```
    File created:    11-Jun-90 16:23:11  {DSK}<usr>local>lde>lispcore>library>HASH.;2

      changes to:    (VARS HASHCOMS)

   previous date:     1-Nov-86 23:03:01  {DSK}<usr>local>lde>lispcore>library>HASH.;1

      Read Table:    INTERLISP

         Package:    INTERLISP

          Format:     XCCS
```

```
(RPAQQ HASHCOMS
       ((COMS                                                   ; User Functions
              (FNS CLEARHASHFILES CLOSEHASHFILE COLLECTKEYS COPYHASHFILE COPYHASHITEM CREATEHASHFILE GETHASHFILE
                   GETHASHTEXT HASHBEFORECLOSE HASHFILEDATA HASHFILENAME HASHFILEP HASHFILEPROP HASHFILESPLST
                   LOOKUPHASHFILE MAPHASHFILE OPENHASHFILE PUTHASHFILE PUTHASHTEXT REHASHFILE))
        (COMS                                                   ; Internal Functions
              (FNS DELETEHASHKEY FIND1STPRIME GETHASHKEY GETPROBE GTHASHFILE HASHFILESPLST1 INSERTHASHKEY
                   MAKEHASHKEY REPLACEHASHKEY SETHASHSTATUS SPLITKEY))
        (COMS                                                   ; System Variables
              (INITVARS (HFGROWTHFACTOR 3)
                     (HASHLOADFACTOR 0.875)
                     (HASHFILEDEFAULTSIZE 512)
                     (HASHSCRATCHCONSCELL (CONS))
                     (HASHTEXTCHAR (CHARACTER (CHARCODE ^A)))
                     (HASHFILERDTBL (COPYREADTABLE 'ORIG))
                     (HASHSCRATCHLST (CONSTANT (to 40 collect NIL)))
                     (HASHBITTABLE (MAKEBITTABLE (LIST HASHTEXTCHAR)))
                     (REHASHGAG T)
                     SYSHASHFILE SYSHASHFILELST)
              (VARS PROBELST HASHACCESSTYPES)
              (ADDVARS (AFTERSYSOUTFORMS (CLEARHASHFILES)))
              (OPTIMIZERS GETHASHFILE HASHFILENAME))
        [COMS                                                   ; System Macros
              (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS ANYEQ CREATEKEY PRINTPTR PRINTSTBYTE READPTR READSTBYTE
                                                      REHASHKEY)

                     ;; etc.

                     (RECORDS HashFile HashTextPtr HashFileEntry DoubleKey)
                     (CONSTANTS (HASH.HEADER.SIZE 8)
                            (HASH.KEY.SIZE 4))
                     (GLOBALVARS HFGROWTHFACTOR HASHLOADFACTOR HASHFILEDEFAULTSIZE HASHSCRATCHCONSCELL
                            HASHTEXTCHAR HASHSCRATCHLST HASHBITTABLE SYSHASHFILE SYSHASHFILELST PROBELST
                            HASHACCESSTYPES HASHFILERDTBL MAX.INTEGER)
                                                                ; For MASTERSCOPE
                     (GLOBALVARS HASH.HEADER.SIZE HASH.KEY.SIZE)
                     (SPECVARS REHASHGAG)
                     (BLOCKS (LOOKUPHASHFILEBLOCK (ENTRIES LOOKUPHASHFILE GETHASHFILE PUTHASHFILE)
                                    LOOKUPHASHFILE GETHASHFILE PUTHASHFILE DELETEHASHKEY GETHASHKEY GETPROBE
                                    INSERTHASHKEY MAKEHASHKEY REPLACEHASHKEY)
                            (OPENHASHFILEBLOCK (ENTRIES CREATEHASHFILE OPENHASHFILE)
                                   CREATEHASHFILE OPENHASHFILE FIND1STPRIME SETHASHSTATUS)
                            (MAPHASHFILEBLOCK (ENTRIES COLLECTKEYS COPYHASHFILE COPYHASHITEM HASHFILESPLST
                                                     MAPHASHFILE REHASHFILE)
                                   (SPECVARS REHASHGAG)
                                   COLLECTKEYS COPYHASHFILE COPYHASHITEM HASHFILESPLST HASHFILESPLST1
                                   MAPHASHFILE REHASHFILE SPLITKEY]
        (PROP FILETYPE HASH)))
```

