课程设计—人机输入显示系统

PS/2键盘输入，LCD显示

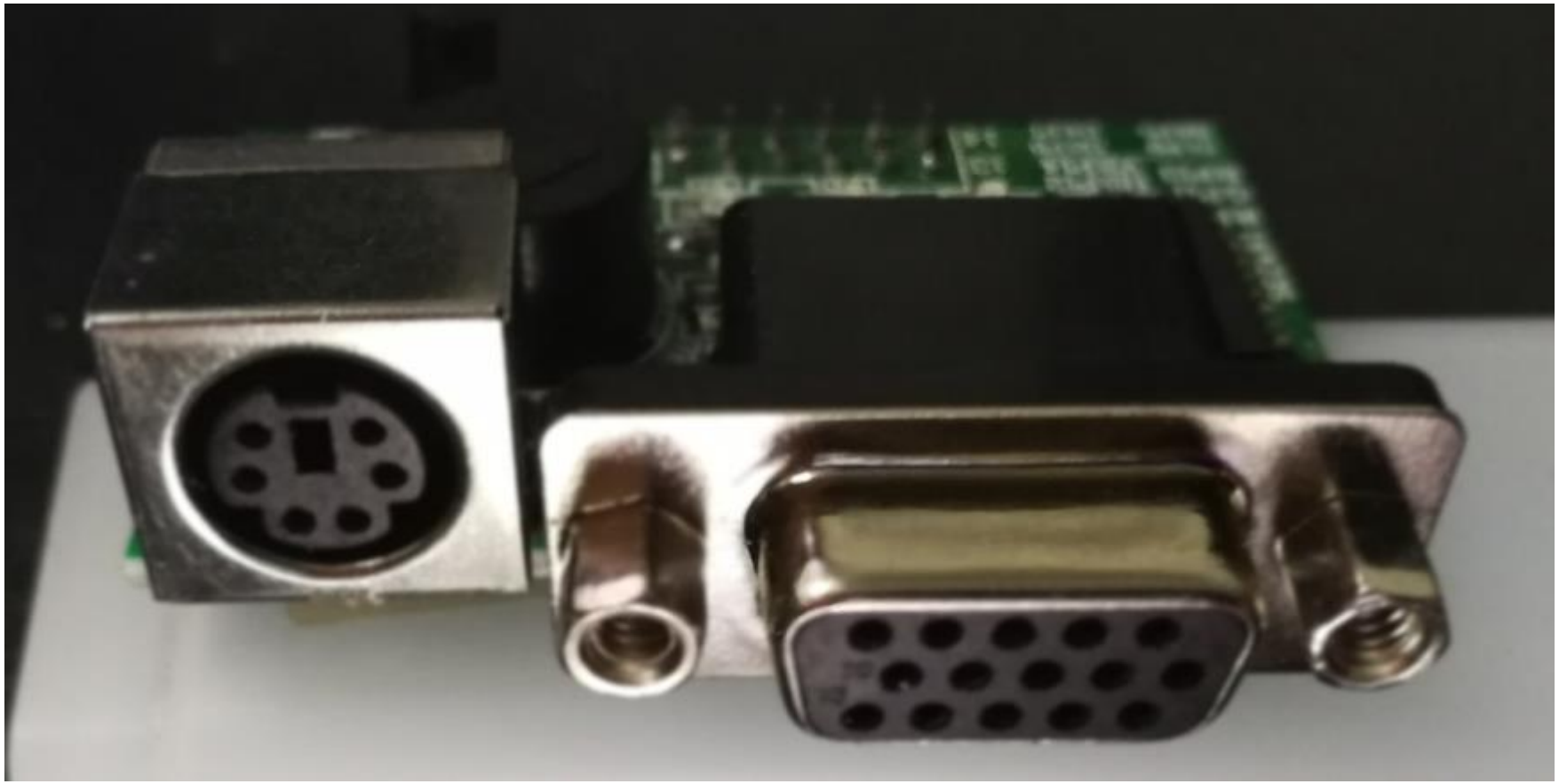
功能要求：

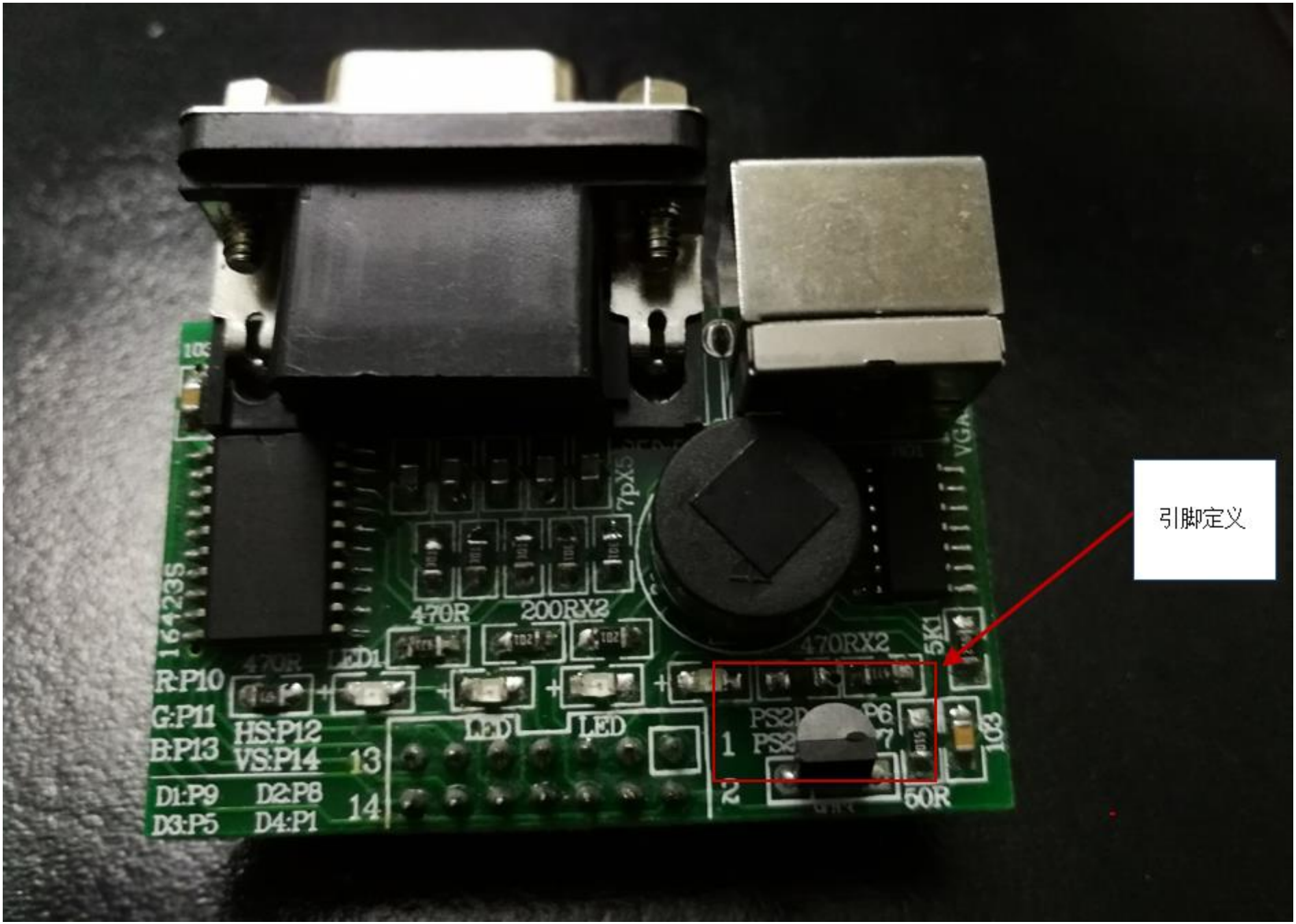
- 实现PS/2键盘的输入
- 将键盘输入的内容显示在液晶显示器上
- 键盘的功能实现自定，例如是否考虑功能键，是否使用连击等
- 显示格式自定，例如是否使用多行，是否使用大小写等

