```
-- KeyStreams.Mesa  Edited by Sandman on May 12, 1978  2:20 PM

DIRECTORY
  InlineDefs: FROM "inlinedefs" USING [BITOR, BITSHIFT],
  Keyboard: FROM "keyboard" USING [
    CDT, charactersAvailable, Cursor, cursorTracking, InputBufferEmpty,
    KeyboardPriority, Keys, KeyTable, ks, monitor, Mouse, ProcessKeyboard,
    ReadChar, wakeup],
  KeyDefs: FROM "keydefs" USING [KeyItem, KeyName, Keys],
  ProcessDefs: FROM "processdefs" USING [
    CV, DIW, GetPriority, InterruptLevel, Priority, SetPriority],
  StreamDefs: FROM "streamdefs" USING [
    KeyboardHandle, KeyBufChars, StreamError, StreamHandle, StreamObject],
  SystemDefs: FROM "systemdefs" USING [AllocateHeapNode, FreeHeapNode];

KeyStreams: MONITOR LOCKS Keyboard.monitor
  IMPORTS Keyboard, ProcessDefs, StreamDefs, SystemDefs
  EXPORTS KeyDefs, StreamDefs SHARES StreamDefs =

BEGIN OPEN StreamDefs;

-- The Stream part:

KS: PUBLIC Keyboard StreamObject ← StreamObject [
  ClearInputBuffer, Keyboard.ReadChar, PutBackChar,
  WriteChar, Keyboard.InputBufferEmpty, DestroyKey,
  Keyboard[0,0,]];

GetDefaultKey: PUBLIC PROCEDURE RETURNS [KeyboardHandle] =
  BEGIN
  RETURN[@KS];
  END;

GetCurrentKey: PUBLIC PROCEDURE RETURNS [KeyboardHandle] =
  BEGIN
  RETURN[Keyboard.ks];
  END;

CreateKeyStream: PUBLIC PROCEDURE RETURNS [KeyboardHandle] =
  BEGIN OPEN SystemDefs;
  s: KeyboardHandle ←
    SystemDefs.AllocateHeapNode[SIZE[Keyboard StreamObject]];
  s↑ ← KS;
  s.in ← s.out ← 0;
  RETURN[s];
  END;

ControlDELtyped: PUBLIC PROCEDURE RETURNS [BOOLEAN] =
  BEGIN RETURN[Keyboard.CDT] END;

ResetControlDEL: PUBLIC PROCEDURE =
  BEGIN Keyboard.CDT ← FALSE END;

OpenKeyStream: PUBLIC ENTRY PROCEDURE [stream:StreamHandle] =
  BEGIN
  WITH s:stream SELECT FROM
    Keyboard => Keyboard.ks ← @s;
    ENDCASE => SIGNAL StreamError[stream,StreamType ! UNWIND => NULL];
  RETURN;
  END;

ClearInputBuffer: ENTRY PROCEDURE [stream:StreamHandle] =
  BEGIN
  WITH s:stream SELECT FROM
    Keyboard => s.in ← s.out ← 0;
    ENDCASE => SIGNAL StreamError[stream,StreamType ! UNWIND => NULL];
  RETURN;
  END;

PutBackChar: ENTRY PROCEDURE [stream:StreamHandle, char:UNSPECIFIED] =
  BEGIN
  newout: CARDINAL;
  WITH s:stream SELECT FROM
    Keyboard =>
      BEGIN
      newout ← s.out;
```

```
      newout ← IF newout = 0 THEN KeyBufChars-1 ELSE newout-1;
      IF newout # s.in THEN
        BEGIN
        s.out ← newout;
        s.buffer[s.out] ← char;
        BROADCAST Keyboard.charactersAvailable;
        END;
      END;
    ENDCASE => SIGNAL StreamError[stream,StreamType ! UNWIND => NULL];
  RETURN;
  END;

WriteChar: PROCEDURE [stream:StreamHandle, char:UNSPECIFIED] =
  BEGIN
  SIGNAL StreamError[stream,StreamAccess];
  RETURN
  END;

CloseKeyStream: PUBLIC ENTRY PROCEDURE [stream:StreamHandle] =
  BEGIN
