

Lab5 键盘接口实验

一、实验目的

- 1) 掌握Nexys A7-100T连接USB接口键盘的方法。
- 2) 掌握根据键盘扫描码转换成ASCII码的方法。
- 3) 掌握组合按键输入识别方法。
- 4) 了解开发板上USB鼠标接口连接方法。

二、实验环境

1. Vivado 开发环境
2. Xilinx A7-100T 实验板

三、实验原理

键盘和鼠标是常见的计算机输入设备，主要用于把数据、程序和命令等输入到计算机主机中。常用的键盘按键工作原理有机械式按键和电容式按键两种，键盘上按键数有 83 键、87 键、93 键、96 键、101 键、102 键、104 键等不同情况。常规键盘具有 CapsLock（字母大小写锁定）、NumLock（数字小键盘锁定）、ScrollLock（滚动锁定键）三个指示灯，标志键盘的当前状态。

键盘由一组排列成 $m \times n$ 矩阵方式的按键开关组成，通常定义为 8 行 \times 16 列，分成主键区、功能键区、控制键区、数字键区和状态指示区。键盘内部由控制器周期性扫描行、列，根据扫描信号线的结果，判断按键的位置，并把相应的键位码输入到计算机中去。鼠标是控制显示器屏幕上光标位置的输入设备。当鼠标与计算机连通时，显示屏上会出现一个光标。鼠标在桌面上移动时，其底部的圆球带动传感器把运动的方向和距离检测出来，送入计算机主机中，控制光标做相应的运动。移动光标对准屏幕上的命令或图形，按下鼠标左右键可输入相应命令。

USB HID（Human Interface Device）是 USB 接口的人机交互操作的设备，包括鼠标、键盘等，HID 设备除了传送数据给主机外，它也会从主机接收数据。PS/2 接口通过一个 6 芯接口插座与主机相接，引脚定

义分别是:电源 (VCC)、地 (GND)、串行时钟线 CLK 和串行数据线 DATA, 还有 2 根未用。PS/2 接口可以与 USB 接口互转。

实验板上的微控制器 PIC24 为 Nexys A7 提供 USB HID Host 功能。进入编程状态, 微控制器处于 USB HID Host 设备主机状态, 可以驱动连接到 Nexys A7 的 A 型 USB 接口上的键盘或鼠标。PIC24 将多个信号传输到 FPGA, 键盘的时钟端 (PS2_CLK) 与数据端 (PS2_DAT) 分别和 FPGA 的 F4 与 B2 引脚相连, 实现与键盘通信。实验时需要在缺省 XDC 约束文件中找到 USB HID (PS/2) 部分的两个引脚定义, 取消注释即可在设计文件中获取 PS2_CLK 及 PS2_DAT 的输入了。

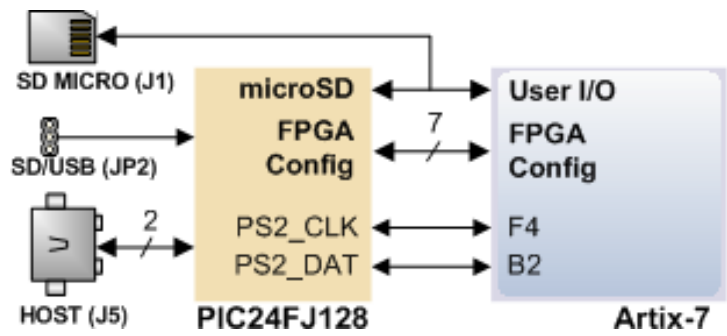


图 5.1 Nexys A7 实验板 PS/2 与 USB 转接口

微控制器 PIC24 在模拟 PS/2 设备主机与键盘或鼠标进行串行通信时, 使用 PS2_CLK 信号线来传输时钟, 指示数据线上的比特位在什么时候是有效; 使用 PS2_DAT 信号线来传输数据。主机与键盘、鼠标之间以每帧 11 位的格式串行传输数据, 其中包括起始位 0、数据位 (8 位/1 个字节, 低位在前)、奇校验位和停止位 1, 如图 5.2 所示。例如传输 “12h”, 传送顺序为: 起始位 (1'b0) + 八位数据位 (8'b010 0100 由低到高排列) + 奇校验位 (1'b1) + 停止位 (1'b1), 则从 PS2_DAT 端送出的数据顺序应该为 “0010 0100 011”。

