File created: 21-Mar-2021 21:33:01 **{DSK}<home>larry>ilisp>medley>sources>LLFAULT.;10**

changes to: (VARS LLFAULTCOMS)

previous date: 16-Mar-2021 20:27:50 {DSK}<home>larry>ilisp>medley>sources>LLFAULT.;8

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982-1993, 2021 by Venue & Xerox Corporation.

(RPAQQ LLFAULTCOMS
       [(VARS (FAULTTEST T))
        (COMS                                                ; Bootstrap code, run once when an image is booted
              (FNS \FAULTINIT \D01.FAULTINIT \D01.ASSIGNBUFFERS \MAIKO.FAULTINIT \MAIKO.NEWFAULTINIT
                   \MAIKO.ASSIGNBUFFERS \M-VMEMSAVE \MAIKO.NEWPAGE)
              ;; For setting up (and maybe eventually removing?) MAIKO-specific versions of the generic low-levle functions:
              (FNS \MAIKO.DO.MOVDS)
              (ADDVARS (\MAIKO.MOVDS (TRUE \LOCKEDPAGEP)
                                     (\MAIKO.NEWPAGE \NEWPAGE)
                                     (\MAIKO.NEWPAGE \DONEWPAGE)
                                     (NILL \LOCKPAGES)
                                     (NILL \DOLOCKPAGES)
                                     (NILL \DOTEMPLOCKPAGES)
                                     (NILL \TEMPUNLOCKPAGES)
                                     (NILL \UNLOCKPAGES)
                                     (NILL \WRITEDIRTYPAGE)
                                     (NILL \DIRTYBACKGROUND)
                                     (ZERO \COUNTREALPAGES)
                                     (NILL \SHOWPAGETABLE)
                                     (NILL CHECKPAGEMAP)
                                     (EVQ \PAGEFAULT)
                                     (EVQ \LOADVMEMPAGE)
                                     (NILL \LOADVMEMPAGE)
                                     (TRUE \MOVEVMEMFILEPAGE)
                                     (TRUE \VALIDADDRESSP)))
              (FNS \DOVE.FAULTINIT \DL.FAULTINIT \DL.NEWFAULTINIT \DL.UNMAPPAGES \DL.MARK.PAGES.UNAVAILABLE
                   \DL.ASSIGNBUFFERS \CHAIN.UP.RPT))
        (COMS                                                ; Pagefault handler
              (FNS \FAULTHANDLER \PAGEFAULT \INVALIDADDR \INVALIDVP \FLUSHPAGE \LOADVMEMPAGE \MOVEREALPAGE
                   \LOOKUPPAGEMAP \VALIDADDRESSP \LOCKEDPAGEP \SELECTREALPAGE \SPECIALRP \TRANSFERPAGE
                   \UPDATECHAIN))
        (COMS                                                ; Allocating and locking new pages
              (FNS \NEWPAGE \DONEWPAGE \ASSURE.FPTOVP.PAGE \MAKESPACEFORLOCKEDPAGE \MOVEVMEMFILEPAGE
                   \NEWEPHEMERALPAGE \DONEWEPHEMERALPAGE \LOCKPAGES \DOLOCKPAGES \TEMPLOCKPAGES \DOTEMPLOCKPAGES
                   \TEMPUNLOCKPAGES \UNLOCKPAGES))
        (COMS                                                ; Writing out the vmem
              (FNS \DOFLUSHVM \RELEASEWORKINGSET \WRITEDIRTYPAGE \WRITEDIRTYPAGE1 \COUNTREALPAGES))
        (COMS                                                ; VMEM.PURE.STATE hack
              (FNS \DOCOMPRESSVMEM VMEM.PURE.STATE))
        (COMS                                                ;; Handling the backing store getting too full--keep running, but if we overflow, we can never \FLUSHVM because there is no place to
                                                             ;; write some pages
              (FNS 32MBADDRESSABLE \SET.VMEM.FULL.STATE \SET.LASTVMEMFILEPAGE \DOVMEMFULLINTERRUPT \FLUSHVMOK?))
        (INITVARS (\UPDATECHAINFREQ 100)
                  (\PAGEFAULTCOUNTER 0)
                  (\DIRTYPAGECOUNTER 0)
                  (\DIRTYPAGEHINT 0)
                  (\LASTACCESSEDVMEMPAGE 0)
                  (\MAXSHORTSEEK 1000)
                  (\MINSHORTSEEK 20)
                  (\MAXCLEANPROBES 20)
                  (\VMEM.INHIBIT.WRITE)
                  (\VMEM.PURE.LIMIT)
                  (\VMEM.FULL.STATE)
                  (\GUARDVMEMFULL 500)
                  (VMEM.COMPRESS.FLG)
                  (\DOFAULTINIT 0)
                  (\VMEMACCESSFN)
                  (\SYSTEMCACHEVARS)
                  (\MAXSWAPBUFFERS 1)
                  (\EXTENDINGVMEMFILE)
                  (\MaxScreenPage 0)
                  (\NEWVMEMPAGEADDED))
        (INITVARS (\LASTDIRTYCNT)
                  (\LASTDIRTYFOUND)
                  (\LASTDIRTYSCANPTR)
                  (\DIRTYSEEKMAX 50))
        (COMS                                                ; Errors signaled in the maintenance panel
              (FNS \MP.ERROR))
        (COMS                                                ; Debugging code.  Some of this also runs renamed for extra
                                                             ; TeleRaid help
```

```
            (FNS \ACTONVMEMFILE \SHOWPAGETABLE CHECKPAGEMAP CHECKFPTOVP CHECKFPTOVP1 \PRINTFPTOVP \PRINTVP))
      (E (RESETSAVE (RADIX 8)))
      (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS \ACTONVMEMFILE .VMEM.CONSISTENTP. .LOCKABLERP.)
            (COMS                                                ; Virtual page flags
                  (CONSTANTS \VMAP.DIRTY \VMAP.CLEAN \VMAP.REF \VMAP.VACANT \VMAP.FLAGS \VMAP.NOTFLAGS)
                  (RECORDS VMEMFLAGS)
                  (MACROS LOGNOT16))
            (COMS                                                ; RPT constants
                  (CONSTANTS \RPT.EMPTY \RPT.UNAVAILABLE \PAGETABLESTOPFLG \RPTENTRYLENGTH)
                  (RECORDS RPT RPT1)
                  (MACROS RPFROMRPT RPTFROMRP NPAGESMACRO))
            (COMS                                                ; Virtual to file pagemap
                  (EXPORT (CONSTANTS \MAXFILEPAGE))
                  (CONSTANTS \EMPTYPMTENTRY)
                  (RECORDS VP)
                  (MACROS .PAGEMAPBASE.))
            (COMS                                                ; FP to VP stuff
                  (RECORDS FPTOVP)
                  (CONSTANTS \NO.VMEM.PAGE)
                  (MACROS DLRPFROMFP DLFPFROMRP))
            (PROP DOPVAL \TOUCHPAGE TIMES3)
            (COMS                                                ; Locked page table
                  (MACROS .LOCKEDVPBASE. .LOCKEDVPMASK.))
            (CONSTANTS \MAXDIRTYSCANCOUNT \MINVMEMSPAREPAGES \DLBUFFERPAGES)
            (CONSTANTS 2MBPAGES)
            (GLOBALVARS \UPDATECHAINFREQ \REALPAGETABLE \RPTLAST \RPOFFSET \RPTSIZE \LOCKEDPAGETABLE
                  \EMBUFBASE \EMBUFVP \EMBUFRP \PAGEFAULTCOUNTER \LASTDIRTYCNT \LASTDIRTYFOUND
                  \LASTDIRTYSCANPTR \MACHINETYPE \LASTACCESSEDVMEMPAGE \MAXSHORTSEEK \MAXCLEANPROBES
                  \MINSHORTSEEK \DIRTYSEEKMAX \DIRTYPAGECOUNTER \DIRTYPAGEHINT \VMEM.INHIBIT.WRITE
                  \VMEM.PURE.LIMIT \VMEM.FULL.STATE \GUARDVMEMFULL VMEM.COMPRESS.FLG \KBDSTACKBASE
                  \MISCSTACKBASE \DOFAULTINIT \FPTOVP \VMEMACCESSFN \SYSTEMCACHEVARS \LASTVMEMFILEPAGE
                  \EXTENDINGVMEMFILE \MaxScreenPage \NEWVMEMPAGEADDED)
            (GLOBALVARS \#SWAPBUFFERS \#EMUBUFFERS \#DISKBUFFERS \MAXSWAPBUFFERS \EMUSWAPBUFFERS \EMUBUFFERS
                  \TELERAIDBUFFER \EMUDISKBUFFERS \EMUDISKBUFEND)
            (MACROS RWMufMan)
            (CONSTANTS (DOLOCKCHECKS NIL)))
      [COMS
```

;;; MAKEINIT stuff

```
            (FNS ADDPME CHECKIFPAGE DUMPINITPAGES MAKEROOMFORPME MAPPAGES READPAGEMAP READPAGEMAPBLOCK
                 SETUPPAGEMAP)
            (DECLARE%: DONTCOPY (MACROS CHECKIF)
                  (ADDVARS (INEWCOMS (FNS DUMPINITPAGES)
                                     (VARS INITCONSTANTS)
                                     (FNS SETUPPAGEMAP ADDPME MAKEROOMFORPME MAPPAGES))
                           (RDCOMS (FNS READPAGEMAP READPAGEMAPBLOCK CHECKIFPAGE \LOCKEDPAGEP \LOOKUPPAGEMAP
                                        CHECKPAGEMAP CHECKFPTOVP CHECKFPTOVP1 \SHOWPAGETABLE \PRINTFPTOVP))
                           (EXPANDMACROFNS CHECKIF .LOCKEDVPBASE. .LOCKEDVPMASK. .PAGEMAPBASE.)
                           (MKI.SUBFNS (\NEWPAGE . MKI.NEWPAGE)
                                  (\LOCKPAGES . MKI.LOCKPAGES))
                           (RD.SUBFNS (\NEWPAGE . VNEWPAGE)
                                  (\LOCKPAGES . VLOCKPAGES))
                           (RDPTRS (\REALPAGETABLE))
                           (RDVALS (\RPTSIZE)))
                        EVAL@COMPILE
                        (ADDVARS (DONTCOMPILEFNS DUMPINITPAGES SETUPPAGEMAP ADDPME MAKEROOMFORPME MAPPAGES
                                      READPAGEMAP READPAGEMAPBLOCK CHECKIFPAGE]
      (FNS \LOCKFN \LOCKCODE \LOCKVAR \LOCKCELL \LOCKWORDS)
      [DECLARE%: DONTCOPY
            (ADDVARS (INEWCOMS (FNS \LOCKFN \LOCKVAR \LOCKCELL \LOCKWORDS \LOCKCODE)
                               (ALLOCAL (ADDVARS (LOCKEDFNS \FAULTHANDLER \FAULTINIT \DOVE.FAULTINIT
                                                            \D01.FAULTINIT \DL.FAULTINIT \CHAIN.UP.RPT
                                                            \MAKESPACEFORLOCKEDPAGE \PAGEFAULT \WRITEMAP
                                                            \LOOKUPPAGEMAP \LOCKEDPAGEP \LOADVMEMPAGE \MOVEREALPAGE
                                                            \INVALIDADDR \INVALIDVP \SELECTREALPAGE \TRANSFERPAGE
                                                            \SPECIALRP \UPDATECHAIN \MARKPAGEVACANT \FLUSHPAGE
                                                            \CLEARWORDS \FLUSHVM \DONEWPAGE \ASSURE.FPTOVP.PAGE
                                                            \DONEWEPHEMERALPAGE \WRITEDIRTYPAGE1 \COPYSYS0
                                                            \COPYSYS0SUBR \RELEASEWORKINGSET \DOFLUSHVM \DOLOCKPAGES
                                                            \DOTEMPLOCKPAGES \TEMPUNLOCKPAGES \MP.ERROR RAID
                                                            \DL.NEWFAULTINIT \DL.MARK.PAGES.UNAVAILABLE
                                                            \DL.UNMAPPAGES \DL.ASSIGNBUFFERS \D01.ASSIGNBUFFERS
                                                            \DOCOMPRESSVMEM \MOVEVMEMFILEPAGE \SET.VMEM.FULL.STATE
                                                            \HINUM \LONUM \ATOMCELL SETTOPVAL)
                                                (LOCKEDVARS \REALPAGETABLE \RPTLAST \PAGEFAULTCOUNTER
                                                       \UPDATECHAINFREQ \RPOFFSET \RPTSIZE \LOCKEDPAGETABLE
                                                       \EMBUFBASE \EMBUFVP \EMBUFRP \LASTACCESSEDVMEMPAGE
                                                       \MAXSHORTSEEK \MAXCLEANPROBES \MINSHORTSEEK
                                                       \DIRTYPAGECOUNTER \DIRTYPAGEHINT \VMEM.INHIBIT.WRITE
                                                       \VMEM.PURE.LIMIT \VMEM.FULL.STATE \GUARDVMEMFULL
                                                       VMEM.COMPRESS.FLG \KBDSTACKBASE \MISCSTACKBASE
                                                       \DOFAULTINIT \FPTOVP \MACHINETYPE \VMEMACCESSFN
                                                       \TELERAIDBUFFER \EMUDISKBUFFERS \EMUDISKBUFEND
                                                       \MAXSWAPBUFFERS \EMUBUFFERS \#EMUBUFFERS \#SWAPBUFFERS
                                                       \#DISKBUFFERS \RCLKSECOND \RCLKMILLISECOND \VALSPACE
                                                       \EMUSWAPBUFFERS \EM.CURSORBITMAP \PAGEMAP \PageMapTBL
```

```
                                                    \IOCBPAGE \IOPAGE \MISCSTATS \DEFSPACE \InterfacePage
                                                    \LASTVMEMFILEPAGE \DoveIORegion \MaxScreenPage
                                                    \NEWVMEMPAGEADDED]
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA)
                      (NLAML)
                      (LAMA CHECKPAGEMAP \SHOWPAGETABLE VMEM.PURE.STATE \COUNTREALPAGES \WRITEDIRTYPAGE
                            \UNLOCKPAGES \TEMPUNLOCKPAGES \DOTEMPLOCKPAGES \DOLOCKPAGES \LOCKPAGES \LOADVMEMPAGE
                            ])

(RPAQQ FAULTTEST T)

;; Bootstrap code, run once when an image is booted

(DEFINEQ

(\FAULTINIT
  [LAMBDA NIL                                                    ; Edited 30-Mar-88 17:30 by Snow

;;; retrieves some constants from Interface page for the swapper and performs other initialization that must happen immediately.  Called when starting
;;; up, and also when \FAULTHANDLER context starts, in case init hasn't happened yet, as e.g.  from MAKEINIT

    (SETQ \MACHINETYPE (fetch MachineType of \InterfacePage))
    [PROG NIL
          (COND
             ((EQ \MACHINETYPE \MAIKO)
              (\MAIKO.FAULTINIT)
              (RETURN NIL)))
          (OR (NEQ (fetch FPTOVPStart of \InterfacePage)
                   0)
              (\MP.ERROR \MP.OBSOLETEVMEM "No FPTOVP"))
          (COND
             ((AND (NEQ 0 (fetch (IFPAGE FullSpaceUsed) of \InterfacePage))
                   (SELECTC \MACHINETYPE
                        (\DORADO NIL)
                        (\DANDELION (EQ 0 (fetch (IFPAGE DL24BitAddressable) of \InterfacePage)))
                        (\DAYBREAK NIL)
                        T))
              (\MP.ERROR \MP.32MBINUSE "Sysout contains virtual pages not addressable by machine")))
          (SETQ \LASTDIRTYSCANPTR)
          (SELECTC \MACHINETYPE
               (\DANDELION (\DL.FAULTINIT))
               (\DAYBREAK (\DOVE.FAULTINIT))
               (\D01.FAULTINIT))                                 ; Have to set \EM.CURSORBITMAP before faults can happen
     ;; But you can't call \SETIOPOINTERS on a Daybreak until after the Dove IO Region is mapped, which happens in \DL.NEWFAULTINIT
          (\SETIOPOINTERS)
          (COND
             ((IGREATERP (fetch (IFPAGE NActivePages) of \InterfacePage)
                     (IDIFFERENCE \LASTVMEMFILEPAGE \GUARDVMEMFULL))
                                                                   ; Vmem getting full!
              (\SET.VMEM.FULL.STATE]
    (COND
       ((EQ (PROG1 \DOFAULTINIT (SETQ \DOFAULTINIT NIL))
            T)                                                     ; true after \FLUSHVM.  Need to rebuild some contexts
        (replace (IFPAGE KbdFXP) of \InterfacePage with (\MAKEFRAME (COND
                                                                        ((fetch (LITATOM CCODEP) of '\KEYHANDLER)
                                                                         (FUNCTION \KEYHANDLER))
                                                                        (T '\DUMMYKEYHANDLER))
                                                                \KBDSTACKBASE
                                                                (IPLUS \KBDSTACKBASE \StackAreaSize)
                                                                0 0))
        (replace (IFPAGE MiscFXP) of \InterfacePage with (\MAKEFRAME (FUNCTION \DOMISCAPPLY)
                                                                \MISCSTACKBASE
                                                                (IPLUS \MISCSTACKBASE \StackAreaSize)
                                                                0 0))
        T])

(\D01.FAULTINIT
  [LAMBDA NIL                                                      (* bvm%: "20-Oct-86 18:19")
    (SETQ \VMEMACCESSFN (FUNCTION \M44ACTONVMEMFILE))
    (SETQ \REALPAGETABLE (fetch (IFPAGE REALPAGETABLEPTR) of \InterfacePage))
    ;; Note: these SETQ's do not reference count, since the values are all smallp's and emulator addresses (in atom space)
    (SETQ \RPOFFSET (SIGNED (fetch (IFPAGE RPOFFSET) of \InterfacePage)
                            BITSPERWORD))
    (SETQ \RPTSIZE (fetch (IFPAGE RPTSIZE) of \InterfacePage))     ; Initialize the software clocks from alto emulator
    (\BLT (LOCF (fetch SECONDSCLOCK of \MISCSTATS))
          (EMADDRESS \RTCSECONDS)
          (UNFOLD 3 WORDSPERCELL))
    [SETQ \RCLKMILLISECOND (CONSTANT (OR (SMALLP \ALTO.RCLKMILLISECOND)
                                         (ERROR \ALTO.RCLKMILLISECOND "\ALTO.RCLKMILLISECOND isn't a SMALLP???"]
```

;;; \ALTO.RCLKMILLISECOND must be a SMALLP here so as not to cause any refcnt or pagefault activity. \RCLKSECOND is large and has to live on
;;; \MISCSTATS, since there is no convenient way to lock a random cell.

