File created: 3-Jan-2024 16:10:17 **{MEDLEY}<SOURCES>LLKEY.;4**

edit by: mth

changes to: (FNS \DECODETRANSITION SHIFTDOWNP)
(VARS LLKEYCOMS)
(RECORDS KEYBOARDEVENT)

previous date: 3-Jan-2024 12:32:52 {MEDLEY}<SOURCES>LLKEY.;3

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ LLKEYCOMS
       [(COMS                                                    ; Access to keyboard
              (FNS BKSYSCHARCODE \CLEARSYSBUF \GETKEY \NSYSBUFCHARS \SAVESYSBUF \SYSBUFP \GETSYSBUF \PUTSYSBUF
                   \PEEKSYSBUF)
              (INITVARS (\LONGSYSBUF))
              (INITVARS (\\KEYBOARDWAITBOX.GLOBALRESOURCE))
              (DECLARE%: DONTCOPY (RESOURCES \KEYBOARDWAITBOX))
              (DECLARE%: DONTCOPY (CONSTANTS (\SYSBUFSIZE 200))
                     (MACROS \GETREALSYSBUF)))
        [DECLARE%: DOCOPY DONTEVAL@LOAD (COMS                    ; Here because it must be done in init before PROC loaded
                                          (P (MOVD? 'NILL 'CARET]
        (COMS                                                    ; Key handler
              (FNS \KEYBOARDINIT \KEYBOARDEVENTFN \ALLOCLOCKED \SETIOPOINTERS \KEYBOARDOFF \KEYBOARDON
                   \KEYHANDLER \KEYHANDLER1 \RESETKEYBOARD \DOMOUSECHORDING \DOTRANSITIONS \DECODETRANSITION
                   MOUSECHORDWAIT \TRACKCURSOR)
              (CONSTANTS (\SUN.TYPE3KEYBOARD 0)
                     (\SUN.TYPE4KEYBOARD 1)
                     (\SUN.JLEKEYBOARD 2)
                     (\TOSHIBA.JIS 7))
              (INITVARS (\MOUSECHORDTICKS)
                     (\MOUSECHORDMILLISECONDS 50)
                     (SHIFTXORLOCKFLG NIL))
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\KEYBOARDINIT)))
              [DECLARE%: DONTCOPY (MACROS .NOTELASTUSERACTION)
                     (CONSTANTS ALLUP \CTRLMASK \METABIT)
                     (CONSTANTS * DLMOUSEBITS)
                     (CONSTANTS * DLMOUSESTATES)
                     (CONSTANTS * TRANSITIONFLAGS)
                     (MACROS \TRANSINDEX ARMEDCODE TRANSITIONALTGRCODE TRANSITIONSHIFTCODE TRANSITIONCODE
                            TRANSITIONFLAGS TRANSITIONDEADLIST CHECKFORDEADKEY)
                     (EXPORT (RECORDS KEYACTION)
                            (CONSTANTS \NKEYS))
                     (RECORDS RING)
                     (COMS                                       ; can get rid of shiftstate after clients have been fixed
                           (RECORDS SHIFTSTATE)
                           (GLOBALVARS \SHIFTSTATE \MOUSETIMERTEMP))
                     (CONSTANTS NRINGINDEXWORDS)
                     (CONSTANTS (\SYSBUFFER.FIRST (UNFOLD NRINGINDEXWORDS BYTESPERWORD))
                            (\SYSBUFFER.LAST (IPLUS \SYSBUFFER.FIRST (SUB1 \SYSBUFSIZE]
              (DECLARE%: EVAL@COMPILE (VARS \KEYNAMES))
              ;; \maikokeyactions does not contain keyactions of the form "2,50" because it breaks the loadup process on the sun.
              (VARS \ORIGKEYACTIONS \DLIONKEYACTIONS \DLIONOSDKEYACTIONS \DORADOKEYACTIONS \DOVEKEYACTIONS
                    \DOVEOSDKEYACTIONS \MAIKOKEYACTIONS \MAIKOKEYACTIONST4 \MAIKO-JLE-KEYACTIONS
                    \TOSHIBA-KEYACTIONS)
              (VARS (KEYBOARD.APPLICATION-SPECIFIC-KEYACTIONS NIL))
              (INITVARS (\KEYBOARD.META 256)
                     (\MODIFIED.KEYACTIONS))
              (DECLARE%: EVAL@COMPILE (ADDVARS (GLOBALVARS \RCLKSECOND \LASTUSERACTION \LASTKEYSTATE)))
              (GLOBALVARS \SYSBUFFER \LONGSYSBUF \INTERRUPTSTATE \MODIFIED.KEYACTIONS \MOUSECHORDTICKS
                     \KEYBOARDEVENTQUEUE \KEYBUFFERING \CURRENTKEYACTION \COMMANDKEYACTION \DEFAULTKEYACTION
                     \TIMER.INTERRUPT.PENDING \ORIGKEYACTIONS \KEYBOARD.META \MOUSECHORDMILLISECONDS
                     \DORADOKEYACTIONS \DLIONKEYACTIONS \DLIONOSDKEYACTIONS \DOVEKEYACTIONS \DOVEOSDKEYACTIONS
                     SHIFTXORLOCKFLG))
        (COMS                                                    ; Key interpretation
              (FNS KEYACTION KEYACTIONTABLE KEYBOARDTYPE RESETKEYACTION \KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS
                   \KEYACTION1 KEYDOWNP KEYNUMBERP \KEYNAMETONUMBER \KEYNUMBERTONAME MODIFY.KEYACTIONS METASHIFT
                   SHIFTDOWNP)
                                                                 ; To support office style 1108 & 1186 keyboards
              (FNS SETUP.OFFICE.KEYBOARD)
              (OPTIMIZERS)
              (MACROS \TEMPCOPYTIMER)
                                                                 ; Don't copy this optimizer since it expands out to \getbasebit, but
                                                                 ; do exportit.
              (DECLARE%: DONTCOPY (EXPORT (OPTIMIZERS KEYDOWNP)))
              (EXPORT (MACROS XKEYDOWNP KEYDOWNP1 \NEWKEYDOWNP)))
        (COMS                                                    ; A raw keyboard device/stream
              (FNS \INIT.KEYBOARD.STREAM)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\INIT.KEYBOARD.STREAM)))
              (EXPORT (GLOBALVARS \KEYBOARD.DEVICE \KEYBOARD.STREAM)))
        (COMS                                                    ; Hook for a periodic interrupt
              (FNS \DOBUFFEREDTRANSITIONS \TIMER.INTERRUPTFRAME \PERIODIC.INTERRUPTFRAME)
```

```
            (INITVARS (\KEYBUFFERING)
                  (\PERIODIC.INTERRUPT)
                  (\TIMER.INTERRUPT.PENDING)
                  (\PERIODIC.INTERRUPT.FREQUENCY 77)))
      (LOCALVARS . T)
      [COMS                                                    ; cursor and mouse related functions.
            (FNS \HARDCURSORUP \HARDCURSORPOSITION \HARDCURSORDOWN)
            (FNS CURSOR.INIT \CURSORDESTINATION \SOFTCURSORUP \SOFTCURSORUPCURRENT \SOFTCURSORPOSITION
                 \SOFTCURSORDOWN CURSORPROP GETCURSORPROP PUTCURSORPROP \CURSORBITSPERPIXEL
                 \CURSORIMAGEPROPNAME \CURSORMASKPROPNAME)
            (FNS CURSORCREATE CURSOR \CURSOR-VALID-P \CURSORUP \CURSORPOSITION \CURSORDOWN
                 ADJUSTCURSORPOSITION CURSORPOSITION CURSORSCREEN CURSOREXIT FLIPCURSOR FLIPCURSORBAR
                 LASTMOUSEX LASTMOUSEY CREATEPOSITION POSITIONP CURSORHOTSPOT)
            (PROPS (CURSORPROP ARGNAMES))
            (INITVARS (\CURSORHOTSPOTX 0)
                  (\CURSORHOTSPOTY 0)
                  (\CURRENTCURSOR NIL)
                  (\SOFTCURSORWIDTH NIL)
                  (\SOFTCURSORHEIGHT NIL)
                  (\SOFTCURSORP NIL)
                  (\SOFTCURSORUPP NIL)
                  (\SOFTCURSORUPBM NIL)
                  (\SOFTCURSORDOWNBM NIL)
                  (\SOFTCURSORBBT1 NIL)
                  (\SOFTCURSORBBT2 NIL)
                  (\SOFTCURSORBBT3 NIL)
                  (\SOFTCURSORBBT4 NIL)
                  (\SOFTCURSORBBT5 NIL)
                  (\SOFTCURSORBBT6 NIL)
                  (\CURSORSCREEN NIL)
                  (\CURSORDESTINATION NIL)
                  (\CURSORDESTHEIGHT 808)
                  (\CURSORDESTWIDTH 1024)
                  (\CURSORDESTRASTERWIDTH 64)
                  (\CURSORDESTLINE 0)
                  (\CURSORDESTLINEBASE NIL))
            (GLOBALVARS \CURSORHOTSPOTX \CURSORHOTSPOTY \CURRENTCURSOR \SOFTCURSORWIDTH \SOFTCURSORHEIGHT
                  \SOFTCURSORP \SOFTCURSORUPP \SOFTCURSORUPBM \SOFTCURSORDOWNBM \SOFTCURSORBBT1
                  \SOFTCURSORBBT2 \SOFTCURSORBBT3 \SOFTCURSORBBT4 \SOFTCURSORBBT5 \SOFTCURSORBBT6
                  \CURSORDESTINATION \CURSORDESTHEIGHT \CURSORDESTWIDTH \CURSORDESTRASTERWIDTH
                  \CURSORDESTLINE \CURSORDESTLINEBASE)
            (FNS GETMOUSESTATE \EVENTKEYS)
            [EXPORT (CONSTANTS (HARDCURSORHEIGHT 16)
                               (HARDCURSORWIDTH 16))
                  (DECLARE%: EVAL@COMPILE (ADDVARS (GLOBALVARS LASTMOUSEX LASTMOUSEY LASTSCREEN
                                                               LASTMOUSEBUTTONS LASTMOUSETIME LASTKEYBOARD]
            (DECLARE%: DONTCOPY (EXPORT (MACROS \SETMOUSEXY))
                  (MACROS \XMOUSECOORD \YMOUSECOORD))
            (DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD 'CURSOR 'SETCURSOR)
                                               (MOVD '\CURSORPOSITION '\SETCURSORPOSITION))
                  (VARS (\SFPosition (CREATEPOSITION]
      [COMS (DECLARE%: DONTCOPY (RECORDS KEYBOARDEVENT)
                  (CONSTANTS (\KEYBOARDEVENT.FIRST NRINGINDEXWORDS)
                        \KEYBOARDEVENT.SIZE
                        (\KEYBOARDEVENT.LAST (PLUS \KEYBOARDEVENT.FIRST (TIMES \KEYBOARDEVENT.SIZE 383]
      (COMS (FNS MACHINETYPE SETMAINTPANEL)
                                                               ; DLion beeper
            (FNS BEEPON BEEPOFF))
      (EXPORT (GLOBALVARS \EM.MOUSEX \EM.MOUSEY \EM.CURSORX \EM.CURSORY \EM.UTILIN \EM.REALUTILIN \EM.KBDAD0
                  \EM.KBDAD1 \EM.KBDAD2 \EM.KBDAD3 \EM.KBDAD4 \EM.KBDAD5 \EM.DISPINTERRUPT \EM.DISPLAYHEAD
                  \EM.CURSORBITMAP \MACHINETYPE \DEFAULTKEYACTION \COMMANDKEYACTION \CURRENTKEYACTION
                  \PERIODIC.INTERRUPT \PERIODIC.INTERRUPT.FREQUENCY))
      (FNS WITHOUT-INTERRUPTS)
      (COMS                                                    ; Compile locked fns together for locality
            (BLOCKS (NIL FLIPCURSORBAR \KEYHANDLER \KEYHANDLER1 \TRACKCURSOR \PERIODIC.INTERRUPTFRAME
                  \TIMER.INTERRUPTFRAME \DOBUFFEREDTRANSITIONS \DOTRANSITIONS \DECODETRANSITION
                  \EVENTKEYS \HARDCURSORUP \DOMOUSECHORDING \KEYBOARDOFF \HARDCURSORPOSITION
                  \HARDCURSORDOWN \SOFTCURSORUP \SOFTCURSORUPCURRENT \SOFTCURSORPOSITION
                  \SOFTCURSORDOWN)))
      [DECLARE%: DONTCOPY
             (ADDVARS [INEWCOMS (ALLOCAL (ADDVARS (LOCKEDFNS FLIPCURSORBAR \SETIOPOINTERS \KEYHANDLER
                                                             \KEYHANDLER1 \CONTEXTAPPLY \LOCKPAGES
                                                             \DECODETRANSITION \SMASHLINK \INCUSECOUNT LLSH
                                                             \MAKEFREEBLOCK \DECUSECOUNT \MAKENUMBER \ADDBASE
                                                             \PERIODIC.INTERRUPTFRAME \DOBUFFEREDTRANSITIONS
                                                             \TIMER.INTERRUPTFRAME \CAUSEINTERRUPT
                                                             \DOMOUSECHORDING \KEYBOARDOFF \TRACKCURSOR
                                                             \HARDCURSORUP \HARDCURSORPOSITION \HARDCURSORDOWN
                                                             \SOFTCURSORUP \SOFTCURSORUPCURRENT
                                                             \SOFTCURSORPOSITION \SOFTCURSORDOWN
                                                             \SOFTCURSORPILOTBITBLT)
                                                   (LOCKEDVARS \InterfacePage \CURSORHOTSPOTX \CURSORHOTSPOTY
                                                         \CURRENTCURSOR \SOFTCURSORWIDTH \SOFTCURSORHEIGHT
                                                         \SOFTCURSORP \SOFTCURSORUPP \SOFTCURSORUPBM
                                                         \SOFTCURSORDOWNBM \SOFTCURSORBBT1 \SOFTCURSORBBT2
                                                         \SOFTCURSORBBT3 \SOFTCURSORBBT4 \SOFTCURSORBBT5
                                                         \SOFTCURSORBBT6 \CURSORDESTINATION \CURSORDESTHEIGHT
```

```
                                                \CURSORDESTWIDTH \CURSORDESTRASTERWIDTH \CURSORDESTLINE
                                                \CURSORDESTLINEBASE \PENDINGINTERRUPT
                                                \PERIODIC.INTERRUPT \PERIODIC.INTERRUPT.FREQUENCY
                                                \LASTUSERACTION \MOUSECHORDTICKS \KEYBOARDEVENTQUEUE
                                                \KEYBUFFERING SCREENWIDTH SCREENHEIGHT
                                                \TIMER.INTERRUPT.PENDING \EM.MOUSEX \EM.MOUSEY
                                                \EM.CURSORX \EM.CURSORY \EM.UTILIN \EM.REALUTILIN
                                                \EM.KBDAD0 \EM.KBDAD1 \EM.KBDAD2 \EM.KBDAD3
                                                \EM.DISPINTERRUPT \EM.CURSORBITMAP \EM.KBDAD4
                                                \EM.KBDAD5 \MISCSTATS \RCLKSECOND]
                    (RDCOMS (FNS \SETIOPOINTERS]
        (PROP FILETYPE LLKEY)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML WITHOUT-INTERRUPTS)
                                                                              (LAMA CURSORPROP METASHIFT
                                                                                    MOUSECHORDWAIT])
```

;; Access to keyboard

```
(DEFINEQ
```

(**BKSYSCHARCODE**

```
  [LAMBDA (CHAR)                                                    (* rrb "30-Dec-83 11:56")
    (OR (\PUTSYSBUF CHAR)
        (PROGN (SETQ \LONGSYSBUF (NCONC \LONGSYSBUF (bind C while (SETQ C (\GETREALSYSBUF)) collect C)))
               (\PUTSYSBUF CHAR])
```

(**\CLEARSYSBUF**

```
  [LAMBDA (ALLFLG)                                                  (* mpl "27-Jun-85 20:04")
    (DECLARE (GLOBALVARS \PROCESSES))
    (COND
       ((OR ALLFLG (TTY.PROCESSP))
        (SETQ \LONGSYSBUF)
        (replace (RING READ) of \SYSBUFFER with 0)))
    (COND
       (ALLFLG (for PROC in \PROCESSES do (replace PROCTYPEAHEAD of PROC with NIL)))
       ((THIS.PROCESS)
        (replace PROCTYPEAHEAD of (THIS.PROCESS) with NIL])
```

(**\GETKEY**

```
  [LAMBDA NIL                                                       (* lmm "18-Apr-85 00:07")
    (DECLARE (GLOBALVARS \KEYBOARDWAIT1 \KEYBOARDWAIT2))
    (COND
       [(AND (THIS.PROCESS)
             (fetch PROCTYPEAHEAD of (THIS.PROCESS)))
        (pop (fetch PROCTYPEAHEAD of (THIS.PROCESS]
       (T (WAIT.FOR.TTY)
          (OR (\GETSYSBUF)
              (GLOBALRESOURCE (\KEYBOARDWAITBOX)                    (* Busy-wait loop that gets next character)
                     (\CLOCK0 \KEYBOARDWAITBOX)
                     (bind C do (COND
                                    ((SETQ C (\GETSYSBUF))
                                     (\BOXIPLUS (LOCF (fetch KEYBOARDWAITTIME of \MISCSTATS))
                                           (CLOCKDIFFERENCE \KEYBOARDWAITBOX))
                                     (RETURN C)))
                                (\TTYBACKGROUND)
                                (\WAIT.FOR.TTY])
```

(**\NSYSBUFCHARS**

```
  [LAMBDA NIL                                                       (* JonL " 7-May-84 01:50")
                                                                    (* Tells how many characters can be \GETSYSBUFed.
                                                                    Used by \SAVESYSBUF.)
    (IPLUS (LENGTH \LONGSYSBUF)
           (PROG ((R (fetch (RING READ) of \SYSBUFFER))
                  (W (fetch (RING WRITE) of \SYSBUFFER)))
                 (RETURN (COND
                            ((EQ 0 R)
                             0)
                            ((IGREATERP W R)
                             (IDIFFERENCE W R))
                            (T (IDIFFERENCE W (IDIFFERENCE R \SYSBUFSIZE])
```

(**\SAVESYSBUF**

```
  [LAMBDA NIL                                                       (* JonL " 7-May-84 01:50")
    (DECLARE (GLOBALVARS \SAVEDSYSBUFFER))
    (PROG (TA (BUF \SAVEDSYSBUFFER)
              (NC (\NSYSBUFCHARS))
              (J 0))
          [COND
             ((TTY.PROCESSP)
              [COND
                 ([AND (THIS.PROCESS)
                       (SETQ TA (fetch PROCTYPEAHEAD of (THIS.PROCESS]
```

```
                    (replace PROCTYPEAHEAD of (THIS.PROCESS) with NIL)
                    (add NC (LENGTH TA))
                    [COND
                       ((IGREATERP NC (NCHARS BUF))
                        (SETQ BUF (ALLOCSTRING NC]
                    (for CH in TA do (RPLCHARCODE BUF (add J 1)
                                                      CH)))
                   ((IGREATERP NC (NCHARS BUF))
                    (SETQ BUF (ALLOCSTRING NC]
                (for I from (ADD1 J) to NC do
```

(* Test on J means that we'll ignore extra chars typed since we got the length.
Test on \GETSYSBUF so we don't get screwed if buffer gets cleared while during this loop)

```
                                        (RPLCHARCODE BUF I (OR (\GETSYSBUF)
                                                               (PROGN (SETQ NC (SUB1 I))
                                                                      (RETURN]
          (RETURN (AND (NOT (EQ 0 NC))
                       (SUBSTRING BUF 1 NC])
```

(\SYSBUFP

```
  [LAMBDA NIL                                                    (* JonL " 7-May-84 01:52")
    (OR [AND (TTY.PROCESSP)
             (OR \LONGSYSBUF (NOT (EQ 0 (fetch (RING READ) of \SYSBUFFER]
        (AND (THIS.PROCESS)
             (fetch PROCTYPEAHEAD of (THIS.PROCESS])
```

(\GETSYSBUF

```
  [LAMBDA NIL                                                    (* lmm " 9-JUL-83 00:56")
    (OR (AND \LONGSYSBUF (pop \LONGSYSBUF))
        (\GETREALSYSBUF])
```

(\PUTSYSBUF

```
  [LAMBDA (CHAR)                                                 (* rmk%: "27-Nov-84 17:51")
    (PROG ((R (fetch (RING READ) of \SYSBUFFER))
           (W (fetch (RING WRITE) of \SYSBUFFER)))
          (RETURN (COND
                     ((EQ R W)                                   (* Full)
                      NIL)
                     (T (\PUTBASEFAT \SYSBUFFER W CHAR)
                        (AND (EQ 0 R)
                             (replace (RING READ) of \SYSBUFFER with W))
                                                                 (* Return random non-NIL value to indicate success for
                                                                 BKSYSBUF)
                        [replace (RING WRITE) of \SYSBUFFER with (COND
                                                                   ((EQ \SYSBUFFER.LAST W)
                                                                    \SYSBUFFER.FIRST)
                                                                   (T (ADD1 W]
                        T])
```

(\PEEKSYSBUF

```
  [LAMBDA (STREAM)                                               (* bvm%: " 8-Feb-85 17:50")
    (PROG (R)
      WAIT
          (until (\SYSBUFP) do (BLOCK))
          (RETURN (if (TTY.PROCESSP)
                      then (if \LONGSYSBUF
                               then (CAR \LONGSYSBUF)
                             elseif (NEQ (SETQ R (fetch (RING READ) of \SYSBUFFER))
                                         0)
                               then                              (* Here's the vanilla case)
                                    (\GETBASEFAT \SYSBUFFER R)
                               else                              (* Foo an interrupt could have sneaked in here and gobbled
                                                                 down the remaining characters)
                                    (GO WAIT))
                    elseif (THIS.PROCESS)
                      then (CAR (fetch PROCTYPEAHEAD of (THIS.PROCESS)))
                    else (SHOULDNT])

)

(RPAQ? \LONGSYSBUF )

(RPAQ? \\KEYBOARDWAITBOX.GLOBALRESOURCE )

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[PUTDEF '\KEYBOARDWAITBOX 'RESOURCES '(NEW (CREATECELL \FIXP]
)
)
```

```
(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \SYSBUFSIZE 200)

(CONSTANTS (\SYSBUFSIZE 200))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \GETREALSYSBUF MACRO [NIL (PROG ((R (fetch (RING READ) of \SYSBUFFER)))
                                          (RETURN (AND (NOT (EQ 0 R))
                                                       (PROG1 (\GETBASEFAT \SYSBUFFER R)
                                                              (AND [EQ (fetch (RING WRITE) of \SYSBUFFER)
                                                                       (replace (RING READ) of \SYSBUFFER
                                                                          with (COND
                                                                                  ((EQ \SYSBUFFER.LAST R)
                                                                                   \SYSBUFFER.FIRST)
                                                                                  (T (ADD1 R]
                                                                    (replace (RING READ) of \SYSBUFFER with 0))))]))
)
)

(DECLARE%: DOCOPY DONTEVAL@LOAD
```

;; Here because it must be done in init before PROC loaded

```
(MOVD? 'NILL 'CARET)
)
```

;; Key handler

```
(DEFINEQ
```

(\KEYBOARDINIT

```
  [LAMBDA NIL                                                          ; Edited 19-Nov-87 16:46 by Snow
    (DECLARE (GLOBALVARS \SAVEDSYSBUFFER))                             ; Sets up keyboard decoding tables.
    (SETQ \CURRENTKEYACTION (SETQ \DEFAULTKEYACTION (KEYACTIONTABLE)))
                                                                       ; added \commandkeyaction 11-19-87 WAS
    (SETQ \COMMANDKEYACTION (KEYACTIONTABLE))
    (SETQ \INTERRUPTSTATE (\ALLOCLOCKED 2))
    (PROGN (SETQ \SYSBUFFER (\ALLOCBLOCK (FOLDHI (ADD1 \SYSBUFFER.LAST)
                                                 WORDSPERCELL)))
           (replace (RING READ) of \SYSBUFFER with 0)
           (replace (RING WRITE) of \SYSBUFFER with \SYSBUFFER.FIRST))
    (SETQ \SAVEDSYSBUFFER (ALLOCSTRING \SYSBUFSIZE NIL NIL T))
    (SETQ \LASTUSERACTION (LOCF (fetch LASTUSERACTION of \MISCSTATS)))
    (PROGN (SETQ \KEYBOARDEVENTQUEUE (\ALLOCLOCKED (FOLDHI (PLUS \KEYBOARDEVENT.LAST \KEYBOARDEVENT.SIZE)
                                                           WORDSPERCELL)))
           (replace (RING READ) of \KEYBOARDEVENTQUEUE with 0)
           (replace (RING WRITE) of \KEYBOARDEVENTQUEUE with \KEYBOARDEVENT.FIRST))
    (SETQ \LASTKEYSTATE (create KEYBOARDEVENT))
    (SETQ \SHIFTSTATE (create SHIFTSTATE))
    (SETQ \MOUSETIMERTEMP (SETUPTIMER 0 NIL 'TICKS))
    (MOUSECHORDWAIT \MOUSECHORDMILLISECONDS)
    (\KEYBOARDON])
```

(\KEYBOARDEVENTFN

