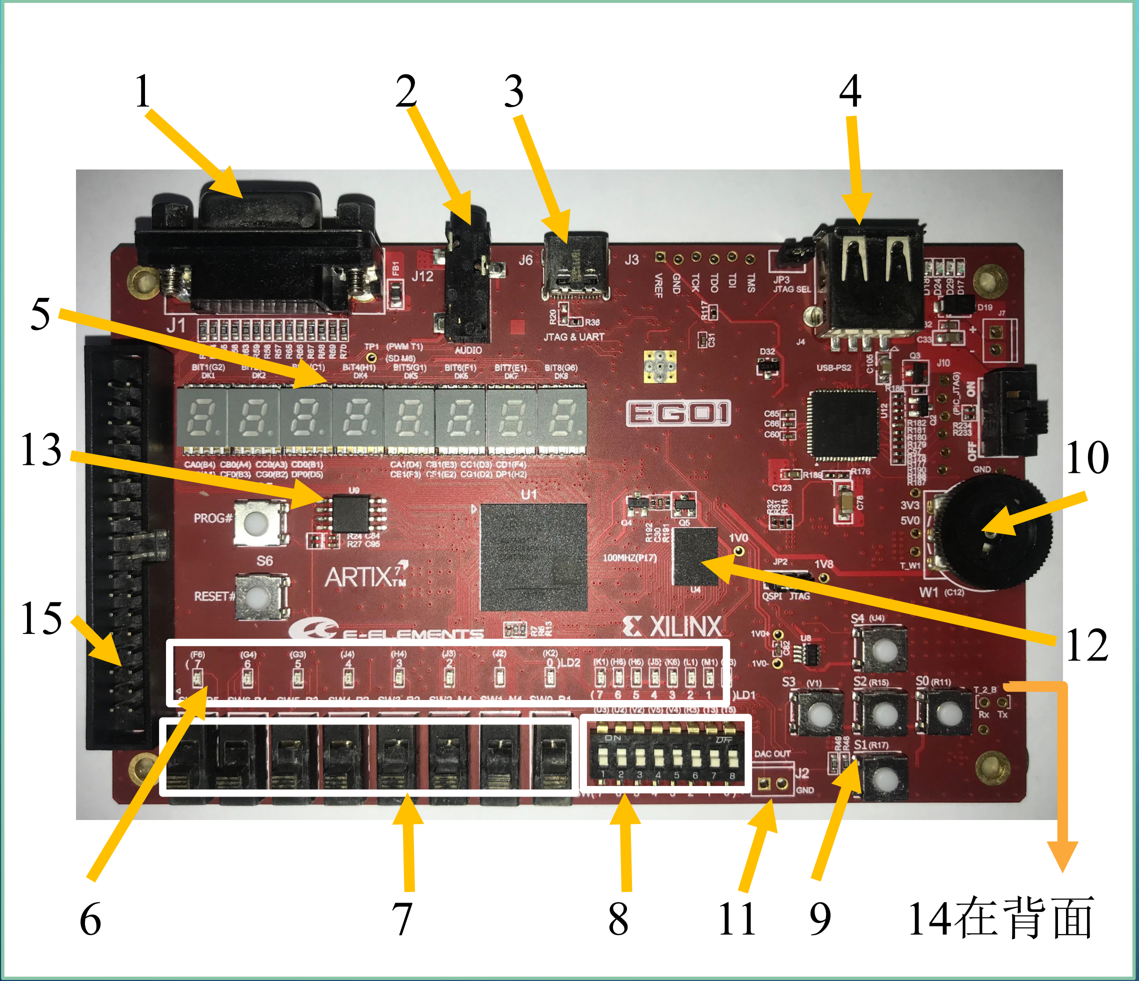
# DIGITAL DESIGN

LAB SUPPLEMENTARY   INTRODUCTION TO PS/2 KEYBOARD

2022 SUMMER TERM

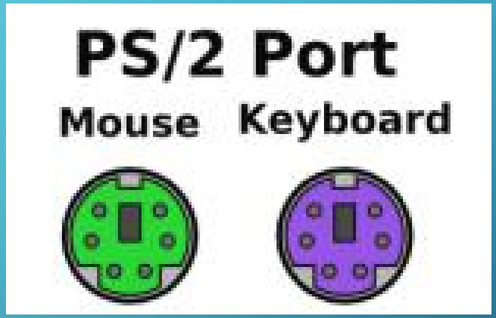
# EGO1



| 编号 | 描述 | 编号 | 描述 |
|---|---|---|---|
| 1 | VGA接口 | 9 | 5个按键 |
| 2 | 音频接口 | 10 | 1个模拟电压输入 |
| 3 | USB转Type-C接口 | 11 | 1个DAC输出接口 |
| 4 | USB接口 | 12 | SRAM存储器 |
| 5 | 2个4位数码管 | 13 | SPI FLASH存储器 |
| 6 | 16个LED灯 | 14 | 蓝牙模块 |
| 7 | 8个拔码开关 | 15 | 通用扩展接口 |
| 8 | 1个8位DIP开关 | | |