实现提示:

- 方法是将PS/2模块与LCD模块连接
- 问题是数据如何传递？
- 利用LCD数据存储器？
- 使用RAM宏模块？

通过扩展模块连接键盘




引脚定义
R:P10
G:P11
B:P13
HS:P12
VS:P14
D1:P9
D2:P8
D3:P5
D4:P1
470R
200RX2
470RX2
LED1
LED
50R
5K1
103
13
14

引脚配置

- PS2DATA → P6 → 芯片引脚
- PS2CLK → P7 →芯片引脚

- 注意
- 两个信号都有反相缓冲，
- 程序设计时要按原逻辑值非处理

要求：

- 希望独立完成（无论实现多少功能）
- 在规定时间规定地点完成（要考勤）
- 功夫花在实验室外（自觉）

要求

- 提交课程设计报告

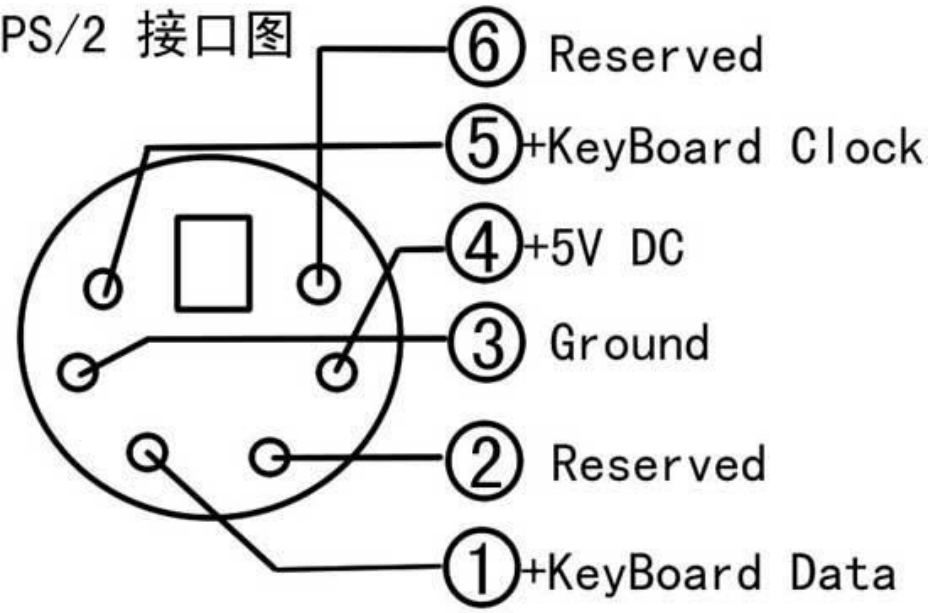
end

PS2键盘协议

- ps2协议是现在大多数鼠标，键盘与PC通讯的标准协议
- 鼠标的通讯更为简单些，只是传送的数据内容不一样而已。

一.电气特性：信号采用5V－TTL电平

从插头看引脚：



1	DATA	KeyData
2	n/c	Notconnected
3	GND	Gnd
4	VCC	Power, +5VDC
5	CLK	Clock
6	n/c	Notconnected

二.数据格式

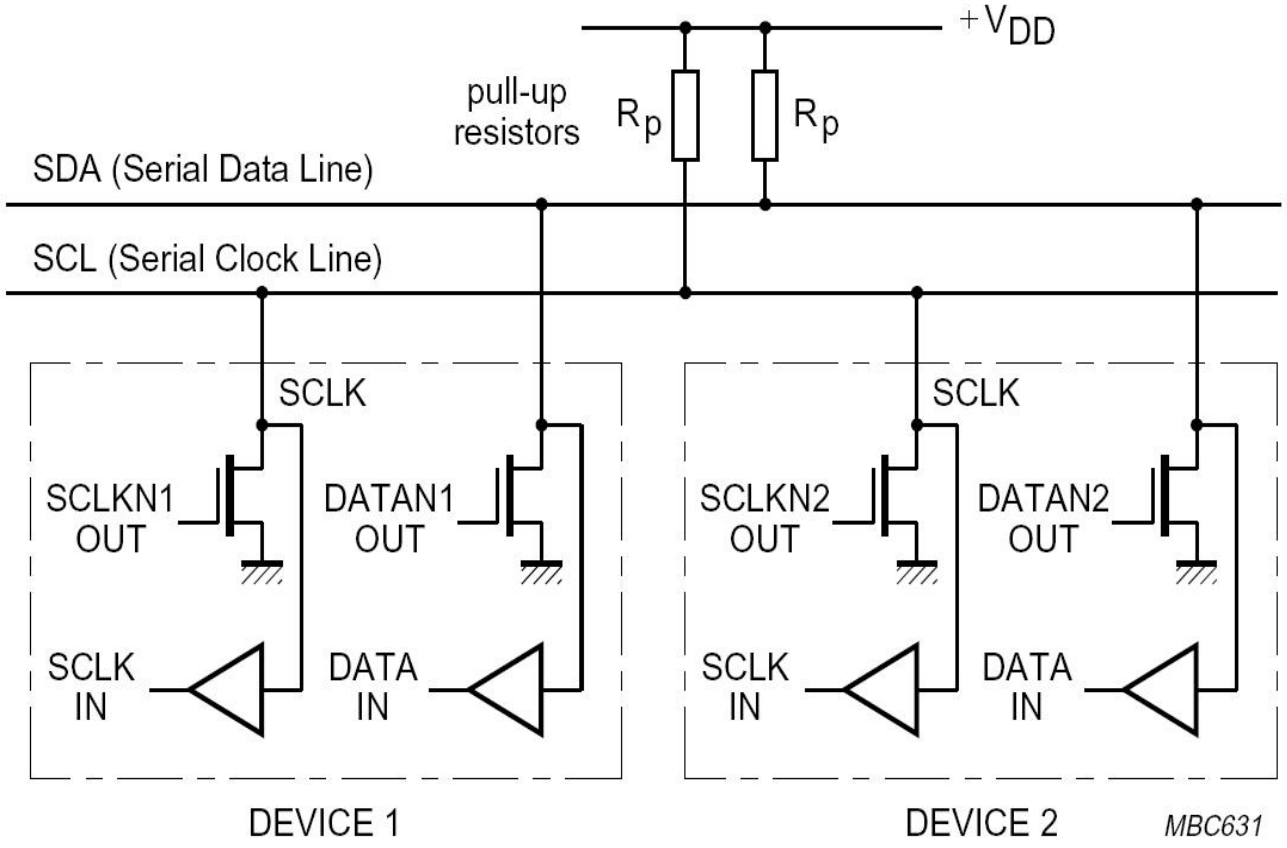
1个起始位	总是逻辑0
8个数据位	（LSB）低位在前
1个奇偶校验位	奇校验
1个停止位	总是逻辑1
1个应答位	仅用在主机对设备的通讯中