```
    (SETQ.NOREF \RCLKSECOND (LOCF (fetch RCLKSECOND of \MISCSTATS)))

;;; Note the SETQ.NOREF for \RCLKSECOND in order to guarantee no refcnt'ing (which might pagefault) Note that these LOADBYTE expressions are
;;; compiled as constants

    (replace (FIXP HINUM) of \RCLKSECOND with (LOADBYTE \ALTO.RCLKSECOND 16 16))
    (replace (FIXP LONUM) of \RCLKSECOND with (LOADBYTE \ALTO.RCLKSECOND 0 16))
    [COND
       ((AND (EQ \MACHINETYPE \DORADO)
             (ILEQ 5124 (fetch RVersion of \InterfacePage)))
        (replace NSHost0 of \InterfacePage with 0)
        (replace NSHost1 of \InterfacePage with 21898)
        (replace NSHost2 of \InterfacePage with (IPLUS (MASK.1'S 15 1)
                                                       (for I (N _ 0) from 1168 to 1175
                                                          do        ; Mufflers '2220Q' thru '2227Q' hold the bits of the basic serial
                                                                    ; number
                                                             [SETQ N (IPLUS (LLSH N 1)
                                                                            (COND
                                                                               ((BITTEST (RWMufMan I)
                                                                                         (MASK.1'S 15 1))
                                                                                0)
                                                                               (T 1]
                                                          finally (RETURN N]
    (\CHAIN.UP.RPT)
    (\D01.ASSIGNBUFFERS])
```

(\D01.ASSIGNBUFFERS

```
  [LAMBDA NIL                                                       (* bvm%: "20-Oct-86 18:21")
    (PROGN                                                          ; Assign swap buffer
           (SETQ \EMBUFVP (fetch (IFPAGE EMBUFVP) of \InterfacePage))
           (SETQ \EMBUFBASE (EMPOINTER (UNFOLD \EMBUFVP WORDSPERPAGE)))
           (SETQ \EMBUFRP (\READRP \EMBUFVP)))
    (PROG ((EMBUF (fetch (IFPAGE EMUBUFFERS) of \InterfacePage))
           (EMLEN (fetch (IFPAGE EMUBUFLENGTH) of \InterfacePage))
           EXTRALEN NPAGES)
          [add EMLEN (IDIFFERENCE EMBUF (SETQ EMBUF (CEIL EMBUF WORDSPERPAGE]
                                                                    ; Round up to a page boundary and throw out the excess
          (SETQ EXTRALEN (IMOD EMLEN WORDSPERPAGE))
          (add EXTRALEN (COND
                           ((ILESSP EXTRALEN 100)
                            (TIMES 2 WORDSPERPAGE))
                           (T WORDSPERPAGE)))
          (SETQ NPAGES (FOLDLO (SETQ EMLEN (IDIFFERENCE EMLEN EXTRALEN))
                               WORDSPERPAGE))
          (OR (IGEQ NPAGES 4)
              (RAID "No swap buffer space"))
          (SETQ \TELERAIDBUFFER (EMPOINTER EMBUF))
          (SETQ \EMUBUFFERS (\ADDBASE \TELERAIDBUFFER WORDSPERPAGE))
          (SETQ \#EMUBUFFERS (SETQ NPAGES (SUB1 NPAGES)))
          (SETQ \#SWAPBUFFERS (IMIN \MAXSWAPBUFFERS (IQUOTIENT NPAGES 2)))
          (SETQ \#DISKBUFFERS (IDIFFERENCE \#EMUBUFFERS \#SWAPBUFFERS))
          (SETQ \EMUDISKBUFFERS \EMUBUFFERS)
          (SETQ \EMUDISKBUFEND (\ADDBASE \EMUDISKBUFFERS (UNFOLD \#DISKBUFFERS WORDSPERPAGE)))
          (SETQ \EMUSWAPBUFFERS \EMUDISKBUFEND)
          (\INITBFS (\ADDBASE \EMUBUFFERS (UNFOLD NPAGES WORDSPERPAGE))
                 EXTRALEN T])
```

(\MAIKO.FAULTINIT

```
  [LAMBDA NIL                                                       ; Edited  2-Jan-93 12:25 by jds
    (SETQ \VMEMACCESSFN (FUNCTION NILL))                            ; This variable must be the name of function that may be
                                                                    ; ACTONVMEMFILE that may write back  from VP to FP. But , in
                                                                    ; Katana type, this function may be required (by tt)
    (SETQ \IOCBPAGE (create POINTER
                            PAGE# _ \VP.IOCBS))
```

;; MOVD all the Maiko-specific low-level functions onto their generic counterparts:

```
    (\MAIKO.DO.MOVDS)
    (\MAIKO.NEWFAULTINIT)
    (SETQ \RCLKMILLISECOND 1000)
    (SETQ \RCLKSECOND 1000000)
    (\RCLK (LOCF (fetch BASECLOCK of \MISCSTATS)))                  ; Reset base clock
    (\PUTBASEPTR (LOCF (fetch SECONDSCLOCK of \MISCSTATS))
           0 NIL)                                                   ; Clear the seconds timer (by tt)
    (\PUTBASEPTR (LOCF (fetch MILLISECONDSCLOCK of \MISCSTATS))
           0 NIL)                                                   ; Clear the milliseconds timer
                                                                    (* SETQ \LASTVMEMFILEPAGE
                                                                    (fetch (IFPAGE DLLastVmemPage) of \InterfacePage))
    (\SETIOPOINTERS])
```

(\MAIKO.NEWFAULTINIT

```
  [LAMBDA NIL                                                       ; Edited 26-Feb-88 14:07 by Osamu Nakamura
```

;; We have just started up on a Katana. Boot code (SYSOUT Loader) may map FP to VP(VP is same map to RP). Therefore, in this function, only
;; done the initialization of the gloval variables (particularly, the variables about Buffers). And, there is not /REALPAGETABLE in Katana.

```
    (PROG ((NBUFFERS (IDIFFERENCE \DLBUFFERPAGES 2)))           ; Allocate buffers
          (\MAIKO.ASSIGNBUFFERS (create POINTER
                                        PAGE# _ \VP.BUFFERS)
                NBUFFERS])
```

(\MAIKO.ASSIGNBUFFERS

```
  [LAMBDA (BASE NPAGES)                                         ; Edited 14-May-88 18:31 by JMTurn
    (PROGN                                                      ; Allocate a page to hold name and password, and perhaps other
                                                                ; ephemeral things
          (replace (IFPAGE UserNameAddr) of \InterfacePage with (\LOLOC (\ADDBASE BASE 1)))
          (replace (IFPAGE UserPswdAddr) of \InterfacePage with (\LOLOC (\ADDBASE BASE 33)))
          (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
          (add NPAGES -1))
    (PROGN                                                      ; Assign swap buffer
          (SETQ \EMBUFBASE BASE)
          (SETQ \EMBUFVP (fetch (POINTER PAGE#) of BASE))
          (SETQ \EMBUFRP \EMBUFVP)
          (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
          (add NPAGES -1))
    (PROGN                                                      ; Assign ether buffers
          (replace (IFPAGE MDSZoneLength) of \InterfacePage with (UNFOLD 2 WORDSPERPAGE))
          (replace (IFPAGE MDSZone) of \InterfacePage with (\LOLOC BASE))
          (SETQ BASE (\ADDBASE BASE (UNFOLD 2 WORDSPERPAGE)))
          (SETQ \TELERAIDBUFFER BASE)
          (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
          (add NPAGES -3))
    (PROGN                                                      ; Divvy up buffer space
          (SETQ \#SWAPBUFFERS (SETQ \#EMUBUFFERS NPAGES))
          (SETQ \#DISKBUFFERS 0)
          (SETQ \EMUSWAPBUFFERS (SETQ \EMUBUFFERS BASE])
```

(\M-VMEMSAVE

```
  [LAMBDA NIL                                                   ; Edited 20-Apr-88 10:28 by MASINTER
    (PROG ((SCRATCHBUF \EMUSWAPBUFFERS))
          (replace (IFPAGE MISCSTACKRESULT) of \InterfacePage with T)
          [COND
             (\VMEM.PURE.LIMIT                                  ; Maintaining file consistency: move high water mark up
                    (COND
                       (VMEM.COMPRESS.FLG (\DOCOMPRESSVMEM)))
                    (SETQ \VMEM.PURE.LIMIT (fetch (IFPAGE NActivePages) of \InterfacePage]
          (COND
             ((.VMEM.CONSISTENTP.)
              (replace (IFPAGE Key) of \InterfacePage with (LOGNOT16 \IFPValidKey))
                                                                ; Invalidate vmem and write out the Interface page
              ;; following form doesn't eval for KATANA (\TRANSFERPAGE IFPVP \FirstVmemBlock (RPTFROMRP (\READRP IFPVP)) T NIL)

              ))
          (replace (IFPAGE Key) of \InterfacePage with \IFPValidKey)
          (\BLT SCRATCHBUF \InterfacePage WORDSPERPAGE)         ; Make its current fx point at user context, i.e. the \FLUSHVM
                                                                ; frame
          (replace (IFPAGE CurrentFXP) of SCRATCHBUF with (fetch (IFPAGE MiscFXP) of \InterfacePage))
          (SUBRCALL VMEMSAVE)
          (RETURN NIL])
```

(\MAIKO.NEWPAGE

```
  [LAMBDA (BASE NOERROR LOCK?)                                  ; Edited 20-Apr-88 10:28 by MASINTER
    (SUBRCALL NEWPAGE BASE])

)
```

;; For setting up (and maybe eventually removing?) MAIKO-specific versions of the generic low-levle functions:

```
(DEFINEQ
```

(\MAIKO.DO.MOVDS

```
  [LAMBDA NIL                                                   ; Edited  2-Nov-92 03:57 by sybalsky:mv:envos
```

;; MOVD all the Maiko-specific low-level functions onto their generic counterparts. This function is called from \MAIKO.FAULTINIT when the ;; system is started up, and called explicitly during the LOADUP process to get everything in a state to run the ethernet.

;; THIS IS WHERE CHANGES SHOULD HAPPEN TO MAKE SUN LOADUPS RUN ON D-MACHINES (BY ADDING A \MAIKO.UNDO.MOVDS ;; CALL AT VMEM SAVING TIME, AND ADDING A SYBMOL TO SAVE THE GENERIC DEFINITION ON TO THE MOVDS LIST.

```
    (FOR PAIR IN \MAIKO.MOVDS DO
```

;; This is like MOVD, but absolutely no consing is done, frame names are not changed, etc. So that no ;; CONSING happens before all the MOVDs are finished -- prevents new-page allocation.

```
(LET [(FROMCELL (fetch (LITATOM DEFINITIONCELL) of (CAR PAIR)))
      (TOCELL (fetch (LITATOM DEFINITIONCELL) of (CADR PAIR]
     (UNINTERRUPTABLY
         (replace (DEFINITIONCELL DEFPOINTER) of TOCELL
            with (fetch (DEFINITIONCELL DEFPOINTER) of FROMCELL))
         (replace (DEFINITIONCELL DEFCELLFLAGS) of TOCELL
            with (fetch (DEFINITIONCELL DEFCELLFLAGS) of FROMCELL))
         (replace (DEFINITIONCELL AUXDEFCELLFLAGS) of TOCELL
            with (fetch (DEFINITIONCELL AUXDEFCELLFLAGS) of FROMCELL)))])
```

```
)

(ADDTOVAR \MAIKO.MOVDS
          (TRUE \LOCKEDPAGEP)
          (\MAIKO.NEWPAGE \NEWPAGE)
          (\MAIKO.NEWPAGE \DONEWPAGE)
          (NILL \LOCKPAGES)
          (NILL \DOLOCKPAGES)
          (NILL \DOTEMPLOCKPAGES)
          (NILL \TEMPUNLOCKPAGES)
          (NILL \UNLOCKPAGES)
          (NILL \WRITEDIRTYPAGE)
          (NILL \DIRTYBACKGROUND)
          (ZERO \COUNTREALPAGES)
          (NILL \SHOWPAGETABLE)
          (NILL CHECKPAGEMAP)
          (EVQ \PAGEFAULT)
          (EVQ \LOADVMEMPAGE)
          (NILL \LOADVMEMPAGE)
          (TRUE \MOVEVMEMFILEPAGE)
          (TRUE \VALIDADDRESSP))

(DEFINEQ
```

(\DOVE.FAULTINIT

```
  [LAMBDA NIL                                                    ; Edited 18-Sep-87 16:01 by bvm:
    (DECLARE (GLOBALVARS \RCLKMILLISECOND \RCLKSECOND))
    (SETQ \VMEMACCESSFN (FUNCTION \DOVE.ACTONVMEMFILE))
    (SETQ \IOCBPAGE (create POINTER
                            PAGE# _ \VP.IOCBS))
    (COND
       ((NOT (.VMEM.CONSISTENTP.))
        (\MP.ERROR \MP.INVALIDVMEM)))
    (SETMAINTPANEL 1188)
    (\DL.NEWFAULTINIT)
    (SETMAINTPANEL 1189)
    (SETQ \RCLKMILLISECOND \DOVE.RCLKMILLISECOND)
    (SETQ.NOREF \RCLKSECOND (LOCF (fetch RCLKSECOND of \MISCSTATS)))
                                                                   ; Unfortunately, \DOVE.RCLKSECOND is not smallp
    (replace (FIXP HINUM) of \RCLKSECOND with (CONSTANT (\HINUM \DOVE.RCLKSECOND)))
    (replace (FIXP LONUM) of \RCLKSECOND with (CONSTANT (\LONUM \DOVE.RCLKSECOND)))
    (\RCLK (LOCF (fetch BASECLOCK of \MISCSTATS)))                 ; Reset base clock
    (\DoveMisc.ReadGMT (LOCF (fetch SECONDSCLOCK of \MISCSTATS)))
    (SETMAINTPANEL 1190)
    (\PUTBASEPTR (LOCF (fetch MILLISECONDSCLOCK of \MISCSTATS))
           0 NIL)                                                  ; Clear the milliseconds timer
    (\DoveMisc.ReadHostID (LOCF (fetch NSHost0 of \InterfacePage)))
    (SETMAINTPANEL 1191)
    [SETQ \LASTVMEMFILEPAGE (COND
                               (NIL                                ; For now, don't assume vmem is any bigger than the part in use
                                                                   ; now.  Local file system init will set it to the truth.
                                    (SETQ \VMEM.FULL.STATE 0)      ; Flag to keep pages from being written off the end.  Setting it
                                                                   ; now prevents bogus vmem full interrupt at startup time.
                                    (fetch (IFPAGE NActivePages) of \InterfacePage))
                               (T                                  ; Microcode is supposed to fill this in
                                  (fetch (IFPAGE DLLastVmemPage) of \InterfacePage]
    (\DoveDisk.Init)
    (SETMAINTPANEL 1192)
    (\DoveDisplay.TurnOn])
```

(\DL.FAULTINIT

```
  [LAMBDA NIL                                                    (* bvm%: "20-Oct-86 18:22")
    (SETQ \VMEMACCESSFN (FUNCTION \DL.ACTONVMEMFILE))
    (SETQ \IOCBPAGE (create POINTER
                            PAGE# _ \VP.IOCBS))
    (COND
       ((NOT (.VMEM.CONSISTENTP.))
        (\MP.ERROR \MP.INVALIDVMEM)))
    (\DL.NEWFAULTINIT)
    (SETQ \RCLKMILLISECOND \DLION.RCLKMILLISECOND)                 ; These are fortunately both small
    (SETQ \RCLKSECOND \DLION.RCLKSECOND)
    (\RCLK (LOCF (fetch BASECLOCK of \MISCSTATS)))                 ; Reset base clock
    [COND
       ((EQ (fetch DLTODVALID of \IOPAGE)
            0)                                                     ; Time not valid, so store zero in the clock
        (\PUTBASEPTR (LOCF (fetch SECONDSCLOCK of \MISCSTATS))
               0 NIL))
       (T (bind TMP (BASE _ (LOCF (fetch SECONDSCLOCK of \MISCSTATS))) do
                                                                   ; Loop until clock reads the same as we wrote, in case it was
                                                                   ; being updated
                                                                          (\PUTBASE BASE 1 (SETQ TMP
                                                                                                 (fetch DLTODLO
                                                                                                    of \IOPAGE)))
                                                                          (\PUTBASE BASE 0 (fetch DLTODHI
                                                                                              of \IOPAGE))
```

```
            repeatuntil (EQ (fetch DLTODLO of \IOPAGE)
                           TMP]
    (\PUTBASEPTR (LOCF (fetch MILLISECONDSCLOCK of \MISCSTATS))
          0 NIL)                                          ; Clear the milliseconds timer
    (repeatwhile (IGEQ (fetch DLPROCESSORCMD of \IOPAGE)
                       \DL.PROCESSORBUSY))                ; Wait for IOP readiness
    (replace DLPROCESSORCMD of \IOPAGE with \DL.READPID)  ; Ask it to give the processor ID (3 words)
    (repeatwhile (IGEQ (fetch DLPROCESSORCMD of \IOPAGE)
                       \DL.PROCESSORBUSY))
    (replace NSHost0 of \InterfacePage with (fetch DLPROCESSOR0 of \IOPAGE))
    (replace NSHost1 of \InterfacePage with (fetch DLPROCESSOR1 of \IOPAGE))
    (replace NSHost2 of \InterfacePage with (fetch DLPROCESSOR2 of \IOPAGE))
    (SETQ \LASTVMEMFILEPAGE (fetch (IFPAGE DLLastVmemPage) of \InterfacePage))
    (\DL.DISKINIT T])
```

(\DL.NEWFAULTINIT

```
  [LAMBDA NIL                                             ; Edited 21-Oct-87 15:40 by bvm:
```

;; We have just started up on a DLion or Daybreak. Boot code has loaded the first n pages of the sysout into pages 2 thru n-3, except for the area
;; covered by the map and IO page, and has built the map accordingly. Our principal task is to build \REALPAGETABLE

```
    (PROG ((NREALPAGES (fetch (IFPAGE NRealPages) of \InterfacePage))
           (FIRSTBUFFERRP \RP.STARTBUFFERS)
           (SCRATCHVP \VP.INITSCRATCH)
           (SCRATCHBASE (create POINTER
                                PAGE# _ \VP.INITSCRATCH))
           FIRSTUSEFULRP IFPAGERP IOCBRP RPTBASE VP RPTPAGES FIRSTRP NDISPLAYPAGES)
          [do (COND
                 ((for I from 0 to (SUB1 \DLBUFFERPAGES) as (FPBASE _ (\ADDBASE \FPTOVP (DLFPFROMRP FIRSTBUFFERRP)
                                                                                    ))
                     by (\ADDBASE FPBASE 1) do (COND
                                                  ([OR (NOT (fetch FPOCCUPIED of FPBASE))
                                                       (\LOCKEDPAGEP (SETQ VP (fetch FPVIRTUALPAGE of FPBASE]
                                                    ;; Can't use as buffer.  This is just a check for consistency;  you should pick
                                                    ;; \RP.STARTBUFFERS so that this isn't a problem
                                                    (RETURN T)))  ; Unmap this page so we can use it for buffers
                                                  (\WRITEMAP VP 0 \VMAP.VACANT))
                                                                  ; Bad starting place, try again
                  (add FIRSTBUFFERRP 1))
                 (T (RETURN]
          (SETQ FIRSTUSEFULRP (+ FIRSTBUFFERRP \DLBUFFERPAGES))
          (PROGN                                              ; Copy vital info that booting left in page 1
                 [COND
                    ((EQ \MACHINETYPE \DAYBREAK)
```

;; Use first buffer page for IOCB page. Used to have to place this in a real page whose page-in-segment number was the
;; same as that of \VP.IOCBS, but that constraint is now lifted for Daybreak.