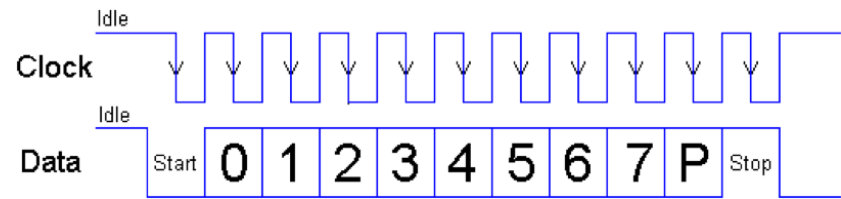


图 5.2 PS/2 传输 1 帧数据

键盘和鼠标的数据包组织方式不同, 键盘允许双向数据传输 (因此主机设备可以点亮键盘上的状态指示灯)。当 PS2_DAT 和 PS2_CLK 信号线都为高电平 (空闲) 时, 键盘可以给主机发送信号。如果主机将 PS2_CLK 信号置低, 键盘将准备接受主机发来的命令。PS/2 总线时序如所示图 5.3。每位数据都在时钟的下降沿有效, 使得数据位能够有足够的建立时间和保持时间来保持信号稳定。

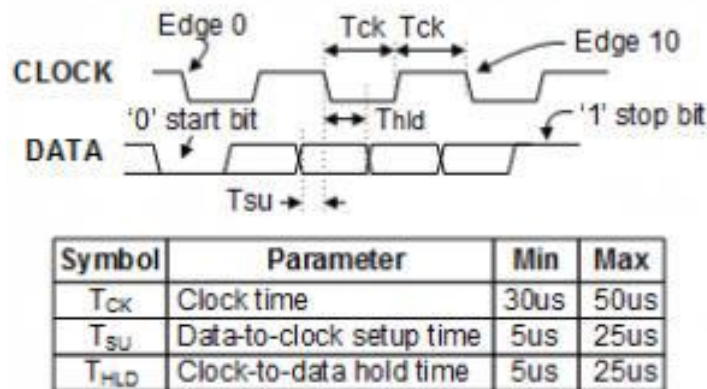


图 5.3 PS/2 设备和主机之间通信时序图

当键盘或鼠标连接到 USB HID Host 上时，FPGA 上将创建 PS/2 的键盘或鼠标接口。键盘或鼠标会向 Host 主机发送“自检通过”指令（0xAA），然后主机就可以向设备发出指令。由于键盘和鼠标使用相同的 PS/2 端口，可以通过读取设备 ID 指令（0xF2）来区分连接的设备类型。鼠标传输是单向的，因此，鼠标在“自检通过”指令后会立即发送其设备 ID（0x00），可以用来区分鼠标还是键盘。

现代键盘使用扫描码来传送按键信息，每个按键都分配了对应的行列扫描码。键盘内部单片机周期性扫描行、列，读回扫描信号线结果，判断是否有键按下；如果有则计算按键的位置以获得扫描码。当有键按下时，键盘控制器分两次将位置扫描码发送到键盘接口；按下时发送的扫描码称为通码（Make Code）；释放时发送的扫描码称为断码（Break Code）。通过扫描位置码可转换为 ASCII 码。送出“断码”的目的是为了能够识别组合键和上、下档键。

现代键盘共有三套的扫描码集，默认使用第二套扫描码，如图 5.4 所示。当按下某键时，键盘大约每隔 100 毫秒就向主机发送一次该键的通码；释放按键时，首先发送代码 0xF0，然后发送按键的通码。例如，按下字母“A”键，则 PS2_DAT 引脚将每隔 100 毫秒输出一“A”键的通码 0x1C；如释放“A”键，则输出的“A”键断码为 0xF01C，分两帧传输。