# INTRODUCTION

- Common keyboard and mouse interfaces
  - PS/2 (traditional)
  - USB (modern)
- EGO1 includes a USB to PS/2 converter
- We will use PS/2 keyboard protocol

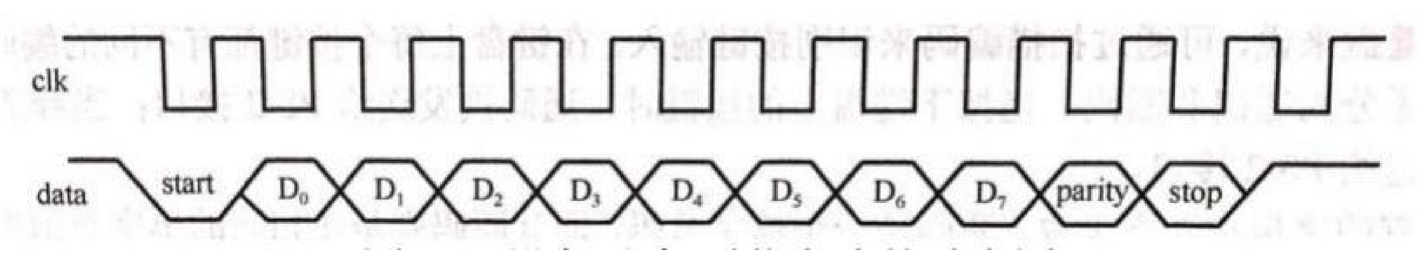


# PS/2 KEYBOARD PROTOCOL

- A bidirectional synchronous serial protocol
- Two directions
  - Keyboard-to-host communication
  - Host-to-keyboard communication
- Clock signal is always generated by the keyboard
- Clock frequency: 10kHZ~16.7kHz