```
  [LAMBDA (FDEV EVENT EXTRA)                                           ; Edited 11-Oct-90 09:49 by jds
    (DECLARE (GLOBALVARS \KEYBOARD.BEFORETYPE \DORADOKEYACTIONS \DLIONKEYACTIONS \MAIKO.BEFOREKEYTYPE))
    (SELECTQ EVENT
        ((BEFORELOGOUT BEFOREMAKESYS BEFORESYSOUT BEFORESAVEVM)
            (SETQ \KEYBOARD.BEFORETYPE \MACHINETYPE)
            (SETQ \MAIKO.BEFOREKEYTYPE (LOGAND 7 (FETCH (IFPAGE DEVCONFIG) OF \InterfacePage)))
            (SETQ \MAIKO.XBEFORE? (SELECTQ (MACHINETYPE)
                                      (MAIKO (EQUAL "X" (UNIX-GETPARM "DISPLAY")))
                                      NIL)))
        ((AFTERLOGOUT AFTERMAKESYS AFTERSYSOUT AFTERSAVEVM)            ; Restarting a world.  If we changed machines, fix up the key
                                                                       ; actions to match the new machine.
                                                                       ; (COND ((NEQ \\MACHINETYPE \\KEYBOARD.BEFORETYPE)
                                                                       ; ; Changed machines.  Change Keyactions. (|for| X |in|
                                                                       ; (\\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS) |do|
                                                                       ; (KEYACTION (CAR X) (CDR X) \\COMMANDKEYACTION)
                                                                       ; (KEYACTION (CAR X) (CDR X) \\DEFAULTKEYACTION))
                                                                       ; (MOUSECHORDWAIT (MOUSECHORDWAIT))))
            [COND
               ((OR (NEQ \MACHINETYPE \KEYBOARD.BEFORETYPE)
                    (NEQ \MAIKO.XBEFORE? (SELECTQ (MACHINETYPE)
                                             (MAIKO (EQUAL "X" (UNIX-GETPARM "DISPLAY")))
                                             NIL)))                    ; Changed machines.  Change Keyactions.
                [COND
                   ((NEQ (MACHINETYPE)
                         'MAIKO)
                                                                       ;; Non-SUN, so just change machine-specific key actions:
```

```
                    (for X in (\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS) do (KEYACTION (CAR X)
                                                                                (CDR X)
                                                                                \COMMANDKEYACTION)
                                                                         (KEYACTION (CAR X)
                                                                                (CDR X)
                                                                                \DEFAULTKEYACTION)))
                    (T ;; On a SUN:  Some keyactions contradict "normal" ones, so reset them all.
                       (for X in (APPEND \ORIGKEYACTIONS (\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS))
                          do (KEYACTION (CAR X)
                                        (CDR X)
                                        \COMMANDKEYACTION)
                             (KEYACTION (CAR X)
                                        (CDR X)
                                        \DEFAULTKEYACTION]
                     (MOUSECHORDWAIT (MOUSECHORDWAIT)))
                    ((EQ (MACHINETYPE)
                         'MAIKO)
                     ;; Same machine type.  SO only worry if we're on SUNs, where the keyboard type can differ between machines.
                     (COND
                        ((NEQ \MAIKO.BEFOREKEYTYPE (LOGAND 7 (fetch (IFPAGE DEVCONFIG) of \InterfacePage)))
                         (for X in (APPEND \ORIGKEYACTIONS (\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS))
                            do (KEYACTION (CAR X)
                                          (CDR X)
                                          \COMMANDKEYACTION)
                               (KEYACTION (CAR X)
                                          (CDR X)
                                          \DEFAULTKEYACTION))
                         (MOUSECHORDWAIT (MOUSECHORDWAIT])
            NIL])
```

(\ALLOCLOCKED

```
  [LAMBDA (NCELLS)                                                    (* lmm "20-Apr-85 13:08")
                                                                      (* allocate a block of NCELLS cells and lock it)
    (PROG [(BLOCK (\ALLOCBLOCK NCELLS NIL (IMIN NCELLS CELLSPERPAGE]
          (\LOCKCELL BLOCK (FOLDHI (IPLUS (fetch (POINTER WORDINPAGE) of BLOCK)
                                          (UNFOLD NCELLS WORDSPERCELL))
                                   WORDSPERPAGE))
          (RETURN BLOCK])
```

(\SETIOPOINTERS

```
  [LAMBDA NIL                                                         ; Edited 28-Apr-88 01:10 by MASINTER
    (SELECTC (SETTOPVAL '\MACHINETYPE (fetch MachineType of \InterfacePage))
        ((LIST \DOLPHIN \DORADO)
             (SETTOPVAL '\EM.MOUSEX (EMADDRESS MOUSEX.EM))
             (SETTOPVAL '\EM.MOUSEY (EMADDRESS MOUSEY.EM))
             (SETTOPVAL '\EM.CURSORX (EMADDRESS CURSORX.EM))
             (SETTOPVAL '\EM.CURSORY (EMADDRESS CURSORY.EM))
             (SETTOPVAL '\EM.REALUTILIN (EMADDRESS UTILIN.EM))
             (SETTOPVAL '\EM.KBDAD0 (EMADDRESS KBDAD0.EM))
             (SETTOPVAL '\EM.KBDAD1 (EMADDRESS KBDAD1.EM))
             (SETTOPVAL '\EM.KBDAD2 (EMADDRESS KBDAD2.EM))
             (SETTOPVAL '\EM.KBDAD3 (EMADDRESS KBDAD3.EM))
             (SETTOPVAL '\EM.KBDAD4 (LOCF (fetch FAKEKBDAD4 of \InterfacePage)))
             (\PUTBASE \EM.KBDAD4 0 ALLUP)
             (SETTOPVAL '\EM.KBDAD5 (LOCF (fetch FAKEKBDAD5 OF \InterfacePage)))
             (\PUTBASE \EM.KBDAD5 0 ALLUP)
             (SETTOPVAL '\EM.DISPINTERRUPT (EMADDRESS DISPINTERRUPT.EM))
             (SETTOPVAL '\EM.CURSORBITMAP (EMADDRESS CURSORBITMAP.EM))
             (SETTOPVAL '\EM.DISPLAYHEAD (EMADDRESS DCB.EM))
             (SETTOPVAL 'SCREENWIDTH (UNFOLD (fetch ScreenWidth of \InterfacePage)
                                             BITSPERWORD)))
        ((LIST \DANDELION \MAIKO)
             (SETTOPVAL '\EM.MOUSEX (fetch DLMOUSEXPTR of \IOPAGE))
             (SETTOPVAL '\EM.MOUSEY (fetch DLMOUSEYPTR of \IOPAGE))
             (SETTOPVAL '\EM.CURSORX (fetch DLCURSORXPTR of \IOPAGE))
             (SETTOPVAL '\EM.CURSORY (fetch DLCURSORYPTR of \IOPAGE))
             (PROGN (SETTOPVAL '\EM.REALUTILIN (fetch DLUTILINPTR of \IOPAGE))
                    ;; Where the hardware bits live, vs.  where the Lisp software sees them after reinterpretation by keyhandler
                    )
             (SETTOPVAL '\EM.KBDAD0 (fetch DLKBDAD0PTR of \IOPAGE))
             (SETTOPVAL '\EM.KBDAD1 (fetch DLKBDAD1PTR of \IOPAGE))
             (SETTOPVAL '\EM.KBDAD2 (fetch DLKBDAD2PTR of \IOPAGE))
             (SETTOPVAL '\EM.KBDAD3 (fetch DLKBDAD3PTR of \IOPAGE))
             (SETTOPVAL '\EM.KBDAD4 (fetch DLKBDAD4PTR of \IOPAGE))
             (SETTOPVAL '\EM.KBDAD5 (fetch DLKBDAD5PTR of \IOPAGE))
             (SETTOPVAL '\EM.DISPINTERRUPT (fetch DLDISPINTERRUPTPTR of \IOPAGE))
             (SETTOPVAL '\EM.CURSORBITMAP (fetch DLCURSORBITMAPPTR of \IOPAGE))
             (SETTOPVAL '\EM.DISPLAYHEAD NIL)
             (SETTOPVAL 'SCREENWIDTH (SELECTC \MACHINETYPE
                                          (\MAIKO (SUBRCALL DSP-SCREENWIDTH))
                                          1024)))
```

```
        (\DAYBREAK (PROG ((KBDBASE (\DoveMisc.GetKBDBase)))
                         (SETTOPVAL '\EM.KBDAD0 (\ADDBASE KBDBASE 1))
                         (SETTOPVAL '\EM.KBDAD1 (\ADDBASE KBDBASE 2))
                         (SETTOPVAL '\EM.KBDAD2 (\ADDBASE KBDBASE 3))
                         (SETTOPVAL '\EM.KBDAD3 (\ADDBASE KBDBASE 4))
                         (SETTOPVAL '\EM.KBDAD4 (\ADDBASE KBDBASE 5))
                         (SETTOPVAL '\EM.KBDAD5 (\ADDBASE KBDBASE 6))
                         (SETTOPVAL '\EM.MOUSEX (\DoveMisc.GetMouseXBase))
                         (SETTOPVAL '\EM.MOUSEY (\DoveMisc.GetMouseYBase))
                         (SETTOPVAL '\EM.CURSORBITMAP (\DoveDisplay.GetCursorBitmapBase))
                                                          (* These three set this way to prevent address faults)
                         (SETTOPVAL '\EM.DISPINTERRUPT (fetch DLDISPINTERRUPTPTR of \IOPAGE))
                         (SETTOPVAL '\EM.CURSORX (fetch DLCURSORXPTR of \IOPAGE))
                         (SETTOPVAL '\EM.CURSORY (fetch DLCURSORYPTR of \IOPAGE))
                         (PROGN (SETTOPVAL '\EM.REALUTILIN KBDBASE)

          (* Where the hardware bits live, vs. where the Lisp software sees them after reinterpretation by keyhandler)

                                )
                         (SETTOPVAL 'SCREENWIDTH (\DoveDisplay.ScreenWidth))))
        (RAID))
    (SETTOPVAL '\EM.UTILIN (LOCF (fetch (IFPAGE FAKEMOUSEBITS) of \InterfacePage])
```

(\KEYBOARDOFF

```
  [LAMBDA NIL                                                ; Edited 20-Apr-88 10:28 by MASINTER
    (\PUTBASE \EM.DISPINTERRUPT 0 (LOGAND (LOGXOR 65535 \LispKeyMask)
                                          (\GETBASE \EM.DISPINTERRUPT 0)))
    (COND
       ((EQ \MACHINETYPE \MAIKO)
        (SUBRCALL KEYBOARDSTATE NIL])
```

(\KEYBOARDON

```
  [LAMBDA (NOCHECK)                                          ; Edited 24-Apr-88 00:03 by MASINTER
    (\SETIOPOINTERS)
    (\PUTBASE \EM.DISPINTERRUPT 0 (LOGOR \LispKeyMask (\GETBASE \EM.DISPINTERRUPT 0)))
    (COND
       ((EQ \MACHINETYPE \MAIKO)
        (SUBRCALL KEYBOARDSTATE T])
```

(\KEYHANDLER

```
  [LAMBDA NIL                                                (* lmm "30-MAR-83 20:40")
    (\KEYHANDLER1])
```

(\KEYHANDLER1

```
  [LAMBDA NIL                                                ; Edited 30-Mar-88 10:40 by Snow
    (PROG ((OLD0 ALLUP)
           (OLD1 ALLUP)
           (OLD2 ALLUP)
           (OLD3 ALLUP)
           (OLD4 ALLUP)
           (OLD5 ALLUP)
           (OLDU ALLUP)
           (OLDFAKEU ALLUP)
           (LOOPCNT 10)
           (PERIODCNT 60)
           (MOUSESTATE \DLMOUSE.UP)
           (MOUSETIMER (LOCF (fetch DLMOUSETIMER of \MISCSTATS)))
           (MOUSETEMP (LOCF (fetch DLMOUSETEMP of \MISCSTATS)))
          CURSORX CURSORY YHOT)
         (SETQ \KEYBUFFERING NIL)
         (replace (RING READ) of \KEYBOARDEVENTQUEUE with 0)
      LP (\CONTEXTSWITCH \KbdFXP)
         [COND
            (\PERIODIC.INTERRUPT                             (* eventually can be replaced with general timer mechanism)
                   (COND
                      ((IGREATERP PERIODCNT 0)               (* Continue counting down to zero)
                       (SETQ PERIODCNT (SUB1 PERIODCNT)))
                      ((\CAUSEINTERRUPT \KbdFXP (FUNCTION \PERIODIC.INTERRUPTFRAME))

          (* When we've counted down, then keep trying to cause the interrupt, and reset the counter when it finally happens)

                       (SETQ PERIODCNT (SUB1 (OR \PERIODIC.INTERRUPT.FREQUENCY 1]
         [COND
            ((OR (NEQ (\GETBASE \EM.MOUSEX 0)
                      CURSORX)
                 (NEQ (\GETBASE \EM.MOUSEY 0)
                      CURSORY))
             (\TRACKCURSOR (SETQ CURSORX (\GETBASE \EM.MOUSEX 0))
                    (SETQ CURSORY (\GETBASE \EM.MOUSEY 0]
         [COND
            ((OR [COND
                    ((OR (NEQ OLDU (\GETBASE \EM.REALUTILIN 0))
                         (COND
```

```
                        ((AND (EQ MOUSESTATE \DLMOUSE.WAITING)
                              (IGREATERP (\BOXIDIFFERENCE (\RCLK MOUSETEMP)
                                                          MOUSETIMER)
                                         0))                    (* Timer expired on seeing both left and right down, so set state
                                                                   to normal)
                         (SETQ MOUSESTATE \DLMOUSE.NORMAL)
                         T)))
                   (SETQ MOUSESTATE (\DOMOUSECHORDING (SETQ OLDU (\GETBASE \EM.REALUTILIN 0))
                                                      MOUSESTATE))
                   (NEQ OLDFAKEU (\GETBASE \EM.UTILIN 0]
               (NEQ OLD0 (\GETBASE \EM.KBDAD0 0))
               (NEQ OLD1 (\GETBASE \EM.KBDAD1 0))
               (NEQ OLD2 (\GETBASE \EM.KBDAD2 0))
               (NEQ OLD3 (\GETBASE \EM.KBDAD3 0))
               (NEQ OLD4 (\GETBASE \EM.KBDAD4 0))
               (NEQ OLD5 (\GETBASE \EM.KBDAD5 0)))
          (COND
             ((EQ 0 (LOGAND (\GETBASE \EM.KBDAD2 0)
                            2114))                              (* Ctrl-shift-DEL panic interrupt --
                                                                   switch to TeleRaid immediately)
              (swap (fetch (IFPAGE TELERAIDFXP) of \InterfacePage)
                    (fetch (IFPAGE KbdFXP) of \InterfacePage))
              (\KEYBOARDOFF)
              (SETQ OLD2 (\GETBASE \EM.KBDAD2 0))
              (GO LP)))
          [PROG ((W (fetch (RING WRITE) of \KEYBOARDEVENTQUEUE))
                 (R (fetch (RING READ) of \KEYBOARDEVENTQUEUE))
                 WPTR)
                (COND
                   ((EQ R W)                                    (* eventqueue full!)
                    (RETURN)))
                (SETQ WPTR (\ADDBASE \KEYBOARDEVENTQUEUE W))
                (\RCLK (LOCF (fetch TIME of WPTR)))
                [with KEYBOARDEVENT WPTR (PROGN (SETQ W0 (SETQ OLD0 (\GETBASE \EM.KBDAD0 0)))
                                                (SETQ W1 (SETQ OLD1 (\GETBASE \EM.KBDAD1 0)))
                                                (SETQ W2 (SETQ OLD2 (\GETBASE \EM.KBDAD2 0)))
                                                (SETQ W3 (SETQ OLD3 (\GETBASE \EM.KBDAD3 0)))
                                                (SETQ W4 (SETQ OLD4 (\GETBASE \EM.KBDAD4 0)))
                                                (SETQ W5 (SETQ OLD5 (\GETBASE \EM.KBDAD5 0)))
                                                (SETQ WU (SETQ OLDFAKEU (\GETBASE \EM.UTILIN 0]
                (COND
                   ((EQ R 0)                                    (* Queue was empty)
                    (replace (RING READ) of \KEYBOARDEVENTQUEUE with W)))
                (replace (RING WRITE) of \KEYBOARDEVENTQUEUE with (COND
                                                                     ((IGEQ W \KEYBOARDEVENT.LAST)
                                                                      \KEYBOARDEVENT.FIRST)
                                                                     (T (IPLUS W \KEYBOARDEVENT.SIZE]
          (OR \KEYBUFFERING (SETQ \KEYBUFFERING T]
      [COND
         [\KEYBUFFERING (COND
                           ((EQ \KEYBUFFERING T)
                            (COND
                               ((\CAUSEINTERRUPT \KbdFXP (FUNCTION \DOBUFFEREDTRANSITIONS))
                                (SETQ \KEYBUFFERING 'STARTED)
                                                                (* don't call until \DOBUFFEREDTRANSITIONS is done)
                                ]
         (T (COND
               (\PENDINGINTERRUPT (COND
                                     ((\CAUSEINTERRUPT \KbdFXP (FUNCTION \INTERRUPTFRAME))
                                      (SETQ \PENDINGINTERRUPT]
      [COND
         ((AND (NEQ \MACHINETYPE \MAIKO)
               (ILEQ (SETQ LOOPCNT (SUB1 LOOPCNT))
                     0))                                        (* Only do this once in a while)
          (SETQ LOOPCNT (COND
                           ((\UPDATETIMERS)

      (* Timer was updated, so do it next time around, too, in case we just came back from RAID or other bcpl code)

                           1)
                           (T 20]
      (COND
         ([AND NIL \TIMER.INTERRUPT.PENDING (IGREATERP (\BOXIDIFFERENCE (\RCLK (LOCF (fetch DLMOUSETEMP
                                                                                       of \MISCSTATS)))
                                                                        (LOCF (fetch DLMOUSETIMER of \MISCSTATS)))
                                                       0)
               (COND
                  ((EQ \TIMER.INTERRUPT.PENDING '\MOUSECHANGE)
                   (SETQ OLDU NIL)
                   T)
                  (T (\CAUSEINTERRUPT \KbdFXP (FUNCTION \TIMER.INTERRUPTFRAME]
          (SETQ \TIMER.INTERRUPT.PENDING)))
      (GO LP])
```

(**RESETKEYBOARD**
  [LAMBDA NIL ; Edited 30-Mar-88 10:07 by Snow

```
(\SETIOPOINTERS)                                    (* Called with lisp keyboard disabled whenever Lisp is resumed
from bcpl logout or copysys.)
(SETQ \KEYBUFFERING NIL)
(COND
   ((OR (EQ \MACHINETYPE \DANDELION)
        (EQ \MACHINETYPE \DAYBREAK)
        (EQ \MACHINETYPE \MAIKO))                   (* Initialize fake mouse bits to all up)
    (\PUTBASE \EM.UTILIN 0 ALLUP)))
(with KEYBOARDEVENT \LASTKEYSTATE (SETQ W0 (\GETBASE \EM.KBDAD0 0))
      (SETQ W1 (\GETBASE \EM.KBDAD1 0))
      (SETQ W2 (\GETBASE \EM.KBDAD2 0))
      (SETQ W3 (\GETBASE \EM.KBDAD3 0))
      (SETQ W4 (\GETBASE \EM.KBDAD4 0))
      (SETQ W5 (\GETBASE \EM.KBDAD5 0))
      (SETQ WU (\GETBASE \EM.REALUTILIN 0))
      (SETQ LOCK (XKEYDOWNP 'LOCK))
      (SETQ 1SHIFT NIL)
      (SETQ 2SHIFT NIL)
      (SETQ CTRL NIL)
      (SETQ META NIL)
      (SETQ FONT NIL)
      (SETQ USERMODE1 NIL)
      (SETQ USERMODE2 NIL)
      (SETQ USERMODE3 NIL)
      (SETQ MOUSESTATE \DLMOUSE.UP))
(SETQ \TIMER.INTERRUPT.PENDING)
(replace (RING READ) of \KEYBOARDEVENTQUEUE with 0)
(replace (RING READ) of \SYSBUFFER with 0)
(SETQ \LONGSYSBUF)
(\DAYTIME0 \LASTUSERACTION)
(\KEYBOARDON])
```

(\DOMOUSECHORDING

```
[LAMBDA (REALUTILIN STATE)                          (* bvm%: " 9-Oct-85 11:24")
```

(* Handles mouse transitions on a DLion. REALUTILIN is the actual util word from the processor.
STATE is our internal state. Sets contents of \EM.UTILIN to reflect the virtual mouse state, which may contain a middle mouse button even where there is only a two-button mouse. Returns new state)

```
(PROG (LRSTATE)
      [COND
         ((OR (NULL \MOUSECHORDTICKS)
              (EQ (SETQ LRSTATE (LOGXOR (LOGAND REALUTILIN \MOUSE.ALLBITS)
                                        \MOUSE.ALLBITS))
                  0))
```

(* Not interpreting chording, or both LEFT and RIGHT are up --
real state and virtual state the same)

```
          (SETQ STATE \DLMOUSE.UP))
         (T                                         (* Either L or R or both are down, so have to decide about
                                                    Middle)
            (SELECTC STATE
                 ((LIST \DLMOUSE.UP \DLMOUSE.WAITING)
                      (SETQ REALUTILIN (LOGOR REALUTILIN \MOUSE.LRBIT))
                                                    (* Turn off the L and/or R bits)
                      (COND
                         ((EQ LRSTATE \MOUSE.LRBIT) (* Both L and R down at once, interpret as MIDDLE without
                                                    waiting)
                          (SETQ REALUTILIN (LOGAND (LOGXOR ALLUP \MOUSE.MIDDLEBIT)
                                                   REALUTILIN))
                          (SETQ STATE \DLMOUSE.MIDDLE))
                         ((NEQ STATE \DLMOUSE.WAITING)
```

(* Only one of L and R down. Set timer, and ignore the down bit for now)

```
                          (\BOXIPLUS (\RCLK (LOCF (fetch DLMOUSETIMER of \MISCSTATS)))
                                \MOUSECHORDTICKS)
                          (SETQ STATE \DLMOUSE.WAITING))))
                 (\DLMOUSE.MIDDLE
```

(* State is middle and at least one of L and R is still down, so consider it to be still only middle)

```
                      (SETQ REALUTILIN (LOGAND (LOGXOR ALLUP \MOUSE.MIDDLEBIT)
                                               (LOGOR REALUTILIN \MOUSE.LRBIT)))
                      (SELECTC LRSTATE
                           (\MOUSE.LEFTBIT           (* Right came up. Henceforth treat right transparently)
                               (SETQ STATE \DLMOUSE.MIDDLE&RIGHT))
                           (\MOUSE.RIGHTBIT          (* Left came up. Henceforth treat left transparently)
                               (SETQ STATE \DLMOUSE.MIDDLE&LEFT))
                           NIL))
                 (\DLMOUSE.MIDDLE&RIGHT             (* Only ignore LEFT)
                      (SETQ REALUTILIN (LOGAND (LOGXOR ALLUP \MOUSE.MIDDLEBIT)
                                               (LOGOR REALUTILIN \MOUSE.LEFTBIT))))
                 (\DLMOUSE.MIDDLE&LEFT              (* Only ignore RIGHT)
                      (SETQ REALUTILIN (LOGAND (LOGXOR ALLUP \MOUSE.MIDDLEBIT)
```

```
                                                    (LOGOR REALUTILIN \MOUSE.RIGHTBIT))))
                    (PROGN
```

(* Remaining state is \DLMOUSE.NORMAL which means treat mouse normally, and the only interesting transition is back to \DLMOUSE.UP)

```
]
          (\PUTBASE \EM.UTILIN 0 REALUTILIN)
          (RETURN STATE])
```

(**DOTRANSITIONS**

```
  [LAMBDA (KEYBASE OLD NEW)                                          ; Edited  1-Feb-92 11:59 by jds
```

;; OLD and NEW are keyboard state words that are known to have changed. KEYBASE is the number in hardware order of the key corresponding ;; to the first bit in these words. This function figures out the indices of transitioning keys and calls the decoder.