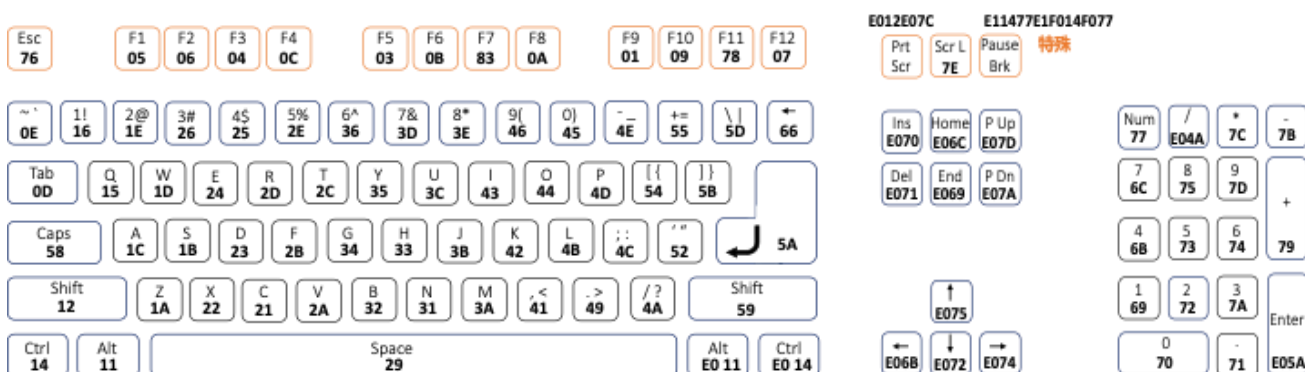


图 5.4 键盘扫描码

机械键盘的按键为机械弹性开关，当机械触点断开、闭合时，由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因而在闭合及断开的瞬间均伴随有一连串的抖动。抖动时间的长短由按键的机械特性决定，一般为 5ms~10ms。为了消除抖动带来的影响，在读取时需要进行按键消抖。

键盘接口通常需要具有消抖、去重、按键识别、键码产生四个基本功能。

如果多个键被同时按下，将按照按下顺序逐个输出按键的扫描码。例如，输入大写字母“A”，先按下左侧“Shift”键、再按下“A”键、释放“A”键、再释放左“Shift”键，则此过程送出的全部扫描码为：0x12, 0x1C, 0xF01C, 0xF012。键盘上有一些按键被称为扩展键，他们的扫描码以 0xE0 开始，码字较长，分多帧进行发送。键盘上各按键的扫描码是随机排列的，实际应用时需要将键盘扫描码转换为 ASCII 码。

发送数据时，键盘控制器生成 11 个时钟转换（20 至 30KHz），在时钟的下降沿上数据有效。

KeyBoardReceiver 模块是用来接收键盘送来的连续 4 个字节的键盘扫描码数据，并进行了消抖和去重处理。

```
module KeyBoardReceiver(
    output [31:0] keycodeout,           //接收到连续 4 个键盘扫描码
    output ready,                       //数据就绪标志位
    input clk,                          //系统时钟
    input kb_clk,                       //键盘 时钟信号
    input kb_data                       //键盘 串行数据
);
    wire kclkf, kdataf;
    reg [7:0] datacur;                   //当前扫描码
    reg [7:0] dataprev;                 //上一个扫描码
    reg [3:0] cnt;                      //收到串行位数
    reg [31:0] keycode;                 //扫描码
    reg flag;                           //接受 1 帧数据
    reg readyflag;
    // reg error;                       //错误标志位
    initial begin                      //初始化
        keycode[7:0] <= 8'b00000000;
        cnt <= 4'b0000;
    end
    debouncer debounce( .clk(clk), .I0(kb_clk), .I1(kb_data), .O0(kclkf), .O1(kdataf)); //消除按键抖动
    always@(negedge(kclkf))begin
        case(cnt)
```

```

        0: readyflag<=1'b0;                //开始位
        1:datacur[0]<=kdataf;
        2:datacur[1]<=kdataf;
        3:datacur[2]<=kdataf;
        4:datacur[3]<=kdataf;
        5:datacur[4]<=kdataf;
        6:datacur[5]<=kdataf;
        7:datacur[6]<=kdataf;
        8:datacur[7]<=kdataf;
        9:flag<=1'b1;                    //已接收 8 位有效数据
        10:flag<=1'b0;                   //结束位
    endcase
    if(cnt<=9) cnt<=cnt+1;
    else if(cnt==10) cnt<=0;
end
always @(posedge flag)begin
    if (dataprev!=datacur)begin          //去除重复按键数据
        keycode[31:24]<=keycode[23:16];
        keycode[23:16]<=keycode[15:8];
        keycode[15:8]<=dataprev;
        keycode[7:0]<=datacur;
        dataprev<=datacur;
        readyflag<=1'b1;                 //数据就绪标志位置 1
    end
end
assign keycodeout=keycode;
assign ready=readyflag;
endmodule
debouncer 模块用来消除按键抖动。
module debouncer(
    input clk,
    input I0,
    input I1,
    output reg O0,
    output reg O1
);
    reg [4:0]cnt0, cnt1;
    reg Iv0=0,Iv1=0;
    reg out0, out1;
    always@(posedge(clk))begin
        if (I0==Iv0)begin
            if (cnt0==19) O0<=I0;        //接收到 20 次相同数据

