARLEN) of ROVER)
                          (fetch (ARRAYBLOCK INUSE) of ROVER)
                          (fetch (ARRAYBLOCK GCTYPE) of ROVER])

)

(RPAQ? ARRAYBLOCKCHECKING )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ARRAYBLOCKCHECKING)
)

;; Basic hunking

(DEFINEQ
```

(\ALLOCHUNK

```
  [LAMBDA (NCELLS GCTYPE INITONPAGE ALIGN)                  (* bvm%: "13-Jun-86 15:21")
    (COND
       ([AND ALIGN (OR (IGREATERP ALIGN \MAX.CELLSPERHUNK)
                       (NOT (FMEMB ALIGN (SELECTC GCTYPE
                                             (UNBOXEDBLOCK.GCT
                                                  (CONSTANT (for X in \HUNK.UNBOXEDSIZES
                                                               when (AND (IGREATERP X 1)
                                                                         (ILEQ X \MAX.CELLSPERHUNK)
                                                                         (POWEROFTWOP X))
                                                               collect X)))
                                             (PTRBLOCK.GCT (CONSTANT (for X in \HUNK.PTRSIZES
                                                                        when (AND (IGREATERP X 1)
                                                                                  (ILEQ X \MAX.CELLSPERHUNK)
                                                                                  (POWEROFTWOP X))
```

```
                                                                    collect X)))
                                                (CODEBLOCK.GCT
                                                    (CONSTANT (LIST CELLSPERQUAD)))
                                               NIL]                 ; Certify that the alignment request is legitimate.
        (ERROR "Oddball alignment request" ALIGN)))
    (PROG ((TYPENUM.TABLE (SELECTC GCTYPE
                                (UNBOXEDBLOCK.GCT
                                    \UNBOXEDHUNK.TYPENUM.TABLE)
                                (CODEBLOCK.GCT
                                    \CODEHUNK.TYPENUM.TABLE)
                                (PTRBLOCK.GCT \PTRHUNK.TYPENUM.TABLE)
                                (SHOULDNT)))
           (FAILCNT 0)
           DTNUMBER HUNK HUNKSIZE ONPAGE STRADDLERS)
      BEG [do (SETQ DTNUMBER (\GETBASEBYTE TYPENUM.TABLE NCELLS))
              (SETQ HUNKSIZE (HUNKSIZEFROMNUMBER DTNUMBER))
          repeatuntil (OR (NOT ALIGN)
                          (EQ 0 (IREMAINDER (FOLDLO (fetch DTDSIZE of (\GETDTD DTNUMBER))
                                                    WORDSPERCELL)
                                            ALIGN))
                          (COND
                             ((IGREATERP (SETQ NCELLS (ADD1 HUNKSIZE))
                                         \MAX.CELLSPERHUNK)
                              (GO LOSE))
                             (T ;; We're allowed to chunk up the size of the request in order to meet the alignment;  ultimately we should top
                                ;; off at \MAX.CELLSPERHUNK
                              NIL]
      LP  (SETQ HUNK (CREATECELL DTNUMBER))
          (COND
             ([OR (NULL INITONPAGE)
                  (ILESSP INITONPAGE (SETQ ONPAGE (IDIFFERENCE CELLSPERPAGE (fetch (POINTER CELLINPAGE)
                                                                                of HUNK]
                                                                    ; Ah, happy case -- all constraints satisfied
              (RETURN HUNK)))
      ;; Sigh, gotta try to get one with more of the initial 'run' of cells on the same page.
          (COND
             (\IN.MAKEINIT                                          ; Lose!  Only code has an INITONPAGE requirement, and
                                                                    ; makeinit does not allocate code via \ALLOCBLOCK
                    (HELP "Call to \ALLOCBLOCK with non-NIL INITONPAGE demand" INITONPAGE))
             (T (COND
                   ([AND (EQ GCTYPE CODEBLOCK.GCT)
                         (ILEQ (IQUOTIENT (ITIMES 10 ONPAGE)
                                          HUNKSIZE)
                               (COND
                                  ((ILEQ HUNKSIZE 24)
                                   60)
                                  ((ILEQ HUNKSIZE 50)
                                   50)
                                  (T 30]
                    ;; If the percentage of the page-straddling codehunk that is on the first page is too small, then just toss this loser into the
                    ;; 'black hole' This heuristic is based on empirical data taken about Sep 1984 which observed the ratio of 'on-page'
                    ;; requirements to code length.
                    (\ADDREF HUNK))
                   (T                                               ; So that a GC doesn't sneak in and put it back on the freelist too
                                                                    ; soon.
                      (push STRADDLERS HUNK)))
                (COND
                   ((IGREATERP (add FAILCNT 1)
                               16)                                  ; Put a limit to this nonsense of trying to find a
                                                                    ; non-page-straddling hunk!
                    (GO LOSE))
                   ((EQ FAILCNT 8)                                  ; After too many failures with this size of hunk, try the next
                                                                    ; container size up.
                    (SETQ NCELLS (ADD1 HUNKSIZE))
                    (AND STRADDLERS (SETQ \HUNKREJECTS (NCONC STRADDLERS \HUNKREJECTS)))
                    (GO BEG)))
                (GO LP)))
      LOSE
          (AND STRADDLERS (SETQ \HUNKREJECTS (NCONC STRADDLERS \HUNKREJECTS)))
          (RETURN])

