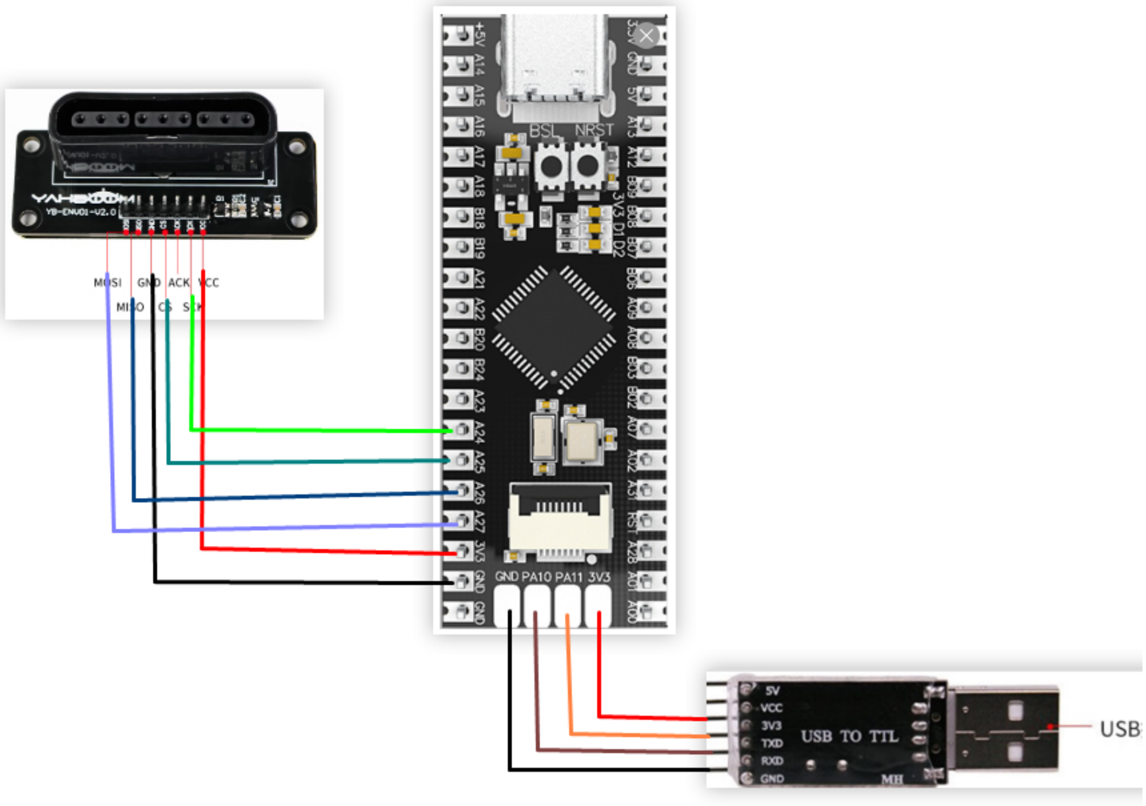
Print data

1. Learning objectives

Print the key value of the PS controller to the computer's serial port assistant through the serial port.

2. Hardware connection

PS2 controller receiving module, USB to TTL and MSPM0G3507 wiring



Note: If there is no TTL module, you can also use the type-c serial port directly

3. Program description

- usart.h

```
#ifndef __USART_H__
#define __USART_H__

#include "ti_msp_dl_config.h"


void USART_Init(void);

void USART_SendData(unsigned char data);

#endif
```