# KEYBOARD-TO-HOST COMMUNICATION

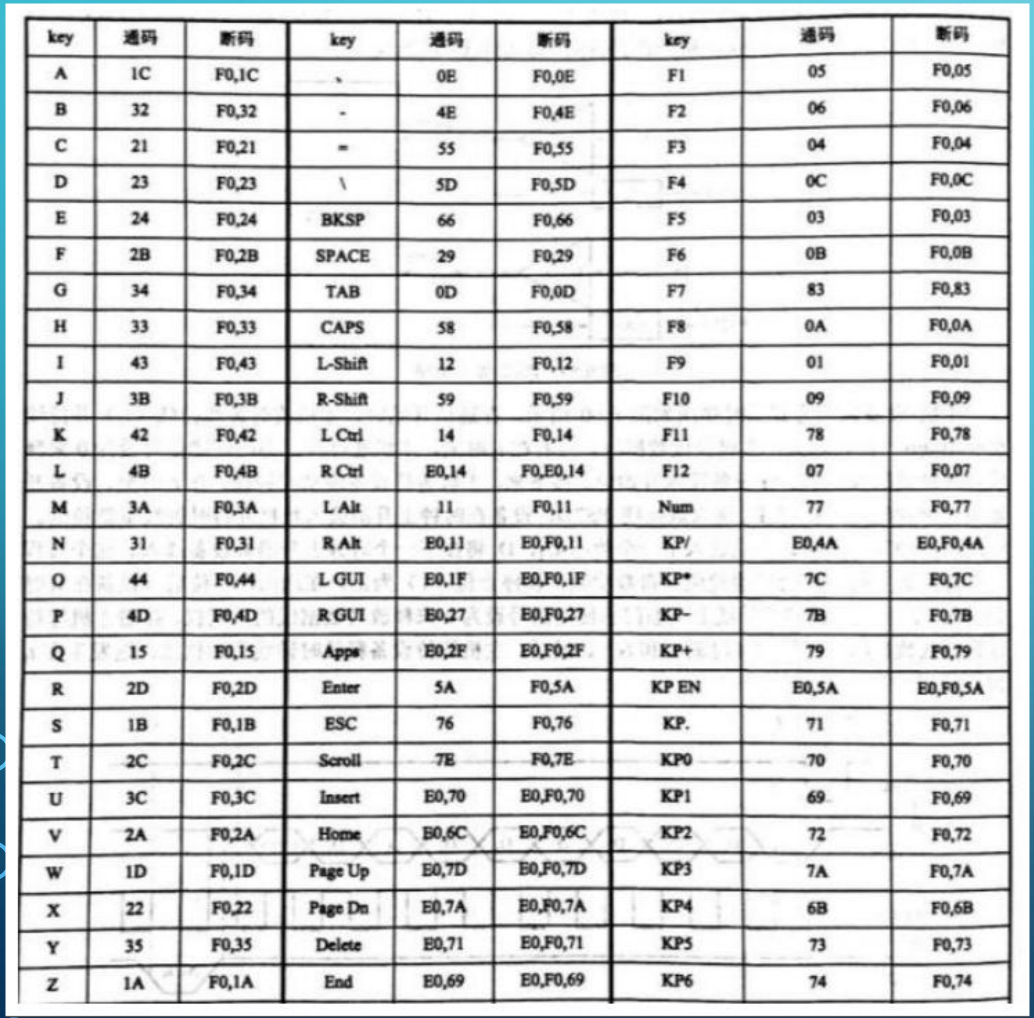
- 11-bit frames
  - 1 start bit, always 0
  - 8 data bits
  - 1 parity bit (odd parity)
  - 1 stop bit, always 1
- Data sent to the host is read on the falling edge of the clock signal



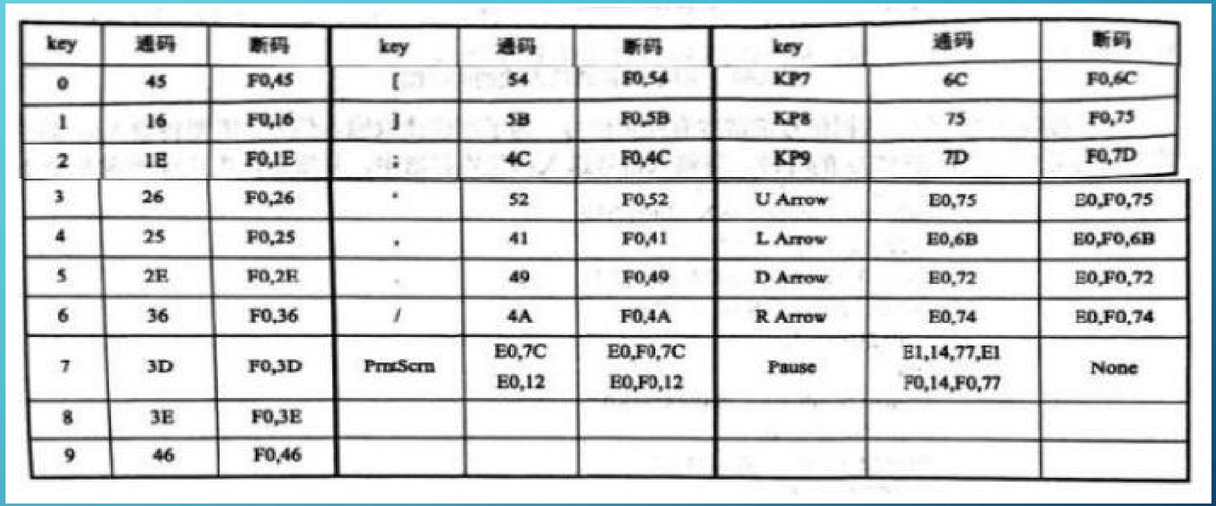
# CODES

- Each key has two kinds of unique codes
  - Make code: when a key is pressed
  - Break code: when a key is released
- Commonly, break codes are two-byte long where the first byte is F0 and the second byte is the make code for that key
  - Some keys have 2-byte long make codes, starting with E0
  - PrintScreen 4-byte long make code
  - Pause 8-byte long make code