;; User Functions

```
(DEFINEQ

(CLEARHASHFILES
  [LAMBDA (CLOSE RELEASE)                                       (* cdl "21-May-86 19:55")

    ;; Called after SYSOUT returns, to clean up any spurious items.  Can also be called to close all hashfiles.

    (if CLOSE
        then [while SYSHASHFILELST do                           ; Do it this way, so the DREMOVE in HASHBEFORECLOSE
                                                                ; doesn't screw up this iteration
                                      (with HashFileEntry (pop SYSHASHFILELST)
                                            (with HashFile HASHFILE (CLOSEF? Stream)
                                                  (SETQ Valid? NIL]; Invalidate anything that was open before the sysout
             (SETQ SYSHASHFILE NIL])

(CLOSEHASHFILE
  [LAMBDA (HASHFILE REOPEN)                                     (* cdl "21-May-86 08:18")
    (if (SETQ HASHFILE (HASHFILEP (OR HASHFILE SYSHASHFILE)))
        then (with HashFile HASHFILE (SETQ File (CLOSEF? Stream))
                   (if REOPEN
                       then                                     ; This flag forces contents of file to exist on disk if we crash,
                                                             ; reuse hashfile datum
                        (OPENHASHFILE File REOPEN NIL NIL HASHFILE)
                   else File])

(COLLECTKEYS
  [LAMBDA (HASHFILE DOUBLE MKSTRING?)                        (* cdl "14-Mar-85 17:01")
    (DECLARE (SPECVARS MKSTRING?))
    (PROG (KEYLST)
          (DECLARE (SPECVARS KEYLST))
          [if DOUBLE
              then (MAPHASHFILE HASHFILE [FUNCTION (LAMBDA (KEY1 KEY2)
                                                   (push KEYLST (CONS (if MKSTRING?
                                                                          then (MKSTRING KEY1)
                                                                        else KEY1)
                                                                      (if MKSTRING?
                                                                          then (MKSTRING KEY2)
                                                                        else KEY2]
                          T)
            else (MAPHASHFILE HASHFILE (FUNCTION (LAMBDA (KEY)
                                                 (push KEYLST (if MKSTRING?
                                                                  then (MKSTRING KEY)
                                                                else KEY]
          (RETURN KEYLST])

(COPYHASHFILE
  [LAMBDA (HASHFILE NEWNAME FN VALUETYPE LEAVEOPEN)          (* cdl "18-Mar-85 09:01")
    (DECLARE (SPECVARS HASHFILE FN))                         ; Copy HashFile by mapping over file hashing items into new file,
                                                             ; slow but lisp independent
    (with HashFile (SETQ HASHFILE (GTHASHFILE HASHFILE))
         (PROG ((ACCESS (HASHFILEPROP HASHFILE 'ACCESS))
                (NEWHASHFILE (CREATEHASHFILE NEWNAME (OR VALUETYPE ValueType)
                                    ItemLength %#Entries NIL ItemCopyFn)))
               (DECLARE (SPECVARS NEWHASHFILE))
               (if (NEQ ACCESS 'INPUT)
                   then                                      ; Close and reopen the hashfile to make sure it is up to date on
                                                             ; the disk
                        (SETQ HASHFILE (CLOSEHASHFILE HASHFILE ACCESS)))
               [MAPHASHFILE HASHFILE (FUNCTION (LAMBDA (KEY)
                                               (COPYHASHITEM KEY HASHFILE NEWHASHFILE FN]
               (RETURN (if (NOT LEAVEOPEN)
                           then (CLOSEHASHFILE NEWHASHFILE)
                         else NEWHASHFILE])

(COPYHASHITEM
  [LAMBDA (KEY HASHFILE NEWHASHFILE USERFN)                  (* cdl "21-May-86 08:18")

    ;; Copy single hash item from old to new hashfile, applying userfn if supplied

    (PROG ((VALUE (GETHASHFILE KEY HASHFILE)))
          (if USERFN
              then (SETQ VALUE (APPLY* USERFN KEY VALUE HASHFILE NEWHASHFILE)))
          (if (type? HashTextPtr VALUE)
              then (with HashTextPtr VALUE (with HashFile HASHFILE (PUTHASHTEXT KEY Stream NEWHASHFILE Start End
                                                                          )))
            else (LOOKUPHASHFILE KEY VALUE NEWHASHFILE 'INSERT])

(CREATEHASHFILE
  [LAMBDA (FILE VALUETYPE ITEMLENGTH %#ENTRIES SMASH COPYFN)  (* cdl "21-May-86 09:32")
    (PROG (STREAM SIZE HASHFILE)
          [SETQ SIZE (FIND1STPRIME (FIX (FTIMES (if %#ENTRIES
                                                    then (MAX %#ENTRIES HASHFILEDEFAULTSIZE)
                                                  else HASHFILEDEFAULTSIZE)
                                                HFGROWTHFACTOR]
          [SETQ STREAM (OPENSTREAM FILE 'OUTPUT 'NEW 8 '((TYPE BINARY]
          (PRINTPTR STREAM 0)
          (PRINTPTR STREAM SIZE)                             (* Put other arguments on file for future expansion)
          [BOUT STREAM (SELECTQ VALUETYPE
                           (TEXT (CHARCODE T))
                           (EXPR (CHARCODE E))
                           (PROGN (SETQ VALUETYPE 'EXPR)
                                  (CHARCODE E]
          (BOUT STREAM (SETQ ITEMLENGTH (if (NUMBERP ITEMLENGTH)