)

(RPAQQ \HUNK.PTRSIZES (2 4 5 6 7 8 10 12 16 24 32 42 64))

;; Compiler needs \HUNK.PTRSIZES for creating closure environments

(DECLARE%: EVAL@COMPILE DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE
```

```
(PUTPROPS HUNKSIZEFROMNUMBER MACRO ((NTYPX)
                                    (FOLDLO (fetch DTDSIZE of (\GETDTD NTYPX))
                                            WORDSPERCELL)))
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

(RPAQQ \HUNK.UNBOXEDSIZES (1 2 3 4 5 6 7 8 9 10 12 14 16 20 24 28 32 40 48 64))

(RPAQQ \HUNK.CODESIZES (12 16 20 24 28 32 36 42 50 64))

(RPAQQ \HUNK.PTRSIZES (2 4 5 6 7 8 10 12 16 24 32 42 64))

(CONSTANTS \HUNK.UNBOXEDSIZES \HUNK.CODESIZES \HUNK.PTRSIZES)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \HUNKING? \UNBOXEDHUNK.TYPENUM.TABLE \CODEHUNK.TYPENUM.TABLE \PTRHUNK.TYPENUM.TABLE)
)
)

;; Keep a list of all the hunks rejected due to poor page-straddling alignment, or to code falling off the end of a doublepage

(RPAQQ \HUNKREJECTS NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \HUNKREJECTS)
)

;; for MAKEINIT

(DEFINEQ

(PREINITARRAYS
  [LAMBDA NIL                                                    (* bvm%: " 9-Jan-85 16:50")
    ;; This is called only at the very beginning of MAKEINIT.  \ARRAYspace and \ARRAYbase are INITCONSTANTS.  This sets up the array allocator
    ;; so that MAKEINIT can do, e.g., string allocations.
    (DECLARE (GLOBALVARS \ArrayFrLst \ArrayFrLst2 \NxtArrayPage))
    (SETQ.NOREF \ArrayFrLst (\VAG2 \FirstArraySegment 0))
    (SETQ.NOREF \ArrayFrLst2 \ARRAYSPACE2)
    (SETQ.NOREF \NxtArrayPage (PAGELOC \ArrayFrLst])