```
    (for I (BITMASK _ (LLSH 1 15)) from 0 to 15 do [OR (EQ 0 (LOGAND BITMASK (LOGXOR OLD NEW)))
                                                       (\DECODETRANSITION (IPLUS I KEYBASE)
                                                              (EQ 0 (LOGAND NEW BITMASK]
                                                    (SETQ BITMASK (LRSH BITMASK 1)))
    T])
```

(**DECODETRANSITION**

```
  [LAMBDA (KEYNUMBER DOWNFLG)                                        ; Edited  3-Jan-2024 16:04 by mth
                                                                     ; Edited 19-Nov-87 16:29 by Snow
```

;; KEYNUMBER is the key number in the hardware keyboard layout, DOWNFLG is T if the key just went down. PENDINGINTERRUPT, bound in ;; \KEYHANDLER, is set to the decoded character if it is an interrupt.

```
    (.NOTELASTUSERACTION)
    (PROG ((TI (\TRANSINDEX KEYNUMBER DOWNFLG))
           (KEYSTATE \LASTKEYSTATE)
           ASCIICODE SHIFTED)
          (SELECTC (TRANSITIONFLAGS \CURRENTKEYACTION TI)
               (IGNORE.TF (RETURN))
               (LOCKSHIFT.TF ;; Take shift action if either Shift or Caps Lock is down.
                             ;; If SHIFTXORLOCKFLG, but not both!
                             (IF SHIFTXORLOCKFLG
                                 THEN (SETQ SHIFTED (fetch (KEYBOARDEVENT SHIFTXORLOCK) of KEYSTATE))
                               ELSE (SETQ SHIFTED (fetch (KEYBOARDEVENT SHIFTORLOCK) of KEYSTATE))))
               (NOLOCKSHIFT.TF ;; Take shift action only when Shift is down
                    (SETQ SHIFTED (fetch (KEYBOARDEVENT SHIFT) of KEYSTATE)))
               (EVENT.TF (RETURN))
               (1SHIFTUP.TF (replace (KEYBOARDEVENT 1SHIFT) of KEYSTATE with NIL)
                            (RETURN))
               (1SHIFTDOWN.TF
                    (replace (KEYBOARDEVENT 1SHIFT) of KEYSTATE with T)
                    (RETURN))
               (2SHIFTUP.TF (replace (KEYBOARDEVENT 2SHIFT) of KEYSTATE with NIL)
                            (RETURN))
               (2SHIFTDOWN.TF
                    (replace (KEYBOARDEVENT 2SHIFT) of KEYSTATE with T)
                    (RETURN))
               (LOCKUP.TF (replace (KEYBOARDEVENT LOCK) of KEYSTATE with NIL)
                          (RETURN))
               (LOCKDOWN.TF (replace (KEYBOARDEVENT LOCK) of KEYSTATE with T)
                            (RETURN))
               (LOCKTOGGLE.TF
                    (replace (KEYBOARDEVENT LOCK) of KEYSTATE with (NOT (fetch (KEYBOARDEVENT LOCK) of KEYSTATE)))
                    (RETURN))
               (CTRLUP.TF (replace (KEYBOARDEVENT CTRL) of KEYSTATE with NIL)
                          (RETURN))
               (CTRLDOWN.TF (replace (KEYBOARDEVENT CTRL) of KEYSTATE with T)
                            (RETURN))
               (METAUP.TF (replace (KEYBOARDEVENT META) of KEYSTATE with NIL)
                          (RETURN))
               (METADOWN.TF (replace (KEYBOARDEVENT META) of KEYSTATE with T)
                            (RETURN))
               (FONTUP.TF (replace (KEYBOARDEVENT FONT) of KEYSTATE with NIL)
                          (RETURN))
               (FONTDOWN.TF (replace (KEYBOARDEVENT FONT) of KEYSTATE with T)
                            (RETURN))
               (FONTTOGGLE.TF
                    (replace (KEYBOARDEVENT FONT) of KEYSTATE with (NOT (fetch (KEYBOARDEVENT FONT) of KEYSTATE)))
                    (RETURN))
               (USERMODE1UP.TF
                    (replace (KEYBOARDEVENT USERMODE1) of KEYSTATE with NIL)
                    (RETURN))
               (USERMODE1DOWN.TF
                    (replace (KEYBOARDEVENT USERMODE1) of KEYSTATE with T)
                    (RETURN))
               (USERMODE1TOGGLE.TF
                    (replace (KEYBOARDEVENT USERMODE1) of KEYSTATE with (NOT (fetch (KEYBOARDEVENT USERMODE1)
                                                                                    of KEYSTATE)))
                    (RETURN))
               (USERMODE2UP.TF
```

```
                    (replace (KEYBOARDEVENT USERMODE2) of KEYSTATE with NIL)
                    (RETURN))
               (USERMODE2DOWN.TF
                    (replace (KEYBOARDEVENT USERMODE2) of KEYSTATE with T)
                    (RETURN))
               (USERMODE2TOGGLE.TF
                    (replace (KEYBOARDEVENT USERMODE2) of KEYSTATE with (NOT (fetch (KEYBOARDEVENT USERMODE2)
                                                                                 of KEYSTATE)))
                    (RETURN))
               (USERMODE3UP.TF
                    (replace (KEYBOARDEVENT USERMODE3) of KEYSTATE with NIL)
                    (RETURN))
               (USERMODE3DOWN.TF
                    (replace (KEYBOARDEVENT USERMODE3) of KEYSTATE with T)
                    (RETURN))
               (USERMODE3TOGGLE.TF
                    (replace (KEYBOARDEVENT USERMODE3) of KEYSTATE with (NOT (fetch (KEYBOARDEVENT USERMODE3)
                                                                                 of KEYSTATE)))
                    (RETURN))
               (SHOULDNT))
```

;; Only the LOCKSHIFT and NOLOCKSHIFT cases make it to here, having set SHIFTED if appropriate.

```
      [SETQ ASCIICODE (COND
                         (SHIFTED (TRANSITIONSHIFTCODE \CURRENTKEYACTION TI))
                         (T (TRANSITIONCODE \CURRENTKEYACTION TI]
      [COND
         ((OR (fetch (KEYBOARDEVENT CTRL) of KEYSTATE)
              (fetch (KEYBOARDEVENT META) of KEYSTATE)
              (fetch (KEYBOARDEVENT FONT) of KEYSTATE))
          [IF (IGREATERP ASCIICODE 127)
              THEN ;; Non-ascii interpretation--what is cntrl/meta supposed to mean?  Try using the original interpretation.  This way we can
                   ;; type ^E or Meta-D even if Russian keyboard is set, but doesn't mess up simple ascii remappings, such as bs->del.
                   (SETQ ASCIICODE (COND
                                      (SHIFTED (TRANSITIONSHIFTCODE \COMMANDKEYACTION TI))
                                      (T (TRANSITIONCODE \COMMANDKEYACTION TI]
          [COND
             ((fetch (KEYBOARDEVENT CTRL) of KEYSTATE)
              (SETQ ASCIICODE (LOGAND ASCIICODE \CTRLMASK]
          (COND
             ((AND (OR (fetch (KEYBOARDEVENT META) of KEYSTATE)
                       (fetch (KEYBOARDEVENT FONT) of KEYSTATE))
                   (ILESSP ASCIICODE \KEYBOARD.META))
              (SETQ ASCIICODE (LOGOR ASCIICODE \KEYBOARD.META]
      (COND
         ((ASSOC ASCIICODE (fetch INTERRUPTLIST of \CURRENTKEYACTION))
          (SETQ PENDINGINTERRUPT T)
          (replace WAITINGINTERRUPT of \INTERRUPTSTATE with T)
          (replace INTCHARCODE of \INTERRUPTSTATE with ASCIICODE))
         (T (\PUTSYSBUF ASCIICODE])
```

(**MOUSECHORDWAIT**

```
  [LAMBDA MSECS                                                          (* MPL "21-Jun-85 16:31")
    (DECLARE (GLOBALVARS \RCLKMILLISECOND))
    (PROG1 (AND \MOUSECHORDTICKS \MOUSECHORDMILLISECONDS)
        [COND
           ((IGREATERP MSECS 0)
            (SETQ \MOUSECHORDTICKS (AND (ARG MSECS 1)
                                        (IMIN MAX.SMALLP (ITIMES (SETQ \MOUSECHORDMILLISECONDS
                                                                   (OR (SMALLP (ARG MSECS 1))
                                                                       50))
                                                                 \RCLKMILLISECOND])])
```

(**\TRACKCURSOR**

```
  [LAMBDA (CURSORX CURSORY)                                              ; Edited 30-Mar-88 11:11 by Snow
    (DECLARE (GLOBALVARS \CURSORDESTHEIGHT \CURSORDESTWIDTH))
    (.NOTELASTUSERACTION)
    [COND
       ((OR [COND
              ((IGEQ CURSORX (IDIFFERENCE \CURSORDESTWIDTH \CURSORHOTSPOTX))

            (* Large cursor values are either out of bounds to the right or are negative values
            (16-bit bcpl signed numbers))

             (COND
                [(IGREATERP CURSORX 32767)                               (* Cursor value is negative)
                 (COND
                    ((ILESSP (IPLUS (SUB1 (IDIFFERENCE CURSORX 65535))
                                    \CURSORHOTSPOTX)
                             0)

            (* Cursor pos + hotspot is still off to the left (the IPLUS is an optimization of
            (\XMOUSECOORD))%, so clip to effective zero)

                     (SETQ CURSORX (COND
```

```
                                        ((EQ \MACHINETYPE \DANDELION)
                                                                (* Temporary workaround)
                                         0)
                                        (T (UNSIGNED (IMINUS \CURSORHOTSPOTX)
                                                     BITSPERWORD]
                    (T (SETQ CURSORX (SUB1 (IDIFFERENCE \CURSORDESTWIDTH \CURSORHOTSPOTX]
               (IGEQ CURSORY (IDIFFERENCE \CURSORDESTHEIGHT HARDCURSORHEIGHT)))
```

(* repeat test so that both X and Y will get clipped each cycle. This keeps the cursor from moving off the screen.)

```
            [COND
               ((IGEQ CURSORY (IDIFFERENCE \CURSORDESTHEIGHT \CURSORHOTSPOTY))
```

(* Large cursor values are either out of bounds to the bottom or are negative values
(16-bit bcpl signed numbers))

```
                  (COND
                     [(IGREATERP CURSORY 32767)                    (* Cursor value is negative)
                      (COND
                         ((ILESSP (IPLUS (SUB1 (IDIFFERENCE CURSORY 65535))
                                         \CURSORHOTSPOTY)
                                  0)
```

(* Cursor pos + hotspot is still off to the top, so clip to effective zero)

```
                           (SETQ CURSORY (COND
                                            ((OR (EQ \MACHINETYPE \DANDELION)
                                                 (EQ \MACHINETYPE \DAYBREAK))
                                                                (* Temporary workaround)
                                             0)
                                            (T (UNSIGNED (IMINUS \CURSORHOTSPOTY)
                                                         BITSPERWORD]
                        (T (SETQ CURSORY (SUB1 (IDIFFERENCE \CURSORDESTHEIGHT \CURSORHOTSPOTY]
```

(* If need to clip mouse, do so here. \SETMOUSEXY MACRO takes dlion complexities into account.)

```
            (COND
               ((NEQ \MACHINETYPE \MAIKO)
                (\SETMOUSEXY CURSORX CURSORY]
         (COND
            (\SOFTCURSORUPP (\SOFTCURSORPOSITION CURSORX CURSORY)))
         (COND
            ((EQ \MACHINETYPE \DAYBREAK)                         (* Have to kick DAYBREAK IOP to track the cursor.
                                                                 *)
             (\DoveDisplay.SetCursorPosition CURSORX CURSORY)))
         (\PUTBASE \EM.CURSORX 0 CURSORX)
         (\PUTBASE \EM.CURSORY 0 CURSORY])

)

(DECLARE%: EVAL@COMPILE

(RPAQQ \SUN.TYPE3KEYBOARD 0)

(RPAQQ \SUN.TYPE4KEYBOARD 1)

(RPAQQ \SUN.JLEKEYBOARD 2)

(RPAQQ \TOSHIBA.JIS 7)

(CONSTANTS (\SUN.TYPE3KEYBOARD 0)
       (\SUN.TYPE4KEYBOARD 1)
       (\SUN.JLEKEYBOARD 2)
       (\TOSHIBA.JIS 7))
)

(RPAQ? \MOUSECHORDTICKS )

(RPAQ? \MOUSECHORDMILLISECONDS 50)

(RPAQ? SHIFTXORLOCKFLG NIL)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\KEYBOARDINIT)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS .NOTELASTUSERACTION MACRO (NIL (\BLT \LASTUSERACTION (LOCF (fetch SECONDSTMP of \MISCSTATS))
                                                WORDSPERCELL)))
)

(DECLARE%: EVAL@COMPILE
```

```
(RPAQQ ALLUP 65535)

(RPAQQ \CTRLMASK 159)

(RPAQQ \METABIT 128)

(CONSTANTS ALLUP \CTRLMASK \METABIT)
)

(RPAQQ DLMOUSEBITS ((\MOUSE.LEFTBIT 4)
                    (\MOUSE.RIGHTBIT 2)
                    (\MOUSE.MIDDLEBIT 1)
                    (\MOUSE.ALLBITS 7)
                    (\MOUSE.LRBIT 6)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \MOUSE.LEFTBIT 4)

(RPAQQ \MOUSE.RIGHTBIT 2)

(RPAQQ \MOUSE.MIDDLEBIT 1)

(RPAQQ \MOUSE.ALLBITS 7)

(RPAQQ \MOUSE.LRBIT 6)

(CONSTANTS (\MOUSE.LEFTBIT 4)
       (\MOUSE.RIGHTBIT 2)
       (\MOUSE.MIDDLEBIT 1)
       (\MOUSE.ALLBITS 7)
       (\MOUSE.LRBIT 6))
)

(RPAQQ DLMOUSESTATES ((\DLMOUSE.UP 0)
                      (\DLMOUSE.WAITING 1)
                      (\DLMOUSE.NORMAL 2)
                      (\DLMOUSE.MIDDLE 3)
                      (\DLMOUSE.MIDDLE&LEFT 4)
                      (\DLMOUSE.MIDDLE&RIGHT 5)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \DLMOUSE.UP 0)

(RPAQQ \DLMOUSE.WAITING 1)

(RPAQQ \DLMOUSE.NORMAL 2)

(RPAQQ \DLMOUSE.MIDDLE 3)

(RPAQQ \DLMOUSE.MIDDLE&LEFT 4)

(RPAQQ \DLMOUSE.MIDDLE&RIGHT 5)

(CONSTANTS (\DLMOUSE.UP 0)
       (\DLMOUSE.WAITING 1)
       (\DLMOUSE.NORMAL 2)
       (\DLMOUSE.MIDDLE 3)
       (\DLMOUSE.MIDDLE&LEFT 4)
       (\DLMOUSE.MIDDLE&RIGHT 5))
)

(RPAQQ TRANSITIONFLAGS
       (ALTGRDOWN.TF ALTGRUP.TF ALTGRTOGGLE.TF CTRLDOWN.TF CTRLUP.TF DEADKEY.TF IGNORE.TF EVENT.TF LOCKDOWN.TF
              LOCKSHIFT.TF LOCKTOGGLE.TF LOCKUP.TF NOLOCKSHIFT.TF 1SHIFTDOWN.TF 1SHIFTUP.TF 2SHIFTDOWN.TF
              2SHIFTUP.TF METADOWN.TF METAUP.TF FONTDOWN.TF FONTUP.TF FONTTOGGLE.TF USERMODE1UP.TF
              USERMODE1DOWN.TF USERMODE1TOGGLE.TF USERMODE2UP.TF USERMODE2DOWN.TF USERMODE2TOGGLE.TF
              USERMODE3UP.TF USERMODE3DOWN.TF USERMODE3TOGGLE.TF))

(DECLARE%: EVAL@COMPILE

(RPAQQ ALTGRDOWN.TF 27)

(RPAQQ ALTGRUP.TF 28)

(RPAQQ ALTGRTOGGLE.TF 29)

(RPAQQ CTRLDOWN.TF 5)

(RPAQQ CTRLUP.TF 4)

(RPAQQ DEADKEY.TF 30)

(RPAQQ IGNORE.TF 0)

(RPAQQ EVENT.TF 1)
```

```
(RPAQQ LOCKDOWN.TF 8)

(RPAQQ LOCKSHIFT.TF 2)

(RPAQQ LOCKTOGGLE.TF 14)

(RPAQQ LOCKUP.TF 7)

(RPAQQ NOLOCKSHIFT.TF 3)

(RPAQQ 1SHIFTDOWN.TF 6)

(RPAQQ 1SHIFTUP.TF 9)

(RPAQQ 2SHIFTDOWN.TF 11)

(RPAQQ 2SHIFTUP.TF 10)

(RPAQQ METADOWN.TF 13)

(RPAQQ METAUP.TF 12)

(RPAQQ FONTDOWN.TF 24)

(RPAQQ FONTUP.TF 25)

(RPAQQ FONTTOGGLE.TF 26)

(RPAQQ USERMODE1UP.TF 15)

(RPAQQ USERMODE1DOWN.TF 16)

(RPAQQ USERMODE1TOGGLE.TF 17)

(RPAQQ USERMODE2UP.TF 18)

(RPAQQ USERMODE2DOWN.TF 19)

(RPAQQ USERMODE2TOGGLE.TF 20)

(RPAQQ USERMODE3UP.TF 21)

(RPAQQ USERMODE3DOWN.TF 22)

(RPAQQ USERMODE3TOGGLE.TF 23)

(CONSTANTS ALTGRDOWN.TF ALTGRUP.TF ALTGRTOGGLE.TF CTRLDOWN.TF CTRLUP.TF DEADKEY.TF IGNORE.TF EVENT.TF
       LOCKDOWN.TF LOCKSHIFT.TF LOCKTOGGLE.TF LOCKUP.TF NOLOCKSHIFT.TF 1SHIFTDOWN.TF 1SHIFTUP.TF 2SHIFTDOWN.TF
       2SHIFTUP.TF METADOWN.TF METAUP.TF FONTDOWN.TF FONTUP.TF FONTTOGGLE.TF USERMODE1UP.TF USERMODE1DOWN.TF
       USERMODE1TOGGLE.TF USERMODE2UP.TF USERMODE2DOWN.TF USERMODE2TOGGLE.TF USERMODE3UP.TF USERMODE3DOWN.TF
       USERMODE3TOGGLE.TF)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \TRANSINDEX MACRO ((KEYNUMBER DOWNFLG)
                             (COND
                                (DOWNFLG (IPLUS \NKEYS KEYNUMBER))
                                (T KEYNUMBER))))

(PUTPROPS ARMEDCODE MACRO ((TABLE CHAR)
                           (\GETBASEBIT (fetch (KEYACTION ARMED)
                                               TABLE)
                                  CHAR)))

(PUTPROPS TRANSITIONALTGRCODE MACRO ((TABLE CHAR)
                                     (\GETBASE (fetch (KEYACTION ALTGRAPHCODES) of TABLE)
                                            CHAR)))

(PUTPROPS TRANSITIONSHIFTCODE MACRO ((TABLE CHAR)
                                     (\GETBASE (fetch (KEYACTION SHIFTCODES)
                                                      TABLE)
                                           CHAR)))

(PUTPROPS TRANSITIONCODE MACRO ((TABLE CHAR)
                                (\GETBASE (fetch (KEYACTION CODES)
                                                 TABLE)
                                      CHAR)))

(PUTPROPS TRANSITIONFLAGS MACRO ((TABLE CHAR)
                                 (\GETBASEBYTE (fetch (KEYACTION FLAGS)
                                                      TABLE)
                                       CHAR)))

(PUTPROPS TRANSITIONDEADLIST MACRO ((TABLE CHAR SHIFTED)
                                    (\GETBASEPTR (fetch (KEYACTION DEADKEYLIST) of TABLE)
```

```
                                        (LLSH (COND
                                                  (SHIFTED (IPLUS CHAR \NKEYS \NKEYS))
                                                  (T CHAR))
                                              1))))

(PUTPROPS CHECKFORDEADKEY MACRO [(KEYCODE TABLE CHAR SHIFTED)
                                 (LET ((CODE KEYCODE))
                                      (COND
                                         [(IEQP CODE 65535)
                                          `(DEADKEY ,(\GETBASEPTR (fetch (KEYACTION DEADKEYLIST)
                                                                     of TABLE)
                                                                  (LLSH (COND
                                                                           (SHIFTED (IPLUS CHAR \NKEYS \NKEYS))
                                                                           (T CHAR))
                                                                        1]
                                         (T CODE])
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

[BLOCKRECORD KEYACTION (;; KEYACTION Table:  For interpreting keystrokes.  Stored as a 8-cell block of untyped pointer hunk storage.
                        FLAGS                                   ; Flag byte per key# (one for down-transtion, 1 for up-.) to
                                                                ; describe whether lockshifting occrrs, you ignore the transition,
                                                                ; etc.
                        CODES                                   ; Table of character codes generated by each key when no shift
                                                                ; key is pressed.
                        SHIFTCODES                              ; Table of character codes generated by each key when the shift
                                                                ; key is pressed.
                        ARMED                                   ; Not sure...
                        INTERRUPTLIST                           ; List of armed interrupts?
                        ALTGRAPHCODES                           ; Table of codes to be generated when the ALT-GRAPH key is
                                                                ; pressed.
                        DEADKEYLIST                             ; Block of dead-key handlers, with the nominal up-transition fields
                                                                ; filled by the shifted-case tables.  Each "table" is an ALIST of
                                                                ; orignal code => accented code.  no entry means punt the
                                                                ; accent..