                                            then (LOGAND ITEMLENGTH 255)
                                          else 0)))          (* Fill the KEY section with zeros and mark end of KEYS, start of
      DATA)
          (to (ADD1 (ITIMES SIZE HASH.KEY.SIZE)) do (BOUT STREAM 0))
                                                             (* Close file and reopen to ensure existance)
          [SELECTQ (SYSTEMTYPE)
              ((TENEX TOPS20)
                   (SETQ FILE (CLOSEF (with STREAM STREAM FULLNAME))))
              (PROGN (SETQ FILE (CLOSEF STREAM]
          (with HashFile (SETQ HASHFILE (if (type? HashFile SMASH)
                                            then SMASH
                                          else (create HashFile)))
                [SETQ ByteStream (OPENSTREAM FILE 'BOTH 'OLD 8 '((TYPE BINARY]
                [SELECTQ (SYSTEMTYPE)
                    ((TENEX TOPS20)
                        (SETQ File (SETQ Stream (with STREAM ByteStream FULLNAME))))
                    (SETQ File (FULLNAME (SETQ Stream ByteStream]
                (SETQ Size SIZE)
                (SETQ %#Entries 0)
                (SETQ Write? T)
                (SETQ ValueType VALUETYPE)
                (SETQ ItemCopyFn COPYFN)
                (SETQ ItemLength ITEMLENGTH))
           (RETURN (SETHASHSTATUS HASHFILE])
```

```
(GETHASHFILE
  [LAMBDA (KEY HASHFILE KEY2)                                (* cdl " 3-Aug-83 15:04")
    (LOOKUPHASHFILE (CREATEKEY KEY KEY2)
           NIL HASHFILE 'RETRIEVE])
```

```
(GETHASHTEXT
  [LAMBDA (KEY HASHFILE DSTFIL)                              (* cdl "21-May-86 08:19")
    (PROG ((HASHTEXTPTR (GETHASHFILE KEY HASHFILE)))
          (if (type? HashTextPtr HASHTEXTPTR)
              then (with HashTextPtr HASHTEXTPTR (with HashFile HASHFILE (RETURN (COPYBYTES Stream DSTFIL Start
                                                                                    End])
```

```
(HASHBEFORECLOSE
  [LAMBDA (FILE)                                             (* cdl "18-Mar-85 10:27")
                                                             (* Called before a hashfile is actually closed)
    (PROG ((ENTRY (ASSOC (FULLNAME FILE)
                         SYSHASHFILELST)))
          (if ENTRY
              then (with HashFileEntry ENTRY (if (EQ HASHFILE SYSHASHFILE)
                                                 then (SETQ SYSHASHFILE NIL))
                                                             (* Mark this datum defunct)
                         (with HashFile HASHFILE (SETQ Valid? NIL)))
                                                             (* Remove from table of open hash files)
                   (SETQ SYSHASHFILELST (DREMOVE ENTRY SYSHASHFILELST])
```

```
(HASHFILEDATA
  [LAMBDA (HASHFILE)                                         (* cdl "22-Aug-83 12:12")
    (with HashFile (GTHASHFILE HASHFILE)
         (LIST File ValueType ItemLength %#Entries])
```

```
(HASHFILENAME
  [LAMBDA (HASHFILE)                                         (* gbn " 7-Nov-84 16:34")
    (HASHFILEPROP HASHFILE 'NAME])
```

```
(HASHFILEP
  [LAMBDA (HASHFILE WRITE)                                   (* cdl "18-Mar-85 10:52")
    (if [AND [OR (type? HashFile HASHFILE)
                 (AND HASHFILE (LITATOM HASHFILE)
                      (SETQ HASHFILE (FULLNAME HASHFILE))
                      (SETQ HASHFILE (CDR (ASSOC HASHFILE SYSHASHFILELST]
             (with HashFile HASHFILE (AND Valid? (OR (NOT WRITE)
                                                     Write?]
        then HASHFILE])
```

```
(HASHFILEPROP
  [LAMBDA (HASHFILE PROP VALUE)                              (* cdl "21-May-86 09:43")
    (with HashFile (GTHASHFILE HASHFILE)
         (SELECTQ PROP
              (VALUETYPE ValueType)
              (ACCESS (GETFILEINFO Stream 'ACCESS))
              (NAME File)
              (COPYFN (PROG1 ItemCopyFn
                             (if VALUE
                                 then (SETQ ItemCopyFn VALUE))))
              (STREAM Stream)
              (SIZE Size)
              (%#ENTRIES %#Entries)
              (ITEMLENGTH ItemLength)
              NIL])
```

```
(HASHFILESPLST
  [LAMBDA (HASHFILE XWORD)                                   (* cdl "15-Mar-85 08:51")
    (DECLARE (SPECVARS . T))                                 (* Just create an Interlisp generator that returns each hash key)
    (if (SETQ HASHFILE (GTHASHFILE HASHFILE))
        then (GENERATOR (HASHFILESPLST1 HASHFILE XWORD])
(LOOKUPHASHFILE
  [LAMBDA (KEY VALUE HASHFILE CALLTYPE KEY2)                    (* Pavel "24-Sep-86 12:31")
    (PROG (RETVAL RETFLG (KEYVAL MAX.INTEGER)
                (INDEX (CREATEKEY KEY KEY2)))
          (SETQ HASHFILE (GTHASHFILE HASHFILE (ANYEQ '(REPLACE DELETE INSERT)
                                                     CALLTYPE)))
          (SETQ KEYVAL (GETHASHKEY INDEX HASHFILE (EQMEMB 'INSERT CALLTYPE)
                             KEYVAL))
          (COND
             ((MINUSP KEYVAL)
              (if (EQMEMB 'INSERT CALLTYPE)
                  then (INSERTHASHKEY (SETQ KEYVAL (IMINUS KEYVAL))
                             INDEX VALUE HASHFILE)))
             (T (if (EQMEMB 'RETRIEVE CALLTYPE)
                    then (SETQ RETFLG T)
                         (SETQ RETVAL (READ (fetch Stream of HASHFILE)
                                            HASHFILERDTBL)))
                (if (EQMEMB 'REPLACE CALLTYPE)
