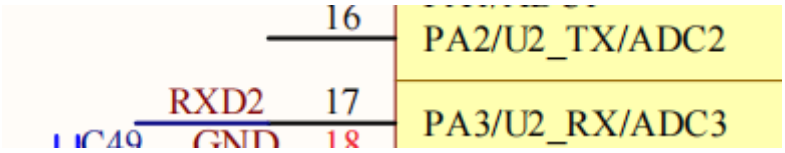
# 6. SBUS model airplane remote control

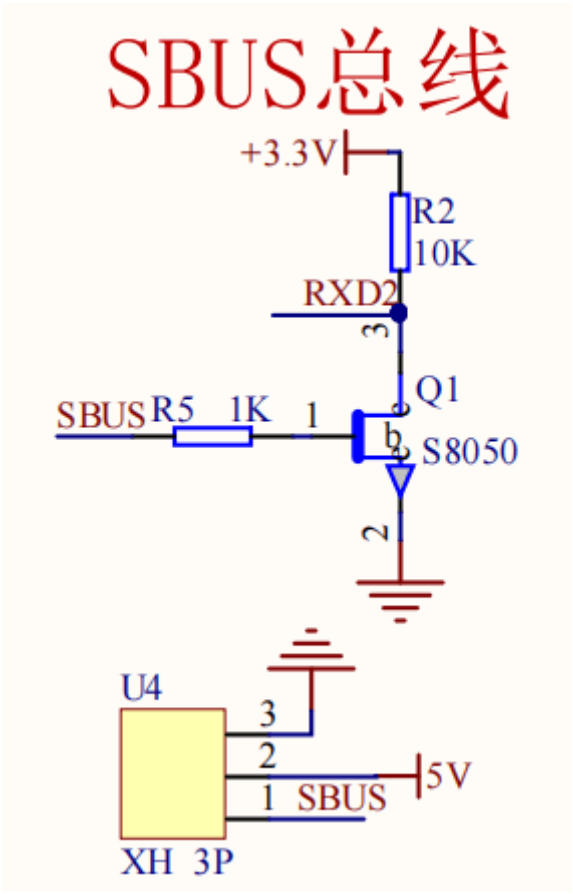


## 6.1. Experimental purpose

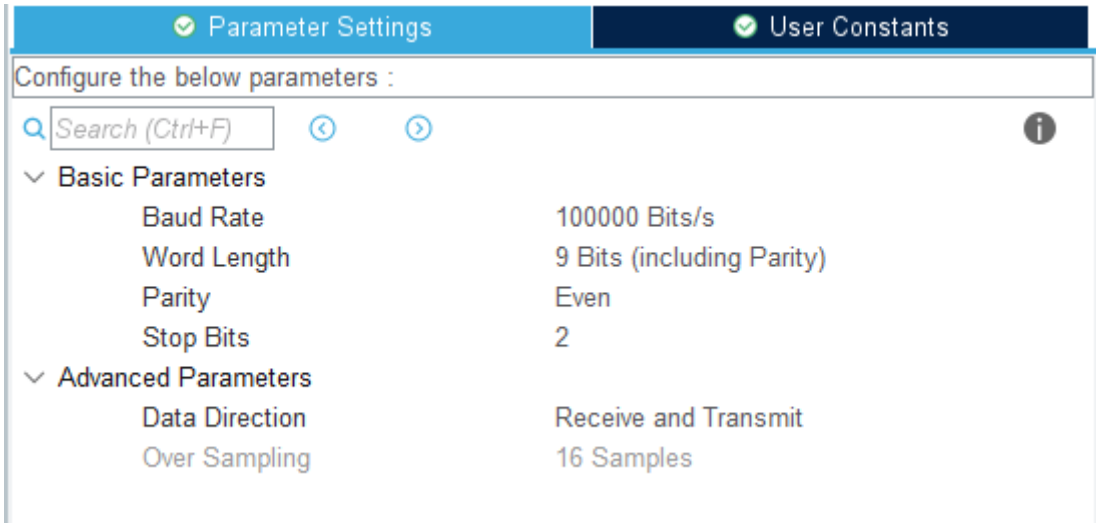
Using STM32 serial port communication, the SBUS protocol data transmitted by remote control transmitter of aircraft model is analyzed, and the values of each channel are printed.