如果数据位中 1 的个数为偶数，校验位就为 1；

如果数据位中 1 的个数为奇数，校验位就为 0；

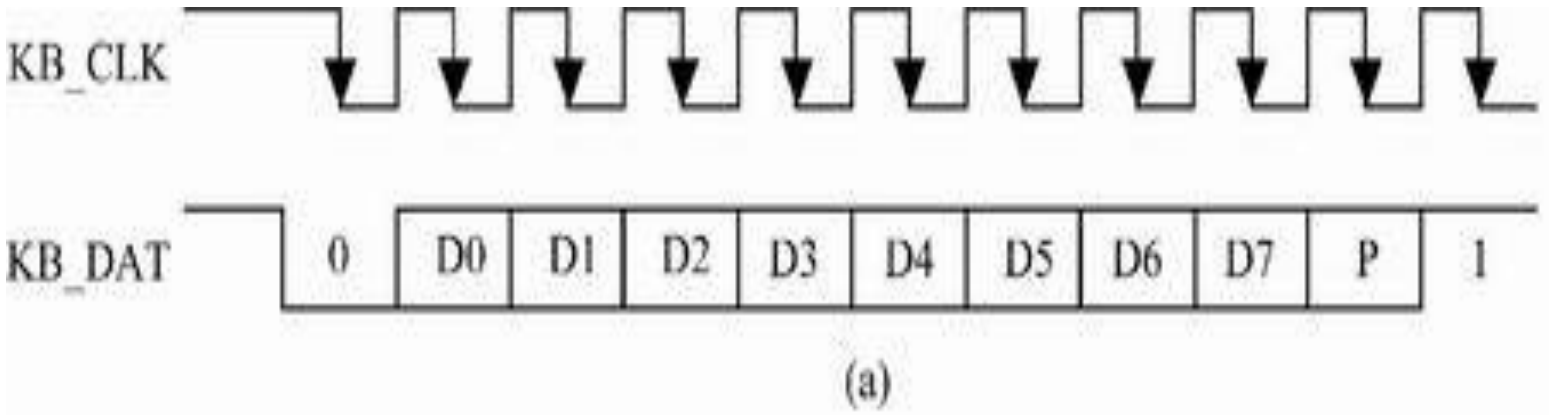
总之，数据位中 1 的个数加上校验位中 1 的个数总为奇数，因此称为奇校验。

ps2设备的clock和data都是集电极开路的，Open Collector, OC 平时都是高电平。

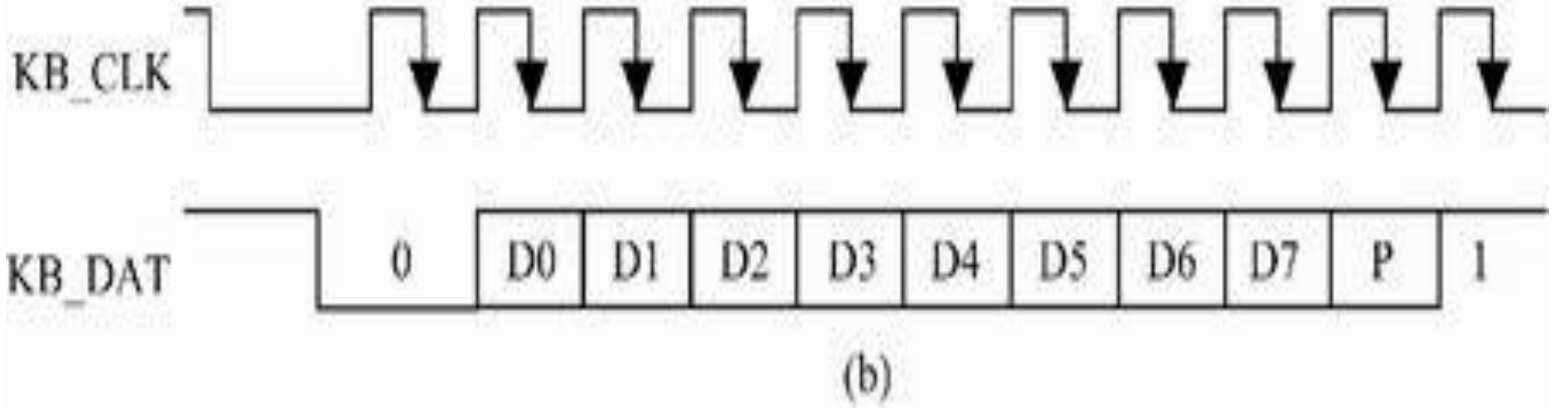


- 当ps2设备等待发送数据时，它首先检查clock是否为高。
  - 如果clock为低，则认为PC抑制了通讯，此时它缓冲数据直到获得总线的控制权。
  - 如果clock为高电平，ps2则开始向PC发送数据。
- 一般都是由ps2设备产生时钟信号。发送按帧格式。数据位在clock为高电平时准备好，在clock下降沿被PC读入。
- 数据从键盘/鼠标发送到主机或从主机发送到键盘/鼠标，时钟都是PS2设备产生。