                    then (REPLACEHASHKEY KEYVAL INDEX VALUE HASHFILE)
                  elseif (EQMEMB 'DELETE CALLTYPE)
                    then (DELETEHASHKEY KEYVAL HASHFILE))
                (RETURN (if RETFLG
                            then RETVAL
                          elseif KEYVAL
                            then T])

(MAPHASHFILE
  [LAMBDA (HASHFILE MAPFN DOUBLE)                               (* Pavel "24-Sep-86 12:30")
    (with HashFile (SETQ HASHFILE (GTHASHFILE HASHFILE))
          (bind KEY VALUE HASHKEY (BOTH _ (IGREATERP (OR (NARGS MAPFN)
                                                         0)
                                                     (if DOUBLE
                                                         then 2
                                                       else 1)))
             to Size as ADR from HASH.HEADER.SIZE by HASH.KEY.SIZE when (PROGN (SETFILEPTR Stream ADR)
                                                                            (READSTBYTE ByteStream
                                                                                   'USED))
             do (SETQ HASHKEY (READPTR ByteStream))
                (SETFILEPTR Stream HASHKEY)
                (SETQ KEY (READ Stream HASHFILERDTBL))
                (if BOTH
                    then (SETQ VALUE (READ Stream HASHFILERDTBL)))
                (if DOUBLE
                    then                                        ; Two key hashing so split up key, userfn takes two key
                                                                ; arguments
                         (with DoubleKey (SPLITKEY KEY)
                               (APPLY* MAPFN Key1 Key2 VALUE))
                  else (APPLY* MAPFN KEY VALUE])

(OPENHASHFILE
  [LAMBDA (FILE ACCESS ITEMLENGTH %#ENTRIES SMASH)              (* cdl "21-May-86 11:30")
    [SETQ ACCESS (for ENTRY in HASHACCESSTYPES thereis (MEMB ACCESS ENTRY) finally (RETURN (CAR ENTRY]
    (if (OR ITEMLENGTH %#ENTRIES (EQ ACCESS 'CREATE))
        then                                                    (* This is really a createhashfile call, the original hash package
                                                                used openhashfile for both)
             (CREATEHASHFILE FILE NIL ITEMLENGTH %#ENTRIES SMASH)
      else (PROG [(HASHFILE (CDR (ASSOC (FULLNAME FILE)
                                        SYSHASHFILELST]
                 [if HASHFILE
                     then (with HashFile HASHFILE (if (EQ ACCESS (GETFILEINFO Stream 'ACCESS))
                                                      then      (* This is the NO-OP case)
                                                            (RETURN HASHFILE]
                 [with HashFile (SETQ HASHFILE (if (type? HashFile SMASH)
                                                   then SMASH
                                                 else (create HashFile)))
                       [SETQ ByteStream (OPENSTREAM FILE ACCESS 'OLD 8 '((TYPE BINARY]
                       (SETQ %#Entries (READPTR ByteStream))
                       (SETQ Size (READPTR ByteStream))
                       (SETQ ValueType (SELCHARQ (BIN ByteStream)
                                            (T 'TEXT)
                                            (E 'EXPR)
                                            'EXPR))
                       (SETQ ItemLength (BIN ByteStream))
                       (SETQ Write? (EQ ACCESS 'BOTH))
                       (SELECTQ (SYSTEMTYPE)
                           ((TENEX TOPS20)
                                (SETQ File (SETQ Stream (with STREAM ByteStream FULLNAME))))
                           (SETQ File (FULLNAME (SETQ Stream ByteStream]
                 (RETURN (SETHASHSTATUS HASHFILE])

(PUTHASHFILE
  [LAMBDA (KEY VALUE HASHFILE KEY2)                             (* cdl "15-Mar-85 08:55")
    (LOOKUPHASHFILE (CREATEKEY KEY KEY2)
           VALUE HASHFILE (if VALUE
                               then '(REPLACE INSERT)
                             else 'DELETE))
    VALUE])


(PUTHASHTEXT
  [LAMBDA (KEY SRCFIL HASHFILE START END)                     (* cdl "21-May-86 08:54")
    (SETQ HASHFILE (GTHASHFILE HASHFILE T))
    (PROG (HASHTEXTPTR)
          [with HashFile HASHFILE (SETFILEPTR Stream -1)
               (with HashTextPtr (SETQ HASHTEXTPTR (create HashTextPtr
                                                          Start _ (GETEOFPTR Stream)))
                    (COPYBYTES SRCFIL Stream START END)
                    (SETQ End (GETEOFPTR Stream]
          (RETURN (PUTHASHFILE KEY HASHTEXTPTR HASHFILE])


(REHASHFILE
  [LAMBDA (HASHFILE NEWNAME VALUETYPE)                        (* cdl "21-May-86 08:23")
    (SETQ HASHFILE (GTHASHFILE HASHFILE))
    (PROG [[NAME (OR NEWNAME (PACKFILENAME 'VERSION NIL 'BODY (HASHFILENAME HASHFILE]
           (ACCESS (HASHFILEPROP HASHFILE 'ACCESS]            (* If rehashgag = T then print out old and new file)
          [with HashFile HASHFILE (if (NOT REHASHGAG)
                                      then (printout NIL "Rehashing" %, File " ... "))
               (SETQ NAME (COPYHASHFILE HASHFILE NAME ItemCopyFn (OR VALUETYPE ValueType]
          (CLOSEHASHFILE HASHFILE)
          (with HashFile (OPENHASHFILE NAME ACCESS NIL NIL HASHFILE)
               (if (NOT REHASHGAG)
                   then (printout NIL File T)))
          (RETURN HASHFILE])

)
```

