```
TITLE "PS/2 INPUT";
%----------------------------------------------------------------------
    PS/2 input module : Communication between PS/2 bus and System
                        bus.
%----------------------------------------------------------------------
INCLUDE "SC_TO_U2.inc";

CONSTANT ps2_addr = H"50";              -- Keyboard address in I/O

SUBDESIGN PS2_IN
(
    GEN                     : input;    -- 20MHz clock
    RESET                   : input;    -- Reset signal from uC

    -- Bus
    ADDR[7..0]              : input;    -- Address bus
    DATA[7..0]              : output;   -- Data bus
    WAIT                    : output;   -- Wait signal
    IORQ                    : input;    -- I/O request signal
    RD                      : input;    -- Read signal

    -- PS/2 debug
    PS2_DEBUG_PORT[7..0]    : output;   -- Output port for debugging

    -- PS/2 connection port
    PDATA_IN[7..0]          : input;    -- PS/2 bus input port
)
VARIABLE
    -- Bus
    CMD_READ                : DFF;      -- Read command from uC
    KEY_Z[7..0]             : DFF;      -- Sign byte from keyboard
    KEY_D0[7..0]            : DFF;      -- First number
    KEY_D1[7..0]            : DFF;      -- Second number
    KEY_D2[7..0]            : DFF;      -- Third number
    SIG_WAIT                : DFF;      -- Wait signal for uC
    DFF_DATA[7..0]          : DFF;      -- Register for final U2 number
    CNT_DATA[7..0]          : DFF;      -- Wait extension counter
    S2U                     : SC_TO_U2; -- Scancode to U2 decoder
    -- Secure input dff's
    DFF_1_PS2_CLK           : DFF;      -- Input dff #1 for clock
    DFF_2_PS2_CLK           : DFF;      -- Input dff #2 for clock
    DFF_1_PS2_DATA          : DFF;      -- Input dff #1 for data
    DFF_2_PS2_DATA          : DFF;      -- Input dff #2 for data
    -- Data recieving
    PS2_FRAME[10..0]        : DFF;      -- Recieved frame
    PS2_CNT_FRAME[3..0]     : DFF;      -- Frame bit's counter
    PS2_CNT_BYTE[1..0]      : DFF;      -- ZDDD byte counter
    PS2_BYTE[7..0]          : DFF;      -- Data from PS/2 frame -
hardwired
    PS2_BYTE_READY          : DFF;      -- Data byte is ready flag
    -- Debug
    D_PS2_DEBUG_PORT[7..0]  : DFF;      -- Debugging register

    -- Machines
```

```
    -- Machine for PS/2 bus communication
    AUT_PS2     : machine of bits (Q[1..0])
                  with states (S0=B"00", S1=B"01",
                               S2=B"10", S3=B"11");

    -- Machine for system bus communication
    AUT_PS2_BUS : machine of bits (QB[2..0])
                  with states (SPB0=B"000", SPB1=B"001", SPB2=B"010",
                               SPB3=B"011", SPB4=B"100", SPB5=B"101",
                               SPB6=B"110", SPB7=B"111");


BEGIN
    % assign global clock %
    DFF_1_PS2_CLK.CLK       = GEN;
    DFF_2_PS2_CLK.CLK       = GEN;
    DFF_1_PS2_DATA.CLK      = GEN;
    DFF_2_PS2_DATA.CLK      = GEN;
    PS2_FRAME[].CLK         = GEN;
    PS2_CNT_FRAME[].CLK     = GEN;
    PS2_CNT_BYTE[].CLK      = GEN;
    D_PS2_DEBUG_PORT[].CLK  = GEN;
    PS2_BYTE[].CLK          = GEN;
    PS2_BYTE_READY.CLK      = GEN;
    CMD_READ.CLK            = GEN;
    KEY_Z[].CLK             = GEN;
    KEY_D0[].CLK            = GEN;
    KEY_D1[].CLK            = GEN;
    KEY_D2[].CLK            = GEN;
    DFF_DATA[].CLK          = GEN;
    SIG_WAIT.CLK            = GEN;
    CNT_DATA[].CLK          = GEN;
    S2U.GEN                 = GEN;
    AUT_PS2.CLK             = GEN;
    AUT_PS2_BUS.CLK         = GEN;

    % assign global reset %
