10. Control serial port servos

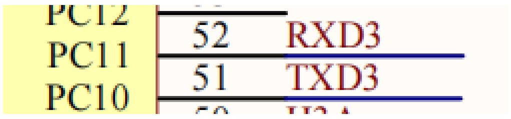
10.1. Purpose of the experiment

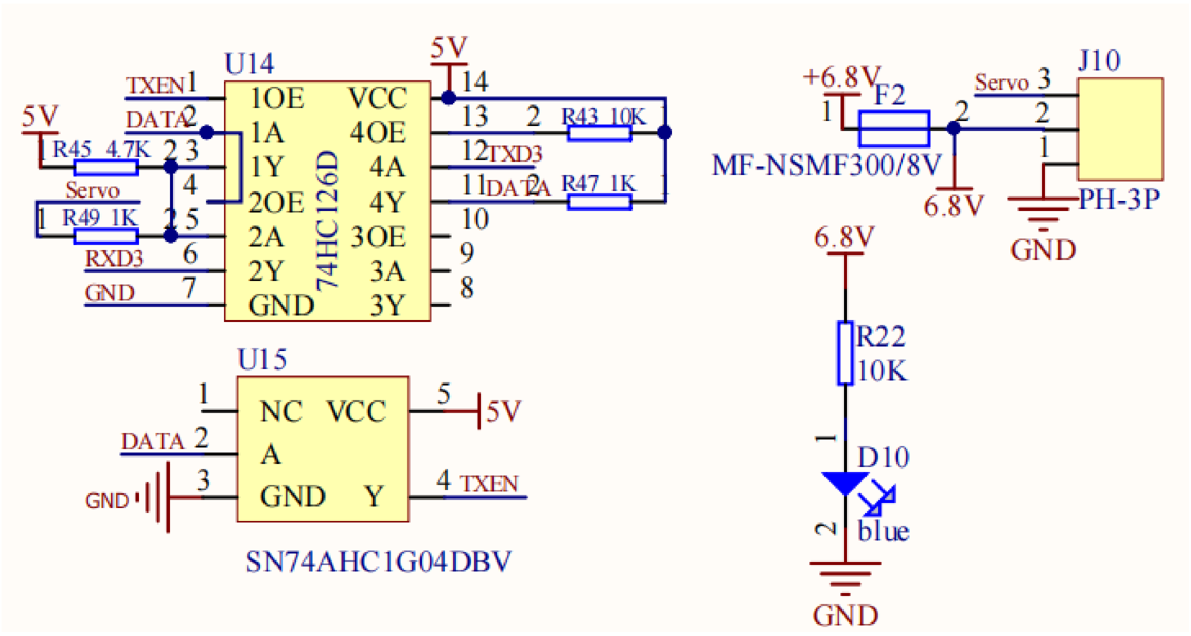
Use the serial port function of STM32 to control the serial port servo and read the position of the serial port servo.