## 6.2. Configure pin information

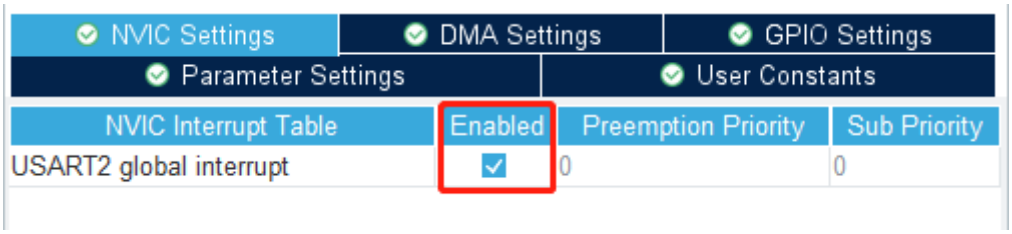


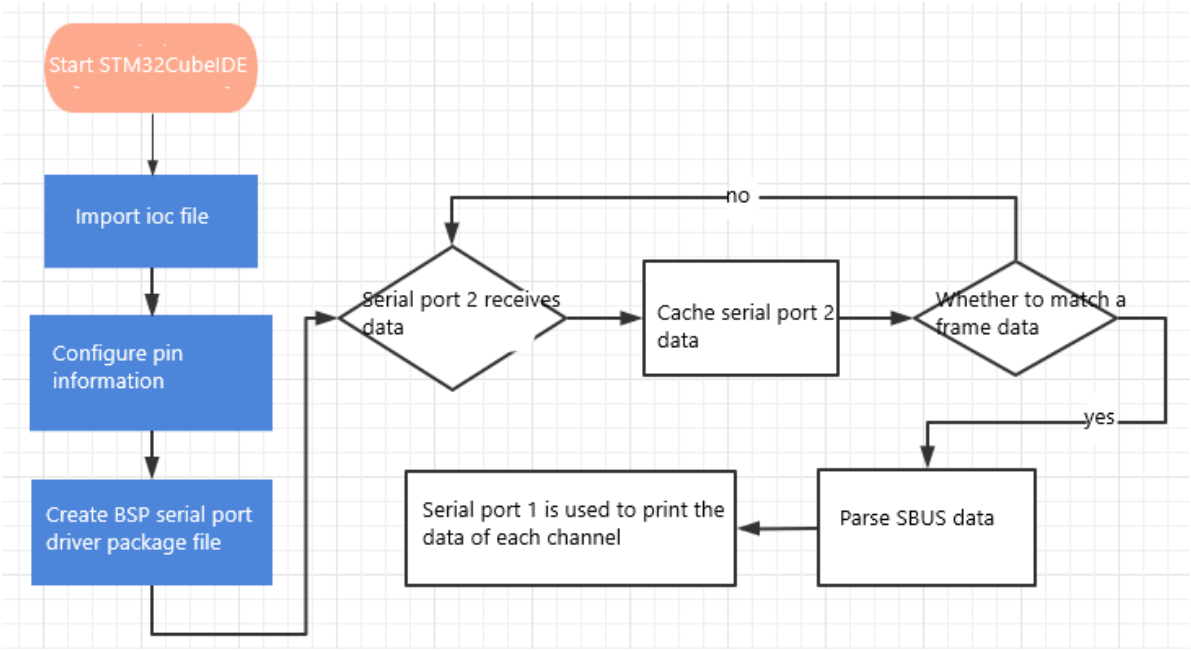

2. Change the mode of serial port 2 to Asynchronous synchronous communication, the baud rate is 100000, the data width is 9 bits, the check is Even, and the stop bit is 2 bits. Serial port 2 Only uses the receiving function, so the Data Direction can be Receive and Transmit or Receive Only.