                        )
       FLAGS _ (\ALLOCBLOCK (FOLDHI (IPLUS \NKEYS \NKEYS)
                                    BYTESPERCELL))
       CODES _ (\ALLOCBLOCK (FOLDHI (PLUS \NKEYS \NKEYS)
                                    WORDSPERCELL))
       SHIFTCODES _ (\ALLOCBLOCK (FOLDHI (PLUS \NKEYS \NKEYS)
                                         WORDSPERCELL))
       ARMED _ (\ALLOCBLOCK (FOLDHI (ADD1 \MAXTHINCHAR)
                                    BITSPERCELL))
       ALTGRAPHCODES _ (\ALLOCBLOCK (FOLDHI (PLUS \NKEYS \NKEYS)
                                           WORDSPERCELL))
       DEADKEYLIST _ (\ALLOCBLOCK (PLUS \NKEYS \NKEYS \NKEYS \NKEYS)
                                  T)
       (CREATE (\ALLOCBLOCK 7 PTRBLOCK.GCT))
       (TYPE? (AND (\BLOCKDATAP DATUM)
                   (IGEQ (\#BLOCKDATACELLS DATUM)
                         5)
                   (OR (NULL (FETCH (KEYACTION INTERRUPTLIST) OF DATUM))
                       (LISTP (FETCH INTERRUPTLIST OF DATUM)))
                   (\BLOCKDATAP (FETCH (KEYACTION FLAGS)
                                       DATUM))
                   (\BLOCKDATAP (FETCH (KEYACTION CODES)
                                       DATUM))
                   (\BLOCKDATAP (FETCH (KEYACTION ARMED)
                                       DATUM]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \NKEYS 112)

(CONSTANTS \NKEYS)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD RING ((READ WORD)
                   (WRITE WORD)))
)

;; can get rid of shiftstate after clients have been fixed

(DECLARE%: EVAL@COMPILE
```

```
[ACCESSFNS SHIFTSTATE [[DUMMYSHIFT (NOT (EQ 0 (LOGAND (\GETBASEBYTE DATUM 0)
                                                       (LOGOR 1 2]
                           [DUMMY1SHIFT [NOT (EQ 0 (LOGAND 1 (\GETBASEBYTE DATUM 0]
                                  (\PUTBASEBYTE DATUM 0 (COND
                                                       (NEWVALUE (LOGOR 1 (\GETBASEBYTE DATUM 0)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 0)
                                                                  (LOGXOR \CHARMASK 1]
                           [DUMMY2SHIFT [NOT (EQ 0 (LOGAND 2 (\GETBASEBYTE DATUM 0]
                                  (\PUTBASEBYTE DATUM 0 (COND
                                                       (NEWVALUE (LOGOR 2 (\GETBASEBYTE DATUM 0)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 0)
                                                                  (LOGXOR \CHARMASK 2]
                           [DUMMYLOCK [NOT (EQ 0 (LOGAND 4 (\GETBASEBYTE DATUM 0]
                                  (\PUTBASEBYTE DATUM 0 (COND
                                                       (NEWVALUE (LOGOR 4 (\GETBASEBYTE DATUM 0)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 0)
                                                                  (LOGXOR \CHARMASK 4]
                           [DUMMYSHIFTORLOCK (NOT (EQ 0 (\GETBASEBYTE DATUM 0)))
                                  (\PUTBASEBYTE DATUM 0 (COND
                                                       (NEWVALUE (HELP " Can't turn on SHIFTORLOCK"))
                                                       (T 0]
                           [DUMMYCTRL (NOT (EQ 0 (\GETBASEBYTE DATUM 1)))
                                  (\PUTBASEBYTE DATUM 1 (COND
                                                       (NEWVALUE 1)
                                                       (T 0]
                           [DUMMYMETA (NOT (EQ 0 (\GETBASEBYTE DATUM 2)))
                                  (\PUTBASEBYTE DATUM 2 (COND
                                                       (NEWVALUE 1)
                                                       (T 0]
                           [DUMMYFONT (NEQ 0 (LOGAND (LLSH 1 3)
                                                     (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 3)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 3]
                           [DUMMYUSERMODE1 (NEQ 0 (LOGAND (LLSH 1 0)
                                                           (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 0)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 0]
                           [DUMMYUSERMODE2 (NEQ 0 (LOGAND (LLSH 1 1)
                                                           (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 1)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 1]
                           [DUMMYUSERMODE3 (NEQ 0 (LOGAND (LLSH 1 2)
                                                           (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 2)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 2]
                           [DUMMYALTGRAPH (NEQ 0 (LOGAND (LLSH 1 4)
                                                          (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 4)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 4]
                           (DUMMYDEADKEYPENDING (NEQ 0 (LOGAND (LLSH 1 5)
                                                                (\GETBASEBYTE DATUM 3)))
                                  (\PUTBASEBYTE DATUM 3 (COND
                                                       (NEWVALUE (LOGOR (LLSH 1 5)
                                                                        (\GETBASEBYTE DATUM 3)))
                                                       (T (LOGAND (\GETBASEBYTE DATUM 3)
                                                                  (LOGXOR \CHARMASK (LLSH 1 5]
       (CREATE (\ALLOCBLOCK (FOLDHI 3 BYTESPERCELL]
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \SHIFTSTATE \MOUSETIMERTEMP)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ NRINGINDEXWORDS 2)

(CONSTANTS NRINGINDEXWORDS)
)

(DECLARE%: EVAL@COMPILE
```

```
(RPAQ \SYSBUFFER.FIRST (UNFOLD NRINGINDEXWORDS BYTESPERWORD))

(RPAQ \SYSBUFFER.LAST (IPLUS \SYSBUFFER.FIRST (SUB1 \SYSBUFSIZE)))

[CONSTANTS (\SYSBUFFER.FIRST (UNFOLD NRINGINDEXWORDS BYTESPERWORD))
       (\SYSBUFFER.LAST (IPLUS \SYSBUFFER.FIRST (SUB1 \SYSBUFSIZE]
)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \KEYNAMES
       ((5 %% FIVE)
        (4 $ FOUR)
        (6 ~ SIX)
        (e E)
        (7 & SEVEN)
        (d D)
        (u U)
        (v V)
        (0 %) ZERO)
        (k K)
        (- %Û)
        (p P)
        (/ ?)
        (\ %| FONT LOOKS)
        (LF SAME)
        (BS <-)
        (3 %# THREE)
        (2 @ TWO)
        (w W)
        (q Q)
        (s S)
        (a A)
        (9 %( NINE)
        (i I)
        (x X)
        (o O)
        (l L)
        (%, <)
        (%’ %")
        (%] })
        (BLANK-MIDDLE OPEN DBK-HELP)
        (BLANK-TOP KEYBOARD DBK-META)
        (1 ! ONE)
        (ESC ESCAPE ->)
        (TAB =>)
        (f F)
        (CTRL PROP’S EDIT)
        (c C)
        (j J)
        (b B)
        (z Z)
        (LSHIFT)
        (%. >)
        (; %:)
        (CR <-%|)
        (_ ^)
        (DEL DELETE)
        (SKIP NEXT)
        (r R)
        (t T)
        (g G)
        (y Y)
        (h H)
        (8 * EIGHT)
        (n N)
        (m M)
        (LOCK)
        (SPACE)
        (%[ {)
        (= +)
        (RSHIFT)
        (BLANK-BOTTOM STOP)
        (MOVE)
        (UNDO)
        (UTIL0 SUN-KEYPAD=)
        (UTIL1 SUN-KEYPAD/)
        (UTIL2 SUPER/SUB)
        (UTIL3 CASE)
        (UTIL4 STRIKEOUT)
        (UTIL5 KEYPAD2)
        (UTIL6 KEYPAD3 PGDN)
        (UTIL7 SUN-LF)
        (PAD1 LEFTKEY CAPSLOCK KEYPAD+)
        (PAD2 LEFTMIDDLEKEY NUMLOCK KEYPAD-)
```

```
        (PAD3 MIDDLEKEY SCROLLLOCK KEYPAD*)
        (PAD4 RIGHTMIDDLEKEY BREAK KEYPAD/ SUN-PAUSE)
        (PAD5 RIGHTKEY DOIT PRTSC)
        (LEFT RED MOUSERED)
        (RIGHT BLUE MOUSEBLUE)
        (MIDDLE YELLOW MOUSEYELLOW)
        (MARGINS)
        (K41 KEYPAD7 HOME)
        (K42 KEYPAD8)
        (K43 KEYPAD9 PGUP)
        (K44 KEYPAD4)
        (K45 KEYPAD5)
        (K46 SUN-LEFT-SPACE)
        (K47 KEYPAD6)
        (K48 RIGHT-COMMAND SUN-RIGHT-SPACE)
        (COPY)
        (FIND)
        (AGAIN)
        (HELP)
        (DEF'N EXPAND)
        (K4E KEYPAD1 END)
        (ALWAYS-ON-1)
        (ALWAYS-ON-2)
        (CENTER)
        (K52 KEYPAD0 INS)
        (BOLD)
        (ITALICS)
        (UNDERLINE)
        (SUPERSCRIPT)
        (SUBSCRIPT)
        (LARGER SMALLER)
        (K59 KEYPAD%| KEYPAD.)
        (K5A KEYPAD\ KEYPAD, SUN-F10)
        (K5B SUN-F11)
        (K5C SUN-F12)
        (DEFAULTS SUN-PROP)
        (K5E SUN-PRTSC)
        (K5F SUN-OPEN)))
)
```

;; \maikokeyactions does not contain keyactions of the form "2,50" because it breaks the loadup process on the sun.

```
(RPAQQ \ORIGKEYACTIONS
       ((0 ("5" "%%" NOLOCKSHIFT))
        (1 ("4" "$" NOLOCKSHIFT))
        (2 ("6" "~" NOLOCKSHIFT))
        (3 ("e" "E" LOCKSHIFT))
        (4 ("7" "&" NOLOCKSHIFT))
        (5 ("d" "D" LOCKSHIFT))
        (6 ("u" "U" LOCKSHIFT))
        (7 ("v" "V" LOCKSHIFT))
        (8 ("0" ")" NOLOCKSHIFT))
        (9 ("k" "K" LOCKSHIFT))
        (10 ("-" "_" NOLOCKSHIFT))
        (11 ("p" "P" LOCKSHIFT))
        (12 ("/" "?" NOLOCKSHIFT))
        (13 ("\" "|" NOLOCKSHIFT))
        (14 ("LF" "`" NOLOCKSHIFT))
        (15 ("Bs" "Bs" NOLOCKSHIFT))
        (16 ("3" "#" NOLOCKSHIFT))
        (17 ("2" "@" NOLOCKSHIFT))
        (18 ("w" "W" LOCKSHIFT))
        (19 ("q" "Q" LOCKSHIFT))
        (20 ("s" "S" LOCKSHIFT))
        (21 ("a" "A" LOCKSHIFT))
        (22 ("9" "(" NOLOCKSHIFT))
        (23 ("i" "I" LOCKSHIFT))
        (24 ("x" "X" LOCKSHIFT))
        (25 ("o" "O" LOCKSHIFT))
        (26 ("l" "L" LOCKSHIFT))
        (27 ("," "<" NOLOCKSHIFT))
        (28 ("'" "%"" NOLOCKSHIFT))
        (29 ("]" "}" NOLOCKSHIFT))
        (30 ("#B" "#B" NOLOCKSHIFT))
        (31 ("#A" "#A" NOLOCKSHIFT))
        (32 ("1" "!" NOLOCKSHIFT))
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (34 ("Tab" "Tab" NOLOCKSHIFT))
        (35 ("f" "F" LOCKSHIFT))
        (36 CTRLDOWN . CTRLUP)
        (37 ("c" "C" LOCKSHIFT))
        (38 ("j" "J" LOCKSHIFT))
        (39 ("b" "B" LOCKSHIFT))
        (40 ("z" "Z" LOCKSHIFT))
        (41 lSHIFTDOWN . lSHIFTUP)
        (42 ("." ">" NOLOCKSHIFT))
        (43 (";" ":" NOLOCKSHIFT))
```

```
        (44 ("CR" "CR" NOLOCKSHIFT))
        (45 ("_" "^" NOLOCKSHIFT))
        (46 ("Del" "Function,^W" NOLOCKSHIFT))
        (47 ("(" "[" NOLOCKSHIFT))
        (48 ("r" "R" LOCKSHIFT))
        (49 ("t" "T" LOCKSHIFT))
        (50 ("g" "G" LOCKSHIFT))
        (51 ("y" "Y" LOCKSHIFT))
        (52 ("h" "H" LOCKSHIFT))
        (53 ("8" "*" NOLOCKSHIFT))
        (54 ("n" "N" LOCKSHIFT))
        (55 ("m" "M" LOCKSHIFT))
        (56 LOCKDOWN . LOCKUP)
        (57 ("Sp" "Sp" NOLOCKSHIFT))
        (58 ("[" "{" NOLOCKSHIFT))
        (59 ("=" "+" NOLOCKSHIFT))
        (60 2SHIFTDOWN . 2SHIFTUP)
        (61 ("#C" "#C" NOLOCKSHIFT))
        (63 (")" "]" NOLOCKSHIFT))
        (77 EVENT . EVENT)
        (78 EVENT . EVENT)
        (79 EVENT . EVENT)
        (102 LOCKDOWN)
        (103 LOCKUP)))

(RPAQQ \DLIONKEYACTIONS
       ((2 ("6" "^" NOLOCKSHIFT))
        (10 ("-" "_" NOLOCKSHIFT))
        (33 ("\" "|" NOLOCKSHIFT))
        (45 ("`" "~" NOLOCKSHIFT))
        (OPEN METADOWN . METAUP)
        (PROP'S CTRLDOWN . CTRLUP)
        (SAME METADOWN . METAUP)
        (FIND ("Function,^C" "Function,#" NOLOCKSHIFT))
        (UNDO ("Function,^D" "Function,$" NOLOCKSHIFT))
        (STOP ("^E" "Bell" NOLOCKSHIFT))
        (MOVE)
        (COPY)
        (AGAIN ("Function,Bs" "Function,(" NOLOCKSHIFT))
        (CENTER ("Function,A" "Function,a" NOLOCKSHIFT))
        (BOLD ("Function,B" "Function,b" NOLOCKSHIFT))
        (ITALICS ("Function,C" "Function,c" NOLOCKSHIFT))
        (UNDERLINE ("Function,F" "Function,f" NOLOCKSHIFT))
        (SUPERSCRIPT ("Function,K" "Function,k" NOLOCKSHIFT))
        (SUBSCRIPT ("Function,L" "Function,l" NOLOCKSHIFT))
        (LARGER ("Function,H" "Function,h" NOLOCKSHIFT))
        (DEFAULTS ("Function,M" "Function,m" NOLOCKSHIFT))
        (93 ("Esc" "Function,64" NOLOCKSHIFT))
        (47 ("Function,^R" "Function,62" NOLOCKSHIFT))
        (31 ("Function,^E" "Function,%%" NOLOCKSHIFT))
        (92 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (80 ("Function,^K" "Function,+" NOLOCKSHIFT))
        (FONT ("Function,J" "Function,j" NOLOCKSHIFT))))

(RPAQQ \DLIONOSDKEYACTIONS ((56 LOCKTOGGLE)))

(RPAQQ \DORADOKEYACTIONS
       ((2 ("6" "~" NOLOCKSHIFT))
        (10 ("-" "-" NOLOCKSHIFT))
        (13 ("\" "|" NOLOCKSHIFT))
        (14 ("LF" "`" NOLOCKSHIFT))
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (45 ("_" "^" NOLOCKSHIFT))))

(RPAQQ \DOVEKEYACTIONS
       ((2 ("6" "^" NOLOCKSHIFT))
        (10 ("-" "_" NOLOCKSHIFT))
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (56 CTRLDOWN . CTRLUP)
        (65 ("Esc" "Esc" NOLOCKSHIFT))
        (71 ("'" "%"" NOLOCKSHIFT))
        (93 ("Function,^T" "Function,64" NOLOCKSHIFT))
        (108 ("`" "~" NOLOCKSHIFT))
        (DBK-META METADOWN . METAUP)
        (DBK-HELP ("Function,^A" "Function,!" NOLOCKSHIFT))
        (SAME METADOWN . METAUP)
        (FIND ("Function,^C" "Function,#" NOLOCKSHIFT))
        (UNDO ("Function,^D" "Function,$" NOLOCKSHIFT))
        (STOP ("^E" "Bell" NOLOCKSHIFT))
        (EDIT ("Function,^E" "Function,%%" NOLOCKSHIFT))
        (MOVE)
        (COPY)
        (AGAIN ("Function,Bs" "Function,(" NOLOCKSHIFT))
        (CENTER ("Function,A" "Function,a" NOLOCKSHIFT))
        (BOLD ("Function,B" "Function,b" NOLOCKSHIFT))
        (ITALICS ("Function,C" "Function,c" NOLOCKSHIFT))
        (CASE ("Function,D" "Function,d" NOLOCKSHIFT))
```

```
        (STRIKEOUT ("Function,E" "Function,e" NOLOCKSHIFT))
        (UNDERLINE ("Function,F" "Function,f" NOLOCKSHIFT))
        (SUPER/SUB ("Function,G" "Function,g" NOLOCKSHIFT))
        (LARGER ("Function,H" "Function,h" NOLOCKSHIFT))
        (MARGINS ("Function,I" "Function,i" NOLOCKSHIFT))
        (LOOKS ("Function,J" "Function,j" NOLOCKSHIFT))
        (CAPSLOCK LOCKTOGGLE)
        (NUMLOCK ("Function,Tab" "-" NOLOCKSHIFT))
        (SCROLLLOCK ("Function,LF" "#4" NOLOCKSHIFT))
        (BREAK ("^B" "#8" NOLOCKSHIFT))
        (DOIT ("Function,^K" "Function,+" NOLOCKSHIFT))
        (KEYPAD7 ("Function,FF" "7" NOLOCKSHIFT))
        (KEYPAD8 ("#-" "8" NOLOCKSHIFT))
        (KEYPAD9 ("Function,CR" "9" NOLOCKSHIFT))
        (KEYPAD4 ("#," "4" NOLOCKSHIFT))
        (KEYPAD5 ("Function,^N" "5" NOLOCKSHIFT))
        (KEYPAD6 ("#." "6" NOLOCKSHIFT))
        (KEYPAD1 ("Function,^O" "1" NOLOCKSHIFT))
        (KEYPAD2 ("#/" "2" NOLOCKSHIFT))
        (KEYPAD3 ("Function,^P" "3" NOLOCKSHIFT))
        (KEYPAD0 ("Function,^Q" "0" NOLOCKSHIFT))
        (KEYPAD%| ("|" "." NOLOCKSHIFT))
        (KEYPAD\ ("\" "," NOLOCKSHIFT))
        (47 ("Function,^R" "Function,62" NOLOCKSHIFT))))

(RPAQQ \DOVEOSDKEYACTIONS ((56 LOCKDOWN . LOCKUP)
                           (36 CTRLDOWN . CTRLUP)
                           (CAPSLOCK ("Function,^E" "Function,%%" NOLOCKSHIFT))))

(RPAQQ \MAIKOKEYACTIONS
       ((61 ("^E" "Bell" NOLOCKSHIFT))
        (91 ("Function,Bs" "Function,(" NOLOCKSHIFT))
        (92 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (30 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (63 ("Function,^D" "Function,$" NOLOCKSHIFT))
        (93 ("Function,^T" "Function,64" NOLOCKSHIFT))
        (62)
        (111 ("Meta,I" "Meta,Bell" NOLOCKSHIFT))
        (89)
        (90 ("Function,^C" "Function,#" NOLOCKSHIFT))
        (73 ("Function,Tab" "Function,Tab" NOLOCKSHIFT))
        (74 ("Function,LF" "Function,LF" NOLOCKSHIFT))
        (75 ("^B" "^B" NOLOCKSHIFT))
        (81 ("Function,FF" "7" NOLOCKSHIFT))
        (82 ("#-" "8" NOLOCKSHIFT))
        (83 ("Function,CR" "9" NOLOCKSHIFT))
        (84 ("#," "4" NOLOCKSHIFT))
        (85 ("Function,^N" "5" NOLOCKSHIFT))
        (87 ("#." "6" NOLOCKSHIFT))
        (94 ("Function,^O" "1" NOLOCKSHIFT))
        (69 ("#/" "2" NOLOCKSHIFT))
        (70 ("Function,^P" "3" NOLOCKSHIFT))
        (98 ("Function,^Q" "0" NOLOCKSHIFT))
        (76 ("Function,^K" "Function,+" NOLOCKSHIFT))
        (72 LOCKTOGGLE)
        (97 ("Function,A" "Function,a" NOLOCKSHIFT))
        (99 ("Function,B" "Function,b" NOLOCKSHIFT))
        (100 ("Function,C" "Function,c" NOLOCKSHIFT))
        (67 ("Function,D" "Function,d" NOLOCKSHIFT))
        (68 ("Function,E" "Function,e" NOLOCKSHIFT))
        (101 ("Function,F" "Function,f" NOLOCKSHIFT))
        (66 ("Function,G" "Function,g" NOLOCKSHIFT))
        (104 ("Function,H" "Function,h" NOLOCKSHIFT))
        (80 ("Function,I" "Function,i" NOLOCKSHIFT))
        (13 ("^W" "^U" NOLOCKSHIFT))
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (65 ("Esc" "Esc" NOLOCKSHIFT))
        (2 ("6" "^" NOLOCKSHIFT))
        (10 ("-" "_" NOLOCKSHIFT))
        (36 CTRLDOWN . CTRLUP)
        (56 LOCKTOGGLE . IGNORE)
        (45 ("`" "~" NOLOCKSHIFT))
        (31 METADOWN . METAUP)
        (14 METADOWN . METAUP)
        (71 ("LF" "LF" NOLOCKSHIFT))
        (47 ("Function,^R" "Function,62" NOLOCKSHIFT))
        (105 ("\" "|" NOLOCKSHIFT))))

(RPAQQ \MAIKOKEYACTIONST4
       ((61 ("^E" "Bell" NOLOCKSHIFT))
        (91 ("Function,Bs" "Function,(" NOLOCKSHIFT))
        (92 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (30 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (109 ("Function,^U" "Function,65" NOLOCKSHIFT))
        (63 ("Function,^D" "Function,$" NOLOCKSHIFT))
        (14 METADOWN . METAUP)
        (93 ("Function,^T" "Function,64" NOLOCKSHIFT))
```

```
        (62)
        (111 ("Meta,o" "Meta,O" NOLOCKSHIFT))
        (89)
        (90 ("Function,^C" "Function,#" NOLOCKSHIFT))
        (73 ("Function,Tab" "Function,Tab" NOLOCKSHIFT))
        (74 ("Function,LF" "Function,LF" NOLOCKSHIFT))
        (75 ("^B" "^B" NOLOCKSHIFT))
        (81 ("Function,FF" "7" NOLOCKSHIFT))
        (82 ("#-" "8" NOLOCKSHIFT))
        (83 ("Function,CR" "9" NOLOCKSHIFT))
        (84 ("#," "4" NOLOCKSHIFT))
        (85 ("Function,^N" "5" NOLOCKSHIFT))
        (87 ("#." "6" NOLOCKSHIFT))
        (94 ("Function,^O" "1" NOLOCKSHIFT))
        (69 ("#/" "2" NOLOCKSHIFT))
        (70 ("Function,^P" "3" NOLOCKSHIFT))
        (98 ("Function,^Q" "0" NOLOCKSHIFT))
        (76 ("Function,^K" "Function,^K" NOLOCKSHIFT))
        (110 ("Function,+" "Function,+" NOLOCKSHIFT))
        (72 LOCKTOGGLE)
        (97 ("Function,A" "Function,a" NOLOCKSHIFT))
        (99 ("Function,B" "Function,b" NOLOCKSHIFT))
        (100 ("Function,C" "Function,c" NOLOCKSHIFT))
        (67 ("Function,D" "Function,d" NOLOCKSHIFT))
        (68 ("Function,E" "Function,e" NOLOCKSHIFT))
        (101 ("Function,F" "Function,f" NOLOCKSHIFT))
        (66 ("Function,G" "Function,g" NOLOCKSHIFT))
        (104 ("Function,H" "Function,h" NOLOCKSHIFT))
        (80 ("Function,I" "Function,i" NOLOCKSHIFT))
        (106 ("Function,K" "Function,k" NOLOCKSHIFT))
        (107 ("Function,L" "Function,l" NOLOCKSHIFT))
        (108 ("Function,M" "Function,m" NOLOCKSHIFT))
        (13 ("^W" "^U" NOLOCKSHIFT))
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (64 IGNORE . IGNORE)
        (65 ("Esc" "Esc" NOLOCKSHIFT))
        (95 IGNORE . IGNORE)
        (96 IGNORE . IGNORE)
        (102 IGNORE . IGNORE)
        (2 ("6" "^" NOLOCKSHIFT))
        (10 ("-" "_" NOLOCKSHIFT))
        (36 CTRLDOWN . CTRLUP)
        (56 LOCKTOGGLE . IGNORE)
        (45 ("`" "~" NOLOCKSHIFT))
        (31 METADOWN . METAUP)
        (71 ("LF" "LF" NOLOCKSHIFT))
        (47 ("Function,^R" "Function,62" NOLOCKSHIFT))
        (86 IGNORE . IGNORE)
        (88 IGNORE . IGNORE)
        (105 ("\" "|" NOLOCKSHIFT))))

(RPAQQ \MAIKO-JLE-KEYACTIONS
       ((2 ("6" "&" NOLOCKSHIFT))
        (4 ("7" "'" NOLOCKSHIFT))
        (8 ("0" "0" NOLOCKSHIFT))
        (10 ("\" "_" NOLOCKSHIFT))
        (13 ("^W" "^U" NOLOCKSHIFT))
        (14 METADOWN . METAUP)
        (15 ("Bs" "Bs" NOLOCKSHIFT))
        (17 ("2" "%"" NOLOCKSHIFT))
        (22 ("9" ")" NOLOCKSHIFT))
        (28 (":" "*" NOLOCKSHIFT))
        (29 ("[" "{" NOLOCKSHIFT))
        (30 ("]" "}" NOLOCKSHIFT))
        (31 METADOWN . METAUP)
        (33 ("Esc" "Esc" NOLOCKSHIFT))
        (36 CTRLDOWN . CTRLUP)
        (43 (";" "+" NOLOCKSHIFT))
        (45 ("^" "~" NOLOCKSHIFT))
        (47 ("Function,^R" "Function,62" NOLOCKSHIFT))
        (53 ("8" "(" NOLOCKSHIFT))
        (56 LOCKTOGGLE . IGNORE)
        (58 ("@" "`" NOLOCKSHIFT))
        (59 ("-" "=" NOLOCKSHIFT))
        (61 ("^E" "Bell" NOLOCKSHIFT))
        (62)
        (63 ("Function,^D" "Function,$" NOLOCKSHIFT))
        (64 ("Function,FF" "7" NOLOCKSHIFT))
        (65 ("Esc" "Esc" NOLOCKSHIFT))
        (66 ("Function,G" "Function,g" NOLOCKSHIFT))
        (67 ("Function,D" "Function,d" NOLOCKSHIFT))
        (69 ("Function,^K" "Function,+" NOLOCKSHIFT))
        (70 ("Function,^P" "3" NOLOCKSHIFT))
        (71 ("LF" "LF" NOLOCKSHIFT))
        (72 ("Function,#~" "Function,#~" NOLOCKSHIFT))
        (73 ("Function,Tab" "Function,Tab" NOLOCKSHIFT))
        (74 ("Function,LF" "Function,LF" NOLOCKSHIFT))
```