```

```

        else cnt0<=cnt0+1;
    end
    else begin
        cnt0<="00000";
        Iv0<=I0;
    end
    if (I1==Iv1)begin
        if (cnt1==19) O1<=I1; //接收到 20 次相同数据
        else cnt1<=cnt1+1;
    end
    else begin
        cnt1<="00000";
        Iv1<=I1;
    end
end
endmodule

```

为了验证键盘接收模块的功能，确认键盘基本通信正常，并测试键码传输的准确性，可将收到的每个键码用七段数码管显示出来。开发板上的 8 个七段数码管可以显示 KeyBoardReceiver 模块中收到的 4 个连续键码。例如按下并放开 “A” 键一次，七段数码管显示上应该显示 “1C F0 1C”。

KeyBoardTest 模块将收到的 4 个键盘码显示在 8 个数码管上。

```

module KeyBoardTest(
    output [6:0]SEG,
    output [7:0]AN,
    output DP,
    output ready,
    input CLK100MHZ,
    input PS2_CLK,
    input PS2_DATA
);
reg CLK50MHZ=0;
wire [31:0]keycode;
always @(posedge(CLK100MHZ)) begin
    CLK50MHZ<=~CLK50MHZ; end
KeyBoardReceiver keyboard_uut (
    .keycodeout(keycode[31:0]),
    .ready(ready),
    .clk(CLK50MHZ),
    .kb_clk(PS2_CLK),
    .kb_data(PS2_DATA));
seg7decimal sevenSeg ( //显示 4 个连续键盘码

```

```

        .x(keycode[31:0]),
        .clk(CLK100MHZ),
        .seg(SEG[6:0]),
        .an(AN[7:0]),
        .dp(DP) );
endmodule

```

配置约束文件 `KeyBoardReceiver.xdc`，然后在实验开发板上进行验证测试。

当接收到一个按键的扫描码后，然后接收到其断码，则表示该键按下然后弹起的单击动作，为了表示具体按键的含义，需要将该键的扫描码转换成对应的 ASCII 码。

`kbcode2ascii` 模块实现根据扫描码查找对应 ASCII 码的功能，如果没有可显示的 ASCII 码，则返回 00。`scancode.txt` 文件中保存了键盘扫描码和对应的 ASCII 码，Verilog 中使用 `readmemh` 系统命令读入数据文件，存储到 FPGA 的存储单元中；考虑到 8 位 ASCII 码，对应到一个 256 个字节数组 `kb_mem [255:0]`。

`kbcode2ascii` 模块参考代码如下：

```

module kbcode2ascii(
    output [7:0] ascii_code,
    input [7:0] kbcode
);
    reg [7:0] kb_mem[255:0];
    initial
    begin
        $readmemh("scancode.txt", kb_mem, 0, 255); //修改 scancode.txt 存放路径
    end
    assign    ascii_code = kb_mem[kbcode];
endmodule

```

连接键盘进行测试过程中，遇到问题有时不能确定键盘本身的故障还是接收模块有问题。因此可以先仿真键盘输入过程，验证键盘接收模块功能的正确性。

`KeyBoardSend` 模块模仿键盘控制器发送键盘扫描码的过程，以每帧 11 位的格式串行传输数据。输出对应键盘串行传输的两个接口信号：键盘时钟和数据位，模块参考代码设计如下：

```

module KeyBoardSend (
    output reg ps2_clk,
    output reg ps2_data
);
    parameter [31:0] kbd_clk_period = 60; //设置模拟键盘码传输时钟周期为 60ns。
    initial ps2_clk = 1'b1;
    task kbd_sendcode;
        input [7:0] code; // 8 位键盘码
    endtask
endmodule

```

```

integer i;
reg[10:0] send_buffer;
begin
    send_buffer[0] = 1'b0; // start bit
    send_buffer[8:1] = code; // code
    send_buffer[9] = ~(^code); // odd parity bit
    send_buffer[10] = 1'b1; // stop bit
    i = 0;
    while( i < 11) begin
        ps2_data = send_buffer[i]; // 传输数据位
        #(kbd_clk_period/2) ps2_clk = 1'b0; // 下降沿
        #(kbd_clk_period/2) ps2_clk = 1'b1;
        i = i + 1;
    end
end
endtask
endmodule

```

Verilog HDL 有类似 C 语言过程和函数的结构：任务 task 和函数 function，Function 用来描述组合逻辑，只能有一个返回值，function 的内部不能包含时序控制。Task 类似 procedure，执行一段 Verilog 代码，task 中可以有任意数量的输入和输出，task 也可以包含时序控制。

KeyBoardSend 模块中主要代码是任务 kbd_sendcode，用来控制键盘接口发送一个键盘码（通码或断码）；将 8 位键盘码转换排列次序输入，添加开始位(1'b0)、奇校验位和停止位(1'b1)。注意：程序设置键盘时钟周期是 kbd_clk_period=60ns，实际的键盘时钟频率没有这么快，这里是为了加速仿真。

键盘接收模块的仿真测试模块 KeyBoardReceiver_tb 中，分别将 KeyBoardSend 和 KeyBoardReceiver 实例化，并连接起来。在 initial 部分，可以直接调用 KeyBoardSend.kbd_sendcode 发送特定的扫描码，修改这部分代码，模拟实验需要的键盘按键序列。

KeyBoardReceiver_tb 模块参考代码设计如下：

```

module KeyBoardReceiver_tb ( );
    parameter [31:0] clock_period = 10; ////设置键盘控制器的时钟周期为 10ns。
    reg clk;
    wire [31:0] data;
    wire ready,overflow;
    wire kbd_clk, kbd_data;
    KeyBoardSend KB_Send(.ps2_clk(kbd_clk),.ps2_data(kbd_data));
    KeyBoardReceiver
    KeyBoardReceiver_uut(.keycodeout(data),.ready(ready),.clk(clk),.kb_clk(kbd_clk),.kb_data(kbd_data));
    initial begin /* clock driver */

```



```

clk = 0;
forever
    #(clock_period/2) clk = ~clk;
end

initial begin
    KB_Send.kbd_sendcode(8'h31); // press 'N'
#10 KB_Send.kbd_sendcode(8'hF0); // break code
#10 KB_Send.kbd_sendcode(8'h31); // release 'N'
#10 KB_Send.kbd_sendcode(8'h12); // 左侧 shiftj 键
#10 KB_Send.kbd_sendcode(8'h3B); // press 'J'
#10 KB_Send.kbd_sendcode(8'hF0); // break code
#10 KB_Send.kbd_sendcode(8'h3B); // release 'J'
#10 KB_Send.kbd_sendcode(8'h3C); // keep pressing 'U'
#10 KB_Send.kbd_sendcode(8'hF0); // break code
#10 KB_Send.kbd_sendcode(8'h3C); // release 'U'
#10
$stop;
end
endmodule

```

注意，下降沿检测时使用输入时钟 clk，该时钟频率应远高于 ps2_clk，可以取 1MHz 或 10MHz。

执行仿真测试模块，检测输出仿真波形，如图 5.5 所示。显示的是接收“N”键和断码的扫描码“31h”和“F0h”的情形。以接收“31h”为例，接口数据传送顺序为：起始位（1'b0）+ 八位数据位（由低到高）+ 奇校验位 + 停止位（1'b1），那么传送“31h”时从 PS2_DAT 端送出的数据顺序应该为“0100 0110 001”。