# CODES



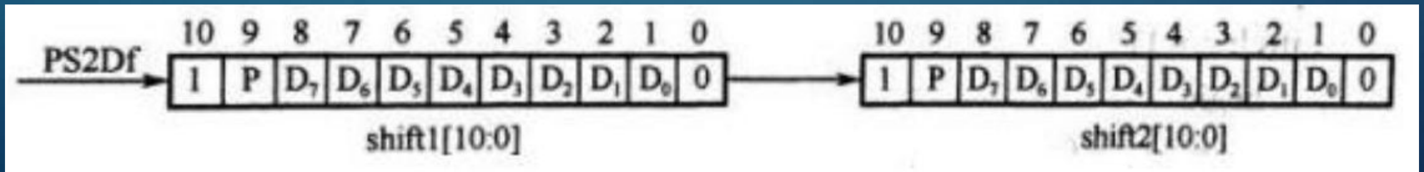
| key | 通码 | 断码 | key | 通码 | 断码 | key | 通码 | 断码 |
|---|---|---|---|---|---|---|---|---|
| A | 1C | F0,1C | ` | 0E | F0,0E | F1 | 05 | F0,05 |
| B | 32 | F0,32 | - | 4E | F0,4E | F2 | 06 | F0,06 |
| C | 21 | F0,21 | = | 55 | F0,55 | F3 | 04 | F0,04 |
| D | 23 | F0,23 | \ | 5D | F0,5D | F4 | 0C | F0,0C |
| E | 24 | F0,24 | BKSP | 66 | F0,66 | F5 | 03 | F0,03 |
| F | 2B | F0,2B | SPACE | 29 | F0,29 | F6 | 0B | F0,0B |
| G | 34 | F0,34 | TAB | 0D | F0,0D | F7 | 83 | F0,83 |
| H | 33 | F0,33 | CAPS | 58 | F0,58 | F8 | 0A | F0,0A |
| I | 43 | F0,43 | L-Shift | 12 | F0,12 | F9 | 01 | F0,01 |
| J | 3B | F0,3B | R-Shift | 59 | F0,59 | F10 | 09 | F0,09 |
| K | 42 | F0,42 | L Ctrl | 14 | F0,14 | F11 | 78 | F0,78 |
| L | 4B | F0,4B | R Ctrl | E0,14 | F0,E0,14 | F12 | 07 | F0,07 |
| M | 3A | F0,3A | L Alt | 11 | F0,11 | Num | 77 | F0,77 |
| N | 31 | F0,31 | R Alt | E0,11 | E0,F0,11 | KP/ | E0,4A | E0,F0,4A |
| O | 44 | F0,44 | L GUI | E0,1F | E0,F0,1F | KP* | 7C | F0,7C |
| P | 4D | F0,4D | R GUI | E0,27 | E0,F0,27 | KP- | 7B | F0,7B |
| Q | 15 | F0,15 | Apps | E0,2F | E0,F0,2F | KP+ | 79 | F0,79 |
| R | 2D | F0,2D | Enter | 5A | F0,5A | KP EN | E0,5A | E0,F0,5A |
| S | 1B | F0,1B | ESC | 76 | F0,76 | KP. | 71 | F0,71 |
| T | 2C | F0,2C | Scroll | 7E | F0,7E | KP0 | 70 | F0,70 |
| U | 3C | F0,3C | Insert | E0,70 | E0,F0,70 | KP1 | 69 | F0,69 |
| V | 2A | F0,2A | Home | E0,6C | E0,F0,6C | KP2 | 72 | F0,72 |
| W | 1D | F0,1D | Page Up | E0,7D | E0,F0,7D | KP3 | 7A | F0,7A |
| X | 22 | F0,22 | Page Dn | E0,7A | E0,F0,7A | KP4 | 6B | F0,6B |
| Y | 35 | F0,35 | Delete | E0,71 | E0,F0,71 | KP5 | 73 | F0,73 |
| Z | 1A | F0,1A | End | E0,69 | E0,F0,69 | KP6 | 74 | F0,74 |