(POSTINITARRAYS
  [LAMBDA (AFTERCODEPTR CODESTARTPAGE CODENEXTPAGE)              (* bvm%: " 7-Feb-85 15:30")
    ;; Called only from MAKEINIT after all code and data has been copied to the new image.  AFTERCODEPTR is a pointer to the first word after the
    ;; last code byte.  CODESTARTPAGE is the page at which MAKEINIT code arrays being.  This function makes sure that any unused space
    ;; between the strings and the beginning of the code gets linked in as free arrayblocks.
    (SETQ \FREEBLOCKBUCKETS (\ALLOCBLOCK (ADD1 \MAXBUCKETINDEX)))
    (PROG [(EXTRACELLS (IDIFFERENCE (UNFOLD CODESTARTPAGE CELLSPERPAGE)
                                    (IPLUS (UNFOLD (fetch SEGMENT# of \ArrayFrLst)
                                                   CELLSPERSEGMENT)
                                           (fetch CELLINSEGMENT of \ArrayFrLst]
      ;; First, tell the makeiniter how many pages were left over in the string space.  He may want to adjust the constants to keep this down to just a
      ;; couple of pages.
          (COND
             ((IGREATERP EXTRACELLS \MaxArrayBlockSize)
              (printout T T T "POSTINITARRAYS:  You pre-allocated too much string space." T 19
                     "MKI.CODESTARTOFFSET on MAKEINIT should be reduced by about " (IDIFFERENCE (FOLDLO
                                                                                                  EXTRACELLS
                                                                                                  CELLSPERPAGE
                                                                                                  )
                                                                                          10)
                     "." T)
              (HELP))
             ((IGEQ EXTRACELLS \MinArrayBlockSize)               ; We don't allow more than one array-block extra.
              (printout T T T "POSTINITARRAYS:  There were " (FOLDLO EXTRACELLS CELLSPERPAGE)
                     " allocated but unused array pages." T T))
             (T (printout T T "POSTINITARRAYS:  String space overflowed into code-arrays" T 19 "You should add
                       at least " (ADD1 (FOLDLO (IMINUS EXTRACELLS)
                                                CELLSPERPAGE))
                     " to MKI.CODESTARTOFFSET on MAKEINIT." T)
                (HELP)))                                         ; Cause those pages to get allocated
          (\LINKBLOCK (\ALLOCBLOCK.NEW EXTRACELLS))
          (SETQ.NOREF \ArrayFrLst AFTERCODEPTR)                  ; \NxtArrayPage is the next page that needs to be NEWPAGEd
          (SETQ.NOREF \NxtArrayPage CODENEXTPAGE)
          (for VP from (PAGELOC \ARRAYSPACE) to (PAGELOC \NxtArrayPage) by (FOLDLO \MDSIncrement WORDSPERPAGE)
             do (\MAKEMDSENTRY VP 0])
```

```
(FILEARRAYBASE
  [LAMBDA NIL                                                    (* rmk%: "15-MAR-82 21:55")
    (\ADDBASE \ARRAYSPACE (LOCAL (IPLUS (UNFOLD MKI.CODESTARTOFFSET WORDSPERPAGE)
                                        (FOLDLO (IDIFFERENCE (GETFILEPTR (OUTPUT))
                                                             MKI.FirstDataByte)
                                                BYTESPERWORD])

(FILEBLOCKTRAILER
  [LAMBDA (BLOCKINFO)                                            (* rmk%: "18-NOV-82 09:49")
    ;; Sets up block trailer, assuming file is currently positioned just past the last dataword
    (BOUT16 OUTX \UsedArrayFlagWord)
    (BOUT16 OUTX BLOCKINFO])

(FILECODEBLOCK
  [LAMBDA (NCELLS INITONPAGE)                                    (* JonL "20-Sep-84 13:29")
    ;; sort of like CODEARRAY at MAKEINIT time for allocating space on the file;  this code borrowed from CODEARRAY and \ALLOCBLOCK.
    ;; Returns ARLEN, which is then passed to FILEBLOCKTRAILER to set trailer length.
    (PROG (PREFIXLEN (ARLEN (IPLUS NCELLS \ArrayBlockOverheadCells)))
     ;; ARLEN is the number of cells in the array .  INITONPAGE is number of cells which must reside on same page
          (COND
             ([NEQ 0 (SETQ PREFIXLEN (\PREFIXALIGNMENT? ARLEN INITONPAGE CELLSPERQUAD CODEBLOCK.GCT (
                                                                                                FILEARRAYBASE
                                                                                                ]
              ;; Check page first, cause if we did segment first and succeeded but then failed on page, we would have to check segment again.
              (FILEPATCHBLOCK PREFIXLEN)))
          (BOUT16 OUTX \CodeArrayFlagWord)
          (BOUT16 OUTX ARLEN)
          (RETURN ARLEN])