- 主机对时钟控制有优先权，即主机想发送控制指令给PS2设备时，可以拉低时钟线至少100μS，然后再下拉数据线，最后释放时钟线为高。
- PS2设备的时钟线和数据线都是集电极开路的，容易实现拉低电平。
- PC在时钟的下降沿读取数据。


KB_CLK
KB_DAT
0
D0
D1
D2
D3
D4
D5
D6
D7
P
1

(a)


KB_CLK
KB_DAT
0
D0
D1
D2
D3
D4
D5
D6
D7
P
1

(b)

从ＰＳ／２向ＰＣ机发送一个字节可按照下面的步骤进行：

1 检测时钟线电平，如果时钟线为低，则延时５０μｓ；

2 检测判断时钟信号是否为高，为高，则向下执行，为低，则转到(１)；

3 检测数据线是否为高，如果为高则继续执行，如果为低，则放弃发送（此时ＰＣ机在向ＰＳ／２设备发送数据，所以ＰＳ／２设备要转移到接收程序处接收数据）；

4 延时２０μｓ（如果此时正在发送起始位，则应延时４０μｓ）；

5 输出起始位（０）到数据线上。这里要注意的是：在送出每一位后都要检测时钟线，以确保ＰＣ机没有抑制ＰＳ／２设备，如果有则中止发送；

6 输出８个数据位到数据线上；

7 输出校验位；

8 输出停止位（１）；

9 延时３０μｓ（如果在发送停止位时释放时钟信号则应延时５０μｓ）

通过以下步骤可发送单个位：

- （１）准备数据位（将需要发送的数据位放到数据线上）
- （２）延时２０μｓ；
- （３）把时钟线拉低；
- （４）延时４０μｓ；
- （５）释放时钟线；
- （６）延时２０μｓ。

ＰＳ／２设备从ＰＣ机接收一个字节

- 由于ＰＳ／２设备能提供串行同步时钟，因此，如果ＰＣ机发送数据，则ＰＣ机要先把时钟线和数据线置为请求发送的状态。
- ＰＣ机通过下拉时钟线大于１００μs来抑制通讯，并且通过下拉数据线发出请求发送数据的信号，然后释放时钟。
- 当ＰＳ／２设备检测到需要接收的数据时，它会产生时钟信号并记录下面８个数据位和一个停止位。
- 主机此时在时钟线变为低时准备数据到数据线，并在时钟上升沿锁存数据
- 而ＰＳ／２设备则要配合ＰＣ机才能读到准确的数据。

具体连接步骤如下：

（１）等待时钟线为高电平。

（２）判断数据线是否为低，为高则错误退出，否则继续执行。

（３）读地址线上的数据内容，共８个ｂｉｔ，每读完一个位，都应检测时钟线是否被PC机拉低，如果被拉低则要中止接收。

（４）读地址线上的校验位内容，１个ｂｉｔ。

（５）读停止位。

（６）如果数据线上为０（即还是低电平），PS/２设备继续产生时钟，直到接收到１且产生出错信号为止（因为停止位是１，如果PS/２设备没有读到停止位，则表明此次传输出错）。