  WITH s:stream SELECT FROM
    Keyboard => Keyboard.ks ← @KS;
    ENDCASE => SIGNAL StreamError[stream,StreamType ! UNWIND => NULL];
  RETURN;
  END;

DestroyKey: --ENTRY-- PROCEDURE [stream:StreamHandle] =
  BEGIN OPEN SystemDefs;
  WITH s:stream SELECT FROM
    Keyboard =>
      BEGIN
      IF @s = Keyboard.ks THEN Keyboard.ks ← @KS;
      IF @s # @KS THEN FreeHeapNode[@s ! UNWIND => NULL];
      END;
    ENDCASE => SIGNAL StreamError[stream,StreamType ! UNWIND => NULL];
  RETURN;
  END;

CursorTrack: PUBLIC PROCEDURE [b: BOOLEAN] =
  BEGIN
  Keyboard.cursorTracking ← b;
  RETURN
  END;


KeyItem: TYPE = KeyDefs.KeyItem;

KeyTable: ARRAY [0..80) OF KeyItem ← [
  -- MEMORY[177033B]    Index [0..15]
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- UNUSED
  KeyItem[FALSE, 0, 0],  -- KeyItemset1
  KeyItem[FALSE, 0, 0],  -- KeyItemset2
  KeyItem[FALSE, 0, 0],  -- KeyItemset3
  KeyItem[FALSE, 0, 0],  -- KeyItemset4
  KeyItem[FALSE, 0, 0],  -- KeyItemset5
  KeyItem[FALSE, 0, 0],  -- Red
  KeyItem[FALSE, 0, 0],  -- Blue
  KeyItem[FALSE, 0, 0],  -- Yellow
  -- MEMORY[177034B]    Index [16..31]
  KeyItem[FALSE,  45B,  65B],  -- %,5
  KeyItem[FALSE,  44B,  64B],  -- $,4
  KeyItem[FALSE, 176B,  66B],  -- ~,6
  KeyItem[TRUE,  105B, 145B],  -- E
  KeyItem[FALSE,  46B,  67B],  -- &,7
  KeyItem[TRUE,  104B, 144B],  -- D
  KeyItem[TRUE,  125B, 165B],  -- U
  KeyItem[TRUE,  126B, 166B],  -- V
  KeyItem[FALSE,  51B,  60B],  -- ),0
```

```
  KeyItem[TRUE,  113B, 153B],  -- K
  KeyItem[FALSE,  30B,  55B],  --
**,-
  KeyItem[TRUE,  120B, 160B],  -- P
  KeyItem[FALSE,  77B,  57B],  -- ?,/
  KeyItem[FALSE, 174B, 134B],  -- |,\
  KeyItem[FALSE,  12B,  12B],  -- LF
  KeyItem[FALSE,  10B,  10B],  -- BS
  -- MEMORY[177035B]    Index [32..47]
  KeyItem[FALSE,  43B,  63B],  -- #,3
  KeyItem[FALSE, 100B,  62B],  -- @,2
  KeyItem[TRUE,  127B, 167B],  -- W
  KeyItem[TRUE,  121B, 161B],  -- Q
  KeyItem[TRUE,  123B, 163B],  -- S
  KeyItem[TRUE,  101B, 141B],  -- A
  KeyItem[FALSE,  50B,  71B],  -- (,9
  KeyItem[TRUE,  111B, 151B],  -- I
  KeyItem[TRUE,  130B, 170B],  -- X
  KeyItem[TRUE,  117B, 157B],  -- O
  KeyItem[TRUE,  114B, 154B],  -- L
  KeyItem[FALSE,  74B,  54B],  -- <,,
  KeyItem[FALSE,  42B,  47B],  -- ",'
  KeyItem[FALSE, 175B, 135B],  -- },]
  KeyItem[FALSE,   0B,   0B],  -- SPARE2
  KeyItem[FALSE,   0B,   0B],  -- SPARE1
  -- MEMORY[177036B]    Index [48..63]
  KeyItem[FALSE,  41B,  61B],  -- !,1
  KeyItem[FALSE,  33B,  33B],  -- ESCAPE
  KeyItem[FALSE,  11B,  11B],  -- TAB
  KeyItem[TRUE,  106B, 146B],  -- F
  KeyItem[FALSE,   0B,  0B],   -- CONTROL
  KeyItem[TRUE,  103B, 143B],  -- C
  KeyItem[TRUE,  112B, 152B],  -- J
  KeyItem[TRUE,  102B, 142B],  -- B
  KeyItem[TRUE,  132B, 172B],  -- Z
  KeyItem[FALSE,   0B,  0B],   -- SHIFT
  KeyItem[FALSE,  76B,  56B],  -- >,.