3. Open the serial port 2 interrupt Settings.




## 6.3. Analysis of experimental flow chart




## 6.4. Core code interpretation

1. Add the following content to bsp_uart.c:

USART1_Init(): initializes content related to the serial port and enables serial ports 1 and 2 to receive one data.

```
// Initialize USART1  初始化串口1
void USART1_Init(void)
{
    HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
    HAL_UART_Receive_IT(&huart2, (uint8_t *)&RxTemp_2, 1);

    printf("start serial\n");
}
```

2. In the serial port interrupt callback, check whether data is received from serial port 2 and determine whether data is received from serial port 1 or serial port 2.

```
// The serial port receiving is interrupted. Procedure  串口接收完成中断
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

    if (huart == &huart1)
    {
        // 测试发送数据，实际应用中不应该在中断中发送数据
        // Test sending data. In practice, data should not be sent during interrupts
        USART1_Send_U8(RxTemp);
        // Continue receiving data  继续接收数据
        HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
    }
    if (huart == &huart2)
    {
        SBUS_Reveive(RxTemp_2);
        // printf("a:%d\n", RxTemp_2);
        HAL_UART_Receive_IT(&huart2, (uint8_t *)&RxTemp_2, 1);
    }
}
```

3. Create bsp_sbus.h and bsp_sbus.c files to manage sbus data parsing. Create the following in bsp_sbus.h:

```
#define SBUS_SIGNAL_OK          0x00
#define SBUS_SIGNAL_LOST        0x01
#define SBUS_SIGNAL_FAILSAFE    0x03
#define SBUS_ALL_CHANNELS       0x00


void SBUS_Reveive(uint8_t data);
void SBUS_Handle(void);
```

SBUS_ALL_CHANNELS controls the number of channels to be resolved. By default, only eight channels are displayed, and if full channels are required, the value is changed to 1.

4.SBUS_Reveive(data) receives serial port data as cache. If it conforms to SBUS communication protocol, a frame data is updated to sbus_data array.

```
// Receives SBUS cache data  接收SBUS的缓存数据
void SBUS_Reveive(uint8_t data)
{
    // If the protocol start flag is met, data is received  如果符合协议开始标志，则开始接收数据
    if (sbus_start == 0 && data == SBUS_START)
    {
        sbus_start = 1;
        sbus_new_cmd = 0;
        sbus_buf_index = 0;
        inBuffer[sbus_buf_index] = data;
        inBuffer[SBUS_RECV_MAX - 1] = 0xff;
    }
    else if (sbus_start)
    {
        sbus_buf_index++;
        inBuffer[sbus_buf_index] = data;
    }

    // Finish receiving a frame of data  完成接收一帧数据
    if (sbus_start & (sbus_buf_index >= (SBUS_RECV_MAX - 1)))
    {
        sbus_start = 0;
        if (inBuffer[SBUS_RECV_MAX - 1] == SBUS_END)
        {
            memcpy(sbus_data, inBuffer, SBUS_RECV_MAX);
            sbus_new_cmd = 1;
        }
    }
}
```