```
        (75 ("^B" "^B" NOLOCKSHIFT))
        (80 ("Function,I" "Function,i" NOLOCKSHIFT))
        (81 ("Function,FF" "7" NOLOCKSHIFT))
        (82 ("#-" "8" NOLOCKSHIFT))
        (83 ("Function,CR" "9" NOLOCKSHIFT))
        (84 ("#," "4" NOLOCKSHIFT))
        (85 ("Function,^N" "5" NOLOCKSHIFT))
        (86 ("Function,#}" "Function,#}" NOLOCKSHIFT))
        (87 ("#." "6" NOLOCKSHIFT))
        (88 ("3,^B" "3,^C" NOLOCKSHIFT))
        (90 ("Function,^C" "Function,#" NOLOCKSHIFT))
        (91 ("Function,Bs" "Function,(" NOLOCKSHIFT))
        (92 ("Function,^A" "Function,!" NOLOCKSHIFT))
        (93 ("Function,^T" "Function,64" NOLOCKSHIFT))
        (96 IGNORE . IGNORE)
        (98 ("Function,^Q" "0" NOLOCKSHIFT))
        (99 ("Function,B" "Function,b" NOLOCKSHIFT))
        (101 ("Function,F" "Function,f" NOLOCKSHIFT))
        (102 IGNORE . IGNORE)
        (103 ("Function,#Del" "3,Null" NOLOCKSHIFT))
        (104 ("Function,H" "Function,h" NOLOCKSHIFT))
        (105 ("\" "|" NOLOCKSHIFT))
        (106 ("Function,K" "Function,k" NOLOCKSHIFT))
        (107 ("Function,L" "Function,l" NOLOCKSHIFT))
        (108 ("Function,M" "Function,m" NOLOCKSHIFT))
        (109 ("3,^A" "3,^A" NOLOCKSHIFT))
        (110 ("Function,+" "Function,+" NOLOCKSHIFT))
        (111 ("Meta,o" "Meta,O" NOLOCKSHIFT))))

(RPAQQ \TOSHIBA-KEYACTIONS
       ((2 ("6" "&" NOLOCKSHIFT))
        (4 ("7" "'" NOLOCKSHIFT))
        (17 ("2" "%"" NOLOCKSHIFT))
        (53 ("8" "(" NOLOCKSHIFT))
        (22 ("9" ")" NOLOCKSHIFT))
        (8 ("0" "0" NOLOCKSHIFT))
        (10 ("-" "=" NOLOCKSHIFT))
        (59 ("^" "~" NOLOCKSHIFT))
        (45 ("\" "|" NOLOCKSHIFT))
        (58 ("@" "`" NOLOCKSHIFT))
        (29 ("[" "{" NOLOCKSHIFT))
        (105 ("]" "}" NOLOCKSHIFT))
        (43 (";" "+" NOLOCKSHIFT))
        (28 (":" "*" NOLOCKSHIFT))
        (15 ("^W" "_" NOLOCKSHIFT))
        (13 ("Bs" "Bs" NOLOCKSHIFT))
        (86 METADOWN . METAUP)
        (73 ("Function,^R" "Function,62" NOLOCKSHIFT))
        (88 ("Function,^T" "Function,64" NOLOCKSHIFT))
        (98 IGNORE . IGNORE)
        (75 ("Function,Tab" "Function,Tab" NOLOCKSHIFT))
        (110 ("Function,LF" "Function,LF" NOLOCKSHIFT))
        (74 ("^B" "^B" NOLOCKSHIFT))
        (64 ("Function,FF" "7" NOLOCKSHIFT))
        (65 ("#-" "8" NOLOCKSHIFT))
        (95 ("Function,CR" "9" NOLOCKSHIFT))
        (81 ("#," "4" NOLOCKSHIFT))
        (82 ("Function,^N" "5" NOLOCKSHIFT))
        (83 ("#." "6" NOLOCKSHIFT))
        (84 ("Function,^O" "1" NOLOCKSHIFT))
        (85 ("#/" "2" NOLOCKSHIFT))
        (87 ("Function,^P" "3" NOLOCKSHIFT))
        (94 ("Function,^Q" "0" NOLOCKSHIFT))
        (69 ("Function,^K" "Function,+" NOLOCKSHIFT))
        (70 LOCKTOGGLE)))

(RPAQQ KEYBOARD.APPLICATION-SPECIFIC-KEYACTIONS NIL)

(RPAQ? \KEYBOARD.META 256)

(RPAQ? \MODIFIED.KEYACTIONS )

(DECLARE%: EVAL@COMPILE

(ADDTOVAR GLOBALVARS \RCLKSECOND \LASTUSERACTION \LASTKEYSTATE)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \SYSBUFFER \LONGSYSBUF \INTERRUPTSTATE \MODIFIED.KEYACTIONS \MOUSECHORDTICKS \KEYBOARDEVENTQUEUE
       \KEYBUFFERING \CURRENTKEYACTION \COMMANDKEYACTION \DEFAULTKEYACTION \TIMER.INTERRUPT.PENDING
       \ORIGKEYACTIONS \KEYBOARD.META \MOUSECHORDMILLISECONDS \DORADOKEYACTIONS \DLIONKEYACTIONS
       \DLIONOSDKEYACTIONS \DOVEKEYACTIONS \DOVEOSDKEYACTIONS SHIFTXORLOCKFLG)
)
```

;; Key interpretation

```
(DEFINEQ
```

(KEYACTION

```
  [LAMBDA (KEYNAME ACTIONS TABLE)                          ; Edited 24-Aug-2021 16:54 by rmk:
    (LET ((NUMB (OR (KEYNUMBERP KEYNAME)
                    (\KEYNAMETONUMBER KEYNAME)))
          (TABLE (OR TABLE \CURRENTKEYACTION)))
         (OR (TYPE? KEYACTION TABLE)
             (\ILLEGAL.ARG TABLE))                         ; Make sure he supplied a valid TABLE argument.
         (CONS (\KEYACTION1 (\TRANSINDEX NUMB T)
                    (AND ACTIONS (OR (CAR ACTIONS)
                                     'IGNORE))
                    TABLE)
               (\KEYACTION1 (\TRANSINDEX NUMB NIL)
                    (AND ACTIONS (OR (CDR ACTIONS)
                                     'IGNORE))
                    TABLE])
```

(KEYACTIONTABLE

```
  [LAMBDA (OLD)                                            ; Edited 23-Mar-92 12:44 by jds
```

;; Create a fresh key action table (or copy OLD so it can be modified without danger). Returns a fresh keyaction table.

```
    (COND
       (OLD ;; He supplied an existing table; create a copy of it:
            (OR (type? KEYACTION OLD)
                (\ILLEGAL.ARG OLD))                        ; Make sure the argument IS a key action table.
            (create KEYACTION copying OLD))
       (T ;; Create a completely fresh table, filled in from \ORIGKEYACTIONS, and the machine-specific exceptions:
          (PROG1 (SETQ OLD (create KEYACTION))
                 (for X in (APPEND (COPY \ORIGKEYACTIONS)
                                   (\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS)
                                   KEYBOARD.APPLICATION-SPECIFIC-KEYACTIONS)
                    do (KEYACTION (CAR X)
                                  (CDR X)
                                  OLD)))])
```

(KEYBOARDTYPE

```
  [LAMBDA NIL                                              ; Edited  6-Nov-95 15:35 by
                                                           ; Edited 17-Feb-95 14:36 by rmk:
                                                           ; Edited 16-Jun-92 11:03 by kaplan
```

;; Returns a symbol identifying the currently connected keyboard type. For now, infers it from the machine type, defaults to NIL (= unknown).

```
    (LET ((MT (MACHINETYPE)))
         (SELECTQ MT
             (MAIKO (OR [CADR (SASSOC (L-CASE (UNIX-GETENV "LDEKBDTYPE"))
                                      '(("type3" SUN3)
                                        ("type4" SUN4)
                                        ("type5" SUN5]
                        (MKATOM (U-CASE (UNIX-GETENV "LDEKBDTYPE")))
                        (AND (STREQUAL "dos" (UNIX-GETPARM "ARCH"))
                             'FULL-IBMPC)))
             ((DORADO DANDELION DOVE)
                 MT)
             NIL])
```

(RESETKEYACTION

```
  [LAMBDA (TABLE FROM RESETINTERRUPTS)                     ; Edited 19-Nov-87 16:55 by Snow
```

;; Resets the actions of key transitions in the keyaction table TABLE, copying in the actions from FROM. If RESETINTERRUPTS is true, also
;; copies the interrupt-character settings from FROM.

```
    (DECLARE (GLOBALVARS \DEFAULTKEYACTION))
```

;; do some type checking first.

```
    (OR (type? KEYACTION TABLE)
        (\ILLEGAL.ARG TABLE))
    (OR FROM (SETQ FROM \DEFAULTKEYACTION))
    (OR (type? KEYACTION FROM)
        (\ILLEGAL.ARG TABLE))
```

;; do the resetting.

```
    (\BLT (fetch (KEYACTION FLAGS) of TABLE)
          (fetch (KEYACTION FLAGS) of FROM)
          (LLSH (\#BLOCKDATACELLS (fetch (KEYACTION FLAGS) of TABLE))
                1))
    (\BLT (fetch (KEYACTION CODES) of TABLE)
          (fetch (KEYACTION CODES) of FROM)
          (LLSH (\#BLOCKDATACELLS (fetch (KEYACTION CODES) of TABLE))
                1))
    (\BLT (fetch (KEYACTION SHIFTCODES) of TABLE)
          (fetch (KEYACTION SHIFTCODES) of FROM)
          (LLSH (\#BLOCKDATACELLS (fetch (KEYACTION SHIFTCODES) of TABLE))
                1))
```

```
    [if RESETINTERRUPTS
        then (\BLT (fetch (KEYACTION ARMED) of TABLE)
                   (fetch (KEYACTION ARMED) of FROM)
                   (LLSH (\#BLOCKDATACELLS (fetch (KEYACTION ARMED) of TABLE))
                         1))
             (replace (KEYACTION INTERRUPTLIST) of TABLE with (COPY (fetch (KEYACTION INTERRUPTLIST) of FROM]
    TABLE])
```

(\KEYBOARD.MACHINE-SPECIFIC-KEYACTIONS

```
  [LAMBDA NIL                                                    ; Edited 18-Sep-90 22:36 by jds
    ;; Return a list of machine-specific keyactions appropriate to the machine you're running on.
    ;; Also take account (on Maiko implementations) of whether we're running under X or not -- the CAPS-LOCK key works differently.
    (LET [(CAPS-LOCK-ACTIONS (COND
                               ((EQUAL (UNIX-GETPARM "DISPLAY")
                                       "X")
                                '((56 LOCKTOGGLE . IGNORE)
                                  (72 LOCKDOWN . LOCKUP]
         ;; seems like X defaults to not handling lock these days, so I changed the defaulet handling of LOCK 56 -- LMM 2/13/2021
         ;; If we're running under X windows, CAPS-LOCK-ACTIONS, appended to the normal keyactions, will reset the keyboard appropriately.
         (COND
            ((EQUAL \SUN.TYPE3KEYBOARD (LOGAND 7 (fetch (IFPAGE DEVCONFIG) of \InterfacePage)))
             (APPEND \MAIKOKEYACTIONS CAPS-LOCK-ACTIONS))
            ((EQUAL \SUN.TYPE4KEYBOARD (LOGAND 7 (fetch (IFPAGE DEVCONFIG) of \InterfacePage)))
             (APPEND \MAIKOKEYACTIONST4 CAPS-LOCK-ACTIONS))
            ((EQUAL \SUN.JLEKEYBOARD (LOGAND 7 (fetch (IFPAGE DEVCONFIG) of \InterfacePage)))
             \MAIKO-JLE-KEYACTIONS)
            ((EQUAL \TOSHIBA.JIS (LOGAND 7 (fetch (IFPAGE DEVCONFIG) of \InterfacePage)))
                                                                 ; Toshiba JIS
             (APPEND \MAIKOKEYACTIONST4 \TOSHIBA-KEYACTIONS))
            (T                                                   ; default is type3
               \MAIKOKEYACTIONS])
```

(\KEYACTION1

```
  [LAMBDA (TI ACTION TABLE)                                      ; Edited  9-Jun-2021 20:18 by rmk:
    (PROG1 (SELECTC (TRANSITIONFLAGS TABLE TI)
                (IGNORE.TF 'IGNORE)
                ((LIST LOCKSHIFT.TF NOLOCKSHIFT.TF)
                     [LET (CODE)
                          (LIST (CHECKFORDEADKEY (TRANSITIONCODE TABLE TI)
                                                 TABLE TI NIL)
                                (CHECKFORDEADKEY (TRANSITIONSHIFTCODE TABLE TI)
                                                 TABLE TI T)
                                (TRANSITIONALTGRCODE TABLE TI)
                                (COND
                                   ((EQ LOCKSHIFT.TF (TRANSITIONFLAGS TABLE TI))
                                    'LOCKSHIFT)
                                   (T 'NOLOCKSHIFT])
                (EVENT.TF 'EVENT)
                (CTRLDOWN.TF 'CTRLDOWN)
                (CTRLUP.TF 'CTRLUP)
                (DEADKEY.TF (LIST 'DEADKEY (TRANSITIONDEADLIST TABLE TI)
                                  (TRANSITIONDEADLIST TABLE TI T)))
                (1SHIFTDOWN.TF
                     '1SHIFTDOWN)
                (1SHIFTUP.TF '1SHIFTUP)
                (2SHIFTDOWN.TF
                     '2SHIFTDOWN)
                (2SHIFTUP.TF '2SHIFTUP)
                (LOCKDOWN.TF 'LOCKDOWN)
                (LOCKUP.TF 'LOCKUP)
                (LOCKTOGGLE.TF
                     'LOCKTOGGLE)
                (METADOWN.TF 'METADOWN)
                (METAUP.TF 'METAUP)
                (FONTUP.TF 'FONTUP)
                (FONTDOWN.TF 'FONTDOWN)
                (FONTTOGGLE.TF
                     'FONTTOGGLE)
                (USERMODE1UP.TF
                     'USERMODE1UP)
                (USERMODE1DOWN.TF
                     'USERMODE1DOWN)
                (USERMODE1TOGGLE.TF
                     'USERMODE1TOGGLE)
                (USERMODE2UP.TF
                     'USERMODE2UP)
                (USERMODE2DOWN.TF
                     'USERMODE2DOWN)
                (USERMODE2TOGGLE.TF
                     'USERMODE2TOGGLE)
                (USERMODE3UP.TF
                     'USERMODE3UP)
```

```
            (USERMODE3DOWN.TF
                'USERMODE3DOWN)
            (USERMODE3TOGGLE.TF
                'USERMODE3TOGGLE)
            (ALTGRUP.TF 'ALTGRUP)
            (ALTGRDOWN.TF 'ALTGRDOWN)
            (ALTGRTOGGLE.TF
                'ALTGRTOGGLE)
            (SHOULDNT))
        [SELECTQ ACTION
             ((NIL NOCHANGE))
             (IGNORE (change (TRANSITIONFLAGS TABLE TI)
                             IGNORE.TF))
             (EVENT (change (TRANSITIONFLAGS TABLE TI)
                            EVENT.TF))
             (CTRLUP (change (TRANSITIONFLAGS TABLE TI)
                             CTRLUP.TF))
             (CTRLDOWN (change (TRANSITIONFLAGS TABLE TI)
                               CTRLDOWN.TF))
             (1SHIFTUP (change (TRANSITIONFLAGS TABLE TI)
                               1SHIFTUP.TF))
             (1SHIFTDOWN (change (TRANSITIONFLAGS TABLE TI)
                                 1SHIFTDOWN.TF))
             (2SHIFTUP (change (TRANSITIONFLAGS TABLE TI)
                               2SHIFTUP.TF))
             (2SHIFTDOWN (change (TRANSITIONFLAGS TABLE TI)
                                 2SHIFTDOWN.TF))
             (LOCKUP (change (TRANSITIONFLAGS TABLE TI)
                             LOCKUP.TF))
             (LOCKDOWN (change (TRANSITIONFLAGS TABLE TI)
                               LOCKDOWN.TF))
             (LOCKTOGGLE (change (TRANSITIONFLAGS TABLE TI)
                                 LOCKTOGGLE.TF))
             (METAUP (change (TRANSITIONFLAGS TABLE TI)
                             METAUP.TF))
             (METADOWN (change (TRANSITIONFLAGS TABLE TI)
                               METADOWN.TF))
             (FONTUP (change (TRANSITIONFLAGS TABLE TI)
                             FONTUP.TF))
             (FONTDOWN (change (TRANSITIONFLAGS TABLE TI)
                               FONTDOWN.TF))
             (FONTTOGGLE (change (TRANSITIONFLAGS TABLE TI)
                                 FONTTOGGLE.TF))
             (USERMODE1UP (change (TRANSITIONFLAGS TABLE TI)
                                  USERMODE1UP.TF))
             (USERMODE1DOWN
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE1DOWN.TF))
             (USERMODE1TOGGLE
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE1TOGGLE.TF))
             (USERMODE2UP (change (TRANSITIONFLAGS TABLE TI)
                                  USERMODE2UP.TF))
             (USERMODE2DOWN
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE2DOWN.TF))
             (USERMODE2TOGGLE
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE2TOGGLE.TF))
             (USERMODE3UP (change (TRANSITIONFLAGS TABLE TI)
                                  USERMODE3UP.TF))
             (USERMODE3DOWN
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE3DOWN.TF))
             (USERMODE3TOGGLE
                  (change (TRANSITIONFLAGS TABLE TI)
                          USERMODE3TOGGLE.TF))
             (ALTGRUP (change (TRANSITIONFLAGS TABLE TI)
                              ALTGRUP.TF))
             (ALTGRDOWN (change (TRANSITIONFLAGS TABLE TI)
                                ALTGRDOWN.TF))
             (ALTGRTOGGLE (change (TRANSITIONFLAGS TABLE TI)
                                  ALTGRTOGGLE.TF))
             (PROG (CODE SHIFTCODE ALTGRCODE ACT DEAD SHIFTDEAD)
                   (COND
                      ([AND [OR (AND (AND (LISTP (CAR (LISTP ACTION)))
                                          (EQ (CAAR (LISTP ACTION))
                                              'DEADKEY))
                                     [SETQ DEAD (for PAIR in (CADAR (LISTP ACTION))
                                                   collect
                                                      ;; Make sure we'll take string charcode specs in the deadkey list.
                                                      (CONS (OR (AND (\CHARCODEP (CAR PAIR))
                                                                     (CAR PAIR))
                                                                (CHARCODE.DECODE (CAR PAIR)))
                                                            (OR (AND (\CHARCODEP (CDR PAIR))
                                                                     (CDR PAIR))
```

```
                                                    (CHARCODE.DECODE (CDR PAIR]
                    (SETQ CODE 65535))
                [\CHARCODEP (SETQ CODE (\GETCHARCODE (CAR (LISTP ACTION]
                (SETQ CODE (CHARCODE.DECODE (CAR (LISTP ACTION]
           [OR (AND (AND (LISTP (CADR (LISTP ACTION)))
                         (EQ (CAADR (LISTP ACTION))
                             'DEADKEY))
                    [SETQ SHIFTDEAD (for PAIR in (CADADR (LISTP ACTION))
                                       collect (CONS (OR (AND (\CHARCODEP (CAR PAIR))
                                                              (CAR PAIR))
                                                         (CHARCODE.DECODE (CAR PAIR)))
                                                     (OR (AND (\CHARCODEP (CDR PAIR))
                                                              (CDR PAIR))
                                                         (CHARCODE.DECODE (CDR PAIR]
                    (SETQ SHIFTCODE 65535)
                    (SETQ ACT (CDR ACTION)))
                [\CHARCODEP (SETQ SHIFTCODE (\GETCHARCODE (CAR (SETQ ACT (LISTP (CDR ACTION]
                (SETQ SHIFTCODE (CHARCODE.DECODE (CAR ACT]
           (OR (NULL (SETQ ACT (CDR ACT)))
                (LISTP ACT))
           (SELECTQ (CAR ACT)
                ((LOCKSHIFT T)
                    (change (TRANSITIONFLAGS TABLE TI)
                            LOCKSHIFT.TF))
                ((NOLOCKSHIFT NIL)
                    (change (TRANSITIONFLAGS TABLE TI)
                            NOLOCKSHIFT.TF))
                (AND [OR [\CHARCODEP (SETQ ALTGRCODE (\GETCHARCODE (CAR ACT]
                         (SETQ ALTGRCODE (CHARCODE.DECODE (CAR ACT]
                     (OR (NULL (SETQ ACT (CDR ACT)))
                         (LISTP ACT))
                     (SELECTQ (CAR ACT)
                         ((LOCKSHIFT T)
                             (change (TRANSITIONFLAGS TABLE TI)
                                     LOCKSHIFT.TF))
                         ((NOLOCKSHIFT NIL)
                             (change (TRANSITIONFLAGS TABLE TI)
                                     NOLOCKSHIFT.TF))
                         NIL]
       (change (TRANSITIONCODE TABLE TI)
               CODE)
       (change (TRANSITIONSHIFTCODE TABLE TI)
               SHIFTCODE)
       (\RPLPTR (fetch (KEYACTION DEADKEYLIST) of TABLE)
                (LLSH TI 1)
               DEAD)
       (\RPLPTR (fetch (KEYACTION DEADKEYLIST) of TABLE)
                (LLSH (IPLUS \NKEYS \NKEYS TI)
                      1)
               SHIFTDEAD)
       (AND ALTGRCODE (change (TRANSITIONALTGRCODE TABLE TI)
                              ALTGRCODE)))
      (T (\ILLEGAL.ARG ACTION])])
```

(**KEYDOWNP**

```
[LAMBDA (KEYNAME)                                   (* lmm "18-Apr-85 02:09")
                                                    (* T if the indicated key is instantaneously down.)
  (\NEWKEYDOWNP (\KEYNAMETONUMBER KEYNAME])
```

(**KEYNUMBERP**

```
[LAMBDA (X)                                         ; Edited 16-Jan-96 13:16 by rmk
  (AND (SMALLP X)
       (IGEQ X 0)
       (ILESSP X \NKEYS)
       X])
```

(**\KEYNAMETONUMBER**

```
[LAMBDA (KEYNAME)                                   (* rmk%: " 2-SEP-83 10:29")
  (DECLARE (GLOBALVARS \KEYNAMES))                  (* The fast case is when KEYNAME is lower-case)
  (for X N in \KEYNAMES as I from 0 when (EQMEMB KEYNAME X) do (RETURN I)
     finally (RETURN (OR (AND (NEQ KEYNAME (SETQ N (L-CASE KEYNAME)))
                              (for Y in \KEYNAMES as I from 0 when (EQMEMB N Y) do (RETURN I)))
                         (\ILLEGAL.ARG KEYNAME])
```

(**\KEYNUMBERTONAME**

```
[LAMBDA (KEYNUMBER)
  (DECLARE (GLOBALVARS \KEYNAMES))                  ; Edited 24-Aug-2021 16:03 by rmk:
  (CAR (NTH \KEYNAMES (ADD1 KEYNUMBER])
```

(**MODIFY.KEYACTIONS**

```
[LAMBDA (KeyActions SaveCurrent?)                   ; Edited  2-Feb-89 15:38 by GADENER
  (PROG1 [if SaveCurrent?
```

```
            then (SETQ \MODIFIED.KEYACTIONS (for ITEM in KeyActions collect (CONS (CAR ITEM)
                                                                                  (KEYACTION (CAR ITEM]
        [for action in KeyActions do (for table in '(\CURRENTKEYACTION \COMMANDKEYACTION)
                                        do (KEYACTION (CAR action)
                                                      (CDR action)
                                                      (EVAL table])])
```

(METASHIFT

```
  [LAMBDA FLG                                                          ; Edited 19-Nov-87 16:59 by Snow
    ;; Sets interpretation of swat key to first arg, where T means meta-shift, NIL means original setting.  Returns previous setting
    (PROG ((METASTATUS '(METADOWN . METAUP))
           OLDSETTING)
          [SETQ OLDSETTING (KEYACTION 'BLANK-BOTTOM (AND (IGREATERP FLG 0)
                                                         (COND
                                                            ((EQ (ARG FLG 1)
                                                                 T)
                                                             METASTATUS)
                                                            (T (OR (ARG FLG 1)
                                                                   (CDR (ASSOC 'BLANK-BOTTOM \ORIGKEYACTIONS]
          (RETURN (COND
                     ((EQUAL OLDSETTING METASTATUS)
                      T)
                     (T OLDSETTING])
```

(SHIFTDOWNP

```
  [LAMBDA (SHIFT)                                                      ; Edited  3-Jan-2024 00:09 by mth
                                                                       (* lmm "18-Apr-85 01:07")
                                                                       (* Tells whether a given shift is down)
    (SELECTQ SHIFT
         (LOCK (fetch (KEYBOARDEVENT LOCK) of \LASTKEYSTATE))
         (META (fetch (KEYBOARDEVENT META) of \LASTKEYSTATE))
         (SHIFT (OR (fetch (KEYBOARDEVENT 1SHIFT) of \LASTKEYSTATE)
                    (fetch (KEYBOARDEVENT 2SHIFT) of \LASTKEYSTATE)))
         (1SHIFT (fetch (KEYBOARDEVENT 1SHIFT) of \LASTKEYSTATE))
         (2SHIFT (fetch (KEYBOARDEVENT 2SHIFT) of \LASTKEYSTATE))
         (SHIFTORLOCK (OR (fetch (KEYBOARDEVENT 1SHIFT) of \LASTKEYSTATE)
                          (fetch (KEYBOARDEVENT 2SHIFT) of \LASTKEYSTATE)
                          (fetch (KEYBOARDEVENT LOCK) of \LASTKEYSTATE)))
         (SHIFTXORLOCK (NEQ (NULL (OR (fetch (KEYBOARDEVENT 1SHIFT) of \LASTKEYSTATE)
                                      (fetch (KEYBOARDEVENT 2SHIFT) of \LASTKEYSTATE)))
                            (NULL (fetch (KEYBOARDEVENT LOCK) of \LASTKEYSTATE))))
         (CTRL (fetch (KEYBOARDEVENT CTRL) of \LASTKEYSTATE))
         (FONT (fetch (KEYBOARDEVENT FONT) of \LASTKEYSTATE))
         (USERMODE1 (fetch (KEYBOARDEVENT USERMODE1) of \LASTKEYSTATE))
         (USERMODE2 (fetch (KEYBOARDEVENT USERMODE2) of \LASTKEYSTATE))
         (USERMODE3 (fetch (KEYBOARDEVENT USERMODE3) of \LASTKEYSTATE))
         (\ILLEGAL.ARG SHIFT])

)

;; To support office style 1108 & 1186 keyboards

(DEFINEQ

(SETUP.OFFICE.KEYBOARD
  [LAMBDA NIL                                                          (* jds " 8-Oct-85 16:27")
    (SELECTQ (MACHINETYPE)
         (DANDELION (MODIFY.KEYACTIONS \DLIONOSDKEYACTIONS))
         (DOVE (MODIFY.KEYACTIONS \DOVEOSDKEYACTIONS))
         NIL])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \TEMPCOPYTIMER MACRO ((X)
                                (PROGN (\BLT \MOUSETIMERTEMP (LOCF X)
                                             WORDSPERCELL)
                                       \MOUSETIMERTEMP)))
)

;; Don't copy this optimizer since it expands out to \getbasebit, but do exportit.