(FILEPATCHBLOCK
  [LAMBDA (ARLEN)                                                (* rmk%: "18-NOV-82 09:50")
    ;; like \PATCHBLOCK for array allocation on files at MAKEINIT time
    (LOCAL (BOUT16 OUTX \FreeArrayFlagWord))                     ; in-use bit off, password set
    (LOCAL (BOUT16 OUTX ARLEN))                                  ; number of cells in this block
    [COND
       ((IGREATERP ARLEN \ArrayBlockHeaderCells)                 ; Assumes that header and trailer look alike, so that we only
                                                                 ; need one instance for a tiny block.
        (LOCAL (BOUTZEROS (UNFOLD (IDIFFERENCE ARLEN \ArrayBlockOverheadCells)
                                  BYTESPERCELL)))                ; zeros for data words
        (LOCAL (BOUT16 OUTX \FreeArrayFlagWord))                 ; Set up trailer
        (LOCAL (BOUT16 OUTX ARLEN]
    NIL])

)

;; Hunk Initialization

(DEFINEQ

(\SETUP.HUNK.TYPENUMBERS
  [LAMBDA NIL                                                    ; Edited  4-Mar-87 11:04 by bvm:

;;; Called before datatype table is initialized.  We add to the list of initial datatypes (\built-in-system-types) entries for all the hunk types we will want.

;;; Note: the compiler knows about the pointer hunk names, so it is important to coordinate any future changes to \HUNK.PTRSIZES with the compiler.

    (SETQ INITIALDTDCONTENTS (APPEND \BUILT-IN-SYSTEM-TYPES (\COMPUTE.HUNK.TYPEDECLS \HUNK.PTRSIZES
                                                                                     PTRBLOCK.GCT '\PTRHUNK)
                                     (\COMPUTE.HUNK.TYPEDECLS \HUNK.UNBOXEDSIZES UNBOXEDBLOCK.GCT
                                          '\UNBOXEDHUNK)
                                     (\COMPUTE.HUNK.TYPEDECLS \HUNK.CODESIZES CODEBLOCK.GCT '\CODEHUNK])

(\COMPUTE.HUNK.TYPEDECLS
  [LAMBDA (SIZELST GCTYPE PREFIX)                                ; Edited  4-Mar-87 11:03 by bvm:
    ;; Add type entries to INITIALDTDCONTENTS for the hunks in SIZELST of type GCTYPE.  PREFIX is the start of the name, e.g., \PTRHUNK.
    ;; Entries are of the form (name size ptrs finalization)
    (ALLOCAL (for HUNKSIZE in SIZELST BIND (FINAL _ (AND (EQ GCTYPE CODEBLOCK.GCT)
                                                         '\RECLAIMCODEBLOCK))
                until (> HUNKSIZE \MAX.CELLSPERHUNK) collect (LIST (PACK* PREFIX HUNKSIZE)
                                                                   (UNFOLD HUNKSIZE WORDSPERCELL)
                                                                   (COND
                                                                      ((EQ GCTYPE PTRBLOCK.GCT)
                                                                   ; Compute DTDPTRS list, i.e., which fields are pointers (all of
                                                                   ; them)
                                                                       (for I from 0 by 2
```

```
                                                    to (SUB1 (UNFOLD HUNKSIZE WORDSPERCELL))
                                                    collect I)))
                                            FINAL])
```

(\TURN.ON.HUNKING

```
  [LAMBDA NIL                                                    (* bvm%: "13-Jun-86 17:27")
```

;;; create all the datatypes, and the tables used to calculate a hunk datatype number from the allocation size request.

```
    (SETQ \UNBOXEDHUNK.TYPENUM.TABLE (\SETUP.TYPENUM.TABLE \HUNK.UNBOXEDSIZES UNBOXEDBLOCK.GCT '\UNBOXEDHUNK))
    (SETQ \CODEHUNK.TYPENUM.TABLE (\SETUP.TYPENUM.TABLE \HUNK.CODESIZES CODEBLOCK.GCT '\CODEHUNK))
    (SETQ \PTRHUNK.TYPENUM.TABLE (\SETUP.TYPENUM.TABLE \HUNK.PTRSIZES PTRBLOCK.GCT '\PTRHUNK))
    (SETQ \HUNKING? T])
```

(\SETUP.TYPENUM.TABLE

```
  [LAMBDA (SIZELST GCTYPE PREFIX)                                ; Edited  5-Mar-87 10:12 by bvm:
```

;;; Create a table that maps from number of cells desired to the closest hunk size that fits for a given GCTYPE. SIZELST is list of sizes in cells.
;;; PREFIX is the datatype name prefix for this kind of hunk.