;; Internal Functions

```
(DEFINEQ

(DELETEHASHKEY
  [LAMBDA (HASHKEY HASHFILE)                                  (* cdl "21-May-86 19:57")
    (with HashFile HASHFILE (SETFILEPTR Stream 0)
          (PRINTPTR ByteStream (SETQ %#Entries (SUB1 %#Entries)))
          (SETFILEPTR Stream HASHKEY)
          (PRINTSTBYTE ByteStream 'DELETED)
          (FORCEOUTPUT Stream])


(FIND1STPRIME
  [LAMBDA (N)                                                 (* cdl "11-Aug-83 08:12")
    (find P from (LOGOR N 1) by 2 suchthat (for I from 3 by 2 never (AND (ILESSP I P)
                                                                         (ZEROP (IREMAINDER P I)))
                                              repeatuntil (ILESSP P (ITIMES I I])


(GETHASHKEY
  [LAMBDA (INDEX HASHFILE DELOK? HASHKEY)                     (* Pavel "24-Sep-86 12:30")
    (with HashFile HASHFILE (bind PROBE DELETED? first (SETQ HASHKEY (MAKEHASHKEY INDEX Size))
                                                        (SETFILEPTR Stream HASHKEY)
                               until (SELCHARQ (BIN ByteStream)
                                         (D (SETQ DELETED? T)
                                            DELOK?)
                                         (NULL 'FREE)
                                         NIL)
                               do (if DELETED?
                                      then (SETQ DELETED? NIL)
                                    else (SETFILEPTR Stream (READPTR ByteStream))
                                         (if (EQUAL INDEX (READ Stream HASHFILERDTBL))
                                             then (RETURN HASHKEY)))
                                  (if (NULL PROBE)
                                      then (SETQ PROBE (GETPROBE INDEX)))
                                  (SETQ HASHKEY (REHASHKEY HASHKEY PROBE Size))
                                  (SETFILEPTR Stream HASHKEY)
                               finally (RETURN (SETQ HASHKEY (IMINUS HASHKEY])


(GETPROBE
  [LAMBDA (KEY)                                               (* cdl "15-Mar-85 09:06")
                                                              (* Get the value to probe by. Probelst contains all the probe
                                                              primes.)
    (CAR (FNTH PROBELST (ADD1 (LOGAND 31 (NTHCHARCODE KEY (ADD1 (LRSH (NCHARS KEY)
                                                                      1])


(GTHASHFILE
  [LAMBDA (HASHFILE WRITE)                                    (* cdl "18-Mar-85 09:55")
    (if (NULL HASHFILE)
        then (SETQ HASHFILE SYSHASHFILE))
        ;; Return hashfile datum for HF, which is a filename or a hashfile datum.  Special cases: if HASHFILE is a filename which is not open, it is opened;
        ;; if HASHFILE is an invalidated hashfile datum (because it was closed), it is reopened;  if HASHFILE is already open for read, but WRITE is set,
        ;; will attempt to close and then open for write

        (if (HASHFILEP HASHFILE WRITE)
            then HASHFILE
          elseif (type? HashFile HASHFILE)
            then (OPENHASHFILE (fetch File of HASHFILE)
                        WRITE NIL NIL HASHFILE)
          elseif (LITATOM HASHFILE)
            then (OPENHASHFILE HASHFILE WRITE)
          else (HELP HASHFILE "NOT A HASHFILE"])

(HASHFILESPLST1
  [LAMBDA (HASHFILE XWORD)                                    (* cdl "15-Mar-85 09:10")
    (DECLARE (SPECVARS XWORD))
    (MAPHASHFILE HASHFILE (FUNCTION (LAMBDA (KEY)