（７输出应答位。

（８）检测奇偶校验位，如果校验失败，则产生错误信号以表明此次传输出现错误。

（９）延时４５μｓ，以便PC机进行下一次传输。

读数据线的步骤如下：

- （１）延时２０μｓ；
- （２）把时钟线拉低
- （３）延时４０μｓ
- （４）释放时钟线
- （５）延时２０μｓ
- （６）读数据线。

下面的步骤可用于发出应答位：

（１）延时１５μｓ；

（２）把数据线拉低；

（３）延时５μｓ；

（４）把时钟线拉低；

（５）延时４０μｓ；

（６）释放时钟线；

（７）延时５μｓ；

（８）释放数据线。

四.键盘返回值

- 键盘的处理器如果发现有键被按下或释放将发送扫描码的信息包到计算机。
- 扫描码有两种不同的类型：通码和断码。
- 当一个键被按下就发送通码，当一个键被释放就发送断码。
- 每个按键被分配了唯一的通码和断码。
- 这样主机通过查找唯一的扫描码就可以测定是哪个按键。
- 每个键一整套的通断码组成了扫描码集。
- 有三套标准的扫描码集：分别是第一套，第二套和第三套。
- 所有现代的键盘默认使用第二套扫描码。

第二套通码

- 虽然多数第二套通码都只有一个字节宽，但也有少数扩展按键的通码是两字节或四字节宽。这类的通码第一个字节总是为E0。
- 正如键按下通码就被发往计算机一样，只要键一释放断码就会被发送
- 每个键都有它自己唯一的通码和断码。
- 幸运的是你不用总是通过查表来找出按键的断码。在通码和断码之间存在着必然的联系。
- 多数第二套断码有两字节长。它们的第一个字节是F0，第二个字节是这个键的通码。
- 扩展按键的断码通常有三个字节，它们前两个字节是E0h,F0h，最后一个字节是这个按键通码的最后一个字节。

例如：下面列出了几个按键的第二套通码和断码

No.	KEY	通码(第二套)	断码(第二套)
1	"A"	1C	F01C
2	"5"	2E	F02E
3	"F10"	09	F009
4	RightArrow	E074	E0F074
5	Right"Ctrl"	E014	E0F014

一个键盘发送字符G值的例子:

- 因为这是一个大写字母，需要发生这样的事件次序：
- 按下Shift键-按下G键-释放G键-释放Shift键。
- 与这些时间相关的扫描码如下：Shift键的通码12h，G键的通码34h，G键的断码F0h34h，Shift键的断码F0h12h。因此发送到你的计算机的数据应该是：
- 12h 34h F0h 34h F0h 12h

第

- 101/102和104键的键盘:

KEY	通码	断码		KEY	通码	断码		KEY	通码	断码
A	1C	F01C		9	46	F046		[	54	F054
B	32	F032		`	0E	F00E		INSERT	E070	E0F070
C	21	F021		-	4E	F04E		HOME	E06C	E0F06C
D	23	F023		=	55	F055		PGUP	E07D	E0F07D
E	24	F024		\	5D	F05D		DELETE	E071	E0F071
F	2B	F02B		BKSP	66	F066		END	E069	E0F069
G	34	F034		SPACE	29	F029		PGDN	E07A	E0F07A
H	33	F033		TAB	0D	F00D		UARROW	E075	E0F075
I	43	F043		CAPS	58	F058		LARROW	E06B	E0F06B
J	3B	F03B		LSHFT	12	F012		DARROW	E072	E0F072
K	42	F042		LCTRL	14	F014		RARROW	E074	E0F074
L	4B	F04B		LGUI	E01F	E0F01F		NUM	77	F077
M	3A	F03A		LALT	11	F011		KP/	E04A	E0F04A
N	31	F031		RSHFT	59	F059		KP*	7C	F07C
O	44	F044		RCTRL	E014	E0F014		KP-	7B	F07B
P	4D	F04D		RGUI	E027	E0F027		KP+	79	F079
Q	15	F015		RALT	E011	E0F011		KPEN	E05A	E0F05A
R	2D	F02D		APPS	E02F	E0F02F		KP	71	F071
S	1B	F01B		ENTER	5A	F05A		KP0	70	F070
T	2C	F02C		ESC	76	F076		KP1	69	F069
U	3C	F03C		F1	05	F005		KP2	72	F072
V	2A	F02A		F2	06	F006		KP3	7A	F07A
W	1D	F01D		F3	04	F004		KP4	6B	F06B
X	22	F022		F4	0C	F00C		KP5	73	F073
Y	35	F035		F5	03	F003		KP6	74	F074
Z	1A	F01A		F6	0B	F00B		KP7	6C	F06C
0	45	F045		F7	83	F083		KP8	75	F075
1	16	F016		F8	0A	F00A		KP9	7D	F07D
2	1E	F01E		F9	01	F001		]	58	F058
3	26	F026		F10	09	F009		;	4C	F04C
4	25	F025		F11	78	F078		'	52	F052
5	2E	F02E		F12	07	F007		,	41	F041
6	36	F036		PRNT SCRN	E012 E07C	E0F0 7CE0 F012		.	49	F049
7	3D	F03D		SCROLL	7E	F0,7E		/	4A	F04A
8	3E	F03E		PAUSE	E11477 E1F014 F077	-NONE-				