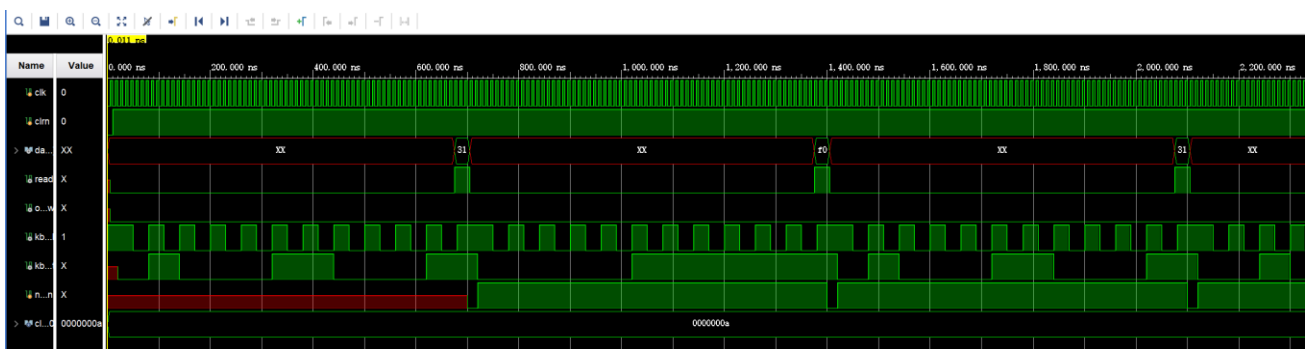


图 5.6 模拟按键 N 并释放的接收信号仿真图

四、实验内容

1、键盘接口实验

实验要求：

- 1、键盘上按键后，在 8 个七段数码管的高两位上显示按键的总次数，中间四位显示上一次按键和当前按键的扫描码，最低两位显示当前按键对应的 ASCII 码，如果按键没有对应的 ASCII 码，则显示 00。按住不放只算一次按键。例如：首次按下“A”键后，七段数码管显示“01 00 1C 61”。
- 2、支持输入大写字符，显示对应的 ASCII 码。
- 3、当按下 CapsLock、NumLock、Shift、Ctrl、Alt 等控制和组合按键时，通过 LED 指示灯高位进行标识。当前按键的 ASCII 码显示在 LED 低 8 位上。
- 4、在键盘发送端模块中建立 8 个字节的缓冲区，当未读取的扫描码超过缓冲区范围时，溢出标志为 1。验证演示键盘缓冲溢出现象。实验板上的 BTNC 按钮作为复位按钮。复位按钮有效时，缓冲区清零。

键盘接口模块端口定义如下：

```
module KeyboardSim(  
    input CLK100MHZ,    //系统时钟信号  
    input PS2_CLK,      //来自键盘的时钟信号  
    input PS2_DATA,     //来自键盘的串行数据位  
    input BTNC,         //Reset  
    output [6:0]SEG,  
    output [7:0]AN,  
    output [15:0] LED   //显示  
);  
  
// Add your code here  
  
endmodule
```

请根据上述描述，按照下列步骤完成实验。

- 1、 使用 Vivado 创建一个新工程。

- ## 2、鼠标接口实验

鼠标采用相对坐标系，鼠标向右移动会生成一个正数 X，向左移动则生成一个负数 X；鼠标向上移动生成一个正数 Y，向下移动则生成一个负数 Y。XS 和 YS 表示符号位，1 表示负数。X 和 Y 数字的大小表示鼠标移动速度，数字越大，鼠标移动速度越快。XY 和 YY 表示移动溢出位，“1”表示发生移动溢出。当鼠标连续地移动时，大约每隔 50ms 就发送 33 位数据到主机。L 和 R 表示是否按下左键和右键，“1”表示按钮按下。

```
module MouseReceiver(
    output [6:0]SEG,
    output [7:0]AN,
    output DP,
    output LeftButton, //左键按下
```

```
output RightButton, //左键按下
input CLK100MHZ,
input PS2_CLK,
input PS2_DATA
);

// Add your code here

endmodule
```

请根据上述描述，按照下列步骤完成实验。

- 1、 使用 Vivado 创建一个新工程。
- 2、 点击添加设计源码文件，加入 lab5.zip 里的 MouseReceiver.v 文件。
- 3、 点击添加约束文件，加入 lab5.zip 里的 MouseReceiver.xdc 文件。
- 4、 根据实验要求，完成源码文件的设计。
- 5、 对工程进行仿真测试，分析输入输出时序波形和控制台信息。
- 6、 仿真通过后，进行综合、实现并生成比特流文件。

生成比特流文件后，加载到实验开发板，进行调试验证，并记录验证过程。

五、思考题

- 1、如何在键盘接收模块 KeyBoardReceiver 中考虑串行数据收到干扰导致传输出错的问题。
- 2、什么是键盘的键位冲突？如何解决？