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DEFOPTIMIZER KEYDOWNP (KEYNAME)
                       `(\NEWKEYDOWNP (\KEYNAMETONUMBER ,KEYNAME)))
)
```

```
;; END EXPORTED DEFINITIONS
;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS XKEYDOWNP MACRO ((KEYNAME)
                           (KEYDOWNP1 (\KEYNAMETONUMBER KEYNAME))))

(PUTPROPS KEYDOWNP1 MACRO [OPENLAMBDA (KEYNUMBER)
                           (DECLARE (GLOBALVARS \EM.KBDAD0 \EM.KBDAD1 \EM.KBDAD2 \EM.KBDAD3 \EM.UTILIN
                                                \EM.KBDAD4 \EM.KBDAD5))
                           (PROG NIL
                                 (RETURN (EQ 0 (LOGAND (LRSH (LLSH 1 15)
                                                              (PROGN
                                                              (* (IMOD KEYNUMBER BITSPERWORD) -
                                                              GETD cause IMOD and BITSPERWORD not exported to user)
                                                                     (LOGAND KEYNUMBER 15)))
                                                       (\GETBASE (SELECTQ (PROGN
                                                              (* (FOLDLO KEYNUMBER BITSPERWORD) GETD follows
                                                              since FOLDLO and BITSPERWORD not exported to user)
                                                                                (LRSH KEYNUMBER 4))
                                                                        (0 \EM.KBDAD0)
                                                                        (1 \EM.KBDAD1)
                                                                        (2 \EM.KBDAD2)
                                                                        (3 \EM.KBDAD3)
                                                                        (4 \EM.UTILIN)
                                                                        (5 (OR \EM.KBDAD4 (RETURN)))
                                                                        (6 (OR \EM.KBDAD5 (RETURN)))
                                                                        (RETURN))
                                                                 0]))

(PUTPROPS \NEWKEYDOWNP MACRO ((KEYNUMBER)
                              (EQ 0 (\GETBASEBIT \LASTKEYSTATE KEYNUMBER))))
)

;; END EXPORTED DEFINITIONS
;; A raw keyboard device/stream

(DEFINEQ

(\INIT.KEYBOARD.STREAM
  [LAMBDA NIL                                                    ; Edited  4-Sep-87 10:25 by jds
    ;; Initialize the "Keyboard" device:  Set up the FDEV and the prototype keyboard stream in their respective global variables.
    (DECLARE (GLOBALVARS \KEYBOARD.DEVICE \KEYBOARD.STREAM))
    [\DEFINEDEVICE 'KEYBOARD (SETQ \KEYBOARD.DEVICE (create FDEV
                                                            DEVICENAME _ 'KEYBOARD
                                                            CLOSEFILE _ (FUNCTION NILL)
                                                            EVENTFN _ (FUNCTION \KEYBOARDEVENTFN)
                                                            BIN _ (FUNCTION \GETKEY)
                                                            PEEKBIN _ (FUNCTION \PEEKSYSBUF)
                                                            READP _ (FUNCTION \SYSBUFP)
                                                            EOFP _ (FUNCTION NILL)
                                                            GETFILENAME _ (FUNCTION (LAMBDA (X MODE)
                                                                                    (if (EQ MODE 'INPUT)
                                                                                        then \KEYBOARD.STREAM]
    (SETQ \KEYBOARD.STREAM (create STREAM
                                   USERCLOSEABLE _ NIL
                                   USERVISIBLE _ NIL
                                   FULLFILENAME _ '{KEYBOARD}
                                   DEVICE _ \KEYBOARD.DEVICE
                                   ACCESS _ 'INPUT])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\INIT.KEYBOARD.STREAM)
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \KEYBOARD.DEVICE \KEYBOARD.STREAM)
)

;; END EXPORTED DEFINITIONS
;; Hook for a periodic interrupt

(DEFINEQ

(\DOBUFFEREDTRANSITIONS
  [LAMBDA (\INTERRUPTABLE)
```

```
(DECLARE (SPECVARS \INTERRUPTABLE))                              ; Edited  1-Feb-92 11:59 by jds
(SETQ \KEYBUFFERING 'INPROGRESS)
(LET ((PENDINGINTERRUPT))
     (DECLARE (SPECVARS PENDINGINTERRUPT))                       ; Used by \DECODETRANSITION
     [bind R RPTR until (EQ 0 (SETQ R (fetch (RING READ) of \KEYBOARDEVENTQUEUE)))
        do (SETQ RPTR (\ADDBASE \KEYBOARDEVENTQUEUE R))          ; get pointer to this event
                                                                 ; handle simple keyboard words by calling \DOTRANSITIONS for
                                                                 ; each word
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W0) of RPTR)
                    (fetch (KEYBOARDEVENT W0) of \LASTKEYSTATE))
               (\DOTRANSITIONS 0 (fetch (KEYBOARDEVENT W0) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W0) of RPTR))
               (replace (KEYBOARDEVENT W0) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W0) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W1) of RPTR)
                    (fetch (KEYBOARDEVENT W1) of \LASTKEYSTATE))
               (\DOTRANSITIONS 16 (fetch (KEYBOARDEVENT W1) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W1) of RPTR))
               (replace (KEYBOARDEVENT W1) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W1) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W2) of RPTR)
                    (fetch (KEYBOARDEVENT W2) of \LASTKEYSTATE))
               (\DOTRANSITIONS 32 (fetch (KEYBOARDEVENT W2) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W2) of RPTR))
               (replace (KEYBOARDEVENT W2) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W2) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W3) of RPTR)
                    (fetch (KEYBOARDEVENT W3) of \LASTKEYSTATE))
               (\DOTRANSITIONS 48 (fetch (KEYBOARDEVENT W3) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W3) of RPTR))
               (replace (KEYBOARDEVENT W3) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W3) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W4) of RPTR)
                    (fetch (KEYBOARDEVENT W4) of \LASTKEYSTATE))
               (\DOTRANSITIONS 80 (fetch (KEYBOARDEVENT W4) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W4) of RPTR))
               (replace (KEYBOARDEVENT W4) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W4) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT W5) of RPTR)
                    (fetch (KEYBOARDEVENT W5) of \LASTKEYSTATE))
               (\DOTRANSITIONS 96 (fetch (KEYBOARDEVENT W5) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT W5) of RPTR))
               (replace (KEYBOARDEVENT W5) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT W5) of RPTR]
           [COND
              ((NEQ (fetch (KEYBOARDEVENT WU) of RPTR)
                    (fetch (KEYBOARDEVENT WU) of \LASTKEYSTATE))
               (\DOTRANSITIONS 64 (fetch (KEYBOARDEVENT WU) of \LASTKEYSTATE)
                      (fetch (KEYBOARDEVENT WU) of RPTR))
               (replace (KEYBOARDEVENT WU) of \LASTKEYSTATE with (fetch (KEYBOARDEVENT WU) of RPTR]

;;; now remove event from queue

           (COND
              ((EQ [replace (RING READ) of \KEYBOARDEVENTQUEUE with (COND
                                                                        ((IGEQ R \KEYBOARDEVENT.LAST)
                                                                         \KEYBOARDEVENT.FIRST)
                                                                        (T (IPLUS \KEYBOARDEVENT.SIZE R]
                   (fetch (RING WRITE) of \KEYBOARDEVENTQUEUE))
               (replace (RING READ) of \KEYBOARDEVENTQUEUE with 0]
     (PROGN                                                      ; update dummy shift state
            (replace DUMMY1SHIFT of \SHIFTSTATE with (fetch (KEYBOARDEVENT 1SHIFT) of \LASTKEYSTATE))
            (replace DUMMY2SHIFT of \SHIFTSTATE with (fetch (KEYBOARDEVENT 2SHIFT) of \LASTKEYSTATE))
            (replace DUMMYLOCK of \SHIFTSTATE with (fetch (KEYBOARDEVENT LOCK) of \LASTKEYSTATE))
            (replace DUMMYCTRL of \SHIFTSTATE with (fetch (KEYBOARDEVENT CTRL) of \LASTKEYSTATE))
            (replace DUMMYMETA of \SHIFTSTATE with (fetch (KEYBOARDEVENT META) of \LASTKEYSTATE))
            (replace DUMMYFONT of \SHIFTSTATE with (fetch (KEYBOARDEVENT FONT) of \LASTKEYSTATE))
            (replace DUMMYUSERMODE1 of \SHIFTSTATE with (fetch (KEYBOARDEVENT USERMODE1) of \LASTKEYSTATE))
            (replace DUMMYUSERMODE2 of \SHIFTSTATE with (fetch (KEYBOARDEVENT USERMODE2) of \LASTKEYSTATE))
            (replace DUMMYUSERMODE3 of \SHIFTSTATE with (fetch (KEYBOARDEVENT USERMODE3) of \LASTKEYSTATE))
            (replace DUMMYALTGRAPH of \SHIFTSTATE with (fetch (KEYBOARDEVENT ALTGRAPH) of \LASTKEYSTATE))
            (replace DUMMYDEADKEYPENDING of \SHIFTSTATE with (fetch (KEYBOARDEVENT DEADKEYPENDING) of
                                                                                                   \LASTKEYSTATE
                                                                  )))

     ;; Note: there is a window between the test of READ above and the setting of \KEYBUFFERING below where a keyboard transition can be
     ;; ignored until the next transition causes \KEYBUFFERING to be set again

     (COND
        ((NOT (OR PENDINGINTERRUPT \PENDINGINTERRUPT))           ; No interrupt noticed this time or on any previous invocation
         (SETQ \KEYBUFFERING NIL))
        ((NOT (\GETBASEPTR (\STKSCAN '\INTERRUPTABLE)
                     0))                                         ; We're not interruptable, so try again later
         (SETQ \PENDINGINTERRUPT T)
         (SETQ \KEYBUFFERING NIL))
        (T (SETQ \PENDINGINTERRUPT NIL)
           (SETQ \KEYBUFFERING NIL)
           (LET ((\INTERRUPTABLE T))
```

```
                    (INTERRUPTED])
```

(\TIMER.INTERRUPTFRAME

```
  [LAMBDA NIL                                                    (* lmm "22-Apr-85 09:47")
                                                                 (* place holder for periodic interrupts)
    (if NIL
        then (APPLY* \PERIODIC.INTERRUPT)
             (if \PERIODIC.INTERRUPT
                 then (SETUPTIMER (QUOTIENT (TIMES \PERIODIC.INTERRUPT.FREQUENCY \RCLKSECOND)
                                            77)
                                  (LOCF (fetch DLMOUSETIMER of \MISCSTATS))
                                  'TICKS)
                      (SETQ \TIMER.INTERRUPT.PENDING T])
```

(\PERIODIC.INTERRUPTFRAME

```
  [LAMBDA NIL
    (DECLARE (GLOBALVARS \PERIODIC.INTERRUPT))                   (* lmm "16-Jul-85 16:22")
    (LET ((FN \PERIODIC.INTERRUPT))
         (AND FN (SPREADAPPLY* FN])

)

(RPAQ? \KEYBUFFERING )

(RPAQ? \PERIODIC.INTERRUPT )

(RPAQ? \TIMER.INTERRUPT.PENDING )

(RPAQ? \PERIODIC.INTERRUPT.FREQUENCY 77)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

;; cursor and mouse related functions.

(DEFINEQ
```

(\HARDCURSORUP

```
  [LAMBDA (NEWCURSOR INVERTFLG)                                  ; Edited  2-Jan-2000 18:10 by kaplan
                                                                 ; version of \CURSORUP that knows about the possibility of the
                                                                 ; cursor being on the color screen.
    (PROG (IMAGE)
          (SETQ \SOFTCURSORP NIL)
          (SETQ \CURRENTCURSOR NEWCURSOR)
          (SETQ IMAGE (fetch (CURSOR CUIMAGE) of NEWCURSOR))
          [COND
             ((NOT (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of IMAGE)
                       (fetch (BITMAP BITMAPBITSPERPIXEL) of \CURSORDESTINATION)))
              (\CURSORBITSPERPIXEL NEWCURSOR (fetch (BITMAP BITMAPBITSPERPIXEL) of \CURSORDESTINATION))
              (SETQ IMAGE (fetch (CURSOR CUIMAGE) of NEWCURSOR]
          (BITBLT IMAGE 0 0 CursorBitMap 0 (IDIFFERENCE HARDCURSORHEIGHT (fetch (BITMAP BITMAPHEIGHT)
                                                                                 of IMAGE))
                HARDCURSORWIDTH HARDCURSORHEIGHT (COND
                                                    (INVERTFLG 'INVERT)
                                                    (T 'INPUT))
                'REPLACE)
          (SELECTC \MACHINETYPE
              (\DAYBREAK (\DoveDisplay.SetCursorShape CursorBitMap))
              (\MAIKO (SUBRCALL DSPCURSOR (fetch (CURSOR CUHOTSPOTX) of NEWCURSOR)
                                (fetch (CURSOR CUHOTSPOTY) of NEWCURSOR)))
              NIL])
```

(\HARDCURSORPOSITION

```
  [LAMBDA (XPOS YPOS)                                            (* kbr%: "13-Jun-85 21:24")

          (* sets cursor position, adjusts for hotspot and tty region limits. XPOS and YPOS are the screen coordinates of the hotspot
          location.)

    (DECLARE (GLOBALVARS \CURSORHOTSPOTX \CURSORHOTSPOTY \CURSORDESTWIDTH \CURSORDESTHEIGHT))

          (* YPOS is reflected around CURSORYMAX because the screen has
          (0,0) as the upper left corner. *)

    (SETQ YPOS (IDIFFERENCE (SUB1 \CURSORDESTHEIGHT)
                            YPOS))                               (* Clip coordinates *)
    (SETQ XPOS (UNSIGNED (IDIFFERENCE (COND
                                        ((ILESSP XPOS 0)
                                         0)
                                        ((IGEQ XPOS \CURSORDESTWIDTH)
                                         (SUB1 \CURSORDESTWIDTH))
                                        (T XPOS))
                                      \CURSORHOTSPOTX)
```

```
                        BITSPERWORD))
    (SETQ YPOS (UNSIGNED (IDIFFERENCE (COND
                                          ((ILESSP YPOS 0)
                                           0)
                                          ((IGEQ YPOS \CURSORDESTHEIGHT)
                                           (SUB1 \CURSORDESTHEIGHT))
                                          (T YPOS))
                                \CURSORHOTSPOTY)
                        BITSPERWORD))
    [COND
       ((EQ \MACHINETYPE \DANDELION)                          (* Temporary workaround)
        (COND
           ((IGREATERP YPOS 32767)
            (SETQ YPOS 0)))
        (COND
           ((IGREATERP XPOS 32767)
            (SETQ XPOS 0]
    (\SETMOUSEXY XPOS YPOS)
    (PROGN
```

(* change the cursor position too so that GETMOUSESTATE will get the correct values if it is called before the next 60 cycle interrupt.)

```
           (\PUTBASE \EM.CURSORX 0 XPOS)
           (\PUTBASE \EM.CURSORY 0 YPOS))
    NIL])
```

(**HARDCURSORDOWN**

```
  [LAMBDA NIL                                                 (* kbr%: "23-Apr-85 18:26")
    (\CLEARBM (CURSORBITMAP])

)

(DEFINEQ
```

(**CURSOR.INIT**

```
  [LAMBDA NIL                                                 (* kbr%: "23-Jan-86 17:34")
    (PROG (DESTBPL)                                           (* Assorted globals for doing the color cursor.
                                                              *)
          (SETQ \CURSORDESTINATION ScreenBitMap)
          (SETQ \SOFTCURSORUPBM NIL)
          (SETQ \SOFTCURSORDOWNBM NIL)
          (SETQ \CURSORDESTLINE 0)
          (SETQ \CURSORDESTLINEBASE (fetch (BITMAP BITMAPBASE) of ScreenBitMap))
          (SETQ \CURSORDESTWIDTH (fetch (BITMAP BITMAPWIDTH) of ScreenBitMap))
          (SETQ \CURSORDESTHEIGHT (fetch (BITMAP BITMAPHEIGHT) of ScreenBitMap))
          (SETQ \CURSORDESTRASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of ScreenBitMap))
                                                              (* Initialize PILOTBBTs. *)
          (SETQ DESTBPL (UNFOLD \CURSORDESTRASTERWIDTH BITSPERWORD))
                                                              (* These PILOTBBTs are the mixing areas for forming the color
     cursor image. *)                                         (* Does SCREEN to DOWNBM via INPUT, REPLACE.
                                                              *)
          (SETQ \SOFTCURSORBBT1 (create PILOTBBT
                                        PBTSOURCEBPL _ DESTBPL
                                        PBTDISJOINT _ T
                                        PBTSOURCETYPE _ 0
                                        PBTOPERATION _ 0))
          (\LOCKCELL \SOFTCURSORBBT1)                         (* Does DOWNBM to UPBM via INPUT, REPLACE.
                                                              *)
          (SETQ \SOFTCURSORBBT2
           (create PILOTBBT
                   PBTDESTBIT _ 0
                   PBTSOURCEBIT _ 0
                   PBTDISJOINT _ T
                   PBTSOURCETYPE _ 0
                   PBTOPERATION _ 0))
          (\LOCKCELL \SOFTCURSORBBT2)                         (* Does MASK to UPBM via INPUT, ERASE.
                                                              *)
          (SETQ \SOFTCURSORBBT3
           (create PILOTBBT
                   PBTDESTBIT _ 0
                   PBTSOURCEBIT _ 0
                   PBTDISJOINT _ T
                   PBTSOURCETYPE _ 1
                   PBTOPERATION _ 1))
          (\LOCKCELL \SOFTCURSORBBT3)                         (* Does IMAGE to UPBM via INPUT, PAINT.
                                                              *)
          (SETQ \SOFTCURSORBBT4
           (create PILOTBBT
                   PBTDESTBIT _ 0
                   PBTSOURCEBIT _ 0
                   PBTDISJOINT _ T
                   PBTSOURCETYPE _ 0
                   PBTOPERATION _ 2))
          (\LOCKCELL \SOFTCURSORBBT4)                         (* Does UPBM to SCREEN via INPUT, REPLACE.
                                                              *)
```

```
        (SETQ \SOFTCURSORBBT5 (create PILOTBBT
                                      PBTDESTBPL _ DESTBPL
                                      PBTDISJOINT _ T
                                      PBTSOURCETYPE _ 0
                                      PBTOPERATION _ 0))
        (\LOCKCELL \SOFTCURSORBBT5)                              (* Does DOWNBM to SCREEN via INPUT, REPLACE.
                                                                 *)
        (SETQ \SOFTCURSORBBT6 (create PILOTBBT
                                      PBTDESTBPL _ DESTBPL
                                      PBTDISJOINT _ T
                                      PBTSOURCETYPE _ 0
                                      PBTOPERATION _ 0))
        (\LOCKCELL \SOFTCURSORBBT6)                              (* Lock things down. *)
    ])
```

(\CURSORDESTINATION

```
  [LAMBDA (DESTINATION)                                          (* kbr%: " 2-Sep-85 20:13")
                                                                 (* Change DESTINATION of \CURRENTCURSOR, assuming it
    is down. *)
    (PROG (DESTBPL)
          (COND
             ((NOT (EQ DESTINATION \CURSORDESTINATION))
              (UNINTERRUPTABLY
                  [COND
                     ((NOT (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (CURSOR CUIMAGE) of \CURRENTCURSOR))
                               (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATION)))
                      (\CURSORBITSPERPIXEL \CURRENTCURSOR (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATION]
                  (\SETMOUSEXY 0 0)
                  (\PUTBASE \EM.CURSORX 0 0)
                  (\PUTBASE \EM.CURSORY 0 0)
                  (SETQ \CURSORDESTLINE 0)
                  (SETQ.NOREF \CURSORDESTLINEBASE (fetch (BITMAP BITMAPBASE) of DESTINATION))
                  (SETQ \CURSORDESTWIDTH (fetch (BITMAP BITMAPWIDTH) of DESTINATION))
                  (SETQ \CURSORDESTHEIGHT (fetch (BITMAP BITMAPHEIGHT) of DESTINATION))
                  (SETQ \CURSORDESTRASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of DESTINATION))
                  (SETQ DESTBPL (UNFOLD \CURSORDESTRASTERWIDTH BITSPERWORD))
                  (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT1 with DESTBPL)
                  (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT5 with DESTBPL)
                  (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT6 with DESTBPL)
                  (SETQ \CURSORDESTINATION DESTINATION))])
```

(\SOFTCURSORUP