Declare the header file for serial port initialization and data transmission

- usart.c

```
//串口发送一个字节
//The serial port sends a byte
void USART_SendData(unsigned char data)
{
    //当串口0忙的时候等待
    //Wait when serial port 0 is busy
    while( DL_UART_isBusy(UART_0_INST) == true );
    //发送
    //send
    DL_UART_Main_transmitData(UART_0_INST, data);
}

//串口的中断服务函数
//Serial port interrupt service function
void UART_0_INST_IRQHandler(void)
{
    uint8_t receivedData = 0;

    //如果产生了串口中断
    //If a serial port interrupt occurs
    switch( DL_UART_getPendingInterrupt(UART_0_INST) )
    {
        case DL_UART_IIDX_RX://如果是接收中断  If it is a receive interrupt

            // 接收发送过来的数据保存  Receive and save the data sent
            receivedData = DL_UART_Main_receiveData(UART_0_INST);

            // 检查缓冲区是否已满    Check if the buffer is full
            if (recv0_length < RE_0_BUFF_LEN_MAX - 1)
            {
                recv0_buff[recv0_length++] = receivedData;
            }
            else
            {
                recv0_length = 0;
            }

            // 标记接收标志   Mark receiving flag
            recv0_flag = 1;

            break;

        default://其他的串口中断   Other serial port interrupts
            break;
    }
}
```

Define the serial port to send a byte function, serial port interrupt service function

- delay.h

```
#ifndef _DELAY_H
#define _DELAY_H

#include <stdint.h>
#include "ti_msp_dl_config.h"

void delay_us(unsigned long __us);
void delay_ms(unsigned long ms);


#endif
```

Declare millisecond and microsecond function header files

- delay.c

```
#include "delay.h"

volatile unsigned int delay_times = 0;

//搭配滴答定时器实现的精确us延时
//Accurate us delay with tick timer
void delay_us(unsigned long __us)
{
    uint32_t ticks;
    uint32_t told, tnow, tcnt = 38;

    // 计算需要的时钟数 = 延迟微秒数 * 每微秒的时钟数
    // Calculate the number of clocks required = delay microseconds * number of 
clocks per microsecond
    ticks = __us * (32000000 / 1000000);

    // 获取当前的SysTick值
    // Get the current SysTick value
    told = SysTick->VAL;

    while (1)
    {
        // 重复刷新获取当前的SysTick值
        // Repeatedly refresh to get the current SysTick value
        tnow = SysTick->VAL;

        if (tnow != told)
        {
            if (tnow < told)
                tcnt += told - tnow;
            else
                tcnt += SysTick->LOAD - tnow + told;

            told = tnow;

            // 如果达到了需要的时钟数，就退出循环
            // If the required number of clocks is reached, exit the loop
            if (tcnt >= ticks)
                break;
        }
    }
}
//搭配滴答定时器实现的精确ms延时
//Accurate ms delay with tick timer
void delay_ms(unsigned long ms)
{
    delay_us( ms * 1000 );
}
```

