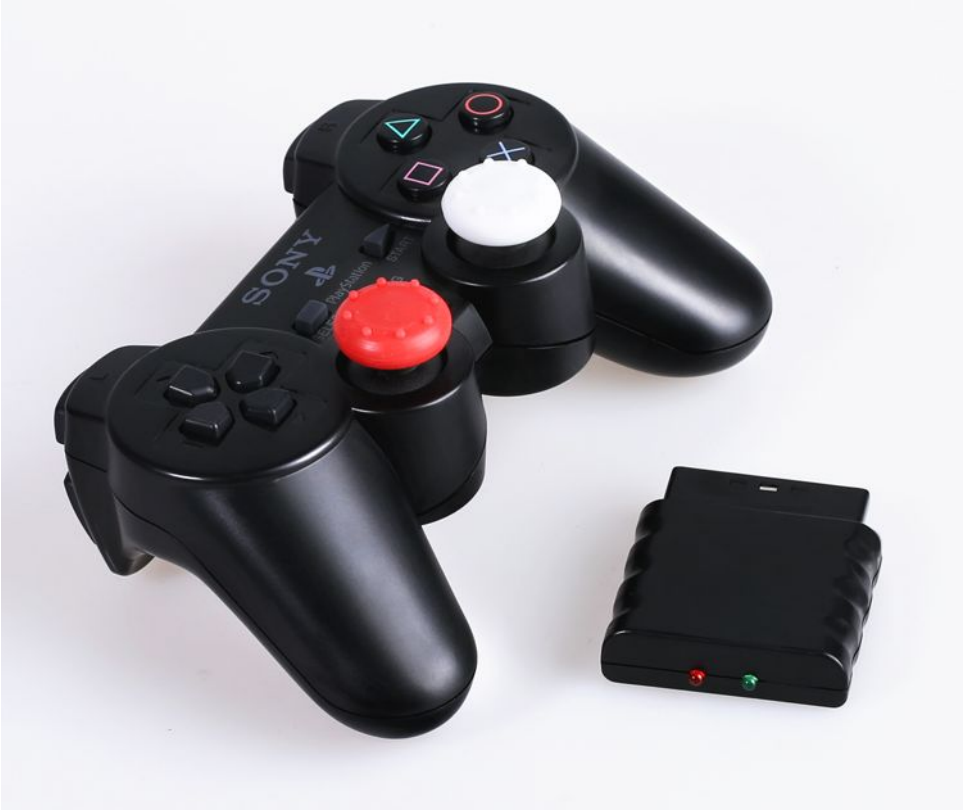
Touch Your Future!
www.yfrobot.com

PS2 解码通讯使用手册 V1.3



YFRobot 荣誉出品

2015 年 7 月 12 日

版 权 声 明

目　录

1 ps2 手柄介绍

ps2 手柄是索尼的 PlayStation2 游戏机的遥控手柄。索尼的 psx 系列游戏主机在全球很是畅销。不知什么时候便有人打起 ps2 手柄的主意，破解了通讯协议，使得手柄可以接在其他器件上遥控使用，比如遥控我们熟悉的机器人。突出的特点是现在这款手柄性价比极高。按键丰富，方便扩展到其它应用中。

ps2 手柄介绍：

ps2 由手柄与接收器两部分组成，手柄主要负责发送按键信息。都接通电源并打开手柄开关时，手柄与接收器自动配对连接，在未配对成功的状态下，接收器绿灯闪烁，手柄上的灯也会闪烁，配对成功后，接收器上绿灯常亮，手柄上灯也常亮，这时可以按“ANALOG”键，选择手柄发送模式，具体区别会在第 4 节实验中详细讲解。