5. According to the data from the analytical sbus_data SBUS communication protocol.

```
// Parses SBUS data into channel values  解析SBUS的数据，转化成通道数值。
static int SBUS_Parse_Data(void)
{
    g_sbus_channels[0]  = ((sbus_data[1] | sbus_data[2] << 8) & 0x07FF);
    g_sbus_channels[1]  = ((sbus_data[2] >> 3 | sbus_data[3] << 5) & 0x07FF);
    g_sbus_channels[2]  = ((sbus_data[3] >> 6 | sbus_data[4] << 2 | sbus_data[5] << 10) & 0x07FF);
    g_sbus_channels[3]  = ((sbus_data[5] >> 1 | sbus_data[6] << 7) & 0x07FF);
    g_sbus_channels[4]  = ((sbus_data[6] >> 4 | sbus_data[7] << 4) & 0x07FF);
    g_sbus_channels[5]  = ((sbus_data[7] >> 7 | sbus_data[8] << 1 | sbus_data[9] << 9) & 0x07FF);
    g_sbus_channels[6]  = ((sbus_data[9] >> 2 | sbus_data[10] << 6) & 0x07FF);
    g_sbus_channels[7]  = ((sbus_data[10] >> 5 | sbus_data[11] << 3) & 0x07FF);
    #ifdef ALL_CHANNELS
    g_sbus_channels[8]  = ((sbus_data[12] | sbus_data[13] << 8) & 0x07FF);
    g_sbus_channels[9]  = ((sbus_data[13] >> 3 | sbus_data[14] << 5) & 0x07FF);
    g_sbus_channels[10] = ((sbus_data[14] >> 6 | sbus_data[15] << 2 | sbus_data[16] << 10) & 0x07FF);
    g_sbus_channels[11] = ((sbus_data[16] >> 1 | sbus_data[17] << 7) & 0x07FF);
    g_sbus_channels[12] = ((sbus_data[17] >> 4 | sbus_data[18] << 4) & 0x07FF);
    g_sbus_channels[13] = ((sbus_data[18] >> 7 | sbus_data[19] << 1 | sbus_data[20] << 9) & 0x07FF);
    g_sbus_channels[14] = ((sbus_data[20] >> 2 | sbus_data[21] << 6) & 0x07FF);
    g_sbus_channels[15] = ((sbus_data[21] >> 5 | sbus_data[22] << 3) & 0x07FF);
    #endif

    // 安全检测，检测是否失联或者数据错误
    // Security detection to check for lost connections or data errors
    failsafe_status = SBUS_SIGNAL_OK;
    if (sbus_data[23] & (1 << 2))
    {
        failsafe_status = SBUS_SIGNAL_LOST;
        printf("SBUS_SIGNAL_LOST\n");
        // lost contact errors  遥控器失联错误
    }
    else if (sbus_data[23] & (1 << 3))
    {
        failsafe_status = SBUS_SIGNAL_FAILSAFE;
        printf("SBUS_SIGNAL_FAILSAFE\n");
        // data loss error  数据丢失错误
    }
    return failsafe_status;
}
```

