# 键盘

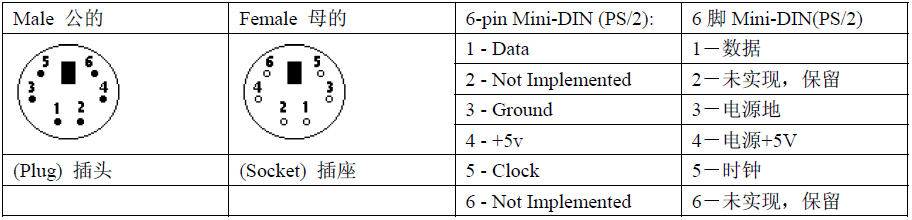
## 5.1 模块功能

设计基于FPGA上的MIPS多时钟CPU架构的键盘输入模块，通过Spartan-3E实验板上的PS/2接口，与标准键盘连接，获取键盘输入信息，并将可显字符以ASCII码形式、其他功能键以标准编码形式存入32位寄存器中，同时通过一条中断信号线与CPU相连，标示按键中断。

## 5.2 设计原理

### 5.2.1 PS/2协议

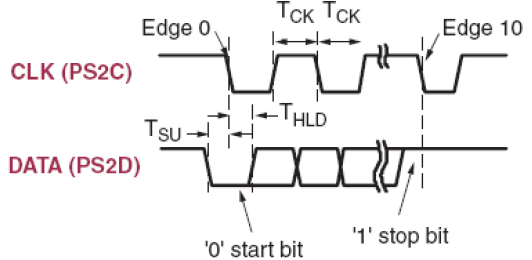
现在一般的PS/2接口为6脚的mini-DIN，其引脚定义为：



可见，在通信上有用的主要是1脚数据和5脚时钟信号。

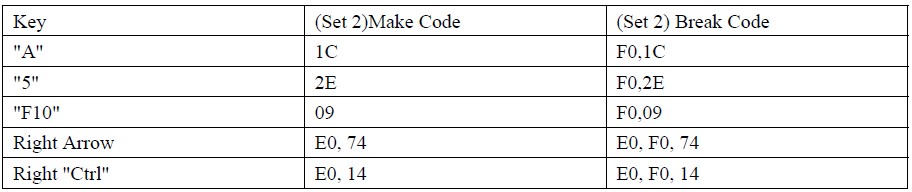
数据和时钟线都是集电极开路结构（正常保持高电平）。当时钟信号保持高电平时，设备可以正常发送数据给主机；当时钟信号保持低电平时，说明主机抑制了设备的信号发送，主机可以通过数据线发送信息给设备。

设备发送数据以一帧11位形式发送，各位数据值意义为：1个起始位总是为0（开始传输数据），8个数据位低位在前，1个校验位，1个停止位总是为1（恢复高电平）。每位数据在时钟下降沿被读取。该过程可以由下图表示：



### 5.2.2 键盘协议

键盘上的电路芯片已将完成各按键检测，将按键信息转为标准扫描码。当按下键时，发送通码；放开按键，发送断码。在最通用的第二套扫描码中，大部分键通码为一个字节，断码为0xF0加上该键的通码，其他扩展按键的通码、断码都比一般按键在最前面多一个字节的0xE0。有些键没有断码，比如”Pause/Break”键。几个按键的第二套扫描码如下：



当一个键一直按着，在一定的机打印时后，键盘以机打速率为频率，重复发生该按键的通码。当有多个键按着时，只有最后一个按键成为机打。有些键不会成为机打，比如”Caps Lock”键。

机打延时、机打速率、扫描码选择等键盘配置信息可以通过主机向键盘发生命令来设置。

## 5.3 具体实现

### 5.3.1 引脚定义

Spartan-3E实验板上的PS/2接口引脚PS2\_DATA定义在G13, PS2\_CLK定义在G14，相应的UCF定义为：

|  |
| --- |
| NET "ps2Clk" LOC = "G14" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8;  NET "ps2Data" LOC= "G13" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8; |

### 5.3.2 获取扫描码

PS2Ctrl模块负责获取单个扫描码，定义为：

|  |
| --- |
| module PS2Ctrl(clk, reset, ps2Clk, ps2Data, scanDone, scanCode);  input clk;  input reset;  input ps2Clk;  input ps2Data;  output scanDone;  output [7:0] scanCode;  reg scanDone; // get a scan code flag  reg [7:0] scanCode; // the scan code  reg [2:0] clkLowFilter;  reg [2:0] clkHighFilter;  reg filterPS2Clk; // the filtered PS2 clock  reg prePS2Clk;  reg [3:0] bitCount; // the buffer length  reg [9:0] buffer; // the data buffer |