```
    (for I from 0 to \MAX.CELLSPERHUNK bind (HUNKSIZE _ -1)
                                            (SIZEL _ SIZELST)
                                            (TABLE _ (\ALLOCBLOCK (FOLDHI (IPLUS 4 \MAX.CELLSPERHUNK)
                                                                          BYTESPERCELL)
                                                                  UNBOXEDBLOCK.GCT))
                                           TNAME DTD DTNUMBER
       do [COND
             ((IGREATERP I HUNKSIZE)                             ; Advance to next quantum range in the SIZELST
              (SETQ HUNKSIZE (OR (FIXP (pop SIZEL))
                                 \MAX.CELLSPERHUNK))
              (SETQ TNAME (PACK* PREFIX HUNKSIZE))
              (COND
                 ((for old DTNUMBER from 1 as TYPE in (LOCAL INITIALDTDCONTENTS)
                     when (EQ (LOCAL (CAR TYPE))
                              TNAME)
                     do ;; Find the type number that has been assigned to this hunk type.  Ordinarily would use \TYPENUMBERFROMNAME,
                        ;; but atoms haven't been initialized yet, so we can only talk locally
                        (RETURN DTNUMBER))
                  (SETQ DTD (\GETDTD DTNUMBER))
                  (replace DTDGCTYPE of DTD with GCTYPE)
                  (replace DTDHUNKP of DTD with T))
                 (T (HELP "No type declaration for" TNAME]
          (\PUTBASEBYTE TABLE I DTNUMBER)
       finally (RETURN TABLE])

)

(DECLARE%: DONTCOPY

(ADDTOVAR INITVALUES (\NxtArrayPage)
                     (\HUNKING?))

(ADDTOVAR INITPTRS (\FREEBLOCKBUCKETS)
                   (\ArrayFrLst)
                   (\ArrayFrLst2)
                   (\UNBOXEDHUNK.TYPENUM.TABLE)
                   (\CODEHUNK.TYPENUM.TABLE)
                   (\PTRHUNK.TYPENUM.TABLE))

(ADDTOVAR INEWCOMS (FNS \#BLOCKDATACELLS \PREFIXALIGNMENT? \ALLOCBLOCK \MAIKO.ALLOCBLOCK \ALLOCBLOCK.NEW
                        \MAKEFREEARRAYBLOCK \MERGEBACKWARD \LINKBLOCK \ALLOCHUNK)
                   (FNS PREINITARRAYS POSTINITARRAYS FILEARRAYBASE FILEBLOCKTRAILER FILECODEBLOCK
                        FILEPATCHBLOCK)
                   (FNS \SETUP.HUNK.TYPENUMBERS \COMPUTE.HUNK.TYPEDECLS \TURN.ON.HUNKING \SETUP.TYPENUM.TABLE))

(ADDTOVAR MKI.SUBFNS (\IN.MAKEINIT . T)
                     (\ALLOCBLOCK.OLD . NILL)
                     (\MERGEFORWARD . NILL)
                     (\FIXCODENUM . I.FIXUPNUM)
                     (\FIXCODESYM . I.FIXUPSYM)
                     (\FIXCODEPTR . I.FIXUPPTR)
                     (\CHECKARRAYBLOCK . NILL)
                     (\ARRAYMERGING PROGN NIL))

(ADDTOVAR EXPANDMACROFNS \ADDBASE2 \ADDBASE4 HUNKSIZEFROMNUMBER BUCKETINDEX FREEBLOCKCHAIN.N)

(ADDTOVAR RDCOMS (FNS \CHECKARRAYBLOCK \PARSEARRAYSPACE \PARSEARRAYSPACE1))

(ADDTOVAR RD.SUBFNS (EQPTR . EQUAL)
                    (ARRAYBLOCKCHECKING . T))

(ADDTOVAR RDPTRS (\FREEBLOCKBUCKETS))