    S2U.RESET               = RESET;
    DFF_1_PS2_CLK.CLRN      = RESET;
    DFF_2_PS2_CLK.CLRN      = RESET;
    DFF_1_PS2_DATA.CLRN     = RESET;
    DFF_2_PS2_DATA.CLRN     = RESET;
    PS2_FRAME[].CLRN        = RESET;
    PS2_CNT_FRAME[].CLRN    = RESET;
    PS2_CNT_BYTE[].CLRN     = RESET;
    D_PS2_DEBUG_PORT[].CLRN = RESET;
    PS2_BYTE[].CLRN         = RESET;
    PS2_BYTE_READY.CLRN     = RESET;
    CMD_READ.CLRN           = RESET;
    KEY_Z[].CLRN            = RESET;
    KEY_D0[].CLRN           = RESET;
    KEY_D1[].CLRN           = RESET;
    KEY_D2[].CLRN           = RESET;
    DFF_DATA[].CLRN         = RESET;
    SIG_WAIT.CLRN           = RESET;
```

```
    CNT_DATA[].CLRN          = RESET;
    AUT_PS2.RESET            = !RESET;
    AUT_PS2_BUS.RESET        = !RESET;

    % connect variables %
    WAIT                     = SIG_WAIT.Q;

    S2U.SC_Z[]               = KEY_Z[];
    S2U.SC_D0[]              = KEY_D0[];
    S2U.SC_D1[]              = KEY_D1[];
    S2U.SC_D2[]              = KEY_D2[];

    case PS2_CNT_BYTE[] is
        when B"00"  => PS2_DEBUG_PORT[] = 0;
        when B"01"  => PS2_DEBUG_PORT[] = 1;
        when B"10"  => PS2_DEBUG_PORT[] = 3;
        when B"11"  => PS2_DEBUG_PORT[] = 7;
    end case;

    DATA[] = DFF_DATA[];

    % Hardwired byte from frame %
    PS2_BYTE[] = (PS2_FRAME8,PS2_FRAME7,PS2_FRAME6,PS2_FRAME5,PS2_FRAME4,
PS2_FRAME3,PS2_FRAME2,PS2_FRAME1);
    % Secure input of ps2 clock %
    DFF_1_PS2_CLK.D = PDATA_IN1;
    DFF_2_PS2_CLK.D = DFF_1_PS2_CLK.Q;

    % Secure input of ps2 data %
    DFF_1_PS2_DATA.D = PDATA_IN0;
    DFF_2_PS2_DATA.D = DFF_1_PS2_DATA.Q;

    % Bus %
    if(ADDR[] == ps2_addr & IORQ == GND & RD == GND) then
            % Command read %
            CMD_READ.D = VCC;
    else
            CMD_READ.D = GND;
    end if;

    case AUT_PS2_BUS is
        % Wait for command from uC %
        when SPB0 => CMD_READ.D = CMD_READ.Q;
                     KEY_Z[] = B"00000000";
                     KEY_D0[] = B"00000000";
                     KEY_D1[] = B"00000000";
                     KEY_D2[] = B"00000000";
                     PS2_CNT_BYTE[] = B"00";
                     CNT_DATA[] = B"00000000";
                     DFF_DATA[] = 0;

                     if(CMD_READ.Q == 1) then
                        SIG_WAIT.D = GND;
                        -- activate reciever
                        AUT_PS2_BUS = SPB1;
```

```
                else
                   -- unblock after reset
                   SIG_WAIT.D = VCC;
                   AUT_PS2_BUS = SPB0;
                end if;
        % Wait for frame %
        when SPB1 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                DFF_DATA[] = DFF_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];

                if(PS2_BYTE_READY.Q == 1) then
                   AUT_PS2_BUS = SPB2;
                else
                   AUT_PS2_BUS = SPB1;
                end if;
        % Check if frame data is "F0" %
        when SPB2 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                DFF_DATA[] = DFF_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];

                if(PS2_BYTE[] == B"11110000") then
                   AUT_PS2_BUS = SPB3;
                else
                   AUT_PS2_BUS = SPB1;
                end if;
        % Wait for data frame ( after "F0" comes scancode of key ) %
        when SPB3 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                DFF_DATA[] = DFF_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];