6. The SBUS_Handle() function is looped in Bsp_Loop() to print the parsed data of each channel through the serial port.

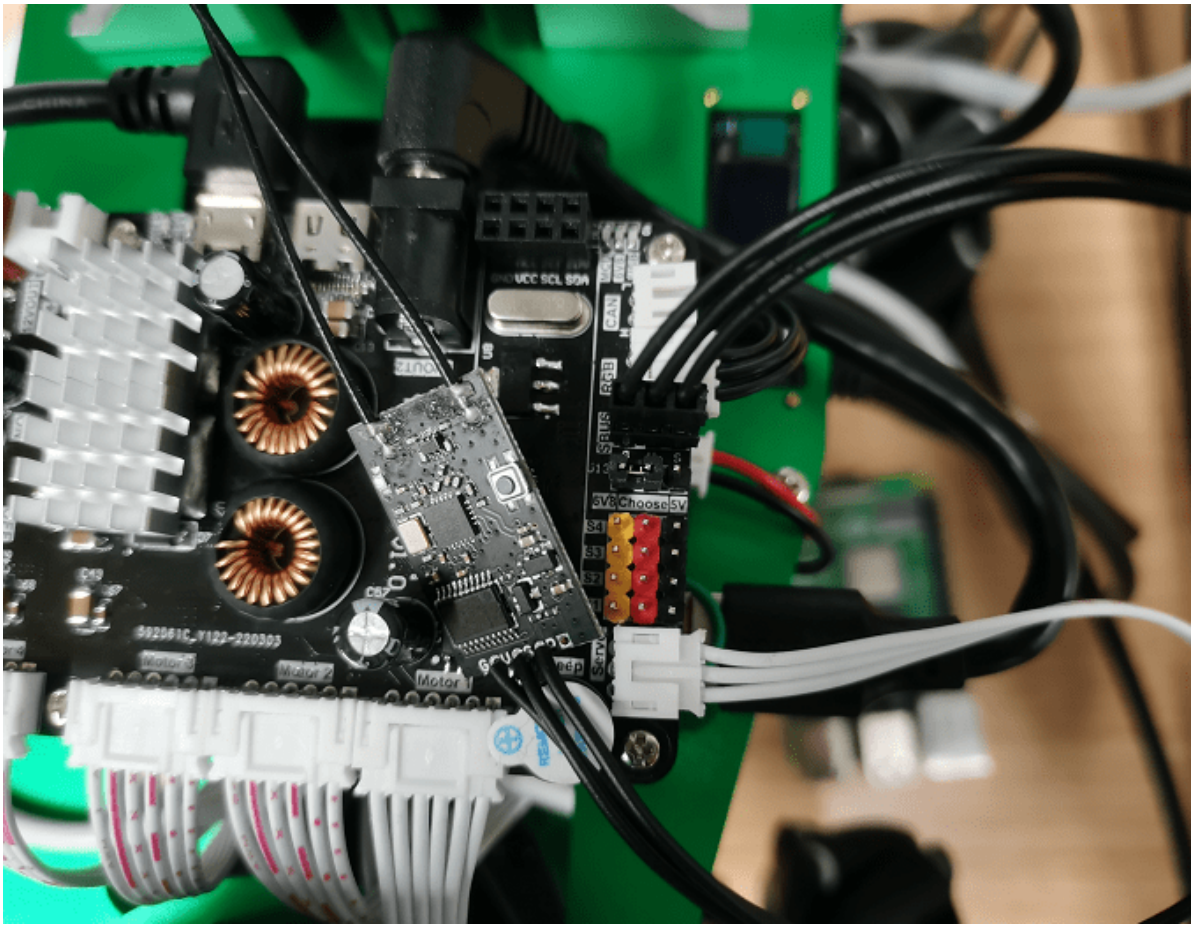
```c
// SBUS receives and processes data handle  SBUS接收处理数据句柄
void SBUS_Handle(void)
{
    if (sbus_new_cmd)
    {
        int res = SBUS_Parse_Data();
        sbus_new_cmd = 0;
        if (res) return;
        #if SBUS_ALL_CHANNELS
        printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
               g_sbus_channels[0], g_sbus_channels[1], g_sbus_channels[2],
               g_sbus_channels[3], g_sbus_channels[4], g_sbus_channels[5],
               g_sbus_channels[6], g_sbus_channels[7], g_sbus_channels[8],
               g_sbus_channels[9], g_sbus_channels[10], g_sbus_channels[11],
               g_sbus_channels[12], g_sbus_channels[13], g_sbus_channels[14],
               g_sbus_channels[15]);
        #else
        printf("%d,%d,%d,%d,%d,%d,%d,%d\r\n",
               g_sbus_channels[0], g_sbus_channels[1], g_sbus_channels[2],
               g_sbus_channels[3], g_sbus_channels[4],g_sbus_channels[5],
               g_sbus_channels[6], g_sbus_channels[7]);
        #endif
    }
}
```

```c
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid multiple modifications to the main.c file
void Bsp_Loop(void)
{
    // Detect button down events   检测按键按下事件
    if (Keyl_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int press = 0;
        press++;