Achieve millisecond and microsecond delays through timer counting

- pstwo.h

```
#ifndef __PSTWO_H
#define __PSTWO_H
#include "ti_msp_dl_config.h"
#include "delay.h"
#include "stdio.h"

#define DI   DL_GPIO_readPins(GPIO_PORT, GPIO_MISO_PIN)

#define DO_H DL_GPIO_setPins(GPIO_PORT, GPIO_MOSI_PIN);  //命令位高 Command bit
high
#define DO_L DL_GPIO_clearPins(GPIO_PORT, GPIO_MOSI_PIN);//命令位低 Command bit
low

#define CS_H DL_GPIO_setPins(GPIO_PORT, GPIO_CS_PIN);   //CS拉高    CS pull high
#define CS_L DL_GPIO_clearPins(GPIO_PORT, GPIO_CS_PIN); //CS拉低    CS pull low

#define CLK_H DL_GPIO_setPins(GPIO_PORT, GPIO_CLK_PIN);  //时钟拉高 Clock high
#define CLK_L DL_GPIO_clearPins(GPIO_PORT, GPIO_CLK_PIN);//时钟拉低 Clock low

#define u8 uint8_t
#define u16 uint16_t
#define u32 uint32_t

//These are our button constants
#define PSB_SELECT      1
#define PSB_L3          2
#define PSB_R3          3
#define PSB_START       4
#define PSB_PAD_UP      5
#define PSB_PAD_RIGHT   6
#define PSB_PAD_DOWN    7
#define PSB_PAD_LEFT    8
#define PSB_L2          9
#define PSB_R2          10
#define PSB_L1          11
#define PSB_R1          12
#define PSB_GREEN       13
#define PSB_RED         14
#define PSB_BLUE        15
#define PSB_PINK        16
#define PSB_TRIANGLE    13
```

```
#define PSB_CIRCLE      14
#define PSB_CROSS       15
#define PSB_SQUARE      26

//#define WHAMMY_BAR        8

//These are stick values
#define PSS_RX 5      //右摇杆X轴数据 Right joystick X-axis data
#define PSS_RY 6
#define PSS_LX 7
#define PSS_LY 8

extern u8 Data[9];
extern u16 MASK[16];
extern u16 Handkey;

u8 PS2_RedLight(void);//判断是否为红灯模式  Determine whether it is red light mode
void PS2_ReadData(void);
void PS2_Cmd(u8 CMD);
u8 PS2_DataKey(void);         //键值读取    Key-value reading
u8 PS2_AnologData(u8 button); //得到一个摇杆的模拟量    Get the analog value of a
joystick
void PS2_ClearData(void);     //清除数据缓冲区  Clear the data buffer

#endif
```

Define the handle key values and IO port control.

- pstwo.c

```
//对读出来的PS2的数据进行处理      只处理了按键部分         默认数据是红灯模式   只有一个按键
按下时
//Process the PS2 data read out. Only the key part is processed. The default data
is red light mode. When only one key is pressed
//按下为0， 未按下为1   Pressed is 0, not pressed is 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 These are 16
keys. If pressed, it is 0. If not pressed, it is 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    No keys pressed
}

//得到一个摇杆的模拟量     范围0~256    Get the analog value of a joystick, range
0~256
u8 PS2_AnologData(u8 button)
{
    return Data[button];
}
```

Realize key value acquisition and analog quantity reading

- empty.c

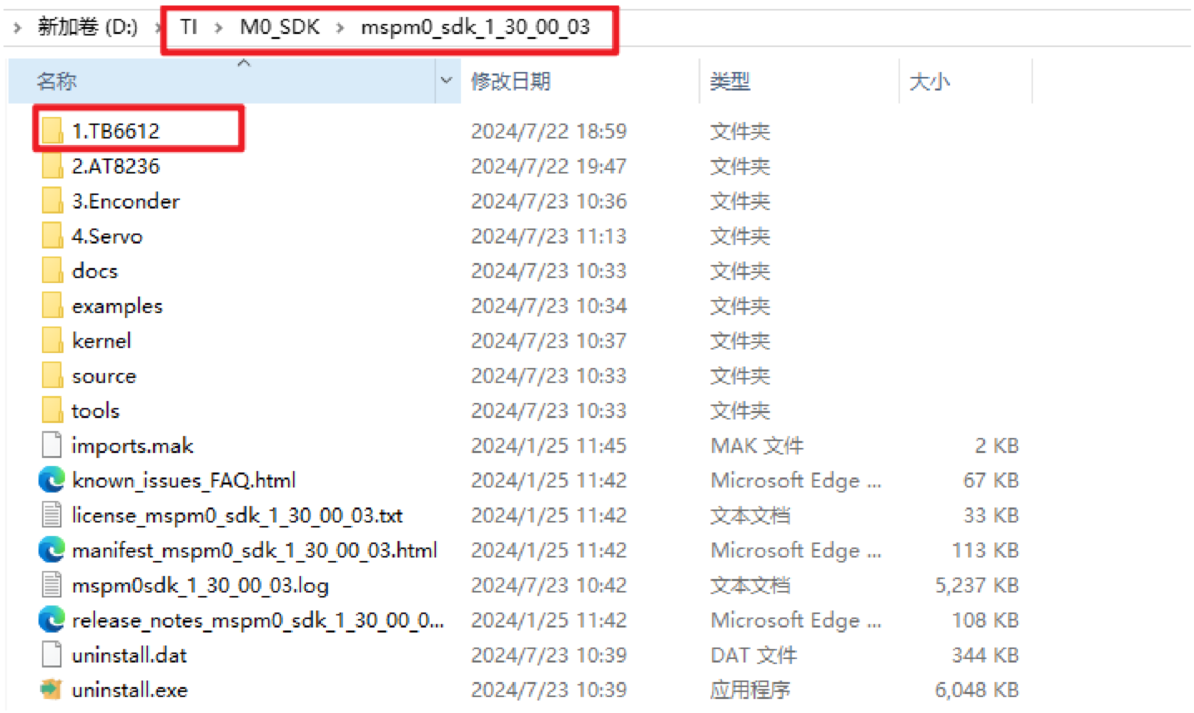
```
int main(void)
{
    int PS2_LX,PS2_LY,PS2_RX,PS2_RY,PS2_KEY;
    USART_Init();
    while(1)
    {
        PS2_LX=PS2_AnologData(PSS_LX);
        PS2_LY=PS2_AnologData(PSS_LY);
        PS2_RX=PS2_AnologData(PSS_RX);
        PS2_RY=PS2_AnologData(PSS_RY);
        PS2_KEY=PS2_DataKey();

        printf("PS2_LX=%d    ",PS2_LX);
        printf("PS2_LY=%d    ",PS2_LY);
        printf("PS2_RX=%d    ",PS2_RX);
        printf("PS2_RY=%d    ",PS2_RY);
        printf("PS2_KEY=%d    \r\n",PS2_KEY);
        delay_ms(10);
    }
}
```