                if(PS2_BYTE_READY.Q == 1) then
                   AUT_PS2_BUS = SPB4;
                else
                   AUT_PS2_BUS = SPB3;
                end if;
        % Write key scancode into correct buffer %
        when SPB4 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
```

```
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                DFF_DATA[] = DFF_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[]+1;

                case PS2_CNT_BYTE[] is
                   when 0 => KEY_Z[] = PS2_BYTE[];
                             AUT_PS2_BUS = SPB1;
                   when 1 => KEY_D2[] = PS2_BYTE[];
                             AUT_PS2_BUS = SPB1;
                   when 2 => KEY_D1[] = PS2_BYTE[];
                             AUT_PS2_BUS = SPB1;
                   when 3 => KEY_D0[] = PS2_BYTE[];
                             AUT_PS2_BUS = SPB5;
                end case;
        % Convert ZDDD to U2 and put it on data bus %
        when SPB5 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];

                DFF_DATA[] = S2U.U2_LS[];

                if( CNT_DATA[] < 20 ) then
                   AUT_PS2_BUS = SPB6;
                else
                   AUT_PS2_BUS = SPB7;
                end if;

        % Extend wait signal %
        when SPB6 => CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[]+1;
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];
                DFF_DATA[] = DFF_DATA[];

                AUT_PS2_BUS = SPB5;
        % Remove Wait, and wait for processor read %
        when SPB7 => SIG_WAIT.D = VCC;
                CMD_READ.D = CMD_READ.Q;
                KEY_Z[] = KEY_Z[];
                KEY_D0[] = KEY_D0[];
                KEY_D1[] = KEY_D1[];
                KEY_D2[] = KEY_D2[];
                CNT_DATA[] = CNT_DATA[];
                PS2_CNT_BYTE[] = PS2_CNT_BYTE[];

                DFF_DATA[] = DFF_DATA[];
```

```
                    if(IORQ == VCC & RD == VCC)then
                        AUT_PS2_BUS = SPB0;
                    else
                        AUT_PS2_BUS = SPB7;
                    end if;

    end case;

    case AUT_PS2 is
        % PS2 clock down - start bit %
        when S0 =>  PS2_CNT_FRAME[] = B"0000";
                    PS2_FRAME[] = B"00000000000";
                    PS2_BYTE_READY.D = GND;

                    if(DFF_1_PS2_CLK.Q == 0 & DFF_2_PS2_CLK.Q == 1) then
                        AUT_PS2 = S2;
                    else
                        AUT_PS2 = S0;
                    end if;
        % Wait for clock falling edge %
        when S1 =>  PS2_CNT_FRAME[] = PS2_CNT_FRAME[];
                    PS2_FRAME[] = PS2_FRAME[];
                    PS2_BYTE_READY.D = GND;

                    if(DFF_1_PS2_CLK.Q == 0 & DFF_2_PS2_CLK.Q == 1) then
                        AUT_PS2 = S2;
                    else
                        AUT_PS2 = S1;
                    end if;
        % Read bit into frame buffer %
        when S2 =>  PS2_CNT_FRAME[] = PS2_CNT_FRAME[]+1;
                    PS2_FRAME[] = (DFF_2_PS2_DATA.Q, PS2_FRAME[10..1]);
                    PS2_BYTE_READY.D = GND;

                    if(PS2_CNT_FRAME[] < 10) then
                        AUT_PS2 = S1;
                    else
                        AUT_PS2 = S3;
                    end if;
        % 11 bit frame is ready %
        when S3 =>  PS2_FRAME[] = PS2_FRAME[];
                    PS2_CNT_FRAME[] = PS2_CNT_FRAME[];

                    PS2_BYTE_READY.D = VCC;
                    -- reciever off
                    AUT_PS2 = S0;
    end case;

END;
```