由于键盘的时钟信号不稳定，加上传输过程中的环境干扰等因素，主机接受到的时钟信号存在波动，需要通过滤波来稳定时钟信号：

|  |
| --- |
| // Get the filter PS2 clock  always @(posedge clk or posedge reset) begin  if(reset) begin  clkLowFilter <= 3'b000;  clkHighFilter <= 3'b000;  filterPS2Clk <= 1'b0;  prePS2Clk <= 1'b0;  end  else begin  prePS2Clk <= filterPS2Clk;  if(ps2Clk) begin  clkLowFilter <= 3'b000;  clkHighFilter <= clkHighFilter + 3'b001;  if(clkHighFilter == 3'b011)  filterPS2Clk <= 1'b1;  end  else if(!ps2Clk) begin  clkHighFilter <= 3'b000;  clkLowFilter <= clkLowFilter + 3'b001;  if(clkLowFilter == 3'b011)  filterPS2Clk <= 1'b0;  end  end  end |

filterPSClk便为滤波后的时钟信号。

使用一个一位寄存器prePS2Clk，保持前一系统时钟的键盘时钟值，与当前键盘时钟值进行比较。若键盘时钟处于下降沿，开始读取键盘发送的数据。当读入8位数据后，再读取一位奇校验码。将8位有效位保存在8位寄存器中，并将scanDone值置位：

|  |
| --- |
| // scan the code  always @(posedge clk or posedge reset) begin  if(reset) begin  bitCount <= 4'b0000;  buffer <= 10'b00\_0000\_0000;  end  else if(prePS2Clk == 1'b1 && filterPS2Clk == 1'b0)  if(bitCount == 4'b0000) begin  if(!ps2Data)  bitCount <= 4'b0001;  end  else begin  bitCount <= bitCount + 4'b0001;  buffer <= {ps2Data, buffer[8:1]};    if(bitCount == 4'b1001) begin  // Get a scan code successfully  scanDone <= 1'b1;  scanCode <= buffer[8:1];  end  else if(bitCount == 4'b1010) begin  // reset the scan state  scanDone <= 1'b0;  scanCode <= 8'b0000\_0000;    bitCount <= 4'b0000;  end  end  end |

### 5.3.3 转换扫描码

获得各按键的扫描码后，再通过查询第二套扫描码表，将相应按键信息转换成ASCII码或其他标准码。模块定义为：

|  |
| --- |
| module PS2Decode(clk, reset, scanDone, scanCode, dataReady, data);  input clk;  input reset;  input scanDone;  input [7:0] scanCode;  output dataReady;  output [31:0] data;  reg dataReady;  reg [31:0] data;  reg capsLock; // the caps lock  reg shift; // shift case flag  reg breakCode; // break code flag  wire upper; // the upper-case flag  assign upper = capsLock ^ shift; |

其中，capsLock寄存器标示当前Caps Lock状态，shift寄存器标示shift是否按下。两个寄存器值合成便可标示当前字符的大小写状态。

|  |
| --- |
| always @(posedge clk or posedge reset) begin  if(reset == 1) begin  dataReady <= 1'b0;  data <= 32'h0000\_0000;    breakCode <= 1'b0;  shift <= 1'b0;  end    // pull down the dataReady signal  else if(scanDone == 1'b0)  dataReady <= 1'b0;    // get a new scan code  else if(preScan == 1'b0 && scanDone == 1'b1)  // break code  if(breakCode) begin  // get the second scan code of a break code  if((scanCode == 8'h12) || (scanCode == 8'h59)) begin  shift <= 1'b0;  end  breakCode <= 1'b0;  end  // get the first scan code of a break code  else if(scanCode == 8'hF0)  breakCode <= 1'b1;    // make code  else  begin  case (scanCode)  // control characters  8'h12, 8'h59: begin // shift  data <= 32'h0000\_0000;  shift <= 1'b1;  end  8'h58: begin // caps lock  data <= 32'h0000\_0000;  capsLock <= ~capsLock;  end  8'h66: // Backspace  data <= 32'h0000\_0008;  ...  default:  data <= 32'h0000\_0000;  endcase    dataReady <= 1'b1;  end  end |

当转换出一个可显字符或功能键时，将该编码存入32位寄存器中，同时置位按键中断，由CPU处理。

### 5.3.4 模块封装

将以上两个功能模块封装，组合成一个完整的键盘输入模块，方便功能调用。

|  |
| --- |
| module PS2Top(clk, ps2Clk, ps2Data, reset, commond, dataReady, data);  input clk;  input ps2Clk;  input ps2Data;  input reset;  input [31:0] commond;  output dataReady;  output [31:0] data;  wire scanDone; // the scanned flag  wire [7:0] scanCode; // the scan code  wire dataReady; // the data ready flag  wire [31:0] data; // the decode data  PS2Ctrl pc(clk, reset, ps2Clk, ps2Data, scanDone, scanCode);  PS2Decode pd(clk, reset, scanDone, scanCode, dataReady, data);  endmodule |

## 5.4 模块测试

模块的输出数据，通过显示测试模块，显示在Spartan-3E实验板上的8盏LED灯上。测试结果如下：

|  |  |  |
| --- | --- | --- |
| 按键 | 结果 | 分析 |
| a |  | 小写’a’，ASCII码0x61 |
| Shift + a |  | 大写’a’，ASCII码0x41 |
| Caps lock下，a |  | 大写’a’，ASCII码0x41 |
| Caps lock下，shift + a |  | 小写’a’，ASCII码0x61 |
| Shift + . |  | 字符’>’，ASCII码0x3E |

经测试，LED显示值正确地反映了当前按键值。本键盘模块能实现按键的检测，并能提供给CPU处理。