Initialize the serial port and handle, and print the obtained handle key value and analog value in the serial port every 10 milliseconds.

Note: The project source code must be placed in the SDK path for compilation,

For example, path: D:\TI\M0_SDK\mspm0_sdk_1_30_00_03\1.TB6612



新加卷 (D:) › TI › M0_SDK › mspm0_sdk_1_30_00_03

名称	修改日期	类型	大小
1.TB6612	2024/7/22 18:59	文件夹	
2.AT8236	2024/7/22 19:47	文件夹	
3.Enconder	2024/7/23 10:36	文件夹	
4.Servo	2024/7/23 11:13	文件夹	
docs	2024/7/23 10:33	文件夹	
examples	2024/7/23 10:34	文件夹	
kernel	2024/7/23 10:37	文件夹	
source	2024/7/23 10:33	文件夹	
tools	2024/7/23 10:33	文件夹	
imports.mak	2024/1/25 11:45	MAK 文件	2 KB
known_issues_FAQ.html	2024/1/25 11:42	Microsoft Edge ...	67 KB
license_mspm0_sdk_1_30_00_03.txt	2024/1/25 11:42	文本文档	33 KB
manifest_mspm0_sdk_1_30_00_03.html	2024/1/25 11:42	Microsoft Edge ...	113 KB
mspm0sdk_1_30_00_03.log	2024/7/23 10:42	文本文档	5,237 KB
release_notes_mspm0_sdk_1_30_00_0...	2024/1/25 11:42	Microsoft Edge ...	108 KB
uninstall.dat	2024/7/23 10:39	DAT 文件	344 KB
uninstall.exe	2024/7/23 10:39	应用程序	6,048 KB

4. Experimental phenomenon

Burn the program to MSPM0G3507 and connect the wires according to the wiring diagram. Close other programs that occupy the serial port, open the serial port assistant on the computer, select the serial port number, and set the baud rate to 115200. Turn on the handle switch and wait for the indicator light to stop flashing. Press the mode button of the handle to switch to the red and green light mode. Press the handle again and you will see the printed key value and analog value in the serial port assistant.