(ADDTOVAR RDVALS (\ArrayFrLst)
```

```
                (\ArrayFrLst2))


(ADDTOVAR DONTCOMPILEFNS PREINITARRAYS POSTINITARRAYS FILEARRAYBASE FILEBLOCKTRAILER FILECODEBLOCK
                                FILEPATCHBLOCK)

(ADDTOVAR DONTCOMPILEFNS \SETUP.HUNK.TYPENUMBERS \COMPUTE.HUNK.TYPEDECLS \TURN.ON.HUNKING \SETUP.TYPENUM.TABLE
)
)
```

;; Debugging aids

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \ArrayFrLst)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \ArrayBlockPassword 5461)

(CONSTANTS \ArrayBlockPassword)
)

(ADDTOVAR DONTCOMPILEFNS \HUNKFIT? \AB.NEXT \AB.BACK)
)

(DEFINEQ
```

(\HUNKFIT?

```
  [LAMBDA (N)                                                    (* JonL "15-Jan-85 00:48")
    ;; Show how an MDS unit of 2 pages would accomodate chunks of size N cells.
    (printout NIL T "Hunk size = " N " cells,  " (IQUOTIENT (FOLDLO \MDSIncrement WORDSPERCELL)
                                                           N)
           " fit in a MDS unit with "
           (IREMAINDER (FOLDLO \MDSIncrement WORDSPERCELL)
                  N)
           " cells left over." T .TAB 8 "('unit' is split with " (IREMAINDER CELLSPERPAGE N)
           " cells kept on first page)" T)
    T])
```

(\AB.NEXT

```
  [LAMBDA (ABHI ABLO)                                            (* JonL "10-Sep-84 05:04")
    ;; ABHI and ABLO form the \HILOC and \LOLOC of some arrayblock which we want to 'go' to the predecessor of;  alternatively, ABHI can be a list
    ;; of these two address parts, or just a random arrayblock address.

    ;; Returns a 4-list;  size of the next block, whether or not it is free, and the \HILOC and the \LOLOC of that block

    [COND
       [(AND (LISTP ABHI)
             (NULL ABLO))
        [COND
           ((AND (EQ 4 (LENGTH ABHI))
                 (FIXP (CAR ABHI))
                 (SELECTQ (CADR ABHI)
                      ((INUSE FREE)
                           T)
                      NIL))                                      ; Result is output of \AB.NEXT itself
            (SETQ ABHI (CDDR ABHI]
        (COND
           ((EQ 2 (LENGTH ABHI))                                 ; A 2-list of \HILOC and \LOLOC
            (SETQ ABLO (CADR ABHI))
            (SETQ ABHI (CAR ABHI]
       ((OR (EQ ABHI \ArrayFrLst)
            (type? ARRAYBLOCK ABHI))
        (SETQ ABLO (\LOLOC ABHI))
        (SETQ ABHI (\HILOC ABHI]
    (OR (IGEQ ABHI 0)
        (ERROR "Negative segment number?" ABHI))
    (AND (IGREATERP ABHI (\HILOC \ArrayFrLst))
         (ERROR "Segment number too high?" ABHI))
    (OR (IGEQ ABLO 0)
        (ERROR "Negative offset number?" ABLO))
    (PROG (PW SIZE SIZE.WORDS (ABADDR (\VAG2 ABHI ABLO)))
          [PROGN                                                 ; Checking on current block
                 (SETQ PW (\GETBASE ABADDR 0))
                 [COND
                    ((NEQ \ArrayBlockPassword (LOADBYTE PW 3 13))
                     (SETQ ABADDR)
                     (ERROR "Array Password not found at this loc" (LIST ABHI ABLO]
                 (SETQ SIZE.WORDS (UNFOLD (SETQ SIZE (\GETBASE ABADDR 1))
                                          WORDSPERCELL))
                 (COND
```

```
                    [(NEQ \ArrayBlockPassword (LOADBYTE (\GETBASE ABADDR (IDIFFERENCE SIZE.WORDS 2))
                                                        3 13))
                     (ERROR "Array Password not found just below this" (PROG1 (LIST ABHI ABLO)
                                                                               (SETQ ABADDR]
                    ((NEQ SIZE (\GETBASE ABADDR (IDIFFERENCE SIZE.WORDS 1)))
                     (ERROR "Header and Trailer lengths disagree" (PROG1 (LIST ABHI ABLO)
                                                                         (SETQ ABADDR]
          (SETQ ABADDR (\ADDBASE ABADDR SIZE.WORDS))
          (SETQ PW (\GETBASE ABADDR 0))
          [COND
             ((NEQ \ArrayBlockPassword (LOADBYTE PW 3 13))
              (SETQ ABADDR)
              (ERROR "Array Password not found at this loc" (LIST ABHI ABLO]
          (RETURN (LIST (\GETBASE ABADDR 1)
                        (COND
                           ((ODDP PW)
                            'INUSE)
                           (T 'FREE))
                        (\HILOC ABADDR)
                        (\LOLOC ABADDR])
```