| key | 通码 | 断码 | key | 通码 | 断码 | key | 通码 | 断码 |
|---|---|---|---|---|---|---|---|---|
| 0 | 45 | F0,45 | [ | 54 | F0,54 | KP7 | 6C | F0,6C |
| 1 | 16 | F0,16 | ] | 5B | F0,5B | KP8 | 75 | F0,75 |
| 2 | 1E | F0,1E | ; | 4C | F0,4C | KP9 | 7D | F0,7D |
| 3 | 26 | F0,26 | ' | 52 | F0,52 | U Arrow | E0,75 | E0,F0,75 |
| 4 | 25 | F0,25 | , | 41 | F0,41 | L Arrow | E0,6B | E0,F0,6B |
| 5 | 2E | F0,2E | . | 49 | F0,49 | D Arrow | E0,72 | E0,F0,72 |
| 6 | 36 | F0,36 | / | 4A | F0,4A | R Arrow | E0,74 | E0,F0,74 |
| 7 | 3D | F0,3D | PrntScrn | E0,7C<br>E0,12 | E0,F0,7C<br>E0,F0,12 | Pause | E1,14,77,E1<br>F0,14,F0,77 | None |
| 8 | 3E | F0,3E |  |  |  |  |  |  |
| 9 | 46 | F0,46 |  |  |  |  |  |  |

# IMPLEMENT

- Read data sent by the keyboard, and display the keycode using leds
- Need to filter clock and data signal
- Filtered data signals are stored by two 11-bit shift registers
  - shift2[8:1]   the first byte
  - shift2[8:1]   the second byte
- xkey[15:0] is the output of 2-byte keycode



PS2Df →

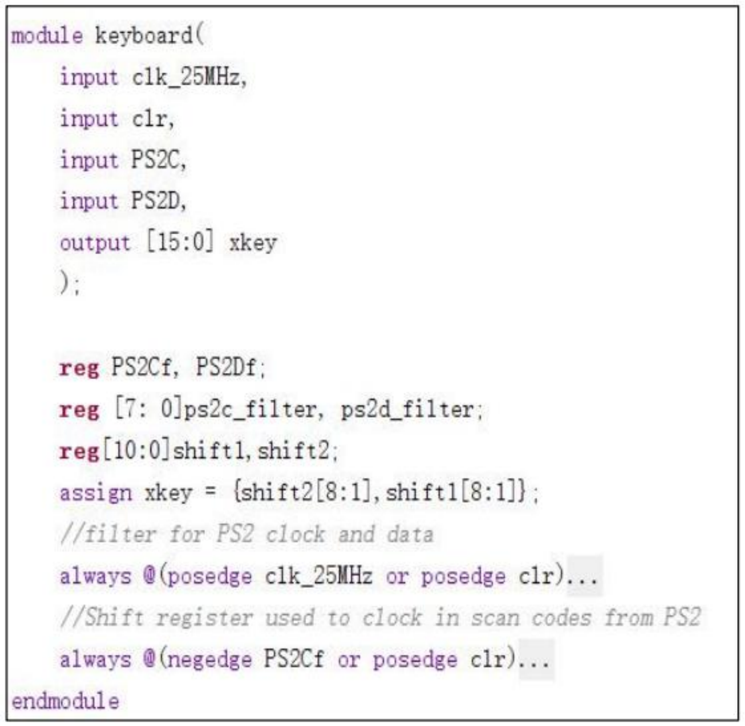
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | P | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | 0 |

shift1[10:0] →

| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | P | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | 0 |

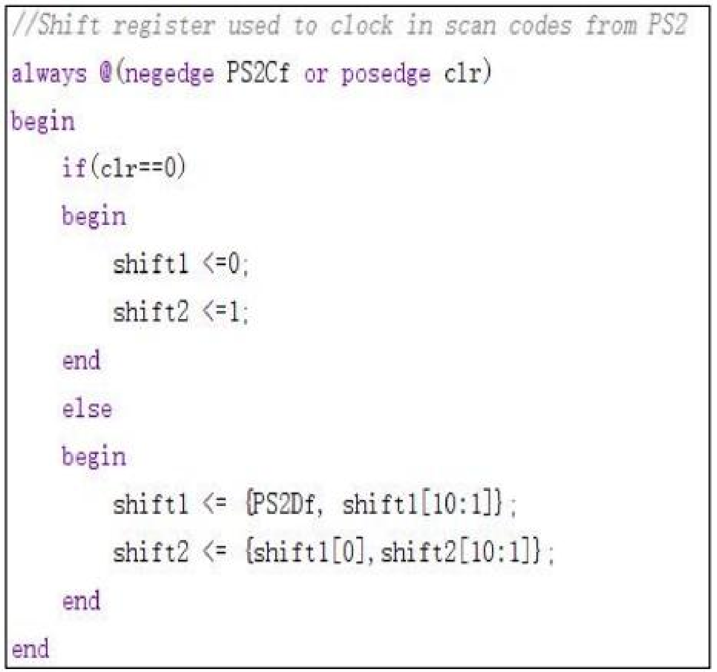
shift2[10:0]