```
                     (SETQ IOCBRP FIRSTBUFFERRP)
                     (add FIRSTBUFFERRP 1))
                    (T (SETQ IOCBRP (+ (LOGAND (SUB1 (IMIN NREALPAGES 3072))
                                               65280)
                                       \VP.IOCBS))
```

;; Put IOCB page near the end of memory, but in the first 1.5 mb so that Burdock can see it. Temporary until Steve fixes
;; swap code to not care what RP contains IOCB's

```
                       [SETQ VP (fetch FPVIRTUALPAGE of (\ADDBASE \FPTOVP (DLFPFROMRP IOCBRP]
                       (COND
                          ((\LOCKEDPAGEP VP)
                           (\MP.ERROR \MP.IOCBPAGE))
                          (T                                  ; Unmap whoever lived in our target page
                             (\WRITEMAP VP 0 \VMAP.VACANT]
                 (\WRITEMAP \VP.IOCBS IOCBRP \VMAP.CLEAN)
                 (\WRITEMAP SCRATCHVP 1 \VMAP.CLEAN)
                 (\BLT \IOCBPAGE SCRATCHBASE WORDSPERPAGE))
          (PROGN                                              ; Copy InterfacePage out of segment zero
                 (\WRITEMAP SCRATCHVP FIRSTBUFFERRP \VMAP.CLEAN)
                 (\BLT SCRATCHBASE \InterfacePage WORDSPERPAGE)
                 (\WRITEMAP \VP.IFPAGE (SETQ IFPAGERP FIRSTBUFFERRP)
                        \VMAP.CLEAN)
                 (add FIRSTBUFFERRP 1))
          [PROGN                                              ; Unmap everything that fell somewhere we can't use
                 (\DL.UNMAPPAGES (ADD1 \FP.IFPAGE)
                        (DLFPFROMRP \RP.IOPAGE))              ; real segment zero, map or IOPAGE
                 (COND
                    ((EQ \MACHINETYPE \DANDELION)
                     (for NEXTBANK0 from 2MBPAGES by 2MBPAGES until (> NEXTBANK0 NREALPAGES)
                        do ;; All the 'shadows of the display bank' in higher memory have restricted use;  take them out of commission for now
                           (\DL.UNMAPPAGES NEXTBANK0 (+ NEXTBANK0 PAGESPERSEGMENT -1]
          (PROGN                                              ; Copy Display into segment zero
                 [SETQ NDISPLAYPAGES (COND
                                        ((EQ \MACHINETYPE \DANDELION)
```

;; Only lock the standard screen's worth of pages on DLion, even if there are more because the
;; sysout came from wide Daybreak. Only this many need to be in the display bank, besides which
;; there is a cursor bank after the display; the rest can be vanilla locked pages.

```
                                    \NP.DISPLAY)
                                   (T (IMAX \NP.DISPLAY (ADD1 \MaxScreenPage]
                                                                    ; Number of display pages in use in this image
          (for I from 0 to (SUB1 NDISPLAYPAGES) do (\WRITEMAP (+ SCRATCHVP I)
                                                              (+ \RP.DISPLAY I)
                                                              \VMAP.CLEAN))
                                                                    ; Point scratch area at real segment zero
          (\BLT SCRATCHBASE (create POINTER
                                    PAGE# _ \VP.DISPLAY)
                 (UNFOLD NDISPLAYPAGES WORDSPERPAGE))       ; Copy display from wherever boot put it
          (for I from 0 to (SUB1 NDISPLAYPAGES) do (\WRITEMAP (+ SCRATCHVP I)
                                                              0 \VMAP.VACANT)
                                                    (\WRITEMAP (+ \VP.DISPLAY I)
                                                              (+ \RP.DISPLAY I)
                                                              \VMAP.CLEAN))
                                                                    ; Display is now where hardware wants it, so enable display
          (replace (IOPAGE DLDISPCONTROL) of \IOPAGE with 0))
    (COND
       ((EQ \MACHINETYPE \DAYBREAK)                             ; If on a daybreak, map the I/O region.  Have to do this before
                                                                ; calling \DoveDisplay.ScreenWidth
        (for I from 0 to (SUB1 \DOVEIORGNSIZE) do (\WRITEMAP (+ \VP.DOVEIORGN I)
                                                             (+ \RP.DOVEIORGN I)
                                                             \VMAP.CLEAN))
        (\DoveIO.InitializeIORegionPtrs)))
    [PROG ((RPSIZE (- NREALPAGES (SETQ \RPOFFSET -1)))
           (FIRSTVP \VP.RPT))
          (SETQ FIRSTRP (COND
                           ((OR (> NDISPLAYPAGES \NP.DISPLAY)
                                (AND (EQ \MACHINETYPE \DAYBREAK)
                                     (EQ (\DoveDisplay.ScreenWidth)
                                         \WIDEDOVEDISPLAYWIDTH)))
                            ;; Sysout was made on a large screen daybreak, or is now being run on one.  Need to make sure there is
                            ;; space for all that display
                            \RP.AFTERDOVEDISPLAY)
                           (T \RP.AFTERDISPLAY)))           ; Construct real page table in segment zero after the display
          [COND
             ((> RPSIZE (CONSTANT (EXPT 2 15)))             ; We only have 15 bits for real page table numbers, so have to
                                                            ; sacrifice the rest of memory
              (SETQ RPSIZE (CONSTANT (EXPT 2 15]
          [SETQ RPTPAGES (PROGN
                           ;; This is a way of computing (FOLDHI RPSIZE*3 WORDSPERPAGE) that won't overflow when
                           ;; memory exceeds 10.6MB -- the first term computes RPSIZE*3/256, the second performs the
                           ;; FOLDHI directly on the now much smaller remainder.
                            (+ (TIMES3 (FOLDLO RPSIZE WORDSPERPAGE))
                               (FOLDHI (TIMES3 (IMOD RPSIZE WORDSPERPAGE))
                                      WORDSPERPAGE]
          (COND
             ((> (+ RPTPAGES FIRSTRP)
                 PAGESPERSEGMENT)
              ;; No space in bank zero, so put RPT in first segment after 2 megabytes, where the first 'shadow' display bank lives.  No
              ;; shadow bank on Daybreak, but this is as good a place as any
              (SETQ FIRSTRP (IMIN 2MBPAGES (- NREALPAGES RPTPAGES)))
                                                            ; IMIN because we could be on a wide-display Daybreak with
                                                            ; small memory
              [COND
                 ((> (+ FIRSTVP RPTPAGES)
                     \VP.BUFFERS)                           ; Move virtual assignment backwards if necessary
                  (SETQ FIRSTVP (COND
                                   ((< RPTPAGES \VP.BUFFERS)
                                    (- \VP.BUFFERS RPTPAGES))
                                   ((<= RPTPAGES PAGESPERSEGMENT)
                                                            ; Can't fit real page table in display bank at all, so overlap
                                                            ; smallneg space
                                    (UNFOLD \SmallNegHi PAGESPERSEGMENT))
                                   (T                       ; Ack, more than 10.6 MB, have to slop over into smallpos space
                                      (- (+ (UNFOLD \SmallNegHi PAGESPERSEGMENT)
                                            PAGESPERSEGMENT)
                                         RPTPAGES]
              (\DL.UNMAPPAGES (DLFPFROMRP FIRSTRP)
                     (DLFPFROMRP (+ FIRSTRP RPTPAGES -1))) ; Unmap the pages in which RPT lives.  This was already done
                                                            ; on DLion, but can't hurt to do it again

              ))
          (for I from 0 to (SUB1 RPTPAGES) do               ; Assign pages to real page table now
                                          (\WRITEMAP (+ FIRSTVP I)
                                                     (+ FIRSTRP I)
                                                     \VMAP.CLEAN))
          (SETQ \REALPAGETABLE (create POINTER
                                       PAGE# _ FIRSTVP))
          (\CLEARWORDS \REALPAGETABLE RPSIZE)
          (\CLEARWORDS (\ADDBASE \REALPAGETABLE RPSIZE)
                 RPSIZE)
          (\CLEARWORDS (\ADDBASE (\ADDBASE \REALPAGETABLE RPSIZE)
                                 RPSIZE)
```

```
                RPSIZE)                                          ; Clear table in three steps, since 3*RPSIZE overflows after
                                                                 ; 10MB
        (SETQ \RPTSIZE RPSIZE)
        (COND
           [(EQ \MACHINETYPE \DANDELION)
            (for NEXTBANK0 from 2MBPAGES by 2MBPAGES until (> NEXTBANK0 NREALPAGES)
               do                                                ; Mark the shadow display bank pages unavailable
                  (\DL.MARK.PAGES.UNAVAILABLE NEXTBANK0 (+ NEXTBANK0 PAGESPERSEGMENT -1]
           (T                                                    ;; RPT itself occupies unavailable pages;  on DLion these were marked unavailable either in segment zero after display or
                                                                 ;; as part of shadow bank
              (\DL.MARK.PAGES.UNAVAILABLE FIRSTRP (+ FIRSTRP RPTPAGES -1))
                                                                 ; Also, Dove IO region is unavailable
              (\DL.MARK.PAGES.UNAVAILABLE \RP.DOVEIORGN (SUB1 (+ \RP.DOVEIORGN \DOVEIORGNSIZE]
    (PROGN                                                       ;; Fill in special cases in RPT -- the display, which is not where FPTOVP says it is, and all the pages that are unavailable for one
                                                                 ;; reason or another.  Note: any page marked unavailable here MUST be unmapped by now, either because booting never put it
                                                                 ;; where FPTOVP says it would be, there's no page there to begin with, or there's an explicit call to \WRITEMAP or
                                                                 ;; \DL.UNMAPPAGES to unmap it above
           (SETQ RPTBASE \REALPAGETABLE)
           [for I from 0 to (SUB1 NDISPLAYPAGES) do (SETQ RPTBASE (\ADDBASE RPTBASE \RPTENTRYLENGTH))
                                                                 ; Fill in Display pages
                                                    (replace (RPT VP) of RPTBASE with (+ \VP.DISPLAY I))
                                                    (replace (RPT FILEPAGE) of RPTBASE
                                                       with (DLFPFROMRP (+ \RP.TEMPDISPLAY I]
           (\DL.MARK.PAGES.UNAVAILABLE NDISPLAYPAGES \RP.IOPAGE)
                                                                 ; Mark rest of segment zero plus Map and IOPAGE unavailable

           )
    [PROGN                                                       ; fill in main part of RPT by reading FPTOVP
           (for I from (ADD1 \RP.IOPAGE) to (SUB1 NREALPAGES) as [FPBASE _ (\ADDBASE \FPTOVP
                                                                                (DLFPFROMRP (ADD1
                                                                                            \RP.IOPAGE
                                                                                            ]
              by (\ADDBASE FPBASE 1) as [RPTBASE _ (fetch RPTRBASE of (RPTFROMRP (ADD1 \RP.IOPAGE]
              by (\ADDBASE RPTBASE \RPTENTRYLENGTH) bind (LASTREALPAGE _ (DLRPFROMFP (fetch (IFPAGE
                                                                                              NActivePages
                                                                                              )
                                                                                    of \InterfacePage))
                                                                        )
              do                                                 ;; Fill in rest of RPT from \FPTOVP.  Could optimize this a little by special casing the area occupied by the display, but
                                                                 ;; this is simpler
                 (COND
                    ((fetch (RPT UNAVAILABLE) of RPTBASE))
                    ((AND (<= I LASTREALPAGE)
                          (fetch FPOCCUPIED of FPBASE)
                          [NOT (fetch (VMEMFLAGS VACANT) of (\READFLAGS (SETQ VP (fetch FPVIRTUALPAGE
                                                                                    of FPBASE]
                          (EQ I (\READRP VP)))
                     ;; There is a VP assigned to this filepage, and it is still there.  False for display that got moved and any real pages
                     ;; that didn't get filled.  LASTREALPAGE is in case the real memory is larger than the sysout -- FPTOVP does not
                     ;; exist all the way
                     (replace (RPT VP) of RPTBASE with VP)
                     (replace (RPT FILEPAGE) of RPTBASE with (DLFPFROMRP I)))
                    (T (replace (RPT EMPTY) of RPTBASE with T]
    (PROGN                                                       ; Touch up RPT with the exceptions
           (SETQ RPTBASE (fetch RPTRBASE of (RPTFROMRP IFPAGERP)))
                                                                 ; Interface Page
           (replace (RPT VP) of RPTBASE with \VP.IFPAGE)
           (replace (RPT FILEPAGE) of RPTBASE with \FP.IFPAGE)
           (replace (RPT UNAVAILABLE) of (fetch RPTRBASE of (RPTFROMRP IOCBRP)) with T)
                                                                 ; \IOCBPAGE
           (\DL.MARK.PAGES.UNAVAILABLE FIRSTBUFFERRP (SUB1 FIRSTUSEFULRP))
                                                                 ; buffer pages unavailable to swapper

           )
    (\CHAIN.UP.RPT)
    (PROG ((NBUFFERS (- FIRSTUSEFULRP FIRSTBUFFERRP)))           ; Allocate buffers
           (for I from 0 to (SUB1 NBUFFERS) do (\WRITEMAP (+ \VP.BUFFERS I)
                                                          (+ FIRSTBUFFERRP I)
                                                          \VMAP.CLEAN))
           (\DL.ASSIGNBUFFERS (create POINTER
                                      PAGE# _ \VP.BUFFERS)
                  NBUFFERS])
```

(\DL.UNMAPPAGES

```
  [LAMBDA (FIRSTFP LASTFP)                                       (* bvm%: "14-Jan-84 14:20")
```

;;; At initialization time, unmap anything that originally lived in filepages FIRSTFP thru LASTFP

```
    (for FP from FIRSTFP to LASTFP as (FPBASE _ (\ADDBASE \FPTOVP FIRSTFP)) by (\ADDBASE FPBASE 1)
       when (fetch FPOCCUPIED of FPBASE) do (\WRITEMAP (fetch FPVIRTUALPAGE of FPBASE)
                                                       0 \VMAP.VACANT])
```

(\DL.MARK.PAGES.UNAVAILABLE

```
  [LAMBDA (FIRSTRP LASTRP)                                          (* bvm%: "14-Jan-84 14:32")
    (for I from FIRSTRP to LASTRP as (RPTBASE _ (fetch RPTRBASE of (RPTFROMRP FIRSTRP)))
       by (\ADDBASE RPTBASE \RPTENTRYLENGTH) do (replace (RPT UNAVAILABLE) of RPTBASE with T])
```

(\DL.ASSIGNBUFFERS

```
  [LAMBDA (BASE NPAGES)                                             (* bvm%: "29-Jan-85 19:05")
    (PROGN                                                          ; Allocate a page to hold name and password, and perhaps other
                                                                    ; ephemeral things
           (\CLEARWORDS BASE WORDSPERPAGE)
           (replace (IFPAGE UserNameAddr) of \InterfacePage with (\LOLOC (\ADDBASE BASE 1)))
           (replace (IFPAGE UserPswdAddr) of \InterfacePage with (\LOLOC (\ADDBASE BASE 33)))
           (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
           (add NPAGES -1))
    (PROGN                                                          ; Assign swap buffer
           (SETQ \EMBUFBASE BASE)
           (SETQ \EMBUFVP (fetch (POINTER PAGE#) of BASE))
           (SETQ \EMBUFRP (\READRP \EMBUFVP))
           (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
           (add NPAGES -1))
    (PROGN                                                          ; Assign ether buffers
           (replace (IFPAGE MDSZoneLength) of \InterfacePage with (UNFOLD 2 WORDSPERPAGE))
           (replace (IFPAGE MDSZone) of \InterfacePage with (\LOLOC BASE))
           (SETQ BASE (\ADDBASE BASE (UNFOLD 2 WORDSPERPAGE)))
           (SETQ \TELERAIDBUFFER BASE)
           (SETQ BASE (\ADDBASE BASE WORDSPERPAGE))
           (add NPAGES -3))
    (PROGN                                                          ; Divvy up buffer space
           (SETQ \#SWAPBUFFERS (SETQ \#EMUBUFFERS NPAGES))
           (SETQ \#DISKBUFFERS 0)
           (SETQ \EMUSWAPBUFFERS (SETQ \EMUBUFFERS BASE])
```

(\CHAIN.UP.RPT

```
  [LAMBDA NIL                                                       (* bvm%: "18-Dec-84 16:07")
```

;;; Maps over the Real Page Table as constructed so far and fleshes it out. Assumes that the table is built, has all its VP and FILEPAGE entries set, and ;;; that the empty and unavailable entries are so marked. Finishes the job by chaining together the available pages and setting the LOCKED bits

```
    (PROG ((RPTBASE \REALPAGETABLE)
           (LASTEMPTY \REALPAGETABLE)
           (LASTUSED (\ADDBASE \REALPAGETABLE 1))
           FIRSTUSED)
          (SETQ FIRSTUSED LASTUSED)
```

;; The 'entry' \REALPAGETABLE is a dummy that points to the least recently used entry. We use the second word of that dummy as a temporary ;; chain head for the used pages, so that we can put all the empty pages at the front of the queue.

```
          [for I from 1 to (SUB1 \RPTSIZE) do (SETQ RPTBASE (\ADDBASE RPTBASE \RPTENTRYLENGTH))
                                              (COND
                                                 ((fetch (RPT UNAVAILABLE) of RPTBASE))
                                                 ((fetch (RPT EMPTY) of RPTBASE)
                                                  (replace (RPT NEXTRP) of LASTEMPTY with I)
                                                  (replace (RPT LOCKED) of RPTBASE with NIL)
                                                  (SETQ LASTEMPTY RPTBASE))
                                                 (T (replace (RPT NEXTRP) of LASTUSED with I)
                                                    (replace (RPT LOCKED) of RPTBASE
                                                       with (\LOCKEDPAGEP (fetch (RPT VP) of RPTBASE)))
                                                    (SETQ LASTUSED RPTBASE]
                                                                    ; Finally, link the end of empty chain to front of in use chain
          (replace (RPT NEXTRP) of LASTEMPTY with (fetch (RPT NEXTRP) of FIRSTUSED))
          (replace (RPT NEXTRP) of (SETQ \RPTLAST LASTUSED) with \PAGETABLESTOPFLG)
          (replace (RPT UNAVAILABLE) of \REALPAGETABLE with T)      ; Dummy first entry

      ])

)

;; Pagefault handler

(DEFINEQ

(\FAULTHANDLER
  [LAMBDA NIL                                                       ; Edited 27-Sep-88 00:47 by jds
    (PROG NIL
      LP  [OR (AND \DOFAULTINIT (\FAULTINIT))
              (\PAGEFAULT (\VAG2 (LOGAND 255 (fetch (IFPAGE FAULTHI) of \InterfacePage))
                                 (fetch (IFPAGE FAULTLO) of \InterfacePage]
          (\CONTEXTSWITCH \FAULTFXP)
          (GO LP])

(\PAGEFAULT
  [LAMBDA (PTR)                                                     (* bvm%: "13-Aug-85 16:38")
    (\CLOCK0 (LOCF (fetch SWAPTEMP0 of \MISCSTATS)))                ; Note time of start
    (PROG ((VP (fetch (POINTER PAGE#) of PTR))
```

```
          FLAGS FILEPAGE)
         (COND
            ((fetch (VP INVALID) of VP)                          ; Map out of bounds on Dolphin always produces -1 as the vp.
                                                                 ; Don't know about other machines
             (\MP.ERROR \MP.MOB "Page Fault: Map out of bounds" (AND (NEQ VP 65535)
                                                                     PTR)
                    T))
            ([NOT (fetch (VMEMFLAGS VACANT) of (SETQ FLAGS (\READFLAGS VP]
             (\MP.ERROR \MP.RESIDENT "Fault on resident page" PTR T))
            ((EQ (SETQ FILEPAGE (\LOOKUPPAGEMAP VP))
                 0)
             (\INVALIDADDR PTR))
            (T (COND
                  ((EQ (\HILOC PTR)
                       \STACKHI)                                 ; should never happen.  For debugging
                   (\MP.ERROR \MP.STACKFAULT "Fault on stack" PTR T)))
               (\LOADVMEMPAGE VP FILEPAGE)))
         (COND
            (\NEWVMEMPAGEADDED                                   ; Only happens if VMEM.PURE.STATE on
                   (\ASSURE.FPTOVP.PAGE)))
         [\BOXIPLUS (LOCF (fetch SWAPWAITTIME of \MISCSTATS))
                (\BOXIDIFFERENCE (\CLOCK0 (LOCF (fetch SWAPTEMP1 of \MISCSTATS)))
                       (LOCF (fetch SWAPTEMP0 of \MISCSTATS]      ; Count the time used.
         (RETURN PTR])
```

(\INVALIDADDR

```
  [LAMBDA (ADDR)                                                 (* bvm%: " 6-AUG-83 22:25")
    (\MP.ERROR \MP.INVALIDADDR "Invalid address" ADDR T])
```

(\INVALIDVP

```
  [LAMBDA (VP)                                                   (* bvm%: " 6-AUG-83 22:25")
    (\MP.ERROR \MP.INVALIDVP "Invalid VP" VP])
```

(\FLUSHPAGE

```
  [LAMBDA (RPTINDEX FROMFLUSHVM)                                 (* bvm%: "13-Aug-85 16:35")
```

;;; Write out real page RPTINDEX if it is dirty.

```
    (PROG ((RPTR (fetch RPTRBASE of RPTINDEX))
           VP FP NEWFP)
          (COND
             ([AND (fetch (RPT OCCUPIED) of RPTR)
                   (fetch (VMEMFLAGS DIRTY) of (\READFLAGS (SETQ VP (fetch (RPT VP) of RPTR]
                                                                 ; Yes, page is dirty
              (SETQ FP (fetch (RPT FILEPAGE) of RPTR))
              [COND
                 [(AND \VMEM.PURE.LIMIT (NOT FROMFLUSHVM))       ; Don't sully vmem;  write page out beyond the original end of
                                                                 ; vmem
                  (COND
                     ((ILEQ FP \VMEM.PURE.LIMIT)
                      (COND
                         ((fetch (RPT LOCKED) of RPTR)
                          (\MP.ERROR \MP.WRITING.LOCKED.PAGE)))
                      (SETQ NEWFP (add (fetch NActivePages of \InterfacePage)
                                       1))
                      (COND
                         ((IGREATERP NEWFP (IDIFFERENCE \LASTVMEMFILEPAGE \GUARDVMEMFULL))
                          (\SET.VMEM.FULL.STATE)))
                      (SETQ \NEWVMEMPAGEADDED T)
                      (\PUTBASE (.PAGEMAPBASE. VP)
                             0 NEWFP)
                      (\PUTBASE \FPTOVP NEWFP VP)
                      (\PUTBASE \FPTOVP FP \NO.VMEM.PAGE)
                      (replace (RPT FILEPAGE) of RPTR with (SETQ FP NEWFP]
                 ((.VMEM.CONSISTENTP.)
                  (replace (IFPAGE Key) of \InterfacePage with (LOGNOT16 \IFPValidKey))
                                                                 ; Invalidate vmem and write out the Interface page
                  (SETQ \DIRTYPAGEHINT 0)                        ; So that the dirty page background writer wakes up
                  (PROG ((IFVP (fetch (POINTER PAGE#) of \InterfacePage)))
                        (\TRANSFERPAGE IFVP \FirstVmemBlock (RPTFROMRP (\READRP IFVP))
                               T NIL]                            ; Write it out
              (COND
                 ((IGREATERP \DIRTYPAGEHINT 0)
                  (add \DIRTYPAGEHINT -1)))
              (\TRANSFERPAGE VP FP RPTINDEX T NIL])
```

(\LOADVMEMPAGE

```
  [LAMBDA (VPAGE FILEPAGE NEWPAGEFLG LOCK? DONTMOVETOPFLG)       (* bvm%: "10-Aug-85 18:08")
```

;; Fault in virtual page VPAGE known to live in FILEPAGE on the vmem. NEWPAGEFLG is true if the page is new, so should just be cleared, not
;; loaded from vmem file. If LOCK? is true, locks down the page as well. In this case, if on Dandelion, we also check for page wanting to live in a
;; particular real page. If DONTMOVETOPFLG is true, the real page we put this page in is not promoted to the front of the LRU queue of pages

```
    (COND
```

```
       ((IGREATERP \PAGEFAULTCOUNTER \UPDATECHAINFREQ)
        (\UPDATECHAIN)))
    (add \PAGEFAULTCOUNTER 1)
    (PROG ((RPTINDEX (\SELECTREALPAGE FILEPAGE LOCK? DONTMOVETOPFLG))
           RPTBASE SPECIALRP)
          (SETQ RPTBASE (fetch RPTRBASE of RPTINDEX))
          [COND
             ((AND LOCK? (OR (EQ \MACHINETYPE \DANDELION)
                             (EQ \MACHINETYPE \DAYBREAK))
                   (SETQ SPECIALRP (\SPECIALRP VPAGE)))          ; Must actually put FILEPAGE into special RP, and thus move
                                                                 ; old contents of SPECIALRP into RPTINDEX
              (LET* ((SRINDEX (RPTFROMRP SPECIALRP))
                     (SRPTR (fetch RPTRBASE of SRINDEX)))
                    (\MOVEREALPAGE SRINDEX SRPTR RPTINDEX RPTBASE)
                    (SETQ RPTINDEX SRINDEX)
                    (SETQ RPTBASE SRPTR]                         ; Fill in new RPTINDEX with appropriate data
          (replace (RPT VP) of RPTBASE with VPAGE)
          (replace (RPT FILEPAGE) of RPTBASE with FILEPAGE)
          (replace (RPT LOCKED) of RPTBASE with LOCK?)
          (COND
             ([AND DOLOCKCHECKS (NOT LOCK?)
                   (EQ (LRSH VPAGE 8)
                       (CONSTANT (\HILOC \PAGEMAP]
              (\MP.ERROR \MP.MAPNOTLOCKED "Page of page map being loaded but not locked" VPAGE)))
          (\TRANSFERPAGE VPAGE FILEPAGE RPTINDEX NIL NEWPAGEFLG])
```

(\MOVEREALPAGE

```
  [LAMBDA (SOURCEINDEX SOURCERPT DESTINDEX DESTRPT)              (* bvm%: "14-Aug-85 13:53")
```

;;; Moves the page, if any, currently living in real page table SOURCEINDEX & SOURCERPT into the page indicated by DESTINDEX & DESTRPT. The ;;; destination is assumed to have been vacated

```
    (CHECK (NOT (fetch (RPT LOCKED) of SOURCERPT)))
    (replace (RPT LOCKED) of DESTRPT with NIL)
    [COND
       ((fetch (RPT OCCUPIED) of SOURCERPT)                      ; Page was not vacant to start with
        (LET* ((SOURCEVP (fetch (RPT VP) of SOURCERPT))
               (SOURCEFLAGS (\READFLAGS SOURCEVP)))
              (replace (RPT VP) of DESTRPT with SOURCEVP)
              (replace (RPT FILEPAGE) of DESTRPT with (fetch (RPT FILEPAGE) of SOURCERPT))
              (\WRITEMAP \EMBUFVP (RPFROMRPT DESTINDEX)
                     0)                                          ; Map buffer to target page
              (\BLT \EMBUFBASE (create POINTER
                                       PAGE# _ SOURCEVP)
                    WORDSPERPAGE)                                ; move data to buffer page
              (\WRITEMAP \EMBUFVP \EMBUFRP 0)                    ; Restore buffer to its proper page
              (\WRITEMAP SOURCEVP (RPFROMRPT DESTINDEX)
                     SOURCEFLAGS)                                ; Set flags and new RP for page

              ]
    DESTINDEX])
```

(\LOOKUPPAGEMAP

```
  [LAMBDA (VP)                                                   (* bvm%: "20-Oct-86 18:26")
```

;; Returns the pagemap entry for VP, which is expected to be in bounds. High bit of result is the lock bit. Zero denotes absence

```
    (LET [(PRIMENTRY (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP]
         (COND
            ((EQ PRIMENTRY \EMPTYPMTENTRY)
             0)
            (T (\GETBASE \PAGEMAP (IPLUS PRIMENTRY (fetch (VP SECONDARYKEY) of VP])
```

(\VALIDADDRESSP

```
  [LAMBDA (BASE)                                                 (* bvm%: "16-Jun-86 11:30")
    (NEQ 0 (\LOOKUPPAGEMAP (fetch (POINTER PAGE#) of BASE])
```

(\LOCKEDPAGEP

```
  [LAMBDA (VP TEMP)                                              (* bvm%: "18-Feb-85 18:08")
```

;;; True if VP is locked. If TEMP is NIL consults only the locked page table; otherwise, also checks for 'temporary' locked page

```
    (OR (NEQ 0 (LOGAND (.LOCKEDVPMASK. VP)
                       (\GETBASE (.LOCKEDVPBASE. VP)
                                 0)))
        (UNLESSRDSYS (AND TEMP (NOT (fetch (VMEMFLAGS VACANT) of (\READFLAGS VP)))
                          (fetch (RPT LOCKED) of (fetch RPTRBASE of (RPTFROMRP (\READRP VP])
```

(\SELECTREALPAGE

```
  [LAMBDA (NEWFP LOCK? DONTMOVETOPFLG)                           (* bvm%: "10-Aug-85 18:08")
```

;; Selects a real page, flushing it if necessary, and returns the RPT index of the page. NEWFP, if supplied, is the filepage that will be read into ;; here. This might influence page choice by minimizing seek time. LOCK? means caller intends to lock the page, which constrains which real ;; pages it can fall into. The selected page is moved to the back of the LRU queue, so that it won't be selected again soon, unless

```
;; DONTMOVETOPFLG is true.  If DONTMOVETOPFLG is REMOVE then the page is spliced out of the chain forever.
(PROG ((TRIES 0)
       (CNTR \MAXCLEANPROBES)
       (DISTANCE \MINSHORTSEEK)
       PREVRPT PREVINDEX RPTINDEX RPTBASE FP FLAGS)
  RETRY
     (SETQ PREVRPT \REALPAGETABLE)
     (until (EQ (SETQ RPTINDEX (fetch (RPT NEXTRP) of PREVRPT))
                \PAGETABLESTOPFLG)
        do (SETQ RPTBASE (fetch RPTRBASE of RPTINDEX))
           [COND
              ((fetch (RPT EMPTY) of RPTBASE)
               (RETURN PREVRPT))
              ((NOT (fetch (RPT OCCUPIED) of RPTBASE))
               (\MP.ERROR \MP.CHAIN.UNAVAIL "UNAVAILABLE page on Chain"))
              ([AND (NOT (fetch (RPT LOCKED) of RPTBASE))
                    [NOT (fetch (VMEMFLAGS REFERENCED) of (SETQ FLAGS (\READFLAGS (fetch (RPT VP)
                                                                                      of RPTBASE]
                    (OR (NOT LOCK?)
                        (.LOCKABLERP. (RPFROMRPT RPTINDEX]
               ;; Page is unlocked and unreferenced, so is good candidate for flushing.  LOCK?  check is to avoid locking a page into a real
               ;; page that might be desired by code that cares about real pages
               (COND
                  ([OR (NOT (fetch (VMEMFLAGS DIRTY) of FLAGS))
                       (PROGN (SETQ FP (fetch (RPT FILEPAGE) of RPTBASE))
                              (COND
                                 ((SELECTQ \VMEM.INHIBIT.WRITE
                                       (NIL [SELECTQ \VMEM.FULL.STATE
                                                 (NIL        ; Normal, can write anything
                                                      T)
                                                 (T          ; Vmem is full and clean, don't write anything
                                                    NIL)
                                                 (PROGN      ; Vmem is full, but sullied, so might as well write anything for
                                                             ; which there is space
                                                        (AND (ILEQ FP \LASTVMEMFILEPAGE)
                                                             (OR (NULL \VMEM.PURE.LIMIT)
                                                                 (IGREATERP FP \VMEM.PURE.LIMIT])
                                       (NEW                  ; Only allowed to write old pages, since new pages might just
                                                             ; have to get moved a second time
                                            (ILEQ FP \VMEM.PURE.LIMIT))
                                       (PROGN                ; We are forbidden from writing any page
                                              NIL))
                                  (COND
                                     ((OR (ILEQ CNTR 0)
                                          (NULL NEWFP)
                                          (ILESSP (IABS (IDIFFERENCE FP NEWFP))
                                                  DISTANCE)) ; Page is near replacement, or we have given up trying for
                                                             ; closeness
                                      T)
                                     (T                      ; Page is too far away from replacement page
                                        (SETQ CNTR (SUB1 CNTR))
                                        [COND
                                           ((ILESSP DISTANCE \MAXSHORTSEEK)
                                                             ; Get more liberal
                                            (SETQ DISTANCE (LLSH DISTANCE 1]
                                        NIL]
                     [COND
                        (DOLOCKCHECKS (COND
                                         ((fetch (RPT LOCKED) of RPTBASE)
                                          (\MP.ERROR \MP.FLUSHLOCKED "Attempt to displace locked page"
                                                 RPTBASE))
                                         ((EQ (fetch (RPT VPSEG) of RPTBASE)
                                              (CONSTANT (\HILOC \PAGEMAP)))
                                          (\MP.ERROR \MP.MAPNOTLOCKED "A page of the page map is not locked"
                                                 RPTBASE]
                     (\FLUSHPAGE RPTINDEX)
                     (\WRITEMAP (fetch (RPT VP) of RPTBASE)
                            0 \VMAP.VACANT)
                     (replace (RPT EMPTY) of RPTBASE with T)
                     (RETURN PREVRPT]
           (SETQ PREVRPT RPTBASE)
           (SETQ PREVINDEX RPTINDEX)
        finally ;; Couldn't find an unreffed page because all pages were touched since last \UPDATECHAIN.  Do another, which clears ref
                ;; bits, and try again
                (COND
                   ((EQ TRIES 0)
                    (SETQ TRIES 1)
                    (\UPDATECHAIN))
                   [(AND (EQ TRIES 1)
                         \VMEM.INHIBIT.WRITE)
                    (SETQ \VMEM.INHIBIT.WRITE)
                    (COND
                       ((AND (NEQ \MACHINETYPE \DANDELION)
                             (NEQ \MACHINETYPE \DAYBREAK))
```

```
                        ;; Don't call RAID on a DLion, since the interface is so bad.  Dorado user might want to know that we're smashing
                        ;; \VMEM.INHIBIT.WRITE

                        (RAID "No clean vmem pages to reuse, must write one.  ^N to continue"]
                    (T (\MP.ERROR \MP.SELECTLOOP "Loop in \SELECTREALPAGE")))
                (GO RETRY))
        (SELECTQ DONTMOVETOPFLG
            (NIL                                                    ; Move this page to head of chain, so that it won't be picked
                                                                    ; again soon
                 (replace (RPT NEXTRP) of PREVRPT with (fetch (RPT NEXTRP) of RPTBASE))
                                                                    ; Splice RPTINDEX out of chain
                 (replace (RPT NEXTRP) of \RPTLAST with RPTINDEX)
                                                                    ; Put new page at end of chain
                 (replace (RPT NEXTRP) of (SETQ \RPTLAST RPTBASE) with \PAGETABLESTOPFLG))
            (REMOVE                                                 ; Splice this page out of chain altogether
                    (replace (RPT NEXTRP) of PREVRPT with (fetch (RPT NEXTRP) of RPTBASE))
                    (replace (RPT NEXTRP) of RPTBASE with \PAGETABLESTOPFLG))
             NIL)
        (RETURN RPTINDEX])
```

(\SPECIALRP

```
[LAMBDA (VP)                                                        (* edited%: " 9-Aug-85 17:14")
```

;; for \DANDELION, some virtual pages must be mapped into special real pages. This function returns the corresponding real page

```
(SELECTC (FOLDLO VP PAGESPERSEGMENT)
     ((FOLDLO \VP.STACK PAGESPERSEGMENT)
         (IPLUS VP (IDIFFERENCE \RP.STACK \VP.STACK)))
     ((FOLDLO \VP.DISPLAY PAGESPERSEGMENT)
         (IPLUS VP (IDIFFERENCE \RP.DISPLAY \VP.DISPLAY)))
    NIL])
```

(\TRANSFERPAGE

```
[LAMBDA (VP FILEPAGE RPTINDEX WRITE? NEWPAGE?)                      (* MPL "27-Jul-85 21:28")
```

;; Transfers virtual page VP between page FILEPAGE of the vmem and real page RPTINDEX. WRITE? indicates direction of transfer. If
;; NEWPAGE?, then page does not exist on file, and is simply cleared

```
(PROG (NEWFLAGS)
      (COND
         (WRITE? (FLIPCURSORBAR 15))
         (T (FLIPCURSORBAR 0)))
      (SETQ NEWFLAGS (COND
                        (NEWPAGE? \VMAP.DIRTY)
                        (WRITE? (LOGAND (\READFLAGS VP)
                                        (LOGNOT16 \VMAP.DIRTY)))
                        (T 0)))
      (COND
         ((AND WRITE? (fetch (RPT LOCKED) of (fetch RPTRBASE of RPTINDEX)))
          ;; Writing a locked page: can't diddle map, because others might die, so do this in the straightforward way
          (\BLT \EMBUFBASE (create POINTER
                                   PAGE# _ VP)
                WORDSPERPAGE)                                       ; Copy page into buffer, then write the buffer out
          (\ACTONVMEMFILE FILEPAGE \EMBUFBASE 1 T)
          (SETQ \LASTACCESSEDVMEMPAGE FILEPAGE))
         ((NOT NEWPAGE?)                                            ; Map the buffer page into the target real page, read/write the
                                                                    ; page, then set the map back
          (\WRITEMAP VP 0 \VMAP.VACANT)                             ; Unmap VP so that we don't have two virtual pages pointing at
                                                                    ; same real page

          (\WRITEMAP \EMBUFVP (RPFROMRPT RPTINDEX)
                 0)                                                 ; Map buffer to target page
          (\ACTONVMEMFILE FILEPAGE \EMBUFBASE 1 WRITE?)             ; Do the i/o
          (\WRITEMAP \EMBUFVP \EMBUFRP 0)                           ; Restore buffer to its proper page
          (SETQ \LASTACCESSEDVMEMPAGE FILEPAGE)))
      (\WRITEMAP VP (RPFROMRPT RPTINDEX)
             NEWFLAGS)                                              ; Set flags for page
      (COND
         (NEWPAGE?                                                  ; Not on file yet, so clear it.  Couldn't do this sooner because the
                                                                    ; flags weren't set
                (\CLEARWORDS (create POINTER
                                     PAGE# _ VP)
                       WORDSPERPAGE)))
      (COND
         (WRITE? (FLIPCURSORBAR 15)
                (\BOXIPLUS (LOCF (fetch SWAPWRITES of \MISCSTATS))
                       1))
         (T (FLIPCURSORBAR 0)
            (\BOXIPLUS (LOCF (fetch PAGEFAULTS of \MISCSTATS))
                   1])
```

(\UPDATECHAIN