  KeyItem[FALSE,  72B,  73B],  -- :,;
  KeyItem[FALSE,  15B,  15B],  -- CR
  KeyItem[FALSE, 136B, 137B],  -- ↑,←
  KeyItem[FALSE, 177B, 177B],  -- DEL
  KeyItem[FALSE,   0B,  0B],   -- NOT USED (FL3)
  -- MEMORY[177037B]    Index [64..79]
  KeyItem[TRUE,  122B, 162B],  -- R
  KeyItem[TRUE,  124B, 164B],  -- T
  KeyItem[TRUE,  107B, 147B],  -- G
  KeyItem[TRUE,  131B, 171B],  -- Y
  KeyItem[TRUE,  110B, 150B],  -- H
  KeyItem[FALSE,  52B,  70B],  -- *,8
  KeyItem[TRUE,  116B, 156B],  -- N
  KeyItem[TRUE,  115B, 155B],  -- M
  KeyItem[FALSE,   0B,   0B],  -- LOCK
  KeyItem[FALSE,  40B,  40B],  -- SPACE
  KeyItem[FALSE, 173B, 133B],  -- {,[
  KeyItem[FALSE,  53B,  75B],  -- +,=
  KeyItem[FALSE,   0B,   0B],  -- SHIFT
  KeyItem[FALSE,   0B,   0B],  -- SPARE3
  KeyItem[FALSE,   0B,   0B],  -- NOT USED (FL4)
  KeyItem[FALSE,   0B,   0B]]; -- NOT USED (FR5)

ChangeKey: PUBLIC PROCEDURE [key: KeyDefs.KeyName, action: KeyDefs.KeyItem]
  RETURNS [oldAction: KeyDefs.KeyItem] =
  BEGIN
  oldAction ← KeyTable[LOOPHOLE[key,CARDINAL]];
  KeyTable[LOOPHOLE[key,CARDINAL]] ← action;
  RETURN
  END;


-- The Process (and initialization) part

Init: PROCEDURE =
  BEGIN OPEN ProcessDefs;
  KeyboardLevel: InterruptLevel = 7;
  KeyboardBit: WORD = InlineDefs.BITSHIFT[1,KeyboardLevel];
  save: Priority = GetPriority[];
```

```
  p: PROCESS;

  START Keyboard;
  ResetControlDEL[];
  CursorTrack[TRUE];
  Keyboard.Keys ← LOOPHOLE[KeyDefs.Keys];
  Keyboard.KeyTable ← @KeyTable;
  Keyboard.ks ← @KS;
  Keyboard.Mouse ← LOOPHOLE[424B];
  Keyboard.Cursor ← LOOPHOLE[426B];
  SetPriority[Keyboard.KeyboardPriority];
  p ← FORK Keyboard.ProcessKeyboard;
  SetPriority[save];
  CV[KeyboardLevel] ← @Keyboard.wakeup;
  DIW↑ ← InlineDefs.BITOR[DIW↑,KeyboardBit];
  END;

Init[];

END.
```