# MODULES (1)

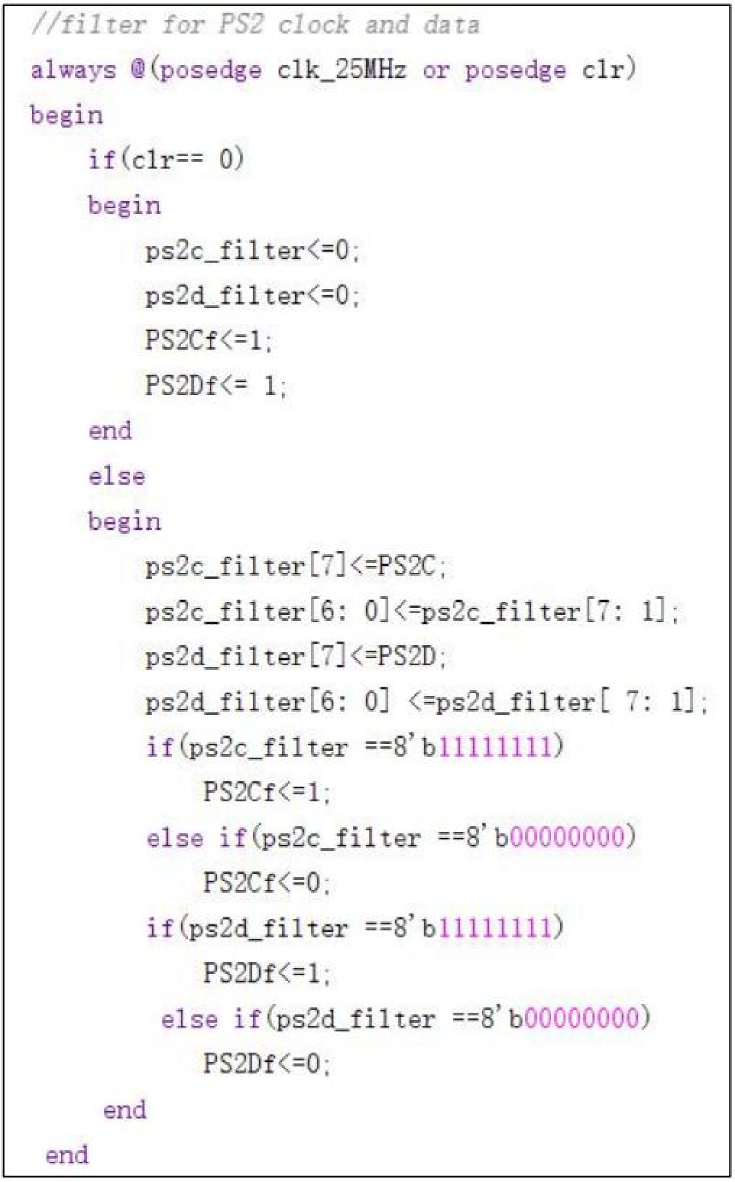


```
module keyboard(
    input clk_25MHz,
    input clr,
    input PS2C,
    input PS2D,
    output [15:0] xkey
    );

    reg PS2Cf, PS2Df;
    reg [7: 0]ps2c_filter, ps2d_filter;
    reg[10:0]shift1,shift2;
    assign xkey = {shift2[8:1],shift1[8:1]};
    //filter for PS2 clock and data
    always @(posedge clk_25MHz or posedge clr)...
    //Shift register used to clock in scan codes from PS2
    always @(negedge PS2Cf or posedge clr)...
endmodule
```

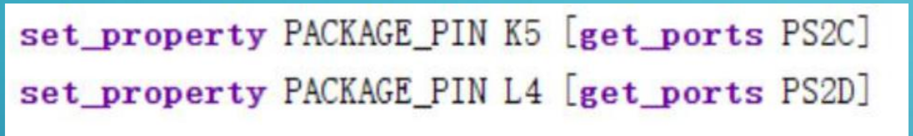


```
//Shift register used to clock in scan codes from PS2
always @(negedge PS2Cf or posedge clr)
begin
    if(clr==0)
    begin
        shift1 <=0;
        shift2 <=1;
    end
    else
    begin
        shift1 <= {PS2Df, shift1[10:1]};
        shift2 <= {shift1[0],shift2[10:1]};
    end
end
```