                                      (if (OR (NULL XWORD)
                                              (STRPOS XWORD KEY 1 NIL T))
                                          then (PRODUCE KEY])

(INSERTHASHKEY
  [LAMBDA (HASHKEY INDEX VALUE HASHFILE)                      (* cdl "21-May-86 09:33")
    (with HashFile HASHFILE (if (GREATERP %#Entries (TIMES Size HASHLOADFACTOR))
                                then (REHASHFILE HASHFILE))
          (SETFILEPTR Stream 0)
          (SETQ %#Entries (ADD1 %#Entries))
          (PRINTPTR ByteStream %#Entries)
          (REPLACEHASHKEY HASHKEY INDEX VALUE HASHFILE])

(MAKEHASHKEY
  [LAMBDA (KEY RANGE)                                         (* cdl "21-May-86 11:28")
    (IPLUS HASH.HEADER.SIZE (ITIMES (for CHARCODE in (DCHCON KEY HASHSCRATCHLST) bind (INDEX _ 1)
                                       do (SETQ INDEX (IMOD (ITIMES INDEX CHARCODE)
                                                            RANGE))
                                       finally (RETURN INDEX))
                                   HASH.KEY.SIZE])

(REPLACEHASHKEY
  [LAMBDA (HASHKEY INDEX VALUE HASHFILE)                      (* bvm%: " 1-Nov-86 22:28")
    (with HashFile HASHFILE (SETFILEPTR Stream HASHKEY)
          (PRINTSTBYTE ByteStream 'USED)
          (PRINTPTR ByteStream (GETEOFPTR Stream))
          (SETFILEPTR Stream -1)
          (PRIN2 INDEX Stream HASHFILERDTBL)
          (SPACES 1 Stream)
          (PRINT VALUE Stream HASHFILERDTBL)
          (FORCEOUTPUT Stream])

(SETHASHSTATUS
  [LAMBDA (HASHFILE)                                          (* cdl "21-May-86 09:13")
    (with HashFile HASHFILE

           (* Fix data structures to know about this file so they get updated when it closes)

          (WHENCLOSE Stream 'BEFORE (FUNCTION HASHBEFORECLOSE))
          (SETQ Valid? T)
          (push SYSHASHFILELST (CONS File HASHFILE)))
    (SETQ SYSHASHFILE HASHFILE])

(SPLITKEY
  [LAMBDA (KEY)                                               (* cdl "14-Mar-85 16:55")
    (PROG ((PTR (STRPOSL HASHBITTABLE KEY)))
          (RETURN (if PTR
                      then (FRPLNODE HASHSCRATCHCONSCELL (SUBATOM KEY 1 (SUB1 PTR))
                                  (SUBATOM KEY (ADD1 PTR)))
                    else (FRPLNODE HASHSCRATCHCONSCELL KEY NIL])
)

;; System Variables

(RPAQ? HFGROWTHFACTOR 3)

(RPAQ? HASHLOADFACTOR 0.875)

(RPAQ? HASHFILEDEFAULTSIZE 512)

(RPAQ? HASHSCRATCHCONSCELL (CONS))

(RPAQ? HASHTEXTCHAR (CHARACTER (CHARCODE ^A)))
```

```
(RPAQ? HASHFILERDTBL (COPYREADTABLE 'ORIG))

(RPAQ? HASHSCRATCHLST (CONSTANT (to 40 collect NIL)))

(RPAQ? HASHBITTABLE (MAKEBITTABLE (LIST HASHTEXTCHAR)))

(RPAQ? REHASHGAG T)

(RPAQ? SYSHASHFILE NIL)

(RPAQ? SYSHASHFILELST NIL)

(RPAQQ PROBELST
       (1 3 5 7 11 11 13 17 17 19 23 23 29 29 29 31 37 37 37 41 41 43 47 47 53 53 53 59 59 59 61 67))

(RPAQQ HASHACCESSTYPES ((INPUT READ OLD NIL RETRIEVE)
                        (BOTH WRITE OUTPUT T INSERT DELETE REPLACE)
                        (CREATE DOUBLE NUMBER STRING PRINT FULLPRINT)))

(ADDTOVAR AFTERSYSOUTFORMS (CLEARHASHFILES))


(DEFOPTIMIZER GETHASHFILE (&REST X)
                          [if (CADDR X)
                              then 'IGNOREMACRO
                            else `(LOOKUPHASHFILE ,(CAR X)
                                         NIL
                                         ,(CADR X)
                                         'RETRIEVE])


(DEFOPTIMIZER HASHFILENAME (HASHFILE)
                           `(HASHFILEPROP ,HASHFILE 'NAME))
```

