-- Keyboard.Mesa  Edited by Johnsson on September 22, 1977 8:18 AM

DIRECTORY
  KeyDefs: FROM "keydefs",
  Mopcodes: FROM "mopcodes",
  StreamDefs: FROM "streamdefs",
  InlineDefs: FROM "inlinedefs",
  ControlDefs: FROM "controldefs",
  ProcessDefs: FROM "processdefs";

DEFINITIONS FROM ProcessDefs, InlineDefs, KeyDefs, StreamDefs;

Keyboard: PROGRAM IMPORTS StreamDefs SHARES ProcessDefs, StreamDefs =

BEGIN
  -- variables set by KeyStreams
  ks: PUBLIC KeyboardHandle;
  CDT: PUBLIC BOOLEAN;
  cursorTracking: PUBLIC BOOLEAN;
  IdleProc: PUBLIC PROCEDURE;
  KeyTable: PUBLIC POINTER TO ARRAY [0..80] OF KeyItem;

  -- The Keyboard part:

  -- fixed addresses for keyboard and mouse
  Keys: POINTER TO KeyArray ~ LOOPHOLE[KeyDefs.Keys];

  Coordinate: TYPE = RECORD [x,y: INTEGER];
  Mouse: POINTER TO Coordinate ~ LOOPHOLE[424B];
  Cursor: POINTER TO Coordinate ~ LOOPHOLE[426B];
  Xmax: CARDINAL = 606-16;
  Ymax: CARDINAL = 808-16;

  ns, os: KeyArray;
  OldState: PUBLIC POINTER TO KeyArray = @os;
  NewState: POINTER TO KeyArray = @ns;

  GetDebugger: MACHINE CODE = INLINE [Mopcodes.zKFCB, ControlDefs.sInterrupt];

ProcessKeyboard: PUBLIC PROCEDURE =

BEGIN
  bitcount, start: [0..15];
  char: [0..377B];
  entry: KeyItem;
  i: [0..SIZE[KeyArray]];
  interruptState: updown = up;
  newin: CARDINAL;
  ph: ProcessHandle;
  pp: ProcessPriority;
  StateWord: WORD:
  stroke: POINTER TO KeyBits = LOOPHOLE[NewState];

  DO
    -- first update the cursor
    IF cursorTracking THEN
      BEGIN
        Mouse.x = Cursor.x = MAX[0.MIN[Xmax,Mouse.x]];
        Mouse.y = Cursor.y = MAX[0.MIN[Ymax,Mouse.y]];
      END;

      NewState* = Keys*;
    -- The following code checks for Ctrl-Swat, the debugger interrupt keys.
    -- This code could be made into a separate process.
    IF stroke.Ctrl = down AND stroke.Spare3 = down THEN
      BEGIN
        IF interruptState = up THEN
          BEGIN
            interruptState = down;
            FOR pp INCREASING IN ProcessPriority DO
              ph = PW[pp];
              IF ph # ProcessNIL AND ph # NIL AND ph.state.instbyte # Mopcodes.BRK THEN
                --
          END;
        END;
      END;
    END;

  END;
BEGIN
  ph.state.instbyte = Mopcodes.ZBRK;
  BLOCK[]: -- try to take breakpoint
  IF PV[pp] = ph AND ph.state.instbyte = 0 THEN EXIT;
  END;
  REPEAT FINISHED => GetDebugger[];
  ENDLOOP;
  NewState* = Keys*;
  END;
END
ELSE interruptState = up;

-- The following code checks for down transitions in the keyboard state
-- and enters characters in the current keystream buffer
FOR i IN [0...SIZE[KeyArray]] DO IF (StateWord ~ BITXOR[OldState[i],NewState[i]]) # 0 THEN BEGIN -- found one or more transitions
  start = 0;
  DO FOR bitcount IN [start...15] DO IF LOOPHOLE[StateWord,INTEGER]<0 THEN EXIT;
     StateWord = BITSHIFT[StateWord,1];
     ENDLOOP;
     entry = KeyTable[i*16 + bitcount];
     IF (char ~ entry.NormalCode)
       AND BITAND[OldState[i].BITSHIFT[10000B,-bitcount]] # 0 THEN
       BEGIN
       SELECT updown[down] FROM
         stroke.Ctrl =>
           IF char = 177B THEN BEGIN
             COT ~ TRUE;
             GOTO skip END
           ELSE char = BITAND[char, 37B];
           stroke.LeftShift, stroke.RightShift =>
             char = entry.ShiftCode;
           stroke.Lock =>
             IF entry.Letter THEN char = entry.ShiftCode;
           ENDCASE;
           IF (new'n~ks.in+l) = KeyBufChars THEN newin = 0;
           IF newin # ks.out THEN BEGIN
             ks.buffer[ks.in] = LOOPHOLE[char];
             ks.in = newin;
             END;
           EXITS skip => NULL;
         END;
     IF (StateWord ~ BITSHIFT[StateWord,1])=0 THEN EXIT;
     start = bitcount+1;
     ENDLOOP;
     END;
ENDLOOP;
OldState* = NewState*;
BLOCK[];
ENDLOOP;
END;

ReadChar: PUBLIC PROCEDURE [stream: StreamHandle] RETURNS [char: UNSPECIFIED] =
BEGIN char ~ 0;
WITH s:stream SELECT FROM
  Keyboard =>
  DO -- until character typed
    IF s.out # s.in THEN BEGIN
      char = s.buffer[s.out];
      s.out +
        IF s.out = KeyBufChars-1
          THEN 0 [
            s.out+1;
            RETURN
          ]
        IF IdleProc#LOOPHOLE[0] THEN IdleProc[];
    END;
  ENDCASE => SIGNAL StreamError[stream.StreamType];
RETURN;
END;

InputBufferEmpty: PUBLIC PROCEDURE [stream:StreamHandle] RETURNS [BOOL] =
BEGIN
WITH s:stream SELECT FROM
  Keyboard => RETURN[s.in = s.out];
ENDCASE => SIGNAL StreamError[stream.StreamType];
  RETURN[FALSE];
END;

OldState+ = Keys+;
END.