10.2. Configuration pin information

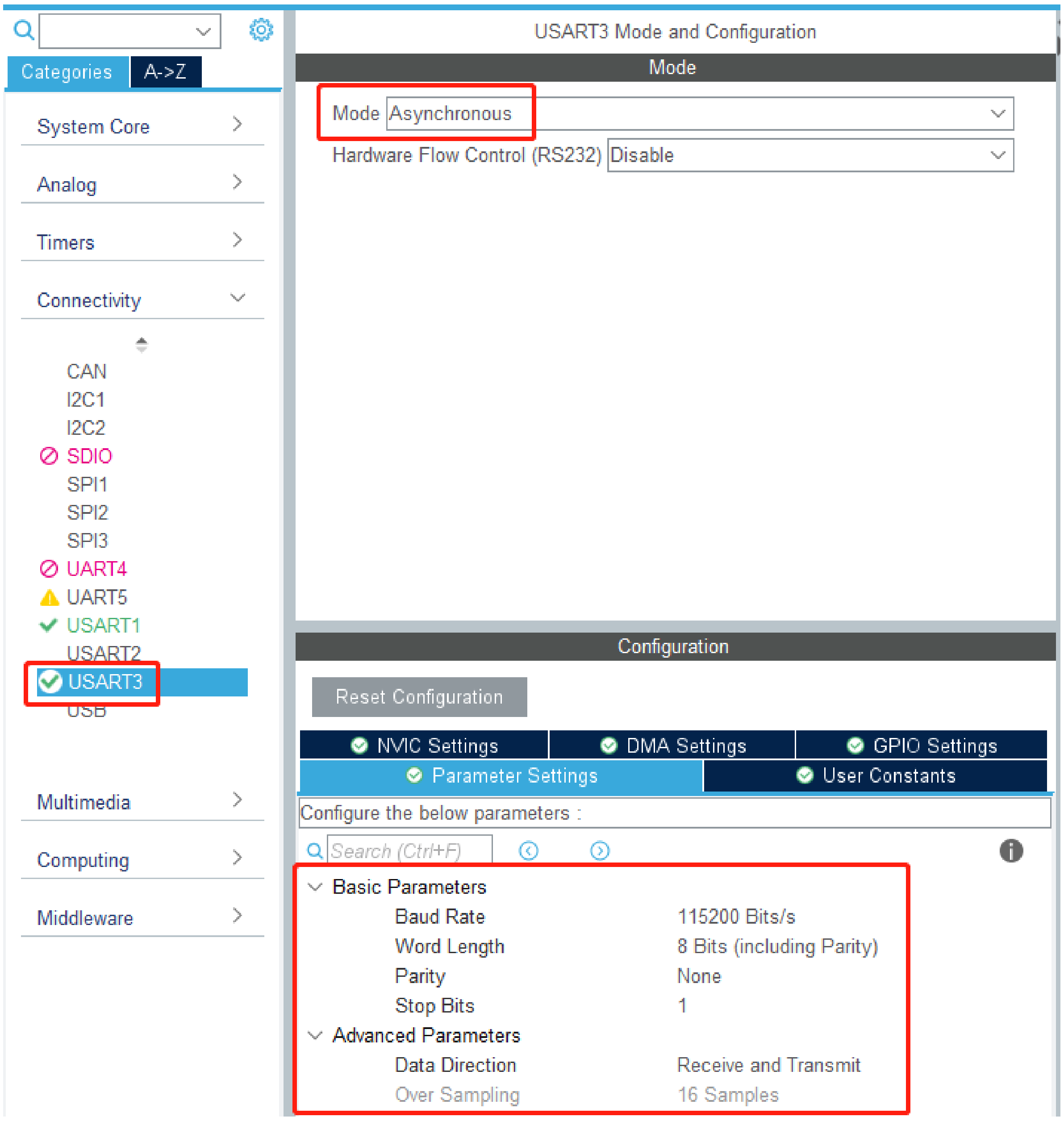
1. Import the ioc file from the Serial project and name it Serial_Servo.

According to the schematic diagram, the serial port servo is connected to the serial port 3, PC10 and PC11 pins.





2. Set serial port 3 to Asynchronous mode, and other parameters are shown in the figure below.



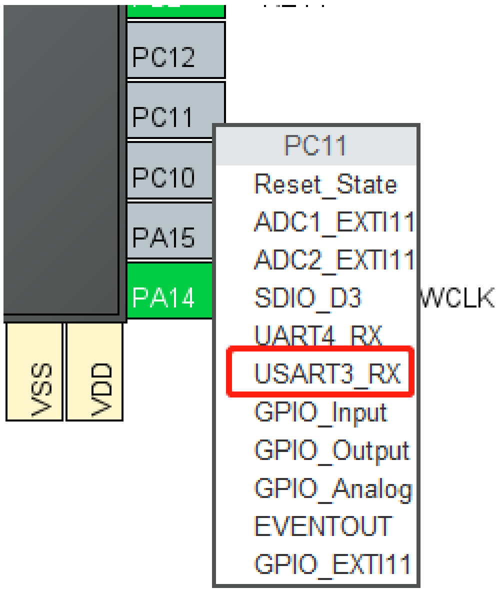
Mode: Asynchronous
Hardware Flow Control (RS232): Disable

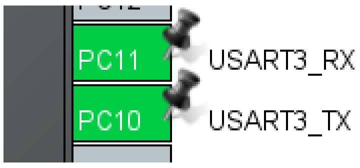
Configure the below parameters :

Basic Parameters
Baud Rate: 115200 Bits/s
Word Length: 8 Bits (including Parity)
Parity: None
Stop Bits: 1
Advanced Parameters
Data Direction: Receive and Transmit
Over Sampling: 16 Samples

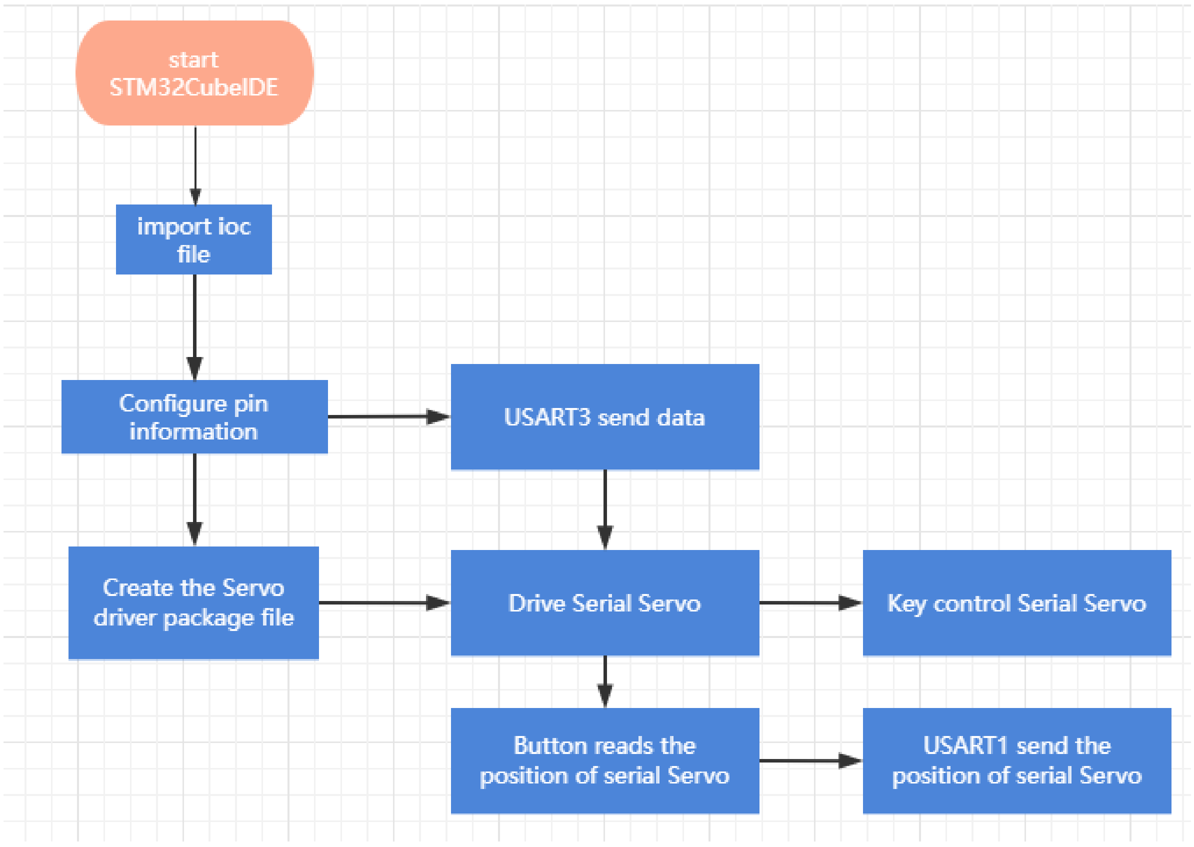
3. Since the default serial port 3 pins are PB10 and PB11, and the serial port 3 of the expansion board schematic diagram is connected to PC10 and PC11, the serial port remapping function is required.

Click the PC11 pin first, and then select USART3_RX. After this operation, the pins of serial port 3 will be remapped to PC10 and PC11.



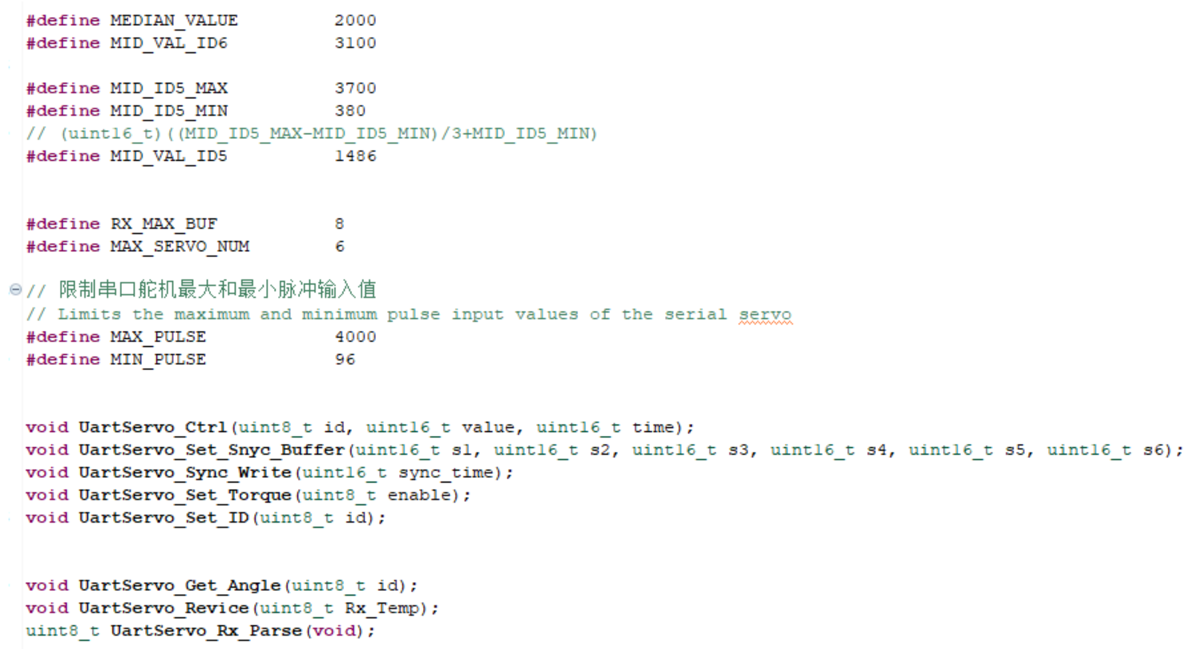


10.3. Analysis of the experimental flow chart


start STM32CubeIDE
import ioc file
Configure pin information
USART3 send data
Create the Servo driver package file
Drive Serial Servo
Key control Serial Servo
Button reads the position of serial Servo
USART1 send the position of serial Servo

10.4. core code explanation

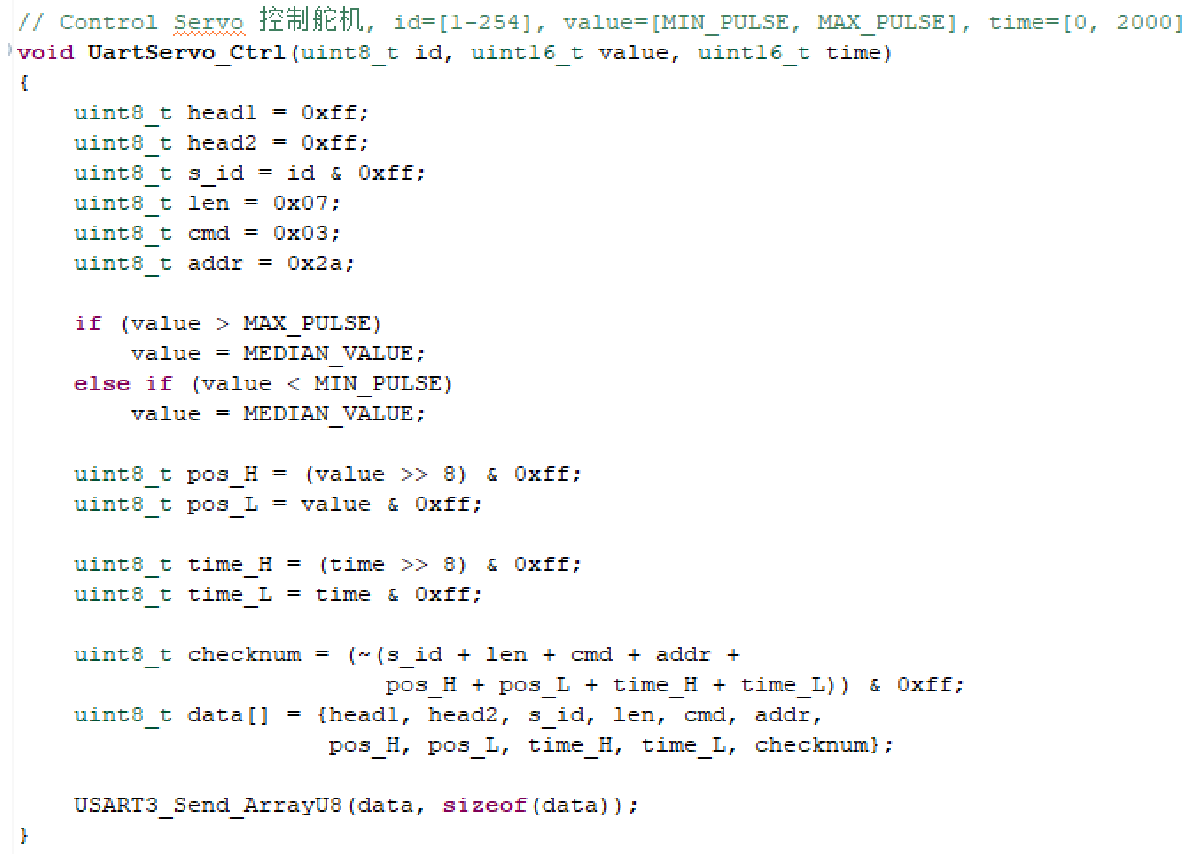
1. Create new bsp_uart_servo.h and bsp_uart_servo.c, and add the following content in bsp_uart_servo.h:



```c
#define MEDIAN_VALUE          2000
#define MID_VAL_ID6           3100

#define MID_ID5_MAX           3700
#define MID_ID5_MIN           380
// (uint16_t)((MID_ID5_MAX-MID_ID5_MIN)/3+MID_ID5_MIN)
#define MID_VAL_ID5           1486


#define RX_MAX_BUF            8
#define MAX_SERVO_NUM         6

// 限制串口舵机最大和最小脉冲输入值
// Limits the maximum and minimum pulse input values of the serial servo
#define MAX_PULSE             4000
#define MIN_PULSE             96


void UartServo_Ctrl(uint8_t id, uint16_t value, uint16_t time);
void UartServo_Set_Snyc_Buffer(uint16_t s1, uint16_t s2, uint16_t s3, uint16_t s4, uint16_t s5, uint16_t s6);
void UartServo_Sync_Write(uint16_t sync_time);
void UartServo