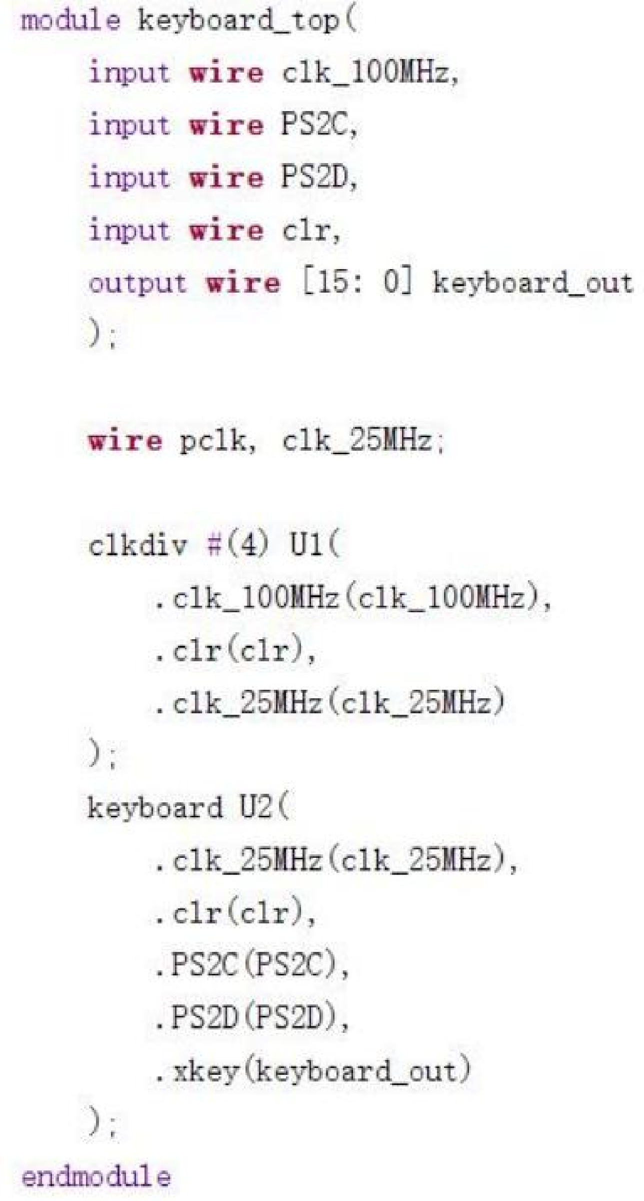


```
 //filter for PS2 clock and data
always @(posedge clk_25MHz or posedge clr)
begin
    if(clr== 0)
    begin
        ps2c_filter<=0;
        ps2d_filter<=0;
        PS2Cf<=1;
        PS2Df<= 1;
    end
    else
    begin
        ps2c_filter[7]<=PS2C;
        ps2c_filter[6: 0]<=ps2c_filter[7: 1];
        ps2d_filter[7]<=PS2D;
        ps2d_filter[6: 0] <=ps2d_filter[ 7: 1];
        if(ps2c_filter ==8'b11111111)
            PS2Cf<=1;
        else if(ps2c_filter ==8'b00000000)
            PS2Cf<=0;
        if(ps2d_filter ==8'b11111111)
            PS2Df<=1;
         else if(ps2d_filter ==8'b00000000)
            PS2Df<=0;
     end
 end
```

# MODULES (2)



```
set_property PACKAGE_PIN K5 [get_ports PS2C]
set_property PACKAGE_PIN L4 [get_ports PS2D]
```



```
module keyboard_top(
    input wire clk_100MHz,
    input wire PS2C,
    input wire PS2D,
    input wire clr,
    output wire [15: 0] keyboard_out
    );

    wire pclk, clk_25MHz;

    clkdiv #(4) U1(
        .clk_100MHz(clk_100MHz),
        .clr(clr),
        .clk_25MHz(clk_25MHz)
    );
    keyboard U2(
        .clk_25MHz(clk_25MHz),
        .clr(clr),
        .PS2C(PS2C),
        .PS2D(PS2D),
        .xkey(keyboard_out)
    );
endmodule
```