接收器和主机（单片机）相连，实现主机与手柄之间的通讯。

接收器引脚输出：

1	2	3	4	5	6	7	8	9
DI/DAT	DO/CMD	NC	GND	VDD	CS/SEL	CLK	NC	ACK

接收器图片

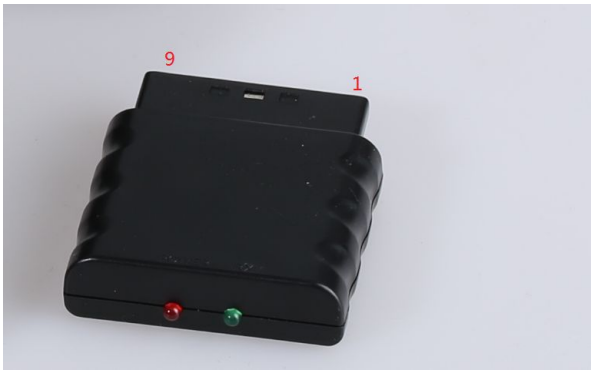


图 1.1 接收器引脚序号

DI/DAT：信号流向，从手柄到主机，此信号是一个 8bit 的串行数据，同步传送于时钟的下降沿。信号的读取在时钟由高到低的变化过程中完成。

DO/CMD：信号流向，从主机到手柄，此信号和 DI 相对，信号是一个 8bit 的串行数据，同步传送于时钟的下降沿。

NC：空端口；

GND：电源地；

VDD：接收器工作电源，电源范围 3~5V；

CS/SEL：用于提供手柄触发信号。在通讯期间，处于低电平；

CLK：时钟信号，由主机发出，用于保持数据同步；

NC：空端口；

ACK：从手柄到主机的应答信号。此信号在每个 8bits 数据发送的最后一个周期变低并且 CS 一直保持低电平，如果 CS 信号不变低，约 60 微秒 PS 主机会试另一个外设。在编程时未使用 ACK 端口。

当主机想读手柄数据时，将会拉低 CS 线电平，并发出一个命令“0x01”；手柄会回复

它的 ID“0x41=模拟绿灯，0x73=模拟红灯”；在手柄发送 ID 的同时，主机将传送 0x42，请求数据；随后手柄发送出 0x5A，告诉主机“数据来了”。

表 1：数据意义对照表

顺序	DO	DI	Bit0、Bit1、Bit2、Bit3、Bit4、Bit5、Bit6、Bit7、
0	0X01	idle	
1	0x42	ID	
2	idle	0x5A	
3	idle	data	SELECT、L3、R3、START、UP、RIGHT、DOWN、LEFT
4	idle	data	L2、R2、L1、R1、△、○、╳、□
5	idle	data	PSS_RX（0x00=left、0xFF=right）
6	idle	data	PSS_RY（0x00=up、0xFF=down）
7	idle	data	PSS_LX（0x00=left、0xFF=right）
8	idle	data	PSS_LY（0x00=up、0xFF=down）

当有按键按下，对应位为“0”，其他位为“1”，例如当键“SELECT”被按下时，Data[3]=11111110B，

红灯模式时：左右摇杆发送模拟值，0x00~0xFF 之间，且摇杆按下的键值值 L3、R3 有效；

绿灯模式时：左右摇杆模拟值为无效，推到极限时，对应发送 UP、RIGHT、DOWN、LEFT、△、○、╳、□，按键 L3、R3 无效。

2 硬件连接方式

接收器与 stm32 连接方式

DI->PB12；

DO->PB13；

CS->PB14；

CLK->PB15。

3 程序设计

完整程序详见工程文件。

这里主要介绍 pstwo.c 文件中的函数。

```
void PS2_Init(void)
{
    //输入  DI->PB12
    RCC->APB2ENR|=1<<3;       //使能 PORTB 时钟
    GPIOB->CRH&=0XFFF0FFFF;//PB12 设置成输入  默认下拉
    GPIOB->CRH|=0X00080000;
    //  DO->PB13      CLC->PB14   CS->PB15
    RCC->APB2ENR|=1<<3;      //使能 PORTB 时钟
    GPIOB->CRH&=0X000FFFFF;
```

```
    GPIOB->CRH|=0X33300000;     //PB13、PB14、PB15 推挽输出
}
```

端口初始化，PB12 为输入，PB13、PB14、PB15 为输出。

```
//向手柄发送命令
void PS2_Cmd(u8 CMD)
{
    volatile u16 ref=0x01;
    for(ref=0x01;ref<0x0100;ref<<=1)
    {
        if(ref&CMD)
         {
            DO_H;                       //输出一位控制位
         }
         else DO_L;
         CLK_H;                         //时钟拉高
         delay_us(50);
         CLK_L;
         delay_us(50);
         CLK_H;
    }
}
//读取手柄数据
void PS2_ReadData()
{
    volatile u8 byte=0;
    volatile u16 ref=0x01;
    CS_L;
    PS2_Cmd(Comd[0]);   //开始命令
    PS2_Cmd(Comd[1]);   //请求数据
    for(byte=2;byte<9;byte++)           //开始接受数据
    {
        for(ref=0x01;ref<0x100;ref<<=1)
        {
            CLK_H;
            CLK_L;
            delay_us(50);
            CLK_H;
              if(DI)
```

```
                Data[byte] = ref|Data[byte];
            }
            delay_us(50);
        }
        CS_H;
    }
```

上面两个函数分别为主机向手柄发送数据、手柄向主机发送数据，并将数据缓存在数组 Data[ ]中，数组中共有 9 个元素，每个元素的意义请见表 1。

```
//对读出来的 PS2 的数据进行处理
//按下为 0， 未按下为 1
u8 PS2_DataKey()
{
    u8 index;
    PS2_ClearData();
    PS2_ReadData();
    Handkey=(Data[4]<<8)|Data[3];       //这是 16 个按键  按下为 0， 未按下为 1
    for(index=0;index<16;index++)
    {
        if((Handkey&(1<<(MASK[index]-1)))==0)
        return index+1;
    }
    return 0;                 //没有任何按键按下
}
```