```
  [LAMBDA (NEWCURSOR)                                            (* kbr%: " 2-Sep-85 20:15")
                                                                 (* Put soft NEWCURSOR up, assuming soft cursor is down.
                                                                 *)
    (PROG (IMAGE MASK WIDTH BWIDTH HEIGHT CURSORBITSPERPIXEL CURSORBPL UPBMBASE DOWNBMBASE)
                                                                 (* Get cursor IMAGE & MASK. *)
          (SETQ IMAGE (fetch (CURSOR CUIMAGE) of NEWCURSOR))
          (SETQ MASK (fetch (CURSOR CUMASK) of NEWCURSOR))
          (SETQ WIDTH (fetch (BITMAP BITMAPWIDTH) of IMAGE))
          (SETQ HEIGHT (fetch (BITMAP BITMAPHEIGHT) of IMAGE))
          (SETQ CURSORBITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of IMAGE))
                                                                 (* Create new UPBM & DOWNBM caches if necessary.
                                                                 *)
          (COND
             ((NOT (AND (type? BITMAP \SOFTCURSORUPBM)
                        (EQ (fetch (BITMAP BITMAPWIDTH) of \SOFTCURSORUPBM)
                            WIDTH)
                        (EQ (fetch (BITMAP BITMAPHEIGHT) of \SOFTCURSORUPBM)
                            HEIGHT)
                        (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of \SOFTCURSORUPBM)
                            CURSORBITSPERPIXEL)))
              (SETQ \SOFTCURSORWIDTH WIDTH)
              (SETQ \SOFTCURSORHEIGHT HEIGHT)
              (SETQ \SOFTCURSORUPBM (BITMAPCREATE WIDTH HEIGHT CURSORBITSPERPIXEL))
              (SETQ \SOFTCURSORDOWNBM (BITMAPCREATE WIDTH HEIGHT CURSORBITSPERPIXEL))
              (SETQ UPBMBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORUPBM))
              (\TEMPLOCKPAGES UPBMBASE 1)
              (SETQ DOWNBMBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORDOWNBM))
              (\TEMPLOCKPAGES DOWNBMBASE 1)
              (SETQ CURSORBPL (UNFOLD (fetch (BITMAP BITMAPRASTERWIDTH) of IMAGE)
                                      BITSPERWORD))
              (SETQ BWIDTH (ITIMES (fetch (BITMAP BITMAPWIDTH) of IMAGE)
                                   (fetch (BITMAP BITMAPBITSPERPIXEL) of IMAGE)))
              (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT1 with CURSORBPL)
              (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT2 with UPBMBASE)
              (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT2 with CURSORBPL)
              (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT2 with DOWNBMBASE)
              (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT2 with CURSORBPL)
              (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT2 with BWIDTH)
              (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT2 with HEIGHT)
              (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT3 with UPBMBASE)
              (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT3 with CURSORBPL)
              (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT3 with CURSORBPL)
              (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT3 with BWIDTH)
```

```
            (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT3 with HEIGHT)
            (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT4 with UPBMBASE)
            (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT4 with CURSORBPL)
            (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT4 with CURSORBPL)
            (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT4 with BWIDTH)
            (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT4 with HEIGHT)
            (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT5 with CURSORBPL)
            (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT6 with CURSORBPL)))
                                                              (* Change PILOTBBTs. *)
        (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT3 with (fetch (BITMAP BITMAPBASE) of MASK))
        (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT4 with (fetch (BITMAP BITMAPBASE) of IMAGE))
                                                              (* Put up new \CURRENTCURSOR.
                                                              *)
        (SETQ \CURRENTCURSOR NEWCURSOR)
        (\TEMPLOCKPAGES \CURRENTCURSOR 1)
        (SETQ \SOFTCURSORP T)
        (\SOFTCURSORUPCURRENT])
```

(\SOFTCURSORUPCURRENT

```
  [LAMBDA NIL                                                 (* kbr%: "18-Aug-85 15:09")
                                                              (* Put soft \CURRENTCURSOR up, assuming soft cursor is
                                                              down. *)
    (PROG (DISPINTERRUPT X Y XBASE YBASE WIDTH HEIGHT BITSPERPIXEL MINUSDESTRASTERWIDTH DEST DESTBIT
                 SOURCEOFFSET UPBMSOURCE DOWNBMSOURCE SOURCEBIT)
          (SETQ DISPINTERRUPT (\GETBASE \EM.DISPINTERRUPT 0))
          (\PUTBASE \EM.DISPINTERRUPT 0 0)
          (SETQ \SOFTCURSORUPP T)                             (* Roughly, we want to (BITBLT CURSOR XBASE YBASE
                                                              SCREEN X Y WIDTH HEIGHT) *)
          (SETQ X (SIGNED (\GETBASE \EM.MOUSEX 0)
                          BITSPERWORD))
          (SETQ Y (SIGNED (\GETBASE \EM.MOUSEY 0)
                          BITSPERWORD))
          (SETQ XBASE 0)
          (SETQ YBASE 0)
          (SETQ WIDTH \SOFTCURSORWIDTH)
          (SETQ HEIGHT \SOFTCURSORHEIGHT)                     (* Clip off screen parts of cursor. *)
          [COND
             ((IGREATERP 0 X)                                 (* Some of cursor is to left of screen.
                                                              *)
              (SETQ XBASE (IMINUS X))
              (SETQ WIDTH (IDIFFERENCE WIDTH XBASE))
              (SETQ X 0))
             ((IGREATERP (IPLUS X WIDTH)
                     \CURSORDESTWIDTH)                        (* Some of cursor is to right of screen.
                                                              *)
              (SETQ WIDTH (IDIFFERENCE \CURSORDESTWIDTH X]
          (COND
             ((ILESSP WIDTH 0)
              (GO EXIT)))
          [COND
             ((IGREATERP 0 Y)                                 (* Some of cursor is to above of screen.
                                                              *)
              (SETQ YBASE (IMINUS Y))
              (SETQ HEIGHT (IDIFFERENCE HEIGHT YBASE))
              (SETQ Y 0))
             ((IGREATERP (IPLUS Y HEIGHT)
                     \CURSORDESTHEIGHT)                       (* Some of cursor is to below of screen.
                                                              *)
              (SETQ HEIGHT (IDIFFERENCE \CURSORDESTHEIGHT Y]
          (COND
             ((ILESSP HEIGHT 0)
              (GO EXIT)))                                     (* These loops reset \CURSORDESTLINEBASE while avoiding
    large number arithmetic. *)
          [COND
             [(IGREATERP \CURSORDESTLINE Y)
              (SETQ MINUSDESTRASTERWIDTH (IMINUS \CURSORDESTRASTERWIDTH))
              (until (EQ \CURSORDESTLINE Y) do (SETQ \CURSORDESTLINE (SUB1 \CURSORDESTLINE))
                                               (SETQ.NOREF \CURSORDESTLINEBASE (\ADDBASE \CURSORDESTLINEBASE
                                                                                        MINUSDESTRASTERWIDTH]
             ((ILESSP \CURSORDESTLINE Y)
              (until (EQ \CURSORDESTLINE Y) do (SETQ \CURSORDESTLINE (ADD1 \CURSORDESTLINE))
                                               (SETQ.NOREF \CURSORDESTLINEBASE (\ADDBASE \CURSORDESTLINEBASE
                                                                                        \CURSORDESTRASTERWIDTH]
                                                              (* Reset PILOTBBTs. *)
          (SETQ BITSPERPIXEL (fetch (CURSOR CUBITSPERPIXEL) of \CURRENTCURSOR))
          (SETQ X (ITIMES BITSPERPIXEL X))
          (SETQ XBASE (ITIMES BITSPERPIXEL XBASE))
          (SETQ WIDTH (ITIMES BITSPERPIXEL WIDTH))
          (SETQ DEST \CURSORDESTLINEBASE)
          (SETQ DESTBIT X)
          (SETQ SOURCEOFFSET (ITIMES YBASE (fetch (BITMAP BITMAPRASTERWIDTH) of \SOFTCURSORUPBM)))
          (SETQ UPBMSOURCE (\ADDBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORUPBM)
                                     SOURCEOFFSET))
          (SETQ DOWNBMSOURCE (\ADDBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORDOWNBM)
                                       SOURCEOFFSET))
          (SETQ SOURCEBIT XBASE)
```

```
        (* TBW%: Most of these fields only need to be set if we are clipping this time or the previous time we put the cursor up.
        *)

        (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT1 with DOWNBMSOURCE)
        (replace (PILOTBBT PBTDESTBIT) of \SOFTCURSORBBT1 with SOURCEBIT)
        (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT1 with DEST)
        (replace (PILOTBBT PBTSOURCEBIT) of \SOFTCURSORBBT1 with DESTBIT)
        (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT1 with WIDTH)
        (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT1 with HEIGHT)
        (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT5 with DEST)
        (replace (PILOTBBT PBTDESTBIT) of \SOFTCURSORBBT5 with DESTBIT)
        (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT5 with UPBMSOURCE)
        (replace (PILOTBBT PBTSOURCEBIT) of \SOFTCURSORBBT5 with SOURCEBIT)
        (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT5 with WIDTH)
        (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT5 with HEIGHT)
        (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT6 with DEST)
        (replace (PILOTBBT PBTDESTBIT) of \SOFTCURSORBBT6 with DESTBIT)
        (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT6 with DOWNBMSOURCE)
        (replace (PILOTBBT PBTSOURCEBIT) of \SOFTCURSORBBT6 with SOURCEBIT)
        (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT6 with WIDTH)
        (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT6 with HEIGHT)
                                                                (* Save background behind cursor.
                                                                *)
        (\PILOTBITBLT \SOFTCURSORBBT1 0)                        (* Compute cursor appearance. UPBM =
                                                                (OR IMAGE (AND DOWNBM (NOT MASK))) *)
        (\PILOTBITBLT \SOFTCURSORBBT2 0)
        (\PILOTBITBLT \SOFTCURSORBBT3 0)
        (\PILOTBITBLT \SOFTCURSORBBT4 0)                        (* Put color cursor up. *)
        (\SOFTCURSORPILOTBITBLT \SOFTCURSORBBT5 0)
    EXIT
        (\PUTBASE \EM.DISPINTERRUPT 0 DISPINTERRUPT])
```

(\SOFTCURSORPOSITION

```
  [LAMBDA (X Y)                                                 (* kbr%: "18-Aug-85 14:50")
                                                                (* Move soft cursor. *)
    (PROG (DISPINTERRUPT)
          (SETQ DISPINTERRUPT (\GETBASE \EM.DISPINTERRUPT 0))
          (\PUTBASE \EM.DISPINTERRUPT 0 0)
          [COND
             ((OR (NOT (EQ (\GETBASE \EM.CURSORX 0)
                           X))
                  (NOT (EQ (\GETBASE \EM.CURSORY 0)
                           Y)))
              (COND
                 (\SOFTCURSORUPP (\SOFTCURSORDOWN)
                         (\SOFTCURSORUPCURRENT]
          (\PUTBASE \EM.DISPINTERRUPT 0 DISPINTERRUPT])
```

(\SOFTCURSORDOWN

```
  [LAMBDA NIL                                                   (* kbr%: " 6-Jul-85 00:09")
                                                                (* Take COLOR cursor down. *)
    (PROG (DISPINTERRUPT)                                       (* \SOFTCURSORUPP must be set to NIL before BITBLTing.
                                                                *)
          (SETQ DISPINTERRUPT (\GETBASE \EM.DISPINTERRUPT 0))
          (\PUTBASE \EM.DISPINTERRUPT 0 0)
          (SETQ \SOFTCURSORUPP NIL)
          (\SOFTCURSORPILOTBITBLT \SOFTCURSORBBT6 0)
          (\PUTBASE \EM.DISPINTERRUPT 0 DISPINTERRUPT])
```

(CURSORPROP

```
  [LAMBDA X                                                     (* kbr%: "11-Jan-86 20:03")
    (COND
       ((IGREATERP X 2)
        (PUTCURSORPROP (ARG X 1)
               (ARG X 2)
               (ARG X 3)))
       ((EQ X 2)
        (GETCURSORPROP (ARG X 1)
               (ARG X 2)))
       (T (\ILLEGAL.ARG NIL])
```

(GETCURSORPROP

```
  [LAMBDA (CURSOR PROP)                                         (* kbr%: "26-Apr-85 11:18")
    (LISTGET (fetch (CURSOR CUDATA) of CURSOR)
           PROP])
```

(PUTCURSORPROP

```
  [LAMBDA (CURSOR PROP VALUE)                                   (* kbr%: "26-Apr-85 11:18")
    (PROG (OLDDATA OLDVALUE)
          (SETQ OLDDATA (fetch (CURSOR CUDATA) of CURSOR))
          [COND
```

```
            [OLDDATA (SETQ OLDVALUE (LISTGET OLDDATA PROP))
                   (COND
                      (VALUE (LISTPUT OLDDATA PROP VALUE))
                      (OLDVALUE (COND
                                  [(EQ (CAR OLDDATA)
                                       PROP)
                                   (replace (CURSOR CUDATA) of CURSOR with (CDDR (fetch (CURSOR CUDATA)
                                                                                        of CURSOR]
                                  (T (FOR TAIL ON (CDR OLDDATA) BY (CDDR TAIL)
                                        WHEN (EQ (CADR TAIL)
                                                 PROP)
                                        DO (FRPLACD TAIL (CDDDR TAIL))
                                           (RETURN]
                      (VALUE (replace (CURSOR CUDATA) of CURSOR with (LIST PROP VALUE]
          (RETURN OLDVALUE])
```

(\CURSORBITSPERPIXEL

```
  [LAMBDA (CURSOR NEWBITSPERPIXEL)                                  (* kbr%: "12-May-85 17:15")
                                                                    (* Swap in NEWBITSPERPIXEL IMAGE and MASK, creating
                                                                    them if necessary. *)
    (PROG (OLDBITSPERPIXEL OLDIMAGE OLDMASK WHITE BLACK NEWIMAGE NEWMASK)
          (SETQ OLDBITSPERPIXEL (fetch (CURSOR CUBITSPERPIXEL) of CURSOR))
          (COND
             ((EQ OLDBITSPERPIXEL NEWBITSPERPIXEL)
              (RETURN)))                                            (* Save OLDIMAGE and OLDMASK.
                                                                    *)
          (SETQ OLDIMAGE (fetch (CURSOR CUIMAGE) of CURSOR))
          (SETQ OLDMASK (fetch (CURSOR CUMASK) of CURSOR))
          (CURSORPROP CURSOR (\CURSORIMAGEPROPNAME OLDBITSPERPIXEL)
                 OLDIMAGE)
          (CURSORPROP CURSOR (\CURSORMASKPROPNAME OLDBITSPERPIXEL)
                 OLDMASK)                                           (* Unsave NEWIMAGE and NEWMASK if possible, otherwise
                                                                    create them. *)
          [COND
             [(SETQ NEWIMAGE (CURSORPROP CURSOR (\CURSORIMAGEPROPNAME NEWBITSPERPIXEL)))
                                                                    (* Use cached NEWIMAGE & NEWMASK.
                                                                    *)
              (SETQ NEWMASK (CURSORPROP CURSOR (\CURSORMASKPROPNAME NEWBITSPERPIXEL]
             (T                                                     (* Create NEWIMAGE & NEWMASK.
                                                                    *)
                (SETQ WHITE (MASK.1'S 0 NEWBITSPERPIXEL))
                (SETQ BLACK 0)
                (SETQ NEWIMAGE (COLORIZEBITMAP (CURSORPROP CURSOR 'IMAGE1)
                                       BLACK WHITE NEWBITSPERPIXEL))
                (SETQ NEWMASK (COLORIZEBITMAP (CURSORPROP CURSOR 'MASK1)
                                      BLACK WHITE NEWBITSPERPIXEL]
          (replace (CURSOR CUIMAGE) of CURSOR with NEWIMAGE)
          (replace (CURSOR CUMASK) of CURSOR with NEWMASK])
```

(\CURSORIMAGEPROPNAME

```
  [LAMBDA (BITSPERPIXEL)                                            (* kbr%: "26-Apr-85 11:18")
    (SELECTQ BITSPERPIXEL
         (1 'IMAGE1)
         (4 'IMAGE4)
         (8 'IMAGE8)
         (SHOULDNT])
```

(\CURSORMASKPROPNAME

```
  [LAMBDA (BITSPERPIXEL)                                            (* kbr%: "26-Apr-85 11:18")
    (SELECTQ BITSPERPIXEL
         (1 'MASK1)
         (4 'MASK4)
         (8 'MASK8)
         (SHOULDNT])

)

(DEFINEQ
```

(CURSORCREATE

```
  [LAMBDA (IMAGE MASK HOTSPOTX HOTSPOTY DATA)                       ; Edited 10-Jul-92 16:32 by cat
                                                                    ; Edited 31-Jul-87 10:01 by jds
    ;; creates a cursor from a bitmap.  HOTSPOTX and HOTSPOTY specify the hotspot.

    ;; INVARIANTS:  the hot spot X and Y must be in the range 0..(width - 1) and 0..(height - 1), respectively.

    (PROG (CURSOR)
          (COND
             ((OR (FIXP MASK)
                  (POSITIONP MASK))
              ;; If Mask is a fixp then we presume this is the old arg list (bitmap x y).  the cursor filepkgtype has been changed to write the new arg
              ;; list.  The other is provided for (dubious) compatibility
              (SETQ HOTSPOTY HOTSPOTX)
              (SETQ HOTSPOTX MASK)
```

```
            (SETQ MASK NIL)))
```

;; Make sure that the image and mask bitmaps are no larger than the hardware cursor, i.e. 16x16 bits [AR 8916 7/31/87]:

```
      (COND
         ((OR (IGREATERP (BITMAPWIDTH IMAGE)
                         16)
              (IGREATERP (BITMAPHEIGHT IMAGE)
                         16))                                  ; IMAGE is too big.
          (\ILLEGAL.ARG IMAGE))
         ((NOT MASK)                                           ; No mask, so it's OK

          )
         ((OR (IGREATERP (BITMAPWIDTH MASK)
                         16)
              (IGREATERP (BITMAPHEIGHT MASK)
                         16))                                  ; MASK is too big.
          (\ILLEGAL.ARG MASK)))
      [COND
         ((POSITIONP HOTSPOTX)
```

;; The hot spot can be specified as a position in one arg, rather than X and Y in two:

```
          (SETQ HOTSPOTY (fetch (POSITION YCOORD) of HOTSPOTX))
          (SETQ HOTSPOTX (fetch (POSITION XCOORD) of HOTSPOTX]
      (SETQ CURSOR (create CURSOR
                           CUIMAGE _ IMAGE
                           CUMASK _ (OR MASK IMAGE)
                           CUHOTSPOTX _ (IMAX 0 (IMIN (SUB1 (BITMAPWIDTH IMAGE))
                                                      (OR (FIXP HOTSPOTX)
                                                          0)))
                           CUHOTSPOTY _ [IMAX 0 (IMIN (SUB1 (BITMAPHEIGHT IMAGE))
                                                      (OR (FIXP HOTSPOTY)
                                                          (SUB1 (BITMAPHEIGHT IMAGE]
                           CUDATA _ DATA))
      (RETURN CURSOR])
```

(CURSOR

```
  [LAMBDA (NEWCURSOR INVERTFLG)                                ; Edited 24-Mar-87 18:30 by jds
```

;; Installs NEWCURSOR as the cursor and returns the old cursor state. If INVERTFLG is non-NIL, the cursor image is inverted during installation.
;; If NEWCURSOR is NIL, just returns the current cursor state.

```
    (DECLARE (GLOBALVARS DEFAULTCURSOR \SOFTCURSORP))
    (PROG (OLDCURSOR)
          (SETQ OLDCURSOR \CURRENTCURSOR)
          (COND
             ((EQ NEWCURSOR T)                                 ; If NEWCURSOR is T, use the system default cursor.
              (SETQ NEWCURSOR DEFAULTCURSOR)))
          (COND
             [(\CURSOR-VALID-P NEWCURSOR \SOFTCURSORP)         ; Only install the cursor if it's a real, valid one.
              (\CURSORDOWN)
              (\CURSORUP NEWCURSOR INVERTFLG)                  ; set after adjustment to avoid confusion about hotspot during
                                                               ; adjustment.
              (SETQ \CURSORHOTSPOTX (fetch (CURSOR CUHOTSPOTX) of NEWCURSOR))
              (SETQ \CURSORHOTSPOTY (IDIFFERENCE (SUB1 (fetch (BITMAP BITMAPHEIGHT) of (fetch (CURSOR CUIMAGE)
                                                                                           of NEWCURSOR)))
                                                 (fetch (CURSOR CUHOTSPOTY) of NEWCURSOR]
             (NEWCURSOR                                        ; NEWCURSOR = NIL means just return the old one, so only
                                                               ; error if one got specified that wasn't valid.
                    (\ILLEGAL.ARG NEWCURSOR)))
          (RETURN OLDCURSOR])
```

(\CURSOR-VALID-P

```
  [LAMBDA (CURSOR SOFT?)                                       ; Edited 25-Mar-87 09:41 by jds
```

;; It returns T if CURSOR is a valid cursor. Validity depends on whether it's meant to be displayed using the cursor hardware or the cursor
;; software.

;; This is really wed to the D-machine display architecture.

```
    (AND (CURSORP CURSOR)
         (COND
            (SOFT? T)
            (T (LET ((IMAGE (fetch (CURSOR CUIMAGE) of CURSOR))
                     (HOTSPOT-X (fetch (CURSOR CUHOTSPOTX) of CURSOR))
                     (HOTSPOT-Y (fetch (CURSOR CUHOTSPOTY) of CURSOR)))
```

;; The bitmap must be <= 16x16, and the hot spot must be within the cursor if we're using hardware cursor.

```
                    (AND (>= 16 (BITMAPWIDTH IMAGE))
                         (>= 16 (BITMAPHEIGHT IMAGE))
                         (<= 0 HOTSPOT-X)
                         (< HOTSPOT-X 16)
                         (<= 0 HOTSPOT-Y)
                         (< HOTSPOT-Y 16])
```

(\CURSORUP

```
  [LAMBDA (NEWCURSOR INVERTFLG)                                (* kbr%: "18-Aug-85 14:38")
    (UNINTERRUPTABLY
```

```
        (\CURSORBITSPERPIXEL NEWCURSOR (fetch (BITMAP BITMAPBITSPERPIXEL) of \CURSORDESTINATION))
        (COND
           ((AND (EQ (fetch (CURSOR CUIMAGE) of NEWCURSOR)
                     (fetch (CURSOR CUMASK) of NEWCURSOR))
                 (ILEQ (fetch (BITMAP BITMAPWIDTH) of (fetch (CURSOR CUIMAGE) of NEWCURSOR))
                       HARDCURSORWIDTH)
                 (ILEQ (fetch (BITMAP BITMAPHEIGHT) of (fetch (CURSOR CUIMAGE) of NEWCURSOR))
                       HARDCURSORHEIGHT)
                 (EQ \CURSORDESTINATION ScreenBitMap))
            (\HARDCURSORUP NEWCURSOR INVERTFLG))
           (T (\SOFTCURSORUP NEWCURSOR)))
        (ADJUSTCURSORPOSITION (IDIFFERENCE \CURSORHOTSPOTX (fetch (CURSOR CUHOTSPOTX) of NEWCURSOR))
              (IDIFFERENCE (IDIFFERENCE (SUB1 (fetch (BITMAP BITMAPHEIGHT) of (fetch (CURSOR CUIMAGE)
                                                                                     of NEWCURSOR)))
                                        (fetch (CURSOR CUHOTSPOTY) of NEWCURSOR))
                           \CURSORHOTSPOTY)))])
```

(\CURSORPOSITION

```
  [LAMBDA (XPOS YPOS)                                   ; Edited 19-Mar-98 14:41 by jds
```

(* sets cursor position, adjusts for hotspot and tty region limits. XPOS and YPOS are the screen coordinates of the hotspot location.)

```
    (DECLARE (GLOBALVARS \CURSORHOTSPOTX \CURSORHOTSPOTY \CURSORDESTWIDTH \CURSORDESTHEIGHT))
```

(* YPOS is reflected around CURSORYMAX because the screen has (0,0) as the upper left corner. *)

```
    (SETQ YPOS (IDIFFERENCE (SUB1 \CURSORDESTHEIGHT)
                            YPOS))                      (* Clip coordinates *)
    (SETQ XPOS (UNSIGNED (IDIFFERENCE (COND
                                      (NIL ;; Removed 2000/1/3 JDS so mousr cursors work.
                                           (ILESSP XPOS 0)
                                           0)
                                      ((IGEQ XPOS \CURSORDESTWIDTH)
                                       (SUB1 \CURSORDESTWIDTH))
                                      (T XPOS))
                                   \CURSORHOTSPOTX)
                            BITSPERWORD))
    (SETQ YPOS (UNSIGNED (IDIFFERENCE (COND
                                      (NIL (ILESSP YPOS 0)
                                           0)
                                      ((IGEQ YPOS \CURSORDESTHEIGHT)
                                       (SUB1 \CURSORDESTHEIGHT))
                                      (T YPOS))
                                   \CURSORHOTSPOTY)
                            BITSPERWORD))
    [COND
       ((EQ \MACHINETYPE \DANDELION)                    (* Temporary workaround)
        (COND
           ((IGREATERP YPOS 32767)
            (SETQ YPOS 0)))
        (COND
           ((IGREATERP XPOS 32767)
            (SETQ XPOS 0]
    (\SETMOUSEXY XPOS YPOS)
    (COND
       (\SOFTCURSORP (\SOFTCURSORPOSITION XPOS YPOS)))
    [PROGN
```

(* change the cursor position too so that GETMOUSESTATE will get the correct values if it is called before the next 60 cycle interrupt.)