```
[LAMBDA NIL                                                         (* bvm%: "30-Jul-85 15:20")
                                                                    ; Sorts the page chain by reference bit

  (CHECK (NOT \INTERRUPTABLE))
  (PROG ((RPTINDEX (fetch (RPT NEXTRP) of \REALPAGETABLE))
         (CHAIN0 \REALPAGETABLE)
```

```
          (CHAIN1 (\ADDBASE \REALPAGETABLE 2))
          RPTR VP FLAGS HEAD1)
         (SETQ HEAD1 CHAIN1)
```

;; HEAD1 = CHAIN1 is just a holding cell for the second Chain we temporarily create inside here. Use the unused third word of the dummy ;; header entry of \REALPAGETABLE

```
        (replace (RPT NEXTRP) of CHAIN0 with \PAGETABLESTOPFLG)
        (replace (RPT NEXTRP) of CHAIN1 with \PAGETABLESTOPFLG)
        (do (SETQ RPTR (fetch RPTRBASE of RPTINDEX))
            (SETQ VP (fetch (RPT VP) of RPTR))
            [SETQ FLAGS (COND
                           ((fetch (RPT EMPTY) of RPTR)
                            0)
                           (T (\READFLAGS VP]
            (COND
               ((OR (fetch (RPT LOCKED) of RPTR)
                    (PROGN (COND
                              ([AND DOLOCKCHECKS (EQ (fetch (RPT VPSEG) of RPTR)
                                                     (CONSTANT (\HILOC \PAGEMAP]
                               (\MP.ERROR \MP.MAPNOTLOCKED "A page of the page map is not locked" RPTR)))
                           (fetch (VMEMFLAGS REFERENCED) of FLAGS)))
                                                                  ; Page referenced or locked, put on CHAIN1
                (\WRITEMAP VP (RPFROMRPT RPTINDEX)
                       (LOGAND FLAGS (LOGNOT16 \VMAP.REF)))       ; Turn off ref bit
                (replace (RPT NEXTRP) of CHAIN1 with RPTINDEX)
                (SETQ CHAIN1 RPTR))
               (T                                                 ; Page was not referenced recently, put on CHAIN0
                  (replace (RPT NEXTRP) of CHAIN0 with RPTINDEX)
                  (SETQ CHAIN0 RPTR)))
            (SETQ RPTINDEX (fetch (RPT NEXTRP) of RPTR))          ; Look at next page in old chain
           repeatuntil (EQ RPTINDEX \PAGETABLESTOPFLG))
        (replace (RPT NEXTRP) of CHAIN1 with \PAGETABLESTOPFLG)   ; End of the line
        (replace (RPT NEXTRP) of CHAIN0 with (fetch (RPT NEXTRP) of HEAD1))
                                                                  ; Link end of CHAIN0 to beginning of CHAIN1
        (SETQ \RPTLAST (COND
                          ((EQ HEAD1 CHAIN1)                      ; Nothing on CHAIN1 ??!!
                           CHAIN0)
                          (T CHAIN1)))                            ; Pointer to end of complete chain
        (SETQ \DIRTYPAGECOUNTER (SETQ \PAGEFAULTCOUNTER 0])

)

;; Allocating and locking new pages

(DEFINEQ

(\NEWPAGE
  [LAMBDA (BASE NOERROR LOCK?)                                    ; Edited 24-Oct-92 12:45 by sybalsky:mv:envos

;;; Creates and returns a new page located at virtual addr BASE

    ;; If LOCK?, lock the page into real memory (A NOP on nonXerox machines!)

    (UNINTERRUPTABLY
        (COND
           [(NOT (\MISCAPPLY* (FUNCTION \DONEWPAGE)
                     BASE LOCK?))                                 ; Failed, page exists
            (COND
               ((NOT NOERROR)
                (\MP.ERROR \MP.NEWPAGE "Attempt to allocate already existing page" BASE T)))
            (COND
               (LOCK? (\LOCKPAGES BASE 1]
            ((IGREATERP (fetch (IFPAGE NActivePages) of \InterfacePage)
                   (IDIFFERENCE \LASTVMEMFILEPAGE \GUARDVMEMFULL))
                                                                  ; Vmem getting full!
             (\SET.VMEM.FULL.STATE)))
        BASE)])

(\DONEWPAGE
  [LAMBDA (BASE LOCK? INTERNALFLG)                                (* bvm%: "13-Aug-85 16:32")

;;; Allocates new page at BASE, locking it if LOCK? is true. Returns vmemfile page# on success, NIL if page already exists. Must be run in safe
;;; context! because it can cause vmem activity

    (AND \DOFAULTINIT (\FAULTINIT))                               ; Only an issue when INIT.SYSOUT starts. Perhaps there is a
                                                                  ; better place to put this
    (PROG ((VP (fetch (POINTER PAGE#) of BASE))
           MAPBASE LOCKBASE FILEPAGE NEXTPM ERRCODE)
          (RETURN (COND
                     ((fetch (VP INVALID) of VP)
                      (\INVALIDVP VP)
                      NIL)
                     (T (SETQ MAPBASE (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)))
                        (COND
                           ((EQ MAPBASE \EMPTYPMTENTRY)           ; Need to create a new second-level block
                            (SETQ NEXTPM (fetch (IFPAGE NxtPMAddr) of \InterfacePage))
                            [COND
```

```
                ((EVENP NEXTPM WORDSPERPAGE)
                 ;; Need a new secondary pagemap page.  This recursion is ok, because we know that SETUPPAGEMAP
                 ;; assures that the pagemap pages for all the pages in secondary map space were created at MAKEINIT time
                 (OR (\DONEWPAGE (\ADDBASE \PAGEMAP NEXTPM)
                                 T T)
                     (RETURN (\MP.ERROR \MP.NEWMAPPAGE "\DONEWPAGE failed to allocate new map
                                                       page"]
                (\PUTBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)
                     NEXTPM)
                (replace (IFPAGE NxtPMAddr) of \InterfacePage with (IPLUS NEXTPM \PMblockSize))
                (SETQ MAPBASE NEXTPM)))
           [SETQ MAPBASE (\ADDBASE \PAGEMAP (IPLUS MAPBASE (fetch (VP SECONDARYKEY) of VP]
           (COND
              ((NEQ (\GETBASE MAPBASE 0)
                    0)                                    ; Page exists
               (RETURN NIL)))
           (SETQ FILEPAGE (add (fetch (IFPAGE NActivePages) of \InterfacePage)
                              1))
           (replace (IFPAGE NDirtyPages) of \InterfacePage with FILEPAGE)
                                                          ; Currently a redundant field
           [COND
              (LOCK? (SETQ FILEPAGE (\MAKESPACEFORLOCKEDPAGE VP FILEPAGE))
                     (\PUTBASE (SETQ LOCKBASE (.LOCKEDVPBASE. VP))
                            0
                            (LOGOR (.LOCKEDVPMASK. VP)
                                   (\GETBASE LOCKBASE 0]
           (\PUTBASE \FPTOVP FILEPAGE VP)
           (\PUTBASE MAPBASE 0 FILEPAGE)
           (\LOADVMEMPAGE VP FILEPAGE T LOCK?)
           (COND
              (INTERNALFLG (SETQ \NEWVMEMPAGEADDED T))
              (T                                          ; Make sure \FPTOVP extended if necessary
                 (\ASSURE.FPTOVP.PAGE)))
           FILEPAGE])
```

(\ASSURE.FPTOVP.PAGE

```
[LAMBDA NIL                                               (* bvm%: "13-Aug-85 16:29")
```

;; Called at the end of some swapping operation that added one or more pages to the vmem file, setting \NEWVMEMPAGEADDED true. If we're ;; going to need a new page of \FPTOVP soon, do it now while there's still maneuvering room. The allowance below is for the worst case, which ;; can happen when VMEM.PURE.STATE is on and \NEWPAGE was called needing a new pagemap page as well, in which case we could have ;; as many as the following new vmem pages before we're home safe --- 1: \NEWPAGE added a page --- 2: a page was displaced by the new page ;; and written to the end of the vmem --- 3: a new pagemap page was needed --- 4: it displaced a page to end of vmem --- 5: the new \FPTOVP ;; page below --- 6: a page displaced by same. --- --- Alternatively, it could have been the new \FPTOVP page that needed a new pagemap block. ;; Will never have both needing a new pagemap block, since there are several pagemap blocks per page

```
(LET ((FILEPAGE (fetch (IFPAGE NActivePages) of \InterfacePage)))
     (COND
        ((IGREATERP (IMOD FILEPAGE WORDSPERPAGE)
                    (IDIFFERENCE WORDSPERPAGE 7))         ; This is a no-op if the page has already been allocated
         (\DONEWPAGE (\ADDBASE \FPTOVP (CEIL FILEPAGE WORDSPERPAGE))
                T T)))
     (SETQ \NEWVMEMPAGEADDED NIL])
```

(\MAKESPACEFORLOCKEDPAGE

```
[LAMBDA (VP FILEPAGE)                                     (* bvm%: "29-Jun-86 17:44")
```

;; VP is a page to be locked, FILEPAGE its home. Returns a possibly new file page where VP will now live, after having kicked the former resident ;; of the new file page into VP's old FILEPAGE

```
(PROG (DESIREDFP OLDVP FPBASE)
      [SETQ DESIREDFP (SELECTC (FOLDLO VP PAGESPERSEGMENT)
                               ((FOLDLO \VP.STACK PAGESPERSEGMENT)
                                    (IPLUS VP (IDIFFERENCE (DLFPFROMRP \RP.STACK)
                                                           \VP.STACK)))
                               ((FOLDLO \VP.DISPLAY PAGESPERSEGMENT)
                                                          ; Display lives in a fixed place in file, but does not land there
                                                          ; initially
                                    (IPLUS VP (IDIFFERENCE (DLFPFROMRP \RP.TEMPDISPLAY)
                                                           \VP.DISPLAY)))
                               ((FOLDLO \VP.FPTOVP PAGESPERSEGMENT)
                                                          ; A new page of FPTOVP has to be continguous on file with
                                                          ; other such pages
                                    (IPLUS VP (IDIFFERENCE (DLFPFROMRP \RP.FPTOVP)
                                                           \VP.FPTOVP)))
                               (COND
                                  ((AND (ILEQ FILEPAGE (fetch LastLockedFilePage of \InterfacePage))
                                        (IGREATERP FILEPAGE (DLFPFROMRP \RP.MISCLOCKED)))
                                                          ; Page is in a good place already.  It probably was once locked,
                                                          ; then unlocked
                                   (RETURN FILEPAGE))
                                  (T                      ; Put it after all the other locked pages
                                     (add (fetch LastLockedFilePage of \InterfacePage)
                                          1]
      (COND
         ((AND (fetch FPOCCUPIED of (SETQ FPBASE (\ADDBASE \FPTOVP DESIREDFP)))
```

```
                (NEQ (SETQ OLDVP (fetch FPVIRTUALPAGE of FPBASE))
                     VP))                                    ; Someone else lives here, so move it out
            (\MOVEVMEMFILEPAGE OLDVP DESIREDFP FILEPAGE)))
        (RETURN DESIREDFP])
```

(\MOVEVMEMFILEPAGE

```
  [LAMBDA (VP OLDFP NEWFP)                                   (* bvm%: "18-Nov-84 14:14")
    (PROG ((FLAGS (\READFLAGS VP))
           RP)
          (COND
             ((fetch (VMEMFLAGS VACANT) of FLAGS)            ; Page not resident, so pull it in
              (\LOADVMEMPAGE VP OLDFP)
              (SETQ FLAGS \VMAP.CLEAN))
             ((\LOCKEDPAGEP VP)
              (\MP.ERROR \MP.BADLOCKED "Locked page is in the way" VP)))
          (\WRITEMAP VP (SETQ RP (\READRP VP))
                 (LOGOR FLAGS \VMAP.DIRTY))                  ; Mark page dirty, so that it will eventually be written to its new
                                                             ; home
          (replace (RPT FILEPAGE) of (fetch RPTRBASE of (RPTFROMRP RP)) with NEWFP)
                                                             ; Tell RPT where VP now lives
          (\PUTBASE (.PAGEMAPBASE. VP)
                 0 NEWFP)                                    ; Tell \PAGEMAP about it
          (\PUTBASE \FPTOVP NEWFP VP)                        ; ... and \FPTOVP

      ])
```

(\NEWEPHEMERALPAGE

```
  [LAMBDA (BASE NOERROR)                                     (* bvm%: "26-NOV-82 15:40")
```

;;; Creates and returns a new page located at virtual addr BASE, mapping it permanently into some real page but leaving it out of the vmem file

```
    (\MISCAPPLY* (FUNCTION \DONEWEPHEMERALPAGE)
           BASE NOERROR])
```

(\DONEWEPHEMERALPAGE

```
  [LAMBDA (BASE NOERROR)                                     (* bvm%: "30-Oct-86 16:47")
```

;;; Creates and returns a new page located at virtual addr BASE, mapping it permanently into some real page but leaving it out of the vmem file

```
    (LET ((VP (fetch (POINTER PAGE#) of BASE))
          MAPBASE PREVRP RPTINDEX RPTR)
         (COND
            ((fetch (VP INVALID) of VP)
             (\INVALIDVP VP)
             NIL)
            ([OR (AND (NEQ (SETQ MAPBASE (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)))
                           \EMPTYPMTENTRY)
                      (NEQ (\GETBASE \PAGEMAP (IPLUS MAPBASE (fetch (VP SECONDARYKEY) of VP)))
                           0))
                 (NOT (fetch (VMEMFLAGS VACANT) of (\READFLAGS VP]
                                                             ; Page is in the vmem already, so no hope
             (COND
                ((NOT NOERROR)
                 (\MP.ERROR \MP.NEWPAGE "Page already exists " BASE T)))
             BASE)
            (T (COND
                  ((IGREATERP \PAGEFAULTCOUNTER \UPDATECHAINFREQ)
                   (\UPDATECHAIN)))
               (add \PAGEFAULTCOUNTER 1)
               (SETQ RPTINDEX (\SELECTREALPAGE NIL T 'REMOVE))  ; Find a page to put this in
               (SETQ RPTR (fetch RPTRBASE of RPTINDEX))         ; Fill in new RPTINDEX with appropriate data
               (replace (RPT VP) of RPTR with \RPT.UNAVAILABLE)
               (replace (RPT FILEPAGE) of RPTR with VP)         ; For debugging only
               (FLIPCURSORBAR 0)
               (\WRITEMAP VP (RPFROMRPT RPTINDEX)
                      \VMAP.DIRTY)                           ; Set flags for page
               (\CLEARWORDS (create POINTER
                                    PAGE# _ VP)
                      WORDSPERPAGE)                          ; Clear new page
               (FLIPCURSORBAR 0)
               (\BOXIPLUS (LOCF (fetch PAGEFAULTS of \MISCSTATS))
                      1)
               (COND
                  (\NEWVMEMPAGEADDED (\ASSURE.FPTOVP.PAGE)))
               BASE])
```

(\LOCKPAGES

```
  [LAMBDA (BASE NPAGES)                                      (* bvm%: "26-NOV-82 15:17")
```

;; Needs to be done in safe stack context because might cause vmem transfer

```
    (\MISCAPPLY* (FUNCTION \DOLOCKPAGES)
           BASE NPAGES)
    BASE])
```

(\DOLOCKPAGES

```
[LAMBDA (BASE NPAGES)                                   ; Edited 21-Oct-87 15:49 by bvm:
  (for I from 0 to (SUB1 NPAGES) bind (VP _ (fetch (POINTER PAGE#) of BASE))
                                 FILEPAGE MAPBASE RPTBASE RPINDEX RP MASK LOCKBASE
     do [COND
           ((fetch (VP INVALID) of VP)
            (\INVALIDVP VP))
           [(EQ (SETQ MAPBASE (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)))
                \EMPTYPMENTRY)
            (\INVALIDADDR (ADDBASE BASE (UNFOLD I WORDSPERPAGE]
           (T [SETQ MAPBASE (\ADDBASE \PAGEMAP (IPLUS MAPBASE (fetch (VP SECONDARYKEY) of VP]
              (SETQ FILEPAGE (\GETBASE MAPBASE 0))
              (COND
                 ((EQ 0 (LOGAND (SETQ MASK (.LOCKEDVPMASK. VP))
                                (\GETBASE (SETQ LOCKBASE (.LOCKEDVPBASE. VP))
                                          0)))                    ; Not locked yet
                  (COND
                     ((fetch VACANT of (\READFLAGS VP))           ; Bring locked page into core so we can move it if necessary
                      (\LOADVMEMPAGE VP FILEPAGE NIL T)))
                  [SETQ RPINDEX (RPTFROMRP (SETQ RP (\READRP VP]
                  (SETQ RPTBASE (fetch RPTRBASE of RPINDEX))
                  [COND
                     ((AND (NOT (.LOCKABLERP. RP))
                           (NOT (\SPECIALRP VP)))
                      ;; Page already swapped in, but lives in a real page that might need to get bumped (e.g., for stack), so move it now.  If
                      ;; \SPECIALRP is true then we know that the page got swapped into the right place, so no need to move it.
                      (LET* ((NEWINDEX (\SELECTREALPAGE NIL T))
                             (NEWRPT (fetch RPTRBASE of NEWINDEX)))
                            (\MOVEREALPAGE RPINDEX RPTBASE NEWINDEX NEWRPT)
                            (replace (RPT EMPTY) of RPTBASE with T)
                                                                 ; Mark vacated RPT entry empty
                            (SETQ RPTBASE NEWRPT)
                            (SETQ RP (RPFROMRPT NEWINDEX]
                  (COND
                     ((NEQ FILEPAGE (SETQ FILEPAGE (\MAKESPACEFORLOCKEDPAGE VP FILEPAGE)))
                      ;; Moving to a new page, so have to mark this locked page dirty so that it will eventually get written to its new home
                      (\WRITEMAP VP RP (LOGOR \VMAP.DIRTY \VMAP.REF))
                      (replace (RPT FILEPAGE) of RPTBASE with FILEPAGE)
                      (\PUTBASE \FPTOVP FILEPAGE VP)
                      (\PUTBASE MAPBASE 0 FILEPAGE)))
                  (\PUTBASE LOCKBASE 0 (LOGOR MASK (\GETBASE LOCKBASE 0)))
                                                                 ; Set lock bit in page map
                  (replace (RPT LOCKED) of RPTBASE with T]
        (add VP 1)
     finally (COND
                (\NEWVMEMPAGEADDED                               ; If we had to load or rearrange pages, vmem could have gotten
                                                                 ; bigger if VMEM.PURE.STATE on
                   (\ASSURE.FPTOVP.PAGE])
```

(\TEMPLOCKPAGES

```
[LAMBDA (BASE NPAGES)                                   (* bvm%: "10-Aug-85 18:17")
```

;;; 'Temporarily' locks BASE for NPAGES, i.e. ensures that the swapper will not move the pages. Information vanishes at logout etc.