;; System Macros

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS ANYEQ MACRO [LAMBDA (X Y)
                        (for Z in X thereis (EQMEMB Z Y])

(PUTPROPS CREATEKEY MACRO [LAMBDA (KEY1 KEY2)
                            (if (NULL KEY2)
                                then KEY1
                              else (PACK* KEY1 HASHTEXTCHAR KEY2])

(PUTPROPS PRINTPTR MACRO
          [X
           `(PROGN
             ,@(for I from 2 to 0 by -1
                    collect `(BOUT ,(CAR X)
                                   (LOGAND 255 ,(if (ZEROP I)
                                                    then (CADR X)
                                                  else `(RSH ,(CADR X)
                                                             ,(ITIMES 8 I])

(PUTPROPS PRINTSTBYTE MACRO [X `(BOUT ,(CAR X)
                                      ,(SELECTQ (CADR (CADR X))
                                           ((U USED)
                                                (CHARCODE U))
                                           ((D DELETED)
                                                (CHARCODE D))
                                           ((F FREE)
                                                (CHARCODE F))
                                           NIL])

(PUTPROPS READPTR MACRO
          [X `(IPLUS ,@(for I from 2 to 0 by -1 collect (if (ZEROP I)
                                                            then `(BIN ,(CAR X))
                                                          else `(LLSH (BIN ,(CAR X))
                                                                      ,(ITIMES 8 I])

(PUTPROPS READSTBYTE MACRO [X `(EQ (BIN ,(CAR X))
                                   (CHARCODE ,(SELECTQ (CADR (CADR X))
                                                  (FREE 'NULL)
                                                  (USED 'U)
                                                  (DELETED 'D)
                                                  NIL])

(PUTPROPS REHASHKEY MACRO [LAMBDA (HKEY PROBE RANGE)
```

;; There is a slight conceptual glitch here in that we should subtract off HASH.HEADER.SIZE from HKEY but it
;; would affect existing hashfiles and does not cause any real error due to the IMOD

```
                            (IPLUS HASH.HEADER.SIZE (ITIMES (IMOD (IPLUS PROBE (IQUOTIENT HKEY HASH.KEY.SIZE)
                                                                                 )
                                                                          RANGE)
                                                                   HASH.KEY.SIZE])
)

(DECLARE%: EVAL@COMPILE

(ARRAYRECORD HashFile (File Stream Size %#Entries ValueType ItemLength Valid? Write? ItemCopyFn ByteStream))

(TYPERECORD HashTextPtr (Start . End))

(RECORD HashFileEntry (FILE . HASHFILE))

(RECORD DoubleKey (Key1 . Key2))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ HASH.HEADER.SIZE 8)

(RPAQQ HASH.KEY.SIZE 4)

(CONSTANTS (HASH.HEADER.SIZE 8)
       (HASH.KEY.SIZE 4))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS HFGROWTHFACTOR HASHLOADFACTOR HASHFILEDEFAULTSIZE HASHSCRATCHCONSCELL HASHTEXTCHAR HASHSCRATCHLST
       HASHBITTABLE SYSHASHFILE SYSHASHFILELST PROBELST HASHACCESSTYPES HASHFILERDTBL MAX.INTEGER)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS HASH.HEADER.SIZE HASH.KEY.SIZE)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(SPECVARS REHASHGAG)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: LOOKUPHASHFILEBLOCK (ENTRIES LOOKUPHASHFILE GETHASHFILE PUTHASHFILE)
       LOOKUPHASHFILE GETHASHFILE PUTHASHFILE DELETEHASHKEY GETHASHKEY GETPROBE INSERTHASHKEY MAKEHASHKEY
       REPLACEHASHKEY)

(BLOCK%: OPENHASHFILEBLOCK (ENTRIES CREATEHASHFILE OPENHASHFILE)
       CREATEHASHFILE OPENHASHFILE FIND1STPRIME SETHASHSTATUS)

(BLOCK%: MAPHASHFILEBLOCK (ENTRIES COLLECTKEYS COPYHASHFILE COPYHASHITEM HASHFILESPLST MAPHASHFILE REHASHFILE)
       (SPECVARS REHASHGAG)
       COLLECTKEYS COPYHASHFILE COPYHASHITEM HASHFILESPLST HASHFILESPLST1 MAPHASHFILE REHASHFILE SPLITKEY)
)
)

(PUTPROPS HASH FILETYPE CL:COMPILE-FILE)

(PUTPROPS HASH COPYRIGHT ("Venue & Xerox Corporation" 1984 1985 1986 1990))
```

{MEDLEY}<library>HASH.;1   28-Jun-2024 18:34:03
-- Listed on 30-Jun-2024 13:13:02 --

FUNCTION INDEX

```
CLEARHASHFILES ....1   FIND1STPRIME ......5   HASHFILEDATA ......3   LOOKUPHASHFILE ....4   REPLACEHASHKEY ....6
CLOSEHASHFILE .....1   GETHASHFILE .......3   HASHFILENAME ......3   MAKEHASHKEY .......6   SETHASHSTATUS .....6
COLLECTKEYS .......2   GETHASHKEY ........5   HASHFILEP .........3   MAPHASHFILE .......4   SPLITKEY ..........6
COPYHASHFILE ......2   GETHASHTEXT .......3   HASHFILEPROP ......3   OPENHASHFILE ......4
COPYHASHITEM ......2   GETPROBE ..........5   HASHFILESPLST .....3   PUTHASHFILE .......4
CREATEHASHFILE ....2   GTHASHFILE ........5   HASHFILESPLST1 ....6   PUTHASHTEXT .......5
DELETEHASHKEY .....5   HASHBEFORECLOSE ...3   INSERTHASHKEY .....6   REHASHFILE ........5
```

VARIABLE INDEX

```
AFTERSYSOUTFORMS ........7   HASHFILERDTBL ...........7   HASHTEXTCHAR ............6   SYSHASHFILE .............7
HASHACCESSTYPES .........7   HASHLOADFACTOR ..........6   HFGROWTHFACTOR ..........6   SYSHASHFILELST ..........7
HASHBITTABLE ............7   HASHSCRATCHCONSCELL .....6   PROBELST ................7
HASHFILEDEFAULTSIZE .....6   HASHSCRATCHLST ..........7   REHASHGAG ...............7
```

MACRO INDEX

```
ANYEQ .......7   CREATEKEY ...7   PRINTPTR ....7   PRINTSTBYTE .7   READPTR .....7   READSTBYTE ..7   REHASHKEY ...7
```

RECORD INDEX

```
DoubleKey ...............8   HashFile ................8   HashFileEntry ...........8   HashTextPtr .............8
```

CONSTANT INDEX

```
HASH.HEADER.SIZE ........8   HASH.KEY.SIZE ...........8
```

OPTIMIZER INDEX

```
GETHASHFILE .............7   HASHFILENAME ............7
```

PROPERTY INDEX

```
HASH ....................8
```