ACPI扫描码:

KEY	通码	断码
Power	E0,37	E0,F0,37
Sleep	E0,3F	E0,F0,3F
Wake	E0,5E	E0,F0,5E

WINDOWS 多媒体扫描码:

KEY	通码	断码
NextTrack	E0,4D	E0,F0,4D
PreviousTrack	E0,15	E0,F0,15
Stop	E0,3B	E0,F0,3B
Play/Pause	E0,34	E0,F0,34
Mute	E0,23	E0,F0,23
VolumeUp	E0,32	E0,F0,32
VolumeDown	E0,21	E0,F0,21
MediaSelect	E0,50	E0,F0,50
E-Mail	E0,48	E0,F0,48
Calculator	E0,2B	E0,F0,2b
MyComputer	E0,40	E0,F0,40
WWWSearch	E0,10	E0,F0,10
WWWHome	E0,3A	E0,F0,3a
WWWBack	E0,38	E0,F0,38
WWWForward	E0,30	E0,F0,20
WWWStop	E0,28	E0,F0,28
WWWRefresh	E0,20	E0,F0,20
WWWFavorites	E0,18	E0,F0,18

六、FPGA实现：

1）完整实现PS/2协议

2）部分实现PS/2协议

教材中的示例：p242

- 部分解决方案
- 仅考虑键盘释放

```
always@(posedge clk)
    begin
        kbclkreg <=kb_clk;
        kbclkfall <= kbclkreg & (~kb_clk);
   end
always@(posedge clk)
    begin
        if( kbclkfall == 1'b1 & datacoming == 1'b0 & kb_data == 1'b0)
            begin
                datacoming <= 1'b1;  cnt <= 4'b0000; parity <= 1'b0;
            end
```

```
        else  if( kbclkfall == 1'b1 & datacoming == 1'b1)
            begin
                if( cnt ==9 )
                    begin
                        if( kb_data ==1'b1)
                            begin
                                datacoming <= 1'b0; dataerror <= 1'b0;
                            end
                        else   dataerror <=1'b1;
                        cnt <= cnt +1;
                    end
```

```
                else if ( cnt == 8)
                    begin
                        if( kb_data == parity ) dataerror <= 1’b0;
                        else   dataerror <= 1’b1;
                        cnt <= cnt+1;
                    end
                else
                    begin
                        shiftdata <= {kb_data,
```

```
always @ (posedge clk)
    begin
        if( cnt == 10)
            begin
                if( shiftdata == 8'b11110000)
                    begin     isfo <= 1'b1;   end
                else if( shiftdata != 8'b1110_0000)
                    if( isfo == 1'b1)
                        begin   keyup <=1'b1; keycode <= shiftdata;  end
                    else   begin keydown <=1'b1; keycode <= shiftdata;  end
            end
    end
```

Fin !