```
    (\MISCAPPLY* (FUNCTION \DOTEMPLOCKPAGES)
           BASE NPAGES])
```

(\DOTEMPLOCKPAGES

```
[LAMBDA (BASE NPAGES)                                   ; Edited 21-Oct-87 15:49 by bvm:
```

;; 'Temporarily' locks BASE for NPAGES, i.e. ensures that the swapper will not move the pages. Information vanishes at logout etc. This function
;; must be locked because it manipulates the page table table. Runs in MISC context

```
    (to NPAGES as VP from (fetch (POINTER PAGE#) of BASE) bind RPTBASE RPINDEX RP
       do (\TOUCHPAGE BASE)                                      ; Touch page in case not resident
          [SETQ RPINDEX (RPTFROMRP (SETQ RP (\READRP VP]
          (SETQ RPTBASE (fetch RPTRBASE of RPINDEX))
          [COND
             ((NOT (.LOCKABLERP. RP))
              ;; Page already swapped in, but lives in a real page that might need to get bumped (e.g., for stack), so move it now
              (LET* ((NEWINDEX (\SELECTREALPAGE NIL T))
                     (NEWRPT (fetch RPTRBASE of NEWINDEX)))
                    (\MOVEREALPAGE RPINDEX RPTBASE NEWINDEX NEWRPT)
                    (replace (RPT EMPTY) of RPTBASE with T)      ; Mark vacated RPT entry empty
                    (SETQ RPTBASE NEWRPT]
          (replace (RPT LOCKED) of RPTBASE with T)
          (SETQ BASE (\ADDBASE BASE WORDSPERPAGE])
```

(\TEMPUNLOCKPAGES

```
[LAMBDA (BASE NPAGES)                                   (* bvm%: "30-Jul-85 16:58")
```

```
;; Unlocks pages that were locked by \TEMPLOCKPAGES.  This function must be locked because it manipulates the page table
(while (IGREATERP NPAGES 0) bind (VP _ (fetch (POINTER PAGE#) of BASE))
                            RPTR
   do (UNINTERRUPTABLY
        (\TOUCHPAGE BASE)                                   ; Touch page in case not resident.  Should only happen if page
                                                            ; wasn't locked to begin with
        (COND
           ((AND (NEQ (SETQ RPTR (\READRP VP))
                      0)
                 (EQ [fetch (RPT VP) of (SETQ RPTR (fetch RPTRBASE of (RPTFROMRP RPTR]
                     VP))
            (COND
               ([AND DOLOCKCHECKS (EQ (LRSH VP 8)
                                      (CONSTANT (\HILOC \PAGEMAP]
                (\MP.ERROR \MP.UNLOCKINGMAP "Attempt to unlock map page" VP)))
            (replace (RPT LOCKED) of RPTR with NIL))
           (T (HELP "Page table changed out from under me!" VP))))
      (add VP 1)
      (add NPAGES -1)
      (SETQ BASE (\ADDBASE BASE WORDSPERPAGE])
```

(\UNLOCKPAGES

```
  [LAMBDA (BASE NPAGES)                                     (* bvm%: "30-Jul-85 16:58")
```

;;; Unlocks NPAGES virtual pages from BASE onward

```
    (UNINTERRUPTABLY
        (for I from 0 to (SUB1 NPAGES) bind (VP _ (fetch (POINTER PAGE#) of BASE))
                                       MASK LOCKBASE
           do (COND
                 ((fetch (VP INVALID) of VP)
                  (\INVALIDVP VP))
                 ((NEQ 0 (LOGAND (SETQ MASK (.LOCKEDVPMASK. VP))
                                 (\GETBASE (SETQ LOCKBASE (.LOCKEDVPBASE. VP))
                                        0)))                ; Yes, page was locked, so turn the bit off now
                  (COND
                     ([AND DOLOCKCHECKS (EQ (LRSH VP 8)
                                            (CONSTANT (\HILOC \PAGEMAP]
                      (\MP.ERROR \MP.UNLOCKINGMAP "Attempt to unlock map page" VP)))
                  (\PUTBASE LOCKBASE 0 (LOGXOR MASK (\GETBASE LOCKBASE 0)))
                                                            ; Update pagemap, then update real page table
                  (replace (RPT LOCKED) of (fetch RPTRBASE of (RPTFROMRP (\READRP VP))) with NIL)))
              (add VP 1)))])

)

;; Writing out the vmem

(DEFINEQ
```

(\DOFLUSHVM

```
  [LAMBDA (MAIKO.SYSOUTFILE)                                ; Edited 10-Feb-2021 22:38 by lmm
                                                            ; Edited  6-Jan-89 19:23 by Hayata
```

;;; Write everything out in a resumable way. Value is NIL if returned from directly, T if from saved state. Always invoked via \MISCAPPLY*

```
    (CHECK (NOT \INTERRUPTABLE))                            ; NOTE: need stats gathering off in here.  Also avoid touching
                                                            ; pages
    (PROG ((IFPVP (fetch (POINTER PAGE#) of \InterfacePage))
           (SCRATCHBUF \EMUSWAPBUFFERS)
           IFPRPT)
          (replace (IFPAGE MISCSTACKRESULT) of \InterfacePage with T)
                                                            ; This will make it look like we have returned from BCPL if caller
                                                            ; gets control from the saved state
     ;; update interface pge before writing out sysout
          (replace (IFPAGE CurrentFXP) of \InterfacePage with (fetch (IFPAGE MiscFXP) of \InterfacePage))
          (RETURN (SUBRCALL VMEMSAVE MAIKO.SYSOUTFILE])
```

(\RELEASEWORKINGSET

```
  [LAMBDA NIL                                               (* bvm%: "29-Nov-84 10:56")
    (COND
       ((\FLUSHVM)                                          ; Returning from Lisp startup
        T)
       (T                                                   ; Unmap any unlocked page
          (for RPTINDEX from 1 to (SUB1 \RPTSIZE) bind RPTR when (AND (fetch (RPT OCCUPIED)
                                                                        of (SETQ RPTR (fetch RPTRBASE of RPTINDEX))
                                                                     )
                                                                     (NOT (fetch (RPT LOCKED) of RPTR)))
             do (\WRITEMAP (fetch (RPT VP) of RPTR)
                           (RPFROMRPT RPTINDEX)
                           \VMAP.VACANT)
                (replace (RPT EMPTY) of RPTR with T])
```

(\WRITEDIRTYPAGE

```
[LAMBDA (MINDIRTY)                                          (* bvm%: "13-Aug-85 17:51")
   (COND
      ((OR (NOT (.VMEM.CONSISTENTP.))
           (AND \VMEM.PURE.LIMIT (NEQ \VMEM.PURE.LIMIT -1)
                (NOT \VMEM.FULL.STATE)))
       (PROG ((RPTR (OR \LASTDIRTYSCANPTR \REALPAGETABLE))
              (NUMDIRTY (OR \LASTDIRTYCNT 0))
              (CNT \MAXDIRTYSCANCOUNT)
              RP FP FLAGS)
             [COND
                ((AND (NULL \LASTDIRTYSCANPTR)
                      (IGREATERP (IPLUS (add \DIRTYPAGECOUNTER 1)
                                        \PAGEFAULTCOUNTER)
                                 \UPDATECHAINFREQ))                 ; Take this time to update the page chain instead
                 (RETURN (UNINTERRUPTABLY
                             (\MISCAPPLY* (FUNCTION \UPDATECHAIN))))]
             (OR MINDIRTY (SETQ MINDIRTY 1))
         LP  [COND
                [(EQ (SETQ RP (fetch (RPT NEXTRP) of RPTR))
                     \PAGETABLESTOPFLG)                             ; Hit end of chain.  Write out what we found if enough were dirty
                 (COND
                    ((AND (IGEQ NUMDIRTY MINDIRTY)
                          (NEQ NUMDIRTY 0)
                          (SETQ RP \LASTDIRTYFOUND))
                     (GO GOTPAGE))
                    (T (SETQ \LASTDIRTYSCANPTR (SETQ \LASTDIRTYCNT (SETQ \LASTDIRTYFOUND NIL)))
                       [COND
                          ((AND (NEQ NUMDIRTY 0)
                                (ILESSP \DIRTYSEEKMAX (LRSH MAX.SMALL.INTEGER 1)))
                                                                    ; Failed because page not close enough, so widen the tolerance
                           (SETQ \DIRTYSEEKMAX (LLSH \DIRTYSEEKMAX 1]
                          (RETURN]
                ((fetch (RPT EMPTY) of (SETQ RPTR (fetch RPTRBASE of RP)))
                                                                    ; Page is empty.  Should never happen if key is valid
                 (RETURN))
                ((NOT (fetch (RPT LOCKED) of RPTR))                 ; Don't bother writing out locked pages, since they don't help us
                                                                    ; in our swapping quest
                 (SETQ FLAGS (\READFLAGS (fetch (RPT VP) of RPTR)))
                 (COND
                    ((NOT (fetch (VMEMFLAGS DIRTY) of FLAGS))       ; Page not dirty;  skip

                     )
                    [(PROGN (SETQ FP (fetch (RPT FILEPAGE) of RPTR))
                            (IGREATERP (IABS (IDIFFERENCE (COND
                                                              ((AND \VMEM.PURE.LIMIT (ILESSP FP \VMEM.PURE.LIMIT
                                                                                             ))
                                                                    ; We'd have to write page to a new place, not here
                                                               (fetch (IFPAGE NActivePages) of \InterfacePage))
                                                              (T FP))
                                                          \LASTACCESSEDVMEMPAGE))
                                       \DIRTYSEEKMAX))              ; Page too far away, don't write it
                     (COND
                        ((fetch (VMEMFLAGS REFERENCED) of FLAGS)    ; but still count it
                         (add NUMDIRTY 1]
                    ((IGREATERP FP \LASTVMEMFILEPAGE)               ; Can't write it

                     )
                    ((fetch (VMEMFLAGS REFERENCED) of FLAGS)        ; Page dirty but referenced.  Note it, but keep looking for a better
                                                                    ; one
                     (COND
                        ((EQ NUMDIRTY 0)
                         (SETQ \LASTDIRTYFOUND RP)))
                     (add NUMDIRTY 1))
                    (T                                              ; Dirty, not referenced: do it
                       (GO GOTPAGE]
             (COND
                ((EQ (add CNT -1)
                     0)                                             ; Scanned for long enough;  don't lock user out
                 (SETQ \LASTDIRTYSCANPTR RPTR)
                 (SETQ \LASTDIRTYCNT NUMDIRTY)
                 (RETURN)))
             (GO LP)
         GOTPAGE
             (UNINTERRUPTABLY
                 (SETQ \LASTDIRTYSCANPTR (SETQ RPTR (fetch RPTRBASE of RP)))
                                                                    ; Keep traveling pointer
                 (SETQ \LASTDIRTYCNT (SETQ \LASTDIRTYFOUND NIL))
                 (COND
                    ((ILEQ (IABS (IDIFFERENCE (fetch (RPT FILEPAGE) of RPTR)
                                              \LASTACCESSEDVMEMPAGE))
                           \DIRTYSEEKMAX)                           ; Could fail if swapping since the selection has moved the disk
                                                                    ; arm too far
                     (\MISCAPPLY* (FUNCTION \WRITEDIRTYPAGE1)
                                  RP RPTR)))
                 (SETQ \DIRTYSEEKMAX \MAXSHORTSEEK))
```

```
              (RETURN T])
```

(\WRITEDIRTYPAGE1

```
  [LAMBDA (RP RPTR)                                                    (* bvm%: "13-Aug-85 16:41")
                                                                       ; Write out buffer RP.  This fn is locked and called in the misc
                                                                       ; context
    (COND
       ([AND (NOT (fetch (RPT LOCKED) of RPTR))
             (fetch (VMEMFLAGS DIRTY) of (\READFLAGS (fetch (RPT VP) of RPTR]
                                                                       ; Verify that the page is still a candidate, so previous loop could
                                                                       ; be interruptable
        (\FLUSHPAGE RP)
        (COND
           (\NEWVMEMPAGEADDED (\ASSURE.FPTOVP.PAGE])
```

(\COUNTREALPAGES

```
  [LAMBDA (TYPE)                                                       (* bvm%: "18-Dec-84 15:31")
    (SELECTQ TYPE
         ((DIRTY REF)
             [PROG [(FLAGBITS (COND
                                ((EQ TYPE 'DIRTY)
                                 \VMAP.DIRTY)
                                (T \VMAP.REF]
                    (RETURN (NPAGESMACRO (NEQ (LOGAND (\READFLAGS VP)
                                                      FLAGBITS)
                                              0])
         (LOCKED (NPAGESMACRO (fetch (RPT LOCKED) of RPTR)))
         (OCCUPIED (NPAGESMACRO T))
         (\ILLEGAL.ARG TYPE])

)

;; VMEM.PURE.STATE hack

(DEFINEQ
```

(\DOCOMPRESSVMEM

```
  [LAMBDA NIL                                                          (* bvm%: " 7-Apr-84 17:53")
```

;;; Called underneath \DOFLUSHVM to write the pages above the high water mark back to the places vacated below that mark

```
    (PROG ((EMPTYFP (DLFPFROMRP \RP.GCTABLE))
           (LASTFP (fetch NActivePages of \InterfacePage))
           (OLDVIW \VMEM.INHIBIT.WRITE)
           VP)
          [COND
             ((NULL OLDVIW)
              ;; Encourage \SELECTREALPAGE to select only 'old' file pages for displacement, so that we don't needlessly write the same page
              ;; twice
              (SETQ \VMEM.INHIBIT.WRITE 'NEW]
      LP  (COND
             ((IGEQ EMPTYFP LASTFP)
              (SETQ \VMEM.INHIBIT.WRITE OLDVIW)
              (RETURN)))
          [COND
             ((EQ (\GETBASE \FPTOVP EMPTYFP)
                  \NO.VMEM.PAGE)
              (while (EQ (SETQ VP (\GETBASE \FPTOVP LASTFP))
                         \NO.VMEM.PAGE)
                 do (SETQ LASTFP (SUB1 LASTFP)))
              (\MOVEVMEMFILEPAGE VP LASTFP EMPTYFP)
              (replace NActivePages of \InterfacePage with (SETQ LASTFP (SUB1 LASTFP]
          (add EMPTYFP 1)
          (GO LP])
```

(\VMEM.PURE.STATE

```
  [LAMBDA FLG                                                          (* bvm%: " 7-Apr-84 16:59")
    (PROG1 (NOT (NULL \VMEM.PURE.LIMIT))
        [COND
           ((IGREATERP FLG 0)
            ;; Set \VMEM.PURE.LIMIT appropriately.  If turning on, and it wasn't on before, set it to -1 so that it takes effect only at the next
            ;; FLUSHVM
            (SETQ \VMEM.PURE.LIMIT (AND (ARG FLG 1)
                                        (OR \VMEM.PURE.LIMIT (SETQ \VMEM.PURE.LIMIT -1])])

)

;; Handling the backing store getting too full--keep running, but if we overflow, we can never \FLUSHVM because there is no place to write some pages

(DEFINEQ
```

(32MBADDRESSABLE

```
  [LAMBDA NIL                                                    ; Edited  2-May-88 22:03 by MASINTER
    (SELECTC \MACHINETYPE
         (\DORADO T)
         (\DOLPHIN NIL)
         (\DAYBREAK T)
         (NEQ 0 (fetch (IFPAGE DL24BitAddressable) of \InterfacePage])
```

(\SET.VMEM.FULL.STATE

```
  [LAMBDA NIL                                                    (* bvm%: "13-Feb-85 20:12")
                                                                 ; We are running out of vmem, try to extend file.  Do this at next
                                                                 ; convenient time

    (COND
       ((NOT \VMEM.FULL.STATE)                                   ; Get an interrupt to handle this
        (replace VMEMFULL of \INTERRUPTSTATE with T)
        (SETQ \PENDINGINTERRUPT T)))
    (SETQ \VMEM.FULL.STATE (COND
                              ((ILESSP (fetch (IFPAGE NActivePages) of \InterfacePage)
                                     \LASTVMEMFILEPAGE)          ; Not completely full, allow normal things to happen
                               0)
                              ((.VMEM.CONSISTENTP.)
                               T)
                              (T 'DIRTY])
```

(\SET.LASTVMEMFILEPAGE

```
  [LAMBDA (N)                                                    ; Edited  6-Apr-87 14:09 by bvm:
    ;; Called by disk routines when they discover how long the physical vmem is.  Currently only used by Dove.
    (COND
       ((IGREATERP (fetch (IFPAGE NActivePages) of \InterfacePage)
                (IDIFFERENCE (SETQ \LASTVMEMFILEPAGE N)
                          \GUARDVMEMFULL))                       ; Vmem getting full!
        (\SET.VMEM.FULL.STATE))
       (T                                                        ; Vmem ok now (was earlier set to full for safety's sake)
          (SETQ \VMEM.FULL.STATE NIL)))
    N])
```

(\DOVMEMFULLINTERRUPT

```
  [LAMBDA NIL                                                    ; Edited 21-Oct-87 13:54 by bvm:

;;; Called while interruptable when vmem is full or nearly so.  Tries to extend vmem file, or gives error if it can't

    (COND
       (\EXTENDINGVMEMFILE
              ;; Another interrupt happened while we are extending file.  Don't try to do this one twice, but repost the interrupt in the hopes that it
              ;; will happen after vmem extension is finished
              (SETQ \PENDINGINTERRUPT T))
       (T (RESETVARS ((\EXTENDINGVMEMFILE T))
            ;; Used to have code here that tried to extend the vmem file, but even on those that support extension it's flaky, and rarely what you
            ;; want--people allocate the vmem file to the desired size in the first place, don't want it extended further.
                     (PROG ((HELPFLAG 'BREAK!))
                           (replace VMEMFULL of \INTERRUPTSTATE with NIL)
                                                                 ; Very slight chance of losing the break if ^E right here.  Don't
                                                                 ; know how to fix this
                           (CL:CERROR "Resume the interrupted computation" (CONCAT
                                                                    "Your virtual memory backing file is
"
                                                                    (COND
                                                                       ((>= (fetch (IFPAGE NActivePages)
                                                                               of \InterfacePage)
                                                                            \LASTVMEMFILEPAGE)
                                                                        "complete")
                                                                       (T "near"))
                                                                    "ly full.
                                                                    Save your work & reload a.s.a.p."])
```

(\FLUSHVMOK?

