Lesson 8 PS2 Handle Control

1. Project Purpose

Learn the principle of PS2 handle control and realize the data communication of PS2 handle.

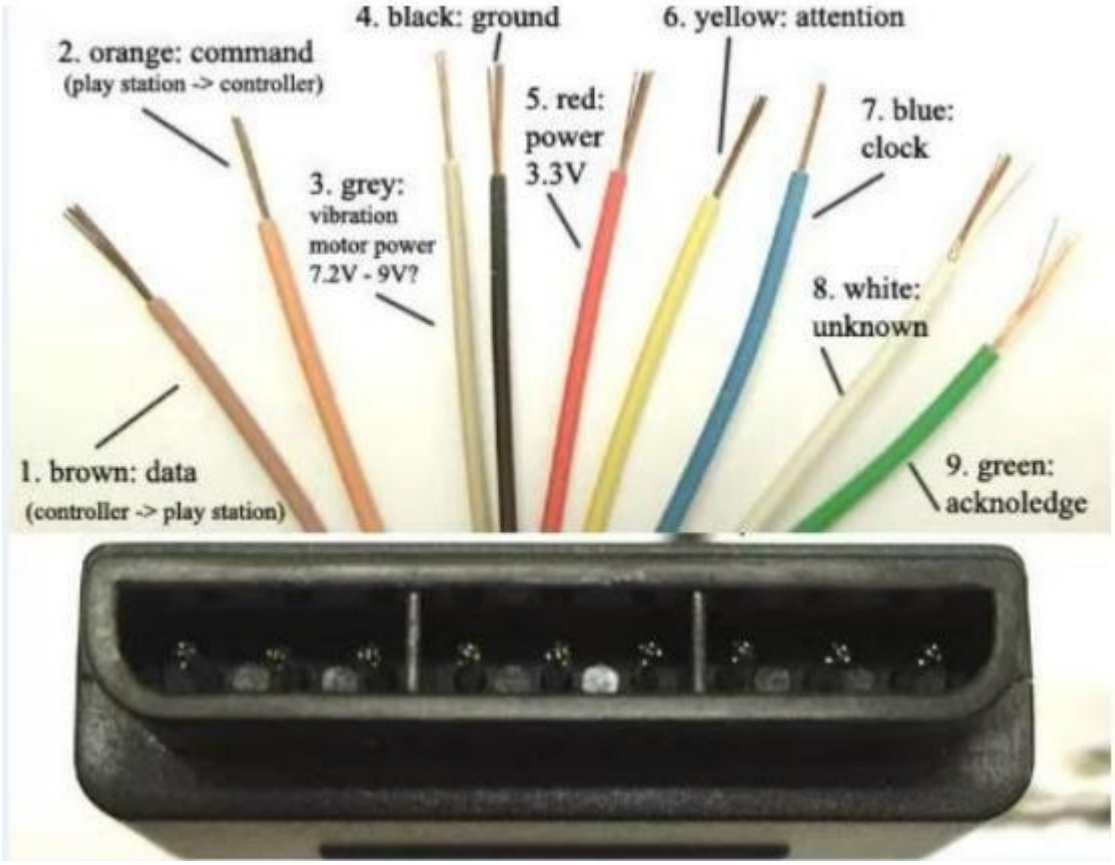
2. Project Principle

The human–computer interface is very important in control system. Handle is easy and convenient to operation and suitable for robot control. In this section, we choose a common PS handle as the control device.

The PS handle requires only four signal wires to communicate with the microcontroller. The communication method between the handle and microcontroller is serial mode, which occupies fewer I/O ports and the communication protocol is simpler. Therefore, it is very suitable for development. The following is the pin definition diagram of the PS handle receiver.

Pin	Definition	Application
1	DATA	The serial data line from the handle to the host, this signal is an 8-bit serial data, synchronously transmitted on the falling edge of the clock (input and output signals change from high to low in the clock signal, and all signals are read from the front edge of the clock to the level Done before the change.)
2	COMMAND	The serial data line from the host to the handle works in the same way as the DATA signal.
3	NC	No use
4	GND	Power ground and signal ground
5	3.3v	Power votage. The effective working voltage is 3V-5V.
6	Attention	Used to provide a handle trigger signal and the signal is at a low level during communication.

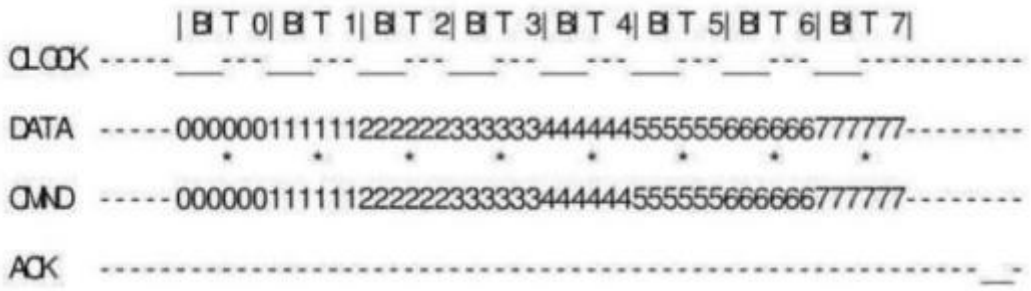
		Equivalent to chip select signal
7	CLOCK	Signal direction: from the host to the handle. Used to keep data in sync
8	NC	No use
9	acknolege	The response signal from the handle to the host. This signal becomes low in the last clock cycle after each 8Bit data is sent, and remains low. If the ACK signal does not go low for about 60us, the PS host will try another handle.