8 位数 Data[3]与 Data[4]，分别对应着 16 个按键的状态，按下为 0，未按下为 1。通过对这两个数的处理，得到按键状态并返回键值。

编写主函数：

```
int main(void)
{
    u8 key;
    Stm32_Clock_Init(9); //系统时钟设置
    delay_init(72);       //延时初始化
    uart_init(72,9600);   //串口 1 初始化
    PS2_Init();
    while(1)
    {
        key=PS2_DataKey();
        if(key!=0)                        //有按键按下
        {
```

```
                printf("  \r\n    %d   is   pressed   \r\n",key);
            }
        printf(" %5d %5d %5d %5d\r\n",PS2_AnologData(PSS_LX),PS2_AnologData(PSS_LY),
                                      PS2_AnologData(PSS_RX),PS2_AnologData(PSS_RY) );
            delay_ms(50);
        }
    }
```

当有按键按下时，输出按键值。

4 下载与测试

编译程序并下载。按 ANALOG 可以改变模式，先选择红灯模式，遥控器上指示灯为红色。串口输出的模拟值为 127 或 128，当晃动摇杆时，相应的模拟值就会改变，这时摇杆按键可以按下，可以输出键值，见图 2。

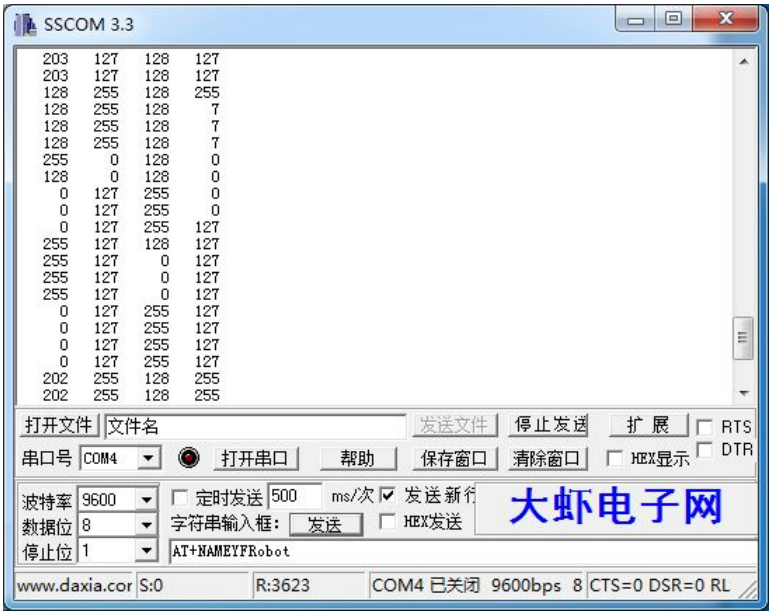


图 1

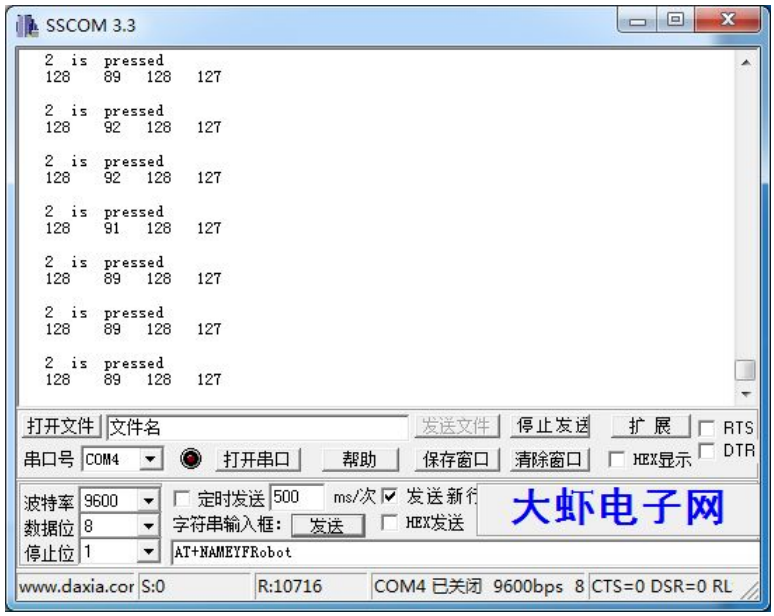


图 2

按下“△”，输出对应的键值“13”。

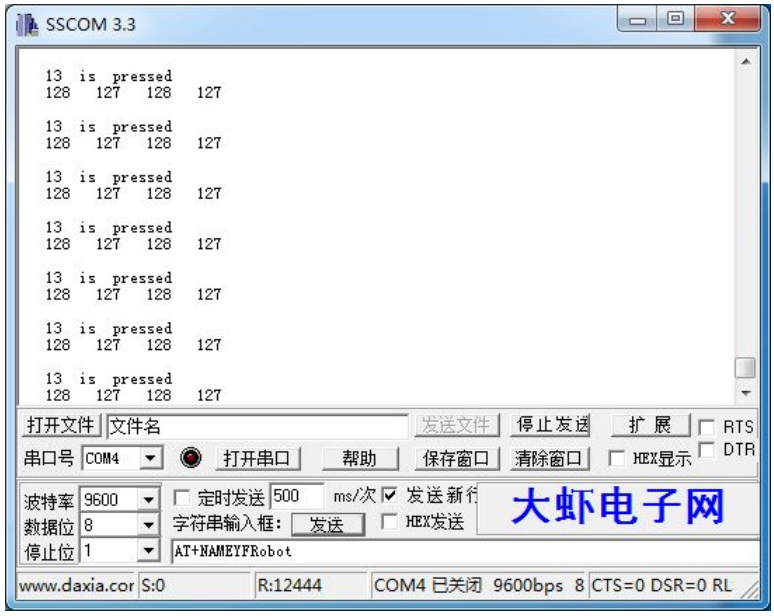


图 3

按“ANALOG”，改为绿灯模式，手柄上指示灯变为“绿色”，串口输出的模拟值为“255”，轻轻晃动摇杆，模拟值不变。

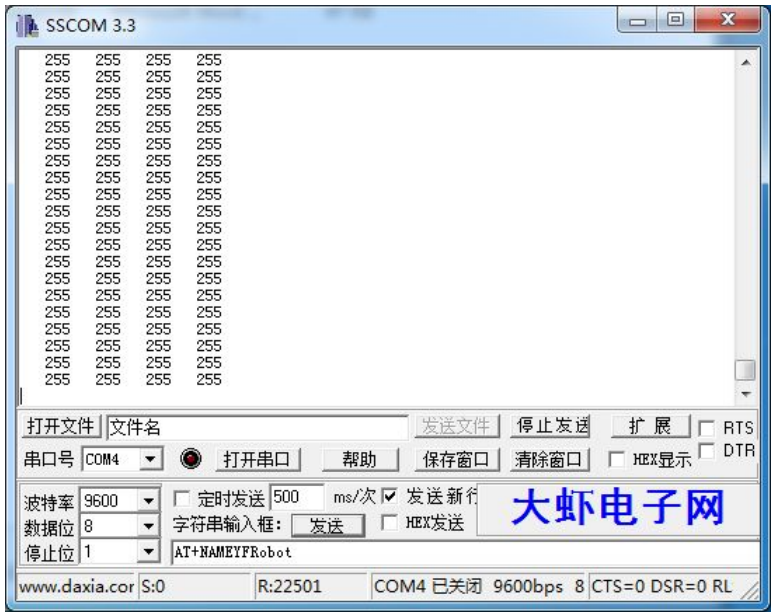


图 4

我们将右摇杆向上推到极限，这时串口输出“13 is pressed”，键值对应“△”，但模拟的值不改变。

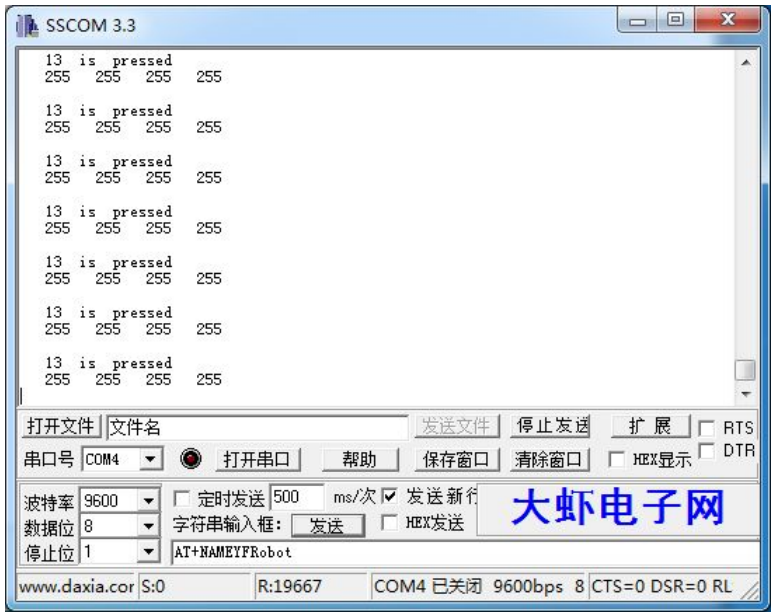


图 5

“红灯模式”和“绿灯模式”的主要区别就在与摇杆模拟值的输出。