```
  [LAMBDA (TYPE NOERROR)                                         ; Edited 10-Feb-2021 21:49 by larry
                                                                 (* bvm%: " 7-Sep-85 10:48")

;;; Called before any attempt to do a \FLUSHVM to make sure it's ok

    T])

)

(RPAQ? \UPDATECHAINFREQ 100)

(RPAQ? \PAGEFAULTCOUNTER 0)

(RPAQ? \DIRTYPAGECOUNTER 0)
```

```
(RPAQ? \DIRTYPAGEHINT 0)

(RPAQ? \LASTACCESSEDVMEMPAGE 0)

(RPAQ? \MAXSHORTSEEK 1000)

(RPAQ? \MINSHORTSEEK 20)

(RPAQ? \MAXCLEANPROBES 20)

(RPAQ? \VMEM.INHIBIT.WRITE )

(RPAQ? \VMEM.PURE.LIMIT )

(RPAQ? \VMEM.FULL.STATE )

(RPAQ? \GUARDVMEMFULL 500)

(RPAQ? VMEM.COMPRESS.FLG )

(RPAQ? \DOFAULTINIT 0)

(RPAQ? \VMEMACCESSFN )

(RPAQ? \SYSTEMCACHEVARS )

(RPAQ? \MAXSWAPBUFFERS 1)

(RPAQ? \EXTENDINGVMEMFILE )

(RPAQ? \MaxScreenPage 0)

(RPAQ? \NEWVMEMPAGEADDED )

(RPAQ? \LASTDIRTYCNT )

(RPAQ? \LASTDIRTYFOUND )

(RPAQ? \LASTDIRTYSCANPTR )

(RPAQ? \DIRTYSEEKMAX 50)
```

;; Errors signaled in the maintenance panel

```
(DEFINEQ

(\MP.ERROR
  [LAMBDA (CODE STRING ARG1 ARG2)                                    (* mpl "20-Jun-85 11:09")
    (COND
       ((OR (EQ \MACHINETYPE \DANDELION)
            (EQ \MACHINETYPE \DAYBREAK))
        ((OPCODES RAID)
         CODE))
       (T (RAID STRING ARG1 ARG2])

)
```

;; Debugging code. Some of this also runs renamed for extra TeleRaid help

```
(DEFINEQ

(\ACTONVMEMFILE
  [LAMBDA (FILEPAGE BUFFER NPAGES WRITEFLAG)                         (* MPL "22-Jun-85 20:18")
    (COND
       ((EQ \MACHINETYPE \DANDELION)
        (\DL.ACTONVMEMFILE FILEPAGE BUFFER NPAGES WRITEFLAG))
       ((EQ \MACHINETYPE \DAYBREAK)
        (\DOVE.ACTONVMEMFILE FILEPAGE BUFFER NPAGES WRITEFLAG))
       (T (\M44ACTONVMEMFILE FILEPAGE BUFFER NPAGES WRITEFLAG])


(\SHOWPAGETABLE
  [LAMBDA (MODE FILE)                                                (* bvm%: "12-Jul-86 16:55")
    (PROG ((*PRINT-BASE* 8)
           (OUTSTREAM (GETSTREAM FILE 'OUTPUT))
           (RPTR \REALPAGETABLE)
           (RP 0)
           FLAGS VP STATE FIRSTONE LASTONE)
          (printout OUTSTREAM "     RP      VP           FilePage  Status" T)
          (until (SELECTQ MODE
                     (CHAIN (EQ (SETQ RP (fetch (RPT NEXTRP) of RPTR))
                                \PAGETABLESTOPFLG))
                     (NIL (add RP 1)
                          (IGEQ RP \RPTSIZE))
                     (\ILLEGAL.ARG MODE))
```

```
            do (SETQ RPTR (fetch RPTRBASE of RP))
               (SETQ VP (fetch (RPT VP) of RPTR))
               (COND
                  ((AND (NULL MODE)
                        (EQ VP STATE))
                   (SETQ LASTONE RP))
                  (T (COND
                        (LASTONE (printout OUTSTREAM "ditto thru " LASTONE T)
                               (SETQ LASTONE NIL)))
                     (SETQ FIRSTONE RP)
                     (SETQ STATE VP)
                     (printout OUTSTREAM .I7.8 (RPFROMRPT RP))
                     [COND
                        ((fetch (RPT EMPTY) of RPTR)
                         (PRIN1 " Empty" OUTSTREAM))
                        ((NOT (fetch (RPT OCCUPIED) of RPTR))
                         (PRIN1 " Unavailable" OUTSTREAM))
                        (T (printout OUTSTREAM .I8.8 VP %,)
                           (\PRINTVP VP OUTSTREAM)
                           (printout OUTSTREAM 28 .I6.8 (fetch (RPT FILEPAGE) of RPTR)
                                  %,,)
                           (COND
                              ((fetch (RPT LOCKED) of RPTR)
                               (COND
                                  ((NOT (\LOCKEDPAGEP VP))     ; not permanently locked
                                   (PRIN1 "Temp" OUTSTREAM)))
                               (PRIN1 "Locked " OUTSTREAM)))
                           (UNLESSRDSYS (PROGN (COND
                                                  ((fetch (VMEMFLAGS REFERENCED) of (SETQ FLAGS (\READFLAGS
                                                                                                  VP)))
                                                   (PRIN1 "Ref " OUTSTREAM)))
                                               (COND
                                                  ((fetch (VMEMFLAGS DIRTY) of FLAGS)
                                                   (PRIN1 "Dirty" OUTSTREAM]
                     (TERPRI OUTSTREAM])
```

(CHECKPAGEMAP

```
[LAMBDA NIL                                                    (* bvm%: "12-Jul-86 16:56")
  (LET ((*PRINT-BASE* 8)
        (NUMOCCUPIED 0)
        (NUMLOCKED 0)
        (CHAINOCCUPIED 0)
        (CHAINLOCKED 0)
        RPTR FPBASE FP VP RP)
       (CHECKFPTOVP)
       [for RPTINDEX from 1 to (SUB1 \RPTSIZE) when (fetch (RPT OCCUPIED) of (SETQ RPTR (fetch RPTRBASE
                                                                                          of RPTINDEX)))
          do (add NUMOCCUPIED 1)
             (SETQ VP (fetch (RPT VP) of RPTR))
             (SETQ FP (fetch (RPT FILEPAGE) of RPTR))
             (COND
                ((CHECKFPTOVP1 FP VP RPTINDEX))
                ([NEQ VP (fetch FPVIRTUALPAGE of (SETQ FPBASE (\ADDBASE \FPTOVP FP]
                 (printout T "RPT for RP " (RPFROMRPT RPTINDEX)
                        " says VP ")
                 (\PRINTVP VP T)
                 (printout T " lives in FP " FP "; but FP Map says that FP contains ")
                 (\PRINTVP (fetch FPVIRTUALPAGE of FPBASE)
                        T)
                 (printout T T))
                ((\LOCKEDPAGEP VP)
                 (add NUMLOCKED 1)
                 (COND
                    ((NOT (fetch (RPT LOCKED) of RPTR))
                     (printout T "VP " VP ", living in RP " (RPFROMRPT RPTINDEX)
                            " should be locked but isn't." T))
                    ((IGREATERP FP (DLRPFROMFP (fetch (IFPAGE LastLockedFilePage) of \InterfacePage)))
                     (printout T "VP " VP " is locked, but living in FP " FP ", which is not in the locked
                            page area" T]
       (PROGN (SETQ RPTR \REALPAGETABLE)                       ; Check pagetable chain
              [while (NEQ (SETQ RP (fetch (RPT NEXTRP) of RPTR))
                          \PAGETABLESTOPFLG)
                 when (fetch (RPT OCCUPIED) of (SETQ RPTR (fetch RPTRBASE of RP)))
                 do (add CHAINOCCUPIED 1)
                    (COND
                       ((fetch (RPT LOCKED) of RPTR)
                        (add CHAINLOCKED 1]
              (COND
                 ((ILESSP CHAINOCCUPIED NUMOCCUPIED)
                  (printout T NUMOCCUPIED " occupied pages, but only " CHAINOCCUPIED " are on page chain.  "
                         NUMLOCKED " pages are permanently locked; " CHAINLOCKED " pages on chain are locked
                         somehow." T])
```

(CHECKFPTOVP

```
[LAMBDA NIL                                                    (* bvm%: "10-Dec-84 12:39")
```

```
(for FP from 1 to (fetch NActivePages of \InterfacePage) as (FPBASE _ (\ADDBASE \FPTOVP 1))
   by (\ADDBASE FPBASE 1) when (fetch FPOCCUPIED of FPBASE) do (CHECKFPTOVP1 FP (fetch FPVIRTUALPAGE
                                                                                  of FPBASE])
```

(CHECKFPTOVP1

```
[LAMBDA (FP VP RPTINDEX)                                    (* bvm%: "10-Dec-84 12:36")
  (PROG ((FP2 (\LOOKUPPAGEMAP VP)))
        (RETURN (COND
                   ((NEQ FP2 FP)
                    (COND
                       ((UNLESSRDSYS RPTINDEX)
                        (printout T "RPT for RP " (RPFROMRPT RPTINDEX)))
                       (T (printout T "FP map")))
                    (printout T " says FP " FP " contains VP ")
                    (\PRINTVP VP T)
                    (printout T "; but PageMap says that page is in FP " FP2 T)
                    T])
```

(\PRINTFPTOVP

```
[LAMBDA (FIRSTPAGE NWORDS TYPEFLG STREAM VPRAWFLG)          (* bvm%: "24-Sep-86 11:44")
  (SETQ STREAM (GETSTREAM STREAM 'OUTPUT))
  (OR FIRSTPAGE (SETQ FIRSTPAGE 1))
  (OR NWORDS (SETQ NWORDS (fetch (IFPAGE NActivePages) of \InterfacePage)))
  (LET ((BASE (\ADDBASE \FPTOVP (SUB1 FIRSTPAGE)))
        (*PRINT-BASE* 8)
        (LASTVP -2)
        (NEXTFP (SUB1 FIRSTPAGE))
       FIRSTFP FIRSTVP NEXTVP LOCKEDP TYPE NEXTLOCKED NEXTTYPE)
      (while (IGEQ NWORDS 0) do (add NEXTFP 1)
                                [COND
                                   ((EQ NWORDS 0)
                                    (SETQ NEXTVP -1))
                                   ((NEQ (SETQ NEXTVP (\GETBASE (SETQ BASE (\ADDBASE BASE 1))
                                                                0))
                                         \NO.VMEM.PAGE)
                                    (SETQ NEXTLOCKED (\LOCKEDPAGEP NEXTVP))
                                    (if TYPEFLG
                                        then (SETQ NEXTTYPE (TYPENAME (create POINTER
                                                                              PAGE# _ NEXTVP)))
                                             (if (NULL NEXTTYPE)
                                                 then (SETQ NEXTTYPE (SELECTC (LRSH NEXTVP 8)
                                                                              ((LIST \PNAME.HI (CL:1+ \PNAME.HI))
                                                                                   "Pnames")
                                                                              ((LIST \DEF.HI (CL:1+ \DEF.HI))
                                                                                   "Definitions")
                                                                              ((LIST \VAL.HI (CL:1+ \VAL.HI))
                                                                                   "Value cells")
                                                                              ((LIST \PLIST.HI (CL:1+ \PLIST.HI))
                                                                                   "Property lists")
                                                                              ((\HILOC \FPTOVP)
                                                                                   "\FPTOVP")
                                                                              (\STACKHI "Stack")
                                                                              ((\HILOC \HTMAIN)
                                                                                   "GC Main table")
                                                                              ((\HILOC \HTOVERFLOW)
                                                                                   "GC Overflow table")
                                                                             NIL]
                                [COND
                                   ([COND
                                       ((EQ NEXTVP \NO.VMEM.PAGE)
                                        (NEQ LASTVP \NO.VMEM.PAGE))
                                       (T (OR (NEQ NEXTVP (ADD1 LASTVP))
                                              (NEQ NEXTLOCKED LOCKEDP)
                                              (NEQ TYPE NEXTTYPE]
                                    [COND
                                       ((IGEQ LASTVP 0)
                                        (COND
                                           (FIRSTFP (printout STREAM FIRSTFP "-")))
                                        (printout STREAM (SUB1 NEXTFP)
                                                12)
                                        (COND
                                           ((EQ LASTVP \NO.VMEM.PAGE)
                                            (printout STREAM "empty"))
                                           (T (COND
                                                 (FIRSTFP (if VPRAWFLG
                                                              then (PRIN1 FIRSTVP STREAM)
                                                            else (\PRINTVP FIRSTVP STREAM))
                                                          (PRIN1 "-" STREAM)))
                                              (if VPRAWFLG
                                                  then (PRIN1 LASTVP STREAM)
                                                else (\PRINTVP LASTVP STREAM))
                                              (COND
                                                 (LOCKEDP (PRIN1 '* STREAM)))
                                              (if TYPE
                                                  then (printout STREAM 32 TYPE]
```

```
                                    (SETQ FIRSTFP)
                                    (TERPRI STREAM)
                                    (SETQ FIRSTVP NEXTVP))
                                   (T                                ; in a run
                                      (OR FIRSTFP (SETQ FIRSTFP (SUB1 NEXTFP]
                                 (SETQ LASTVP NEXTVP)
                                 (SETQ LOCKEDP NEXTLOCKED)
                                 (SETQ TYPE NEXTTYPE)
                                 (add NWORDS -1])
```

(**PRINTVP**

```
  [LAMBDA (VP STREAM)                                                    (* bvm%: "28-MAR-83 12:40")
    (printout STREAM "{" (LRSH VP 8)
            ","
            (LOGAND VP 255)
            "}"])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \ACTONVMEMFILE MACRO ((X . Y)
                                (SPREADAPPLY* \VMEMACCESSFN X . Y)))

(PUTPROPS .VMEM.CONSISTENTP. MACRO (NIL (EQ (fetch (IFPAGE Key) of \InterfacePage)
                                            \IFPValidKey)))

(PUTPROPS .LOCKABLERP. MACRO [(RP)
                              (OR (NEQ (FOLDLO RP PAGESPERSEGMENT)
                                       (FOLDLO \RP.STACK PAGESPERSEGMENT))
                                  (NOT (OR (EQ \MACHINETYPE \DANDELION)
                                           (EQ \MACHINETYPE \DAYBREAK])
)
```

;; Virtual page flags

```
(DECLARE%: EVAL@COMPILE

(RPAQQ \VMAP.DIRTY 4096)

(RPAQQ \VMAP.CLEAN 0)

(RPAQQ \VMAP.REF 32768)

(RPAQQ \VMAP.VACANT 12288)

(RPAQQ \VMAP.FLAGS 61440)

(RPAQQ \VMAP.NOTFLAGS 4095)

(CONSTANTS \VMAP.DIRTY \VMAP.CLEAN \VMAP.REF \VMAP.VACANT \VMAP.FLAGS \VMAP.NOTFLAGS)
)

(DECLARE%: EVAL@COMPILE

[ACCESSFNS VMEMFLAGS ((VACANT (EQ (LOGAND DATUM \VMAP.VACANT)
                                  \VMAP.VACANT))
                      (DIRTY (NEQ (LOGAND DATUM \VMAP.DIRTY)
                                  0))
                      (REFERENCED (NEQ (LOGAND DATUM \VMAP.REF)
                                       0]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS LOGNOT16 MACRO ((X)
                          (LOGXOR X 65535)))
)
```

;; RPT constants

```
(DECLARE%: EVAL@COMPILE

(RPAQQ \RPT.EMPTY 65534)

(RPAQQ \RPT.UNAVAILABLE 65535)

(RPAQQ \PAGETABLESTOPFLG 0)

(RPAQQ \RPTENTRYLENGTH 3)

(CONSTANTS \RPT.EMPTY \RPT.UNAVAILABLE \PAGETABLESTOPFLG \RPTENTRYLENGTH)
)
```

```
(DECLARE%: EVAL@COMPILE

[BLOCKRECORD RPT ((LOCKED FLAG)
                  (NEXTRP BITS 15)
                  (VP WORD)
                  (FILEPAGE WORD))
       (BLOCKRECORD RPT ((NIL BITS 16)
                         (VPSEG BYTE)
                         (VPPAGEINSEG BYTE)))
       (ACCESSFNS RPT ([EMPTY (EQ (fetch (RPT VP) of DATUM)
                                  \RPT.EMPTY)
                              (COND
                                 (NEWVALUE (replace (RPT VP) of DATUM with \RPT.EMPTY))
                                 (T (ERROR "Invalid replace of RPT.EMPTY" DATUM]
                       [UNAVAILABLE (EQ (fetch (RPT VP) of DATUM)
                                        \RPT.UNAVAILABLE)
                              (COND
                                 (NEWVALUE (replace (RPT VP) of DATUM with \RPT.UNAVAILABLE))
                                 (T (ERROR "Invalid replace of RPT.UNAVAILABLE" DATUM]
                       (OCCUPIED (ILESSP (fetch (RPT VP) of DATUM)
                                         \RPT.EMPTY]

(ACCESSFNS RPT1 (RPTRBASE (\ADDBASE (\ADDBASE \REALPAGETABLE (LLSH DATUM 1))
                                    DATUM)))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS RPFROMRPT MACRO ((RPTINDEX)
                           (IPLUS RPTINDEX \RPOFFSET)))

(PUTPROPS RPTFROMRP MACRO ((RP)
                           (IDIFFERENCE RP \RPOFFSET)))

(PUTPROPS NPAGESMACRO MACRO ((FORM)
                             (PROG ((RESULT 0)
                                    (CNTR \RPTSIZE)
                                    (RPTR \REALPAGETABLE)
                                    VP)
                               LP   (COND
                                       ((NEQ (SETQ CNTR (SUB1 CNTR))
                                             0)
                                        (SETQ RPTR (\ADDBASE RPTR \RPTENTRYLENGTH))
                                        (COND
                                           ((AND (fetch (RPT OCCUPIED) of RPTR)
                                                 (PROGN (SETQ VP (fetch (RPT VP) of RPTR))
                                                        FORM))
                                            (add RESULT 1)))
                                        (GO LP)))
                                    (RETURN RESULT))))
)

;; Virtual to file pagemap
;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(RPAQQ \MAXFILEPAGE 65534)

(CONSTANTS \MAXFILEPAGE)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

(RPAQQ \EMPTYPMTENTRY 65535)

(CONSTANTS \EMPTYPMTENTRY)
)

(DECLARE%: EVAL@COMPILE

[ACCESSFNS VP ((PRIMARYKEY (LRSH DATUM 5))
               (SECONDARYKEY (LOGAND DATUM 31))
               (INVALID (PROGN NIL]
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS .PAGEMAPBASE. MACRO [OPENLAMBDA (VPAGE)
                                 (\ADDBASE \PAGEMAP (IPLUS (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY)
                                                                                    of VPAGE))
                                                           (fetch (VP SECONDARYKEY) of VPAGE])
)
```

;; FP to VP stuff

```
(DECLARE%: EVAL@COMPILE

[BLOCKRECORD FPTOVP ((FPVIRTUALPAGE FIXP))
       (ACCESSFNS FPTOVP ((FPOCCUPIED (NEQ (\GETBASE DATUM 0)
                                           \NO.VMEM.PAGE]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \NO.VMEM.PAGE 65535)

(CONSTANTS \NO.VMEM.PAGE)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS DLRPFROMFP MACRO ((FP)
                            (ADD1 FP)))

(PUTPROPS DLFPFROMRP MACRO ((RP)
                            (SUB1 RP)))
)

(PUTPROPS \TOUCHPAGE DOPVAL (1 GETBASE.N 0))