        printf("press:%d\n", press);
    }
    SBUS_Handle();

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out   蜂鸣器超时自动关闭
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

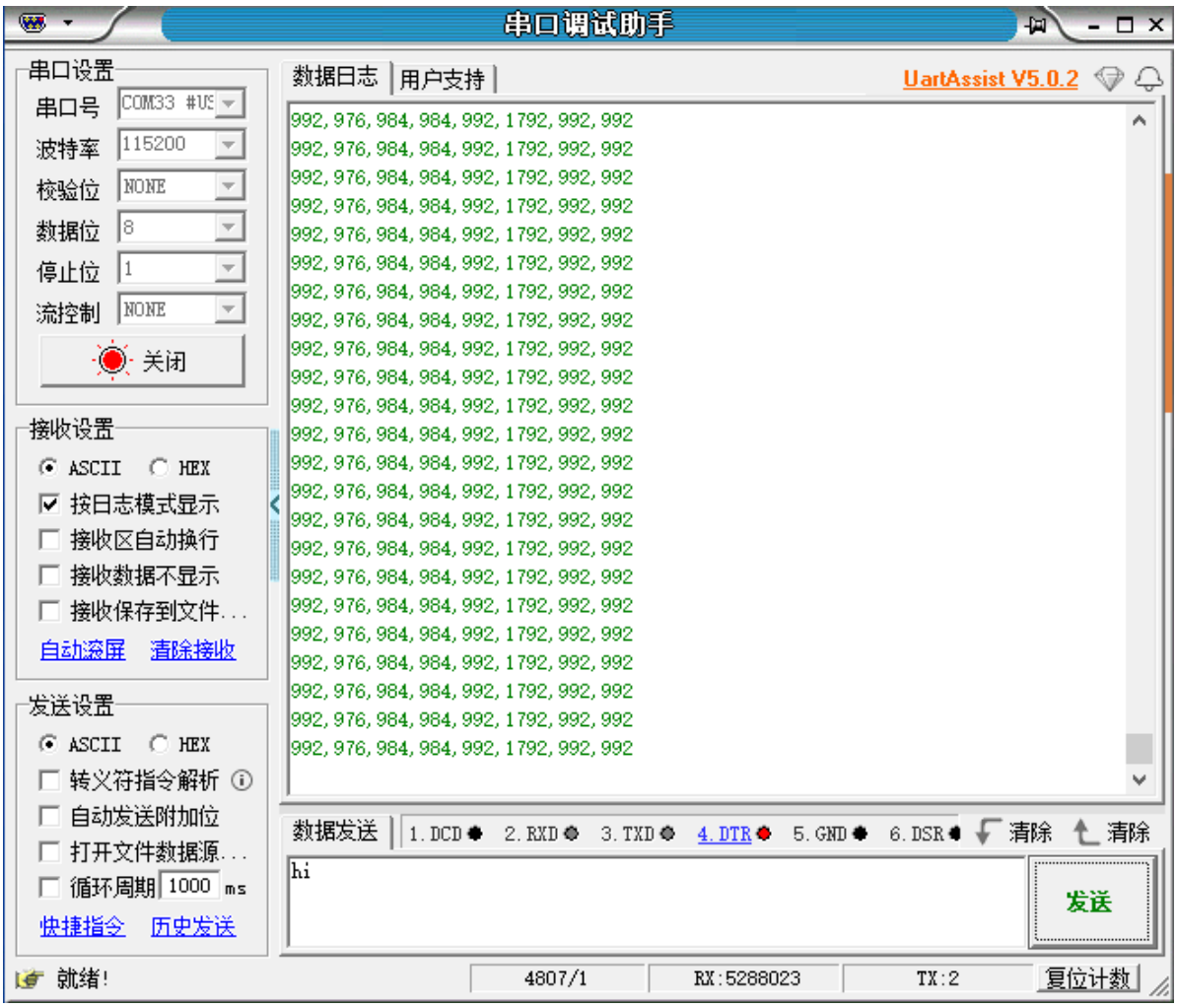
## 6.5. Hardware connection

Because SBUS communication needs to connect the SBUS receiver to the SBUS interface on the expansion board, S is connected to the signal, V is connected to the positive terminal of the power supply, and G is grounded. Therefore, you need to bring your own model remote control and SBUS receiver, pair them well in advance and turn on the power switch.




## 6.6. Experimental effect

After the program is burned, the LED light flashes every 200 milliseconds, the expansion board is connected to the computer through the micro-USB data cable, and the serial port assistant is opened (specific parameters are shown in the figure below). You can see that the serial port assistant has been printing the data of each channel of the model airplane remote control. When we manually flip the rocker or button of the model airplane remote control, the data will change.


串口调试助手
串口设置
串口号
COM33 #US
波特率
115200
校验位
NONE
数据位
8
停止位
1
流控制
NONE
关闭
接收设置
ASCII
HEX
按日志模式显示
接收区自动换行
接收数据不显示
接收保存到文件...
自动滚屏
清除接收
发送设置
ASCII
HEX
转义符指令解析
自动发送附加位
打开文件数据源...
循环周期
1000
ms
快捷指令
历史发送
数据日志
用户支持
UartAssist V5.0.2
992,976,984,984,992,1792,992,992
数据发送
1.DCD
2.RXD
3.TXD
4.DTR
5.GND
6.DSR
清除
清除
hi
发送
就绪!
4807/1
RX:5288023
TX:2
复位计数