3. Program Analyst

1) The host can connect multiple receivers at the same time, and select the designated handle by pulling down the corresponding AttentIOn signal. All communication data of the PS handle receiver is 8-bit serial data, with the least

significant bit first. The following figure shows the timing of sending and receiving a byte when the PS handle receiver communicates:



2) From the above figure, we can see that the data lines (DATA，COMMAND) all change as the CLOCK falls. Therefore, the data reception should be completed from the rising edge of the clock (marked by * in the above diagram) to the time when the level changes again.

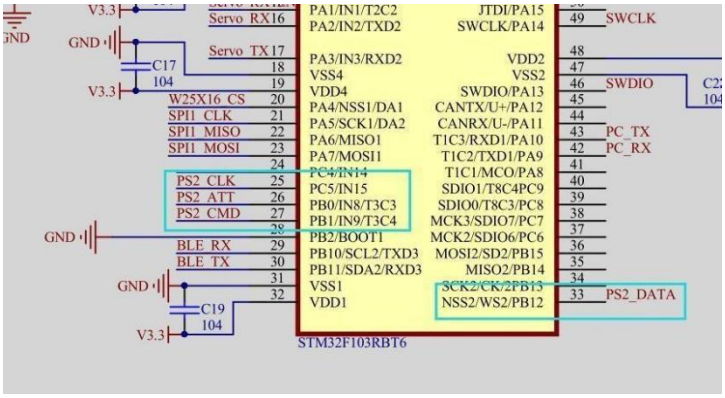
3) After the selected handle receives the COMMAND data bytes, ACK will be pulled down for a clock cycle. If the selected handle does not response, the host will consider that the handle is not connected. Through the information above, we can write a function of the handle receiver to send command, as shown in the following figure:

```
65  // send commands to the PS2 handle
66  void PS2_Cmd(u8 CMD)
67  {
68      volatile u16 ref=0x01;
69      Data[1] = 0;
70      for(ref=0x01;ref<0x0100;ref<<=1)
71      {
72          if(ref&CMD)
73          {
74              DO_H;                   //Control bit
75          }
76          else DO_L;
77
78          Delay(10);
79          CLK_L;
80          Delay(40);
81          CLK_H;
82          if(DI)
83              Data[1] = ref|Data[1];
84          Delay(10);
85      }
86  }
```

4) In the above code, we did not find that the use of the AttentIOn and ACK signals. Because there is only a receiver, we do not need to switch among multiple receivers. Therefore, the ACK signal does not need to be used.

5) The AttentIOn signal is because we manually set the AttentIOn signal before calling PS2_Cmd. But before using the above function, we still need to configure the functions between each I/O port.

6) The AttentIOn signal is because we manually set the AttentIOn signal before calling PS2_Cmd. Before using the above function, we need to configure the functions among each I/O port.



Configure PS2_CLK、PS2_CMD、PS2ATT to push–pull output and PS2_DATA to pull-up input.

```
void InitPS2(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;         //Push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;         //Push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    PS2_SetInit();       // Configuration initialization, configure "red and green lights mode", and select whether it can be modified
}
```

7) According to the timing and protocol, we can write such a function which is used to read the data of the handle button, as shown in the following figure:

```
100  //read handle data
101  void PS2_ReadData(void)
102  {
103      volatile u8 byte;
104      volatile u16 ref;
105
106      CS_L;
107      Delay(10);
108      PS2_Cmd(Comd[0]);  //start command
109      PS2_Cmd(Comd[1]);  //ask data
110      for(byte=2;byte<9;byte++)          //start receiving data
111      {
112          for(ref=0x01;ref<0x100;ref<<=1)
113          {
114
115              CLK_L;
116                  Delay(50);
117              CLK_H;
118                if(DI)
119                {
120                  Data[byte] = ref|Data[byte];
121                }
122                  Delay(20);
123
124          }
125              Delay(40);
126      }
127      CS_H;
128  }
129
```

8) In the above function, we can see the process of data reception: pull down the level of AttentIOn wire first, and send the start command and the request date command. Then start to receive the button data returned from the receiver.

9) The start command is 0x01 and the request command is 0x42. In fact, the reception of each byte is the same as the implementation of PS_Cmd. Therefore, the level of AttentIOn line is pulled up after the reception is completed, which can end the communication.

10) The read data may be different due to the different handle working modes. The working mode of the handle can be set by sending command. Program will set the handle to the red light mode during initialization. The data format in the red light mode is shown in the following table:

BYTES	CMD DATA	DATA	DATA CORRESPONDING MEANING
01	0x01 start command	Null	
02	0×42 request data	0x53	
03	Null	0x5A	
04	Null	Data	Bit0:SELECT,Bit1:L3,Bit2:R3,Bit3:START,Bit4:上,Bit:5右,Bit6:下,Bit7:左
05	Null	Data	Bit0:L2,Bit1:R2,Bit2:L1,Bit3:R1, Bit4:三角,Bit5:圈,Bit6:叉,Bit7:方框
06	Null	Data	右摇杆，0x00时向左，0xFF时向右
07	Null	Data	右摇杆，0x00时向上，0xFF时向下
08	Null	Data	左摇杆，0x00时向左，0xFF时向右
09	Null	Data	左摇杆，0x00时向上，0xFF时向下

11) In the table above, the fourth and the fifth bytes contain the button data of the handle. We can judge whether a button is pressed according to the requirement. The sixth to the ninth bytes are the joystick data. The maximum value is 255 and the minimum value is 0.

12) Finally ,in program, you only need to set the handle data that is periodically read in the handle receiver, and then make the corresponding operation according to the obtained button data to realize the handle remote control.

: