Basys3 键盘操作原理及参考代码

实验原理

BASYS3 开发板集成了一个 USB 鼠标键盘接口转 PS/2 接口的硬件模块 PIC24FJ128（如图 1），这样对于 FPGA 来说，外部连接的鼠标键盘就是 PS/2 接口的。PS/2 接口采用一种双向的同步串行协议。无论是主机-设备通信，还是设备-主机通信，时钟总是由设备产生，频率为 10~16.7KHZ。数据传输方式为每次一字节，用 11 位的帧来传送，它由起始位、8 位数据、奇偶检验位和停止位构成。

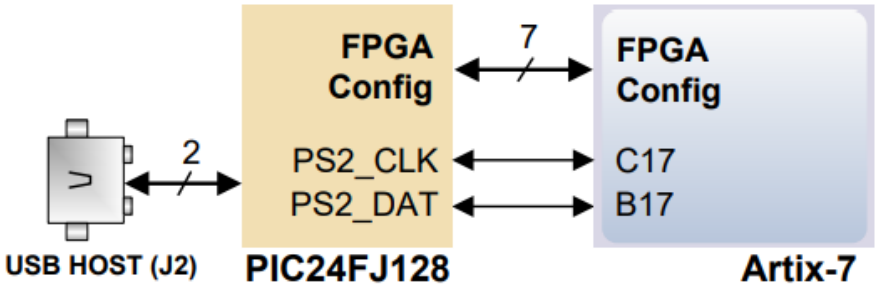

图 1 BASYS3 开发板集成的 PIC24FJ128 模块电路

图 2 给出了设备-主机通讯方式的时序图。由设备产生时钟和数据。在空闲时，时钟和数据线处于高电平。主机在主机时钟下降沿记录从设备发送过来的数据。

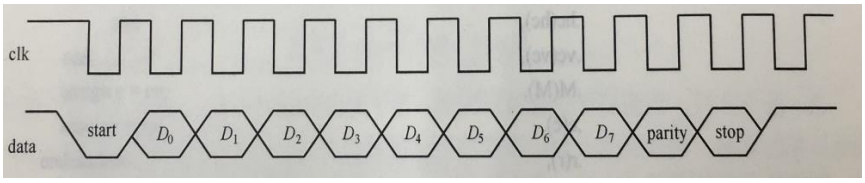

图 2 设备-主机通信方式的时序图

数据帧的对应位表示如下：

Start:起始位，总为‘0’（低电平）；

D0~D7：8 位数据位（地位在前，高位在后）；

Parity：奇偶校验位（为奇校验）；

Stop：停止位，总为‘1’（高电平）。

对于键盘来说，可通过扫描编码来识别按键输入。键盘的每个按键都有

不同的编码。每个按键的编码分为通码和断码。当按下键盘上的按键时，通码被发到 PS/2 接口；当释放案件时，断码被发送到 PS/2 接口。如下表 1 给出了键盘上所有按键的通码和断码。

表 1 PS2 键盘通码断码表

KEY	通码	断码	KEY	通码	断码	KEY	通码	断码
A	1C	F0 1C	9	46	F0 46	[	54	F0 54
B	32	F0 32	`	0E	F0 0E	INSERT	E0 70	E0 F0 70
C	21	F0 21	-	4E	F0 4E	HOME	E0 6C	E0 F0 6C
D	23	F0 23	=	55	F0 55	PG UP	E0 7D	E0 F0 7D
E	24	F0 24	\	5D	F0 5D	DELETE	E0 71	E0 F0 71
F	2B	F0 2B	BKSP	66	F0 66	END	E0 69	E0 F0 69
G	34	F0 34	SPACE	29	F0 29	PG DN	E0 7A	E0 F0 7A
H	33	F0 33	TAB	0D	F0 0D	U ARROW	E0 75	E0 F0 75
I	43	F0 43	CAPS	58	F0 58	L ARROW	E0 6B	E0 F0 6B
J	3B	F0 3B	L SHFT	12	F0 12	D ARROW	E0 72	E0 F0 72
K	42	F0 42	L CTRL	14	F0 14	R ARROW	E0 74	E0 F0 74
L	4B	F0 4B	L GUI	E0 1F	E0 F0 1F	NUM	77	F0 77
M	3A	F0 3A	L ALT	11	F0 11	KP /	E0 4A	E0 F0 4A
N	31	F0 31	R SHFT	59	F0 59	KP *	7C	F0 7C
O	44	F0 44	R CTRL	E0 14	E0 F0 14	KP -	7B	F0 7B
P	4D	F0 4D	R GUI	E0 27	E0 F0 27	KP +	79	F0 79
Q	15	F0 15	R ALT	E0 11	E0 F0 11	KP EN	E0 5A	E0 F0 5A
R	2D	F0 2D	APPS	E0 2F	E0 F0 2F	KP .	71	F0 71
S	1B	F0 1B	ENTER	5A	F0 5A	KP 0	70	F0 70
T	2C	F0 2C	ESC	76	F0 76	KP 1	69	F0 69
U	3C	F0 3C	F1	5	F0 05	KP 2	72	F0 72
V	2A	F0 2A	F2	6	F0 06	KP 3	7A	F0 7A
W	1D	F0 1D	F3	4	F0 04	KP 4	6B	F0 6B
X	22	F0 22	F4	0C	F0 0C	KP 5	73	F0 73
Y	35	F0 35	F5	3	F0 03	KP 6	74	F0 74
Z	1A	F0 1A	F6	0B	F0 0B	KP 7	6C	F0 6C
0	45	F0 45	F7	83	F0 83	KP 8	75	F0 75
1	16	F0 16	F8	0A	F0 0A	KP 9	7D	F0 7D
2	1E	F0 1E	F9	1	F0 01	]	58	F0 58
3	26	F0 26	F10	9	F0 09	;	4C	F0 4C
4	25	F0 25	F11	78	F0 78	’	52	F0 52
5	2E	F0 2E	F12	7	F0 07	,	41	F0 41
6	36	F0 36	PRNTSCRN	E0 12 E0 7C	E0 F0 7C E0 F0 12	.	49	F0 49
7	3D	F0 3D	SCROLL	7E	F0,7E	/	4A	F0 4A
8	3E	F0 3E	PAUSE	E1 14 77 E1 F0 14 F0 77	-NONE-			

主机接收到的时钟、数据信号常常含有噪声信号。为了准确读取输入数据，通常需要将输入信号过滤。本次实验可采用频率为 25MHz 的时钟，通过规定连续出现 8 个“1”为高电平，连续出现 8 个“0”为低电平来对键盘时钟和数据信号进行过滤，并将过滤后的数据信号送入移位寄存器中。

参考代码

```
module keyboard(
input wire clk_25M,
input wire clr,
input wire PS2C,
input wire PS2D,
output wire [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]};
    always@(posedge clk_25M or posedge clr)
    begin
        if(clr==1)
            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
        always@(negedge PS2Cf or posedge clr)
                begin
                        if(clr==1)
                                begin
                                        shift1<=0;
                                        shift2<=1;
                                end
                         else
                                begin
                                        shift1<={PS2Df,shift1[10:1]};
                                        shift2<={shift1[0],shift2[10:1]};
                                end
                end


endmodule

module keyboard_top(
input wire clk_100M,
input wire PS2C,
input wire PS2D,
input wire[3:3] btn,
output wire[6:0]a_to_g,
output wire[7:0]an
);
        wire clk_25M,clk_190,clr;
        wire [15:0]xkey;
        reg [18:0]count1;
        reg [2:0]count2;
        assign clr=btn[3];
        assign an[7:4]=4'b1111;
        always@(posedge clk_100M or posedge clr)
                        if(clr==1)
                          begin
                                count1=0;
                                count2=0;
                           end
                        else
                          begin
                                count1=count1+1;
                                count2=count2+1;
```

```
                end
    assign clk_190=count1[18];
    assign clk_25M=count2[2];
    keyboard U2 (clk_25M,clr,PS2C,PS2D,xkey);
    x7seg4 U3 (xkey,clk_190,clr,a_to_g,an);

endmodule

module x7seg4(
input wire[15:0]s,
input wire clk,clr,
output wire[6:0] a_to_g,
output reg[3:0] ans
    );
    reg [20:0]count;
    reg [3:0]digit;
    always@(posedge clk or posedge clr)
            if(clr==1)
                count=0;
            else
                count=count+1;
    always@(posedge clk)
        case(count[20:19])
        2'b00:
            begin
                digit=s[3:0];
                ans=4'b1110;
            end
        2'b01:
                begin
                    digit=s[7:4];
                    ans=4'b1101;
                end
        2'b10:
                    begin
                        digit=s[11:8];
                        ans=4'b1011;
                    end
         2'b11:
                        begin
                            digit=s[15:12];
                            ans=4'b0111;
                        end
```

```
        seg7 U4(.hex(digit),.segs(a_to_g));

endmodule

module seg7(
input wire[3:0]hex,
output reg[6:0]segs
        );
        always@(*)
        case(hex)
                // abc_defg
                4'h0: segs = 7'b000_0001;
                4'h1: segs = 7'b100_1111;
                4'h2: segs = 7'b001_0010;
                4'h3: segs = 7'b000_0110;
                4'h4: segs = 7'b100_1100;
                4'h5: segs = 7'b010_0100;
                4'h6: segs = 7'b010_0000;
                4'h7: segs = 7'b000_1111;
                4'h8: segs = 7'b000_0000;
                4'h9: segs = 7'b000_1100;
                4'ha: segs = 7'b000_1000;
                4'hb: segs = 7'b110_0000;
                4'hc: segs = 7'b111_0010;
                4'hd: segs = 7'b100_0010;
                4'he: segs = 7'b011_0000;
                4'hf: segs = 7'b011_1000;
                default:
                        segs = 7'b111_1111;
                endcase
endmodule
```