```
        (\PUTBASE \EM.CURSORX 0 XPOS)
        (\PUTBASE \EM.CURSORY 0 YPOS)
        (COND
           ((EQ \MACHINETYPE \DAYBREAK)                 (* Need to notify DAYBREAK IOP to move cursor.
                                                        *)
            (\DoveDisplay.SetCursorPosition XPOS YPOS]
    NIL])
```

(\CURSORDOWN

```
  [LAMBDA NIL                                           (* kbr%: "12-Jun-85 17:21")
    (UNINTERRUPTABLY
        (COND
           (\SOFTCURSORP (\SOFTCURSORDOWN))
           (T (\HARDCURSORDOWN))))])
```

(ADJUSTCURSORPOSITION

```
  [LAMBDA (DELTAX DELTAY)                               (* kbr%: " 6-Jan-86 11:55")
    (COND
       [(POSITIONP DELTAX)
        (\CURSORPOSITION (IPLUS (fetch (POSITION XCOORD) of DELTAX)
```

```
                                (\XMOUSECOORD))
                (IPLUS (fetch (POSITION YCOORD) of DELTAX)
                       (\YMOUSECOORD]
        (T (\CURSORPOSITION (IPLUS (OR DELTAX 0)
                                   (\XMOUSECOORD))
                   (IPLUS (OR DELTAY 0)
                          (\YMOUSECOORD])
```

(CURSORPOSITION

```
[LAMBDA (NEWPOSITION DISPLAYSTREAM OLDPOSITION)                    (* kbr%: "13-Feb-86 15:53")
  (PROG (DD)
        (SETQ DD (\GETDISPLAYDATA DISPLAYSTREAM))
        (OR (type? POSITION OLDPOSITION)
            (SETQ OLDPOSITION (create POSITION)))
        (freplace (POSITION XCOORD) of OLDPOSITION with (\DSPUNTRANSFORMX (\XMOUSECOORD)
                                                                          DD))
        (freplace (POSITION YCOORD) of OLDPOSITION with (\DSPUNTRANSFORMY (\YMOUSECOORD)
                                                                          DD))
        (COND
           ((type? POSITION NEWPOSITION)
            (\CURSORPOSITION (\DSPTRANSFORMX (fetch (POSITION XCOORD) of NEWPOSITION)
                                             DD)
                             (\DSPTRANSFORMY (fetch (POSITION YCOORD) of NEWPOSITION)
                                             DD)))
           ((type? SCREENPOSITION NEWPOSITION)
            (CURSORSCREEN (fetch (SCREENPOSITION SCREEN) of NEWPOSITION)
                   (fetch (SCREENPOSITION XCOORD) of NEWPOSITION)
                   (fetch (SCREENPOSITION YCOORD) of NEWPOSITION)))
           (NEWPOSITION (\ILLEGAL.ARG NEWPOSITION)))
        (RETURN OLDPOSITION])
```

(CURSORSCREEN

```
[LAMBDA (SCREEN XCOORD YCOORD)                                     (* gbn%: "25-Jan-86 16:53")
```

(* * sets up SCREEN to be the current screen, XCOORD %, YCOORD is initial pos of cursor on SCREEN)

```
  (COND
     ((NULL XCOORD)
      (SETQ XCOORD 0)))
  (COND
     ((NULL YCOORD)
      (SETQ YCOORD 0)))
  (PROG (DESTINATION)
        (SETQ DESTINATION (fetch (SCREEN SCDESTINATION) of SCREEN))
        (\CURSORDOWN)
        (SETQ \CURSORSCREEN SCREEN)
        (\CURSORDESTINATION DESTINATION)
        (\CURSORUP \CURRENTCURSOR)
        (\CURSORPOSITION XCOORD YCOORD])
```

(CURSOREXIT

```
[LAMBDA NIL                                                        (* gbn%: "25-Jan-86 16:52")
```

(* * called when cursor moves off the screen edge)

```
  (DECLARE (GLOBALVARS LASTSCREEN LASTMOUSEX LASTMOUSEY))
  (PROG (SCREEN XCOORD YCOORD SCREEN2 XCOORD2 YCOORD2)
        (SETQ SCREEN LASTSCREEN)
        (SETQ XCOORD LASTMOUSEX)
        (SETQ YCOORD LASTMOUSEY)
        (SETQ SCREEN2 (COND
                         ((EQ SCREEN \MAINSCREEN)
                          \COLORSCREEN)
                         (T \MAINSCREEN)))                         (* generalize for more than two screens
                                                                   (or alternate physical arrangement of screens.))
        (COND
           ((EQ XCOORD 0)
            (SETQ XCOORD2 (IDIFFERENCE (fetch (SCREEN SCWIDTH) of SCREEN2)
                                       2)))
           ((EQ XCOORD (SUB1 (fetch (SCREEN SCWIDTH) of SCREEN)))
            (SETQ XCOORD2 1))
           (T (RETURN)))
        [SETQ YCOORD2 (IQUOTIENT (ITIMES YCOORD (SUB1 (fetch (SCREEN SCHEIGHT) of SCREEN2)))
                                 (SUB1 (fetch (SCREEN SCHEIGHT) of SCREEN]
        (CURSORSCREEN SCREEN2 XCOORD2 YCOORD2])
```

(FLIPCURSOR

```
[LAMBDA NIL                                                        ; Edited 24-Apr-88 00:04 by MASINTER
  (PROG (ADDR)
        (COND
           ((NOT \SOFTCURSORP)
            (SETQ ADDR \EM.CURSORBITMAP)
            (FRPTQ HARDCURSORHEIGHT [\PUTBASE ADDR 0 (LOGXOR (\GETBASE ADDR 0)
```

```
                                                                    (CONSTANT (SUB1 (EXPT 2 HARDCURSORWIDTH]
                    (SETQ ADDR (\ADDBASE ADDR 1)))
               (SELECTC \MACHINETYPE
                    (\DAYBREAK (\DoveDisplay.SetCursorShape))
                    (\MAIKO (AND \CURRENTCURSOR (SUBRCALL DSPCURSOR (fetch (CURSOR CUHOTSPOTX) of \CURRENTCURSOR)
                                                          (fetch (CURSOR CUHOTSPOTY) of \CURRENTCURSOR))))
                    NIL])
```

(FLIPCURSORBAR

```
  [LAMBDA (N)                                                       ; Edited 19-Mar-98 14:23 by jds
```

;;; Inverts the Nth line of the cursor, N = 0 being the top

```
    (COND
       ((NOT \SOFTCURSORP)
        (\PUTBASE \EM.CURSORBITMAP N (LOGXOR (\GETBASE \EM.CURSORBITMAP N)
                                             MAX.SMALLP))
        (SELECTC \MACHINETYPE
             (\DAYBREAK                                             ; Notify IOP
                  (\DoveDisplay.SetCursorShape))
             (\MAIKO (AND \CURRENTCURSOR (SUBRCALL DSPCURSOR (fetch (CURSOR CUHOTSPOTX) of \CURRENTCURSOR)
                                                   (fetch (CURSOR CUHOTSPOTY) of \CURRENTCURSOR))))
             NIL])
```

(LASTMOUSEX

```
  [LAMBDA (DS)                                                      (* rmk%: "30-AUG-83 13:07")
                                                                    (* returns the mouse x position in the coordinates of the
                                                                    DisplayStream DS)
    (\DSPUNTRANSFORMX LASTMOUSEX (\GETDISPLAYDATA DS])
```

(LASTMOUSEY

```
  [LAMBDA (DS)                                                      (* rmk%: "30-AUG-83 13:07")
                                                                    (* returns the mouse y position in the coordinates of the
                                                                    DisplayStream DS)
    (\DSPUNTRANSFORMY LASTMOUSEY (\GETDISPLAYDATA DS])
```

(CREATEPOSITION

```
  [LAMBDA (XCOORD YCOORD)                                           (* rmk%: " 6-Aug-84 13:43")
    (create POSITION
           XCOORD _ (OR XCOORD 0)
           YCOORD _ (OR YCOORD 0])
```

(POSITIONP

```
  [LAMBDA (X)                                                       (* rrb "25-AUG-82 11:04")
```

(* is X a position? For now just a cons check but should be made a datatype.)

```
    (AND (LISTP X)
         (NUMBERP (CAR X))
         (NUMBERP (CDR X))
         X])
```

(CURSORHOTSPOT

```
  [LAMBDA (NEWPOSITION)                                             (* gbn%: "26-Jan-86 15:36")
```

(* returns the current cursor hot spot and sets the hot spot to NEWPOSITON if one is given.)

```
    (PROG1 (create POSITION
                   XCOORD _ \CURSORHOTSPOTX
                   YCOORD _ \CURSORHOTSPOTY)
        [COND
           ((POSITIONP NEWPOSITION)
            (SETQ \CURSORHOTSPOTX (fetch (POSITION YCOORD) of NEWPOSITION))
            (SETQ \CURSORHOTSPOTY (fetch (POSITION YCOORD) of NEWPOSITION])])

)

(PUTPROPS CURSORPROP ARGNAMES (NIL (CURSOR PROP {NEWVALUE}) . U))

(RPAQ? \CURSORHOTSPOTX 0)

(RPAQ? \CURSORHOTSPOTY 0)

(RPAQ? \CURRENTCURSOR NIL)

(RPAQ? \SOFTCURSORWIDTH NIL)

(RPAQ? \SOFTCURSORHEIGHT NIL)

(RPAQ? \SOFTCURSORP NIL)
```

```
(RPAQ? \SOFTCURSORUPP NIL)

(RPAQ? \SOFTCURSORUPBM NIL)

(RPAQ? \SOFTCURSORDOWNBM NIL)

(RPAQ? \SOFTCURSORBBT1 NIL)

(RPAQ? \SOFTCURSORBBT2 NIL)

(RPAQ? \SOFTCURSORBBT3 NIL)

(RPAQ? \SOFTCURSORBBT4 NIL)

(RPAQ? \SOFTCURSORBBT5 NIL)

(RPAQ? \SOFTCURSORBBT6 NIL)

(RPAQ? \CURSORSCREEN NIL)

(RPAQ? \CURSORDESTINATION NIL)

(RPAQ? \CURSORDESTHEIGHT 808)

(RPAQ? \CURSORDESTWIDTH 1024)

(RPAQ? \CURSORDESTRASTERWIDTH 64)

(RPAQ? \CURSORDESTLINE 0)

(RPAQ? \CURSORDESTLINEBASE NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \CURSORHOTSPOTX \CURSORHOTSPOTY \CURRENTCURSOR \SOFTCURSORWIDTH \SOFTCURSORHEIGHT \SOFTCURSORP
       \SOFTCURSORUPP \SOFTCURSORUPBM \SOFTCURSORDOWNBM \SOFTCURSORBBT1 \SOFTCURSORBBT2 \SOFTCURSORBBT3
       \SOFTCURSORBBT4 \SOFTCURSORBBT5 \SOFTCURSORBBT6 \CURSORDESTINATION \CURSORDESTHEIGHT \CURSORDESTWIDTH
       \CURSORDESTRASTERWIDTH \CURSORDESTLINE \CURSORDESTLINEBASE)
)

(DEFINEQ
```

(GETMOUSESTATE

```
  [LAMBDA NIL                                                    (* kbr%: " 6-Jul-85 14:16")
                                                                 (* Reads the current state of the mouse and keyboard)
    (SETQ LASTMOUSEX (\XMOUSECOORD))
    (SETQ LASTMOUSEY (\YMOUSECOORD))
    (SETQ LASTMOUSEBUTTONS (LOGXOR (LOGAND (fetch (KEYBOARDEVENT WU) of \LASTKEYSTATE)
                                           \MOUSE.ALLBITS)
                                   \MOUSE.ALLBITS))
    (SETQ LASTKEYBOARD (\EVENTKEYS))
    (SETQ LASTSCREEN \CURSORSCREEN)
    NIL])
```

(\EVENTKEYS

```
  [LAMBDA NIL                                                    (* rmk%: " 4-JUN-81 22:58")
                                                                 (* Returns the state of the various keys that are represented in
                                                                 mouse events)
    (LOGOR (COND
              ((KEYDOWNP 'LOCK)
               128)
              (T 0))
           (COND
              ((KEYDOWNP 'LSHIFT)
               64)
              (T 0))
           (COND
              ((KEYDOWNP 'CTRL)
               32)
              (T 0))
           (COND
              ((KEYDOWNP 'RSHIFT)
               8)
              (T 0))
           (COND
              ((KEYDOWNP 'BLANK-TOP)
               4)
              (T 0))
           (COND
              ((KEYDOWNP 'BLANK-MIDDLE)
               2)
              (T 0))
           (COND
              ((KEYDOWNP 'BLANK-BOTTOM)
               1)
              (T 0])
```

```
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(RPAQQ HARDCURSORHEIGHT 16)

(RPAQQ HARDCURSORWIDTH 16)

(CONSTANTS (HARDCURSORHEIGHT 16)
       (HARDCURSORWIDTH 16))
)

(DECLARE%: EVAL@COMPILE

(ADDTOVAR GLOBALVARS LASTMOUSEX LASTMOUSEY LASTSCREEN LASTMOUSEBUTTONS LASTMOUSETIME LASTKEYBOARD)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \SETMOUSEXY MACRO [(XPOS YPOS)
                             (PROGN (SELECTC \MACHINETYPE
                                          (\DAYBREAK (\DoveMisc.SetMousePosition XPOS YPOS))
                                          (\MAIKO (SUBRCALL SETMOUSEXY XPOS YPOS))
                                          (\DANDELION (do (PROGN (replace (IOPAGE NEWMOUSEX) of \IOPAGE
                                                                    with XPOS)
                                                                 (replace (IOPAGE NEWMOUSEY) of \IOPAGE
                                                                    with YPOS))
                                                                repeatuntil (ILESSP (fetch (IOPAGE NEWMOUSESTATE)
                                                                                       of \IOPAGE)
                                                                                    32768))
                                                                ; smash position until mouse says it is not busy
                                                                (replace (IOPAGE NEWMOUSEX) of \IOPAGE with XPOS)
                                                                (replace (IOPAGE NEWMOUSEY) of \IOPAGE with YPOS)
                                                                (replace (IOPAGE NEWMOUSESTATE) of \IOPAGE with 32768))
                                          NIL)
                                    (PROGN (\PUTBASE \EM.MOUSEX 0 XPOS)
                                           (\PUTBASE \EM.MOUSEY 0 YPOS])
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

(PUTPROPS \XMOUSECOORD MACRO (NIL (IPLUS \CURSORHOTSPOTX (SIGNED (\GETBASE \EM.CURSORX 0)
                                                                 BITSPERWORD))))

(PUTPROPS \YMOUSECOORD MACRO [NIL (IDIFFERENCE (SUB1 \CURSORDESTHEIGHT)
                                               (IPLUS \CURSORHOTSPOTY (SIGNED (\GETBASE \EM.CURSORY 0)
                                                                              BITSPERWORD])
)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD 'CURSOR 'SETCURSOR)

(MOVD '\CURSORPOSITION '\SETCURSORPOSITION)

(RPAQ \SFPosition (CREATEPOSITION))
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD KEYBOARDEVENT ((W0 WORD)
                            (W1 WORD)
                            (W2 WORD)
                            (W3 WORD)
                            (WU WORD)
                            (W4 WORD)
                            (W5 WORD)
                            (TIME FIXP)
                            (MOUSESTATE BITS 3)
                            (1SHIFT FLAG)
                            (2SHIFT FLAG)
```

```
                            (LOCK FLAG)
                            (CTRL FLAG)
                            (META FLAG)
                            (FONT FLAG)
                            (USERMODE1 FLAG)
                            (USERMODE2 FLAG)
                            (USERMODE3 FLAG)
                            (ALTGRAPH FLAG)
                            (DEADKEYPENDING FLAG)              ; T if the last key was a dead (accent) key, and we should
                                                               ; generate an accented character if possible.
                            (NIL BITS 2)
                            (MOUSEX WORD)
                            (MOUSEY WORD)
                            (DEADKEY-ALIST XPOINTER)           ; The ALIST describing accents possible from teh last dead key.

                           )
       (CREATE (\ALLOCBLOCK (FOLDHI \KEYBOARDEVENT.SIZE WORDSPERCELL)))
       W0 _ ALLUP W1 _ ALLUP W2 _ ALLUP W3 _ ALLUP W4 _ ALLUP W5 _ ALLUP WU _ ALLUP MOUSESTATE _ \DLMOUSE.UP
       [ACCESSFNS KEYBOARDEVENT ((SIZE (INDEXF (fetch MOUSEY of DATUM)))
                                 (SHIFT (OR (fetch (KEYBOARDEVENT 1SHIFT)
                                                   DATUM)
                                            (fetch (KEYBOARDEVENT 2SHIFT)
                                                   DATUM)))
                                 (SHIFTORLOCK (OR (fetch (KEYBOARDEVENT SHIFT)
                                                         DATUM)
                                                  (fetch (KEYBOARDEVENT LOCK)
                                                         DATUM)))
                                 (SHIFTXORLOCK (NEQ (NULL (fetch (KEYBOARDEVENT SHIFT)
                                                                 DATUM))
                                                    (NULL (fetch (KEYBOARDEVENT LOCK)
                                                                 DATUM]
       LOCK _ (XKEYDOWNP 'LOCK)
       TIME _ 0 DEADKEYPENDING _ NIL)
)

(DECLARE%: EVAL@COMPILE

(RPAQ \KEYBOARDEVENT.FIRST NRINGINDEXWORDS)

(RPAQQ \KEYBOARDEVENT.SIZE 14)

(RPAQ \KEYBOARDEVENT.LAST (PLUS \KEYBOARDEVENT.FIRST (TIMES \KEYBOARDEVENT.SIZE 383)))

[CONSTANTS (\KEYBOARDEVENT.FIRST NRINGINDEXWORDS)
       \KEYBOARDEVENT.SIZE
       (\KEYBOARDEVENT.LAST (PLUS \KEYBOARDEVENT.FIRST (TIMES \KEYBOARDEVENT.SIZE 383]
)
)

(DEFINEQ
```

(MACHINETYPE

```
  [LAMBDA NIL                                                  ; Edited 30-Mar-88 10:27 by Snow
    (SELECTC (fetch MachineType of \InterfacePage)
         (\DORADO 'DORADO)
         (\DANDELION 'DANDELION)
         (\DAYBREAK                                            (* This is \DAYBREAK internally)
                    'DOVE)
         (\MAIKO 'MAIKO)
         NIL])
```

(SETMAINTPANEL

```
  [LAMBDA (N)                                                  (* mpl "21-Jul-85 18:15")
    (SELECTC \MACHINETYPE
         (\DANDELION (replace DLMAINTPANEL of \IOPAGE with N))
         (\DOLPHIN ((OPCODES MISC1 3)
                    (\DTEST N 'SMALLP)))
         (\DAYBREAK ((OPCODES DOVEMISC 2)
                     (\DTEST N 'SMALLP)))
         NIL])

)
```

;; DLion beeper

```
(DEFINEQ
```

(BEEPON

```
  [LAMBDA (FREQ)                                               ; Edited 10-May-88 18:17 by MASINTER
    (SELECTC \MACHINETYPE
         (\DANDELION (while (IGEQ (fetch DLBEEPCMD of \IOPAGE)
                                  32768)
                        do (BLOCK))
                     (replace DLBEEPFREQ of \IOPAGE with (IQUOTIENT 1843200 (IMAX FREQ 29)))
                     (replace DLBEEPCMD of \IOPAGE with 32768))
         (\DAYBREAK (\DoveMisc.BeepOn FREQ))
```

```
        (\MAIKO (SUBRCALL KEYBOARDBEEP T FREQ))
        (PROGN NIL))
    NIL])
```

(**BEEPOFF**

```
  [LAMBDA NIL                                                    ; Edited 10-May-88 18:17 by MASINTER
    (SELECTC \MACHINETYPE
         (\DANDELION (while (IGEQ (fetch DLBEEPCMD of \IOPAGE)
                                  32768)
                            do (BLOCK))
                     (replace DLBEEPCMD of \IOPAGE with 32769))
         (\DAYBREAK (\DoveMisc.BeepOff))
         (\MAIKO (SUBRCALL KEYBOARDBEEP NIL NIL))
         (PROGN NIL))
    NIL])

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \EM.MOUSEX \EM.MOUSEY \EM.CURSORX \EM.CURSORY \EM.UTILIN \EM.REALUTILIN \EM.KBDAD0 \EM.KBDAD1
       \EM.KBDAD2 \EM.KBDAD3 \EM.KBDAD4 \EM.KBDAD5 \EM.DISPINTERRUPT \EM.DISPLAYHEAD \EM.CURSORBITMAP
       \MACHINETYPE \DEFAULTKEYACTION \COMMANDKEYACTION \CURRENTKEYACTION \PERIODIC.INTERRUPT
       \PERIODIC.INTERRUPT.FREQUENCY)
)

;; END EXPORTED DEFINITIONS

(DEFINEQ
```

(**WITHOUT-INTERRUPTS**

```
  [NLAMBDA (FORM)                                                (* lmm "18-Apr-85 02:53")
    (PROG (VAL)
          (\KEYBOARDOFF)
          (SETQ VAL (DISPLAYDOWN FORM))
          (\KEYBOARDON)
          (RETURN VAL])

)

;; Compile locked fns together for locality

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: NIL FLIPCURSORBAR \KEYHANDLER \KEYHANDLER1 \TRACKCURSOR \PERIODIC.INTERRUPTFRAME \TIMER.INTERRUPTFRAME
       \DOBUFFEREDTRANSITIONS \DOTRANSITIONS \DECODETRANSITION \EVENTKEYS \HARDCURSORUP \DOMOUSECHORDING
       \KEYBOARDOFF \HARDCURSORPOSITION \HARDCURSORDOWN \SOFTCURSORUP \SOFTCURSORUPCURRENT \SOFTCURSORPOSITION
       \SOFTCURSORDOWN)
)

(DECLARE%: DONTCOPY

(ADDTOVAR INEWCOMS
          (ALLOCAL (ADDVARS (LOCKEDFNS FLIPCURSORBAR \SETIOPOINTERS \KEYHANDLER \KEYHANDLER1 \CONTEXTAPPLY
                                    \LOCKPAGES \DECODETRANSITION \SMASHLINK \INCUSECOUNT LLSH \MAKEFREEBLOCK
                                    \DECUSECOUNT \MAKENUMBER \ADDBASE \PERIODIC.INTERRUPTFRAME
                                    \DOBUFFEREDTRANSITIONS \TIMER.INTERRUPTFRAME \CAUSEINTERRUPT \DOMOUSECHORDING
                                    \KEYBOARDOFF \TRACKCURSOR \HARDCURSORUP \HARDCURSORPOSITION \HARDCURSORDOWN
                                    \SOFTCURSORUP \SOFTCURSORUPCURRENT \SOFTCURSORPOSITION \SOFTCURSORDOWN
                                    \SOFTCURSORPILOTBITBLT)
                           (LOCKEDVARS \InterfacePage \CURSORHOTSPOTX \CURSORHOTSPOTY \CURRENTCURSOR
                                  \SOFTCURSORWIDTH \SOFTCURSORHEIGHT \SOFTCURSORP \SOFTCURSORUPP \SOFTCURSORUPBM
                                  \SOFTCURSORDOWNBM \SOFTCURSORBBT1 \SOFTCURSORBBT2 \SOFTCURSORBBT3
                                  \SOFTCURSORBBT4 \SOFTCURSORBBT5 \SOFTCURSORBBT6 \CURSORDESTINATION
                                  \CURSORDESTHEIGHT \CURSORDESTWIDTH \CURSORDESTRASTERWIDTH \CURSORDESTLINE
                                  \CURSORDESTLINEBASE \PENDINGINTERRUPT \PERIODIC.INTERRUPT
                                  \PERIODIC.INTERRUPT.FREQUENCY \LASTUSERACTION \MOUSECHORDTICKS
                                  \KEYBOARDEVENTQUEUE \KEYBUFFERING SCREENWIDTH SCREENHEIGHT
                                  \TIMER.INTERRUPT.PENDING \EM.MOUSEX \EM.MOUSEY \EM.CURSORX \EM.CURSORY
                                  \EM.UTILIN \EM.REALUTILIN \EM.KBDAD0 \EM.KBDAD1 \EM.KBDAD2 \EM.KBDAD3
                                  \EM.DISPINTERRUPT \EM.CURSORBITMAP \EM.KBDAD4 \EM.KBDAD5 \MISCSTATS \RCLKSECOND
                                  ))))

(ADDTOVAR RDCOMS (FNS \SETIOPOINTERS))
)

(PUTPROPS LLKEY FILETYPE :BCOMPL)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML WITHOUT-INTERRUPTS)
```

```
(ADDTOVAR LAMA CURSORPROP METASHIFT MOUSECHORDWAIT)
)
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

{MEDLEY}<sources>LLKEY.;1

CONSTANT INDEX

RECORD INDEX

PROPERTY INDEX

OPTIMIZER INDEX