(PUTPROPS TIMES3 DOPVAL (1 COPY LLSH1 IPLUS2))
```

;; Locked page table

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS .LOCKEDVPBASE. MACRO ((VP)
                                (\ADDBASE \LOCKEDPAGETABLE (FOLDLO VP BITSPERWORD))))

(PUTPROPS .LOCKEDVPMASK. MACRO ((VP)
                                (LLSH 1 (IMOD VP BITSPERWORD))))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \MAXDIRTYSCANCOUNT 100)

(RPAQQ \MINVMEMSPAREPAGES 100)

(RPAQQ \DLBUFFERPAGES 16)

(CONSTANTS \MAXDIRTYSCANCOUNT \MINVMEMSPAREPAGES \DLBUFFERPAGES)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ 2MBPAGES 4096)

(CONSTANTS 2MBPAGES)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \UPDATECHAINFREQ \REALPAGETABLE \RPTLAST \RPOFFSET \RPTSIZE \LOCKEDPAGETABLE \EMBUFBASE \EMBUFVP
       \EMBUFRP \PAGEFAULTCOUNTER \LASTDIRTYCNT \LASTDIRTYFOUND \LASTDIRTYSCANPTR \MACHINETYPE
       \LASTACCESSEDVMEMPAGE \MAXSHORTSEEK \MAXCLEANPROBES \MINSHORTSEEK \DIRTYSEEKMAX \DIRTYPAGECOUNTER
       \DIRTYPAGEHINT \VMEM.INHIBIT.WRITE \VMEM.PURE.LIMIT \VMEM.FULL.STATE \GUARDVMEMFULL VMEM.COMPRESS.FLG
       \KBDSTACKBASE \MISCSTACKBASE \DOFAULTINIT \FPTOVP \VMEMACCESSFN \SYSTEMCACHEVARS \LASTVMEMFILEPAGE
       \EXTENDINGVMEMFILE \MaxScreenPage \NEWVMEMPAGEADDED)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \#SWAPBUFFERS \#EMUBUFFERS \#DISKBUFFERS \MAXSWAPBUFFERS \EMUSWAPBUFFERS \EMUBUFFERS \TELERAIDBUFFER
       \EMUDISKBUFFERS \EMUDISKBUFEND)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS RWMufMan DMACRO ((X)
                           ((OPCODES 120 9)
                            X)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ DOLOCKCHECKS NIL)
```

```
(CONSTANTS (DOLOCKCHECKS NIL))
)
)
```

;;; MAKEINIT stuff

```
(DEFINEQ
```

(ADDPME

```
  [LAMBDA (VP NEWPAGEOK)                                    (* bvm%: " 6-Dec-84 14:07")
```

;; add an entry for VP to the PAGEMAP. Called only under MAKEINIT

```
    (PROG (PX PMP LOCKBASE)
          [COND
             ((IEQ (SETQ PMP (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)))
                   \EmptyPMTEntry)                          ; empty entries in the PageMapTBL have 177777q as their value
              (COND
                 ((EVENP NEXTPM WORDSPERPAGE)               ; must add a new page map page
                  (SETQ PX (\ADDBASE \PAGEMAP NEXTPM))
                  (OR NEWPAGEOK (IGREATERP (PAGELOC PX)
                                           VP)
                      (HELP "page map needs new page after page map written out"))
                  (\NEWPAGE PX NIL T)))
              (\PUTBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)
                     (SETQ PMP NEXTPM))
              (SETQ NEXTPM (IPLUS NEXTPM \PMblockSize]
          (SETQ PX (IPLUS PMP (fetch (VP SECONDARYKEY) of VP)))
          (COND
             ((NEQ (\GETBASE \PAGEMAP PX)
                   0)
              (HELP "page already in pagemap" VP))
             (T (\PUTBASE \PAGEMAP PX NEXTVMEM)
                [COND
                   ((LOCKEDPAGEP VP)                        ; Set lock bit in locked page table
                    (\PUTBASE (SETQ LOCKBASE (.LOCKEDVPBASE. VP))
                           0
                           (LOGOR (.LOCKEDVPMASK. VP)
                                  (\GETBASE LOCKBASE 0]
                (SETQ NEXTVMEM (ADD1 NEXTVMEM]))
```

(CHECKIFPAGE

```
  [LAMBDA NIL                                               (* mjs "19-Jul-84 13:24")
    (CHECKIF Key EQUAL \IFPValidKey "Interface page key"])
```

(DUMPINITPAGES

```
  [LAMBDA (CODEFIRSTPAGE CODENEXTPAGE VERSIONS)             (* bvm%: "14-Jan-85 12:51")
                                                            ; called only under MAKEINIT
    (ADDPME (PAGELOC \InterfacePage)
            T)                                              ; THE INTERFACE PAGE MUST BE THE FIRST PAGE
    (for I from CODEFIRSTPAGE to (SUB1 CODENEXTPAGE) do     ; add the pagemap entries for the pages which were written
                                                            ; directly to the file
                                                       (ADDPME I T))
    (MAPPAGES 0 (ADD1 \MAXVMPAGE)
          (FUNCTION MAKEROOMFORPME))
    (MAPPAGES 0 (ADD1 \MAXVMPAGE)
          (FUNCTION ADDPME))
    (PROGN                                                  ; set interface page locations --- stack pointers already set up IN
                                                            ; SETUPSTACK
          (replace (IFPAGE NxtPMAddr) of \InterfacePage with NEXTPM)
          (replace (IFPAGE NActivePages) of \InterfacePage with (SUB1 NEXTVMEM))
          (replace (IFPAGE NDirtyPages) of \InterfacePage with (SUB1 NEXTVMEM))
          (replace (IFPAGE filePnPMP0) of \InterfacePage with (\GETBASE \PAGEMAP 0))
          (replace (IFPAGE filePnPMT0) of \InterfacePage with (\GETBASE (.PAGEMAPBASE. (PAGELOC \PageMapTBL))
                                                                      0))
          [COND
             (VERSIONS (replace (IFPAGE LVersion) of \InterfacePage with (CAR VERSIONS))
                    (replace (IFPAGE MinBVersion) of \InterfacePage with (CADDR VERSIONS))
                    (replace (IFPAGE MinRVersion) of \InterfacePage with (CADR VERSIONS]
          (replace (IFPAGE Key) of \InterfacePage with \IFPValidKey))
    (MAPPAGES 0 (ADD1 \MAXVMPAGE)
          (FUNCTION DUMPVP))
    (ALLOCAL (PROG ((FILE (OUTPUT)))
                   [COND
                      ((NOT (RANDACCESSP FILE))             ; SYSOUT file is sequential;  have to get it random access for
                                                            ; this
                       (OUTPUT (SETQ FILE (OPENFILE (CLOSEF FILE)
                                                    'BOTH]
                   (SETFILEPTR FILE MKI.Page0Byte)))
    (DUMPVP (PAGELOC \InterfacePage])
```

(MAKEROOMFORPME

```
  [LAMBDA (VP)                                              (* bvm%: "29-MAR-83 17:11")
```

;; make sure that the pagemap-page for page VP exists; we later will want to add it to the pagemap

```
    (COND
       ((IEQ (\GETBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP))
             \EmptyPMTEntry)                                    ; empty entries in the PageMapTBL have 177777q as their value
        (COND
           ((EVENP NEXTPM WORDSPERPAGE)                         ; must add a new page map page
            (\NEWPAGE (\ADDBASE \PAGEMAP NEXTPM)
                   NIL T)))
        (\PUTBASE \PageMapTBL (fetch (VP PRIMARYKEY) of VP)
              NEXTPM)
        (SETQ NEXTPM (IPLUS NEXTPM \PMblockSize])
```

(MAPPAGES

```
  [LAMBDA (BOT TOP FN)                                          ; Edited  5-Nov-92 15:41 by sybalsky:mv:envos
```

;; Map thru all pages from BOT to TOP that exist, skipping the interface page, if it falls into that range. Call FN on the page number.

```
    (PROG ((VP BOT)
           (IVP (PAGELOC \InterfacePage)))
      LP   (COND
              ((AND (SETQ VP (MKI.NEXTPAGE VP))
                    (IGREATERP TOP VP))
               (COND
                  ((NOT (IEQ VP IVP))
                   (APPLY* FN VP)))
               (SETQ VP (ADD1 VP))
               (GO LP])
```

(READPAGEMAP

```
  [LAMBDA NIL                                                   (* bvm%: "10-Dec-84 21:54")
                                                                ; called only under READSYS -- reads in pagemap so that
                                                                ; SETVMPTR can work
    (PROG (D)
          (LOCAL (MAPVMPAGE (fetch (POINTER PAGE#) of \InterfacePage)
                            1))                                 ; Install interface page by magic
                                                                (* PROGN (SETQ FPSTART (fetch
                                                                (IFPAGE LastDominoFilePage) of \InterfacePage))
                                                                (SETQ NPAGES (fetch (IFPAGE NActivePages) of
                                                                \InterfacePage)) (* ; "Note: have to do these fetches before the
                                                                SETFILEPTR since they indirectly do SETFILEPTR
                                                                themselves") (SETFILEPTR VMEMFILE
                                                                (IPLUS (UNFOLD (SUB1 (fetch (IFPAGE FPTOVPStart) of
                                                                \InterfacePage)) BYTESPERPAGE)
                                                                (UNFOLD FPSTART BYTESPERWORD)))
                                                                (for I from FPSTART to NPAGES bind VP when
                                                                (NEQ (SETQ VP (VBIN2)) \NO.VMEM.PAGE) do
                                                                (* ; "Read in all of FPTOVP") (MAPVMPAGE VP
                                                                (SUB1 I))))
          [LOCAL (MAPVMPAGE (PAGELOC \PAGEMAP)
                            (SUB1 (fetch (IFPAGE filePnPMP0) of \InterfacePage]
                                                                ; map in first page of secondary page map, which is where all the
                                                                ; secondary map pages themselves live
          (LOCAL (SETVMPTR \PAGEMAP))
          (for I from 0 to (SUB1 (FOLDHI PAGESPERSEGMENT \PMblockSize)) as VP from (PAGELOC \PAGEMAP) by
                                                                                                \PMblockSize
             do                                                 ; Have to read all the addresses of secondary map pages
                                                                ; themselves before we can read their contents
                (READPAGEMAPBLOCK VP))
          (for J from 0 to (SUB1 \NumPMTpages) as FP from (SUB1 (fetch (IFPAGE filePnPMT0) of \InterfacePage))
             do                                                 ; read in all the primary map table pages
                (LOCAL (MAPVMPAGE (IPLUS (PAGELOC \PageMapTBL)
                                         J)
                                  FP)))
          (for I from 0 to (SUB1 (UNFOLD \NumPMTpages WORDSPERPAGE))
             do (COND
                   ((IEQ (SETQ D (GETBASE \PageMapTBL I))
                         \EmptyPMTEntry))
                   (T (LOCAL (SETVMPTR (ADDBASE \PAGEMAP D)))
                      (READPAGEMAPBLOCK (UNFOLD I \PMblockSize])
```

(READPAGEMAPBLOCK

```
  [LAMBDA (VP)                                                  (* lmm " 4-MAY-82 21:12")
    (PROG ((B VP)
           P)
          (FRPTQ \PMblockSize [COND
                                 ((NEQ (SETQ P (VBIN2))
                                       0)
                                  (LOCAL (MAPVMPAGE B (SUB1 P]
                 (SETQ B (ADD1 B])
```

(SETUPPAGEMAP

```
  [LAMBDA NIL                                                   ; Edited  5-Nov-92 16:03 by sybalsky:mv:envos
                                                                ; called only from MAKEINIT to initialize the page map
    (PROG NIL                                                   ; set up page map
```

```
          (\NEWPAGE \PAGEMAP NIL T)                          ; Create 1 page worth of real page table
          (CREATEPAGES \PageMapTBL \NumPMTpages NIL T)       ; And the segment table.
    ;; init PageMapTBL pages to 177777q:
          (for I from 0 to (SUB1 (UNFOLD \NumPMTpages WORDSPERPAGE)) do (\PUTBASE \PageMapTBL I \EmptyPMTEntry))
          (SETQ NEXTPM 0)
          (for I from 0 to (SUB1 (fetch (VP PRIMARYKEY) of \NumPageMapPages))
             bind (PAGEMAPKEY _ (fetch (VP PRIMARYKEY) of (PAGELOC \PAGEMAP)))
             do ;; Assign pagemap pages to cover all pagemap pages, so that \DONEWPAGE can guarantee that when it needs to allocate a new
                ;; pagemap page, that the pagemap page for the new page already exists
                (\PUTBASE \PageMapTBL (IPLUS PAGEMAPKEY I)
                       NEXTPM)
                (SETQ NEXTPM (IPLUS NEXTPM \PMblockSize)))
          (SETQ NEXTVMEM \FirstVmemBlock)                    ; add entry for InterfacePage which must be on FirstVMemBlock
          (CREATEPAGES \LOCKEDPAGETABLE \NumLPTPages NIL T])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS CHECKIF MACRO [(FLD COMPARISON VALUE STR)
                         (COND
                            ((NOT (COMPARISON VALUE (fetch (IFPAGE FLD) of \InterfacePage)))
                             (printout T "Warning: " STR "= " (PROGN VALUE)
                                    ", but \InterfacePage says "
                                    (fetch (IFPAGE FLD) of \InterfacePage)
                                    T])
)

(ADDTOVAR INEWCOMS (FNS DUMPINITPAGES)
                   (VARS INITCONSTANTS)
                   (FNS SETUPPAGEMAP ADDPME MAKEROOMFORPME MAPPAGES))

(ADDTOVAR RDCOMS (FNS READPAGEMAP READPAGEMAPBLOCK CHECKIFPAGE \LOCKEDPAGEP \LOOKUPPAGEMAP CHECKPAGEMAP
                      CHECKFPTOVP CHECKFPTOVP1 \SHOWPAGETABLE \PRINTFPTOVP))

(ADDTOVAR EXPANDMACROFNS CHECKIF .LOCKEDVPBASE. .LOCKEDVPMASK. .PAGEMAPBASE.)

(ADDTOVAR MKI.SUBFNS (\NEWPAGE . MKI.NEWPAGE)
                     (\LOCKPAGES . MKI.LOCKPAGES))

(ADDTOVAR RD.SUBFNS (\NEWPAGE . VNEWPAGE)
                    (\LOCKPAGES . VLOCKPAGES))

(ADDTOVAR RDPTRS (\REALPAGETABLE))

(ADDTOVAR RDVALS (\RPTSIZE))


(ADDTOVAR DONTCOMPILEFNS DUMPINITPAGES SETUPPAGEMAP ADDPME MAKEROOMFORPME MAPPAGES READPAGEMAP
                         READPAGEMAPBLOCK CHECKIFPAGE)
)

(DEFINEQ

(\LOCKFN
  [LAMBDA (FN)                                                  (* bvm%: "22-NOV-82 17:39")
    [\LOCKCELL (SETQ FN (fetch (LITATOM DEFINITIONCELL) of (EVQ FN]
    (COND
       ((fetch (DEFINITIONCELL CCODEP) of FN)
        (\LOCKCODE (fetch (DEFINITIONCELL DEFPOINTER) of FN])


(\LOCKCODE
  [LAMBDA (CODEBLOCK)                                           (* rmk%: "15-Aug-84 13:35")
    (\LOCKWORDS CODEBLOCK (UNFOLD (\#BLOCKDATACELLS CODEBLOCK)
                                  WORDSPERCELL])


(\LOCKVAR
  [LAMBDA (VAR)                                                 (* lmm " 5-APR-82 00:43")
    (\LOCKCELL (fetch (LITATOM VCELL) of (EVQ VAR])


(\LOCKCELL
  [LAMBDA (X NPGS)                                              (* bvm%: "22-NOV-82 17:54")
    (\LOCKPAGES (PAGEBASE X)
           (OR NPGS 1])


(\LOCKWORDS
  [LAMBDA (BASE NWORDS)                                         (* bvm%: "22-NOV-82 17:35")
    (\LOCKPAGES (PAGEBASE BASE)
           (COND
```

```
              (NWORDS (FOLDHI (IPLUS (fetch (POINTER WORDINPAGE) of BASE)
                                     NWORDS)
                              WORDSPERPAGE))
              (T 1])

)

(DECLARE%: DONTCOPY

(ADDTOVAR INEWCOMS
          (FNS \LOCKFN \LOCKVAR \LOCKCELL \LOCKWORDS \LOCKCODE)
          (ALLOCAL (ADDVARS (LOCKEDFNS \FAULTHANDLER \FAULTINIT \DOVE.FAULTINIT \D01.FAULTINIT \DL.FAULTINIT
                                       \CHAIN.UP.RPT \MAKESPACEFORLOCKEDPAGE \PAGEFAULT \WRITEMAP \LOOKUPPAGEMAP
                                       \LOCKEDPAGEP \LOADVMEMPAGE \MOVEREALPAGE \INVALIDADDR \INVALIDVP
                                       \SELECTREALPAGE \TRANSFERPAGE \SPECIALRP \UPDATECHAIN \MARKPAGEVACANT
                                       \FLUSHPAGE \CLEARWORDS \FLUSHVM \DONEWPAGE \ASSURE.FPTOVP.PAGE
                                       \DONEWEPHEMERALPAGE \WRITEDIRTYPAGE1 \COPYSYS0 \COPYSYS0SUBR
                                       \RELEASEWORKINGSET \DOFLUSHVM \DOLOCKPAGES \DOTEMPLOCKPAGES \TEMPUNLOCKPAGES
                                       \MP.ERROR RAID \DL.NEWFAULTINIT \DL.MARK.PAGES.UNAVAILABLE \DL.UNMAPPAGES
                                       \DL.ASSIGNBUFFERS \D01.ASSIGNBUFFERS \DOCOMPRESSVMEM \MOVEVMEMFILEPAGE
                                       \SET.VMEM.FULL.STATE \HINUM \LONUM \ATOMCELL SETTOPVAL)
                                 (LOCKEDVARS \REALPAGETABLE \RPTLAST \PAGEFAULTCOUNTER \UPDATECHAINFREQ \RPOFFSET
                                        \RPTSIZE \LOCKEDPAGETABLE \EMBUFBASE \EMBUFVP \EMBUFRP \LASTACCESSEDVMEMPAGE
                                        \MAXSHORTSEEK \MAXCLEANPROBES \MINSHORTSEEK \DIRTYPAGECOUNTER \DIRTYPAGEHINT
                                        \VMEM.INHIBIT.WRITE \VMEM.PURE.LIMIT \VMEM.FULL.STATE \GUARDVMEMFULL
                                        VMEM.COMPRESS.FLG \KBDSTACKBASE \MISCSTACKBASE \DOFAULTINIT \FPTOVP
                                        \MACHINETYPE \VMEMACCESSFN \TELERAIDBUFFER \EMUDISKBUFFERS \EMUDISKBUFEND
                                        \MAXSWAPBUFFERS \EMUBUFFERS \#EMUBUFFERS \#SWAPBUFFERS \#DISKBUFFERS
                                        \RCLKSECOND \RCLKMILLISECOND \VALSPACE \EMUSWAPBUFFERS \EM.CURSORBITMAP
                                        \PAGEMAP \PageMapTBL \IOCBPAGE \IOPAGE \MISCSTATS \DEFSPACE \InterfacePage
                                        \LASTVMEMFILEPAGE \DoveIORegion \MaxScreenPage \NEWVMEMPAGEADDED))))
)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA CHECKPAGEMAP \SHOWPAGETABLE VMEM.PURE.STATE \COUNTREALPAGES \WRITEDIRTYPAGE \UNLOCKPAGES
                    \TEMPUNLOCKPAGES \DOTEMPLOCKPAGES \DOLOCKPAGES \LOCKPAGES \LOADVMEMPAGE)
)

(PUTPROPS LLFAULT COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992
                                 1993 2021))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

MACRO INDEX

RECORD INDEX

PROPERTY INDEX