(**AB.BACK**

```
  [LAMBDA (ABHI ABLO)                                                    (* JonL " 9-Sep-84 16:28")
```

;; ABHI and ABLO form the \HILOC and \LOLOC of some arrayblock which we want to 'go' to the predecessor of; alternatively, ABHI can be a list
;; of these two address parts, or just a random arrayblock address.

;; Returns a 4-list; size of the block we are starting from, whether or not it is free, and the \HILOC and the \LOLOC of the predecessor block

```
    [COND
       [(AND (LISTP ABHI)
             (NULL ABLO))
        [COND
           ((AND (EQ 4 (LENGTH ABHI))
                 (FIXP (CAR ABHI))
                 (SELECTQ (CADR ABHI)
                      ((INUSE FREE)
                        T)
                      NIL))                                              ; Result is output of \AB.BACK itself
            (SETQ ABHI (CDDR ABHI]
        (COND
           ((EQ 2 (LENGTH ABHI))                                         ; A 2-list of \HILOC and \LOLOC
            (SETQ ABLO (CADR ABHI))
            (SETQ ABHI (CAR ABHI]
       ((OR (EQ ABHI \ArrayFrLst)
            (type? ARRAYBLOCK ABHI))
        (SETQ ABLO (\LOLOC ABHI))
        (SETQ ABHI (\HILOC ABHI]
    (OR (IGEQ ABHI 0)
        (ERROR "Negative segment number?" ABHI))
    (AND (IGREATERP ABHI (\HILOC \ArrayFrLst))
         (ERROR "Segment number too high?" ABHI))
    (OR (IGEQ ABLO 0)
        (ERROR "Negative offset number?" ABLO))
    (PROG (PW SIZE (ABADDR (\ADDBASE (\VAG2 ABHI ABLO)
                                     -2)))
          (SETQ PW (\GETBASE ABADDR 0))
          [COND
             ((NEQ \ArrayBlockPassword (LOADBYTE PW 3 13))
              (SETQ ABADDR)
              (ERROR "Array Password not found just below this" (LIST ABHI ABLO]
          (SETQ SIZE (\GETBASE ABADDR 1))
          [SETQ ABADDR (\ADDBASE ABADDR (IMINUS (UNFOLD (SUB1 SIZE)
                                                        WORDSPERCELL]
          [COND
             [(NEQ \ArrayBlockPassword (LOADBYTE (\GETBASE ABADDR 0)
                                                 3 13))
              (ERROR "Array Password not found just below this" (PROG1 (LIST ABHI ABLO)
                                                                       (SETQ ABADDR]
             ((NEQ SIZE (\GETBASE ABADDR 1))
              (ERROR "Header and Trailer lengths disagree" (PROG1 (LIST ABHI ABLO)
                                                                  (SETQ ABADDR]
          (RETURN (LIST SIZE (COND
                                ((ODDP PW)
                                 'INUSE)
                                (T 'FREE))
                        (\HILOC ABADDR)
                        (\LOLOC ABADDR])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

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

```
(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA CL::PUTHASH HARRAYPROP)
)

(PUTPROPS LLARRAYELT COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
                                1992 1993))
```

FUNCTION INDEX

CONSTANT INDEX

VARIABLE INDEX

MACRO INDEX

RECORD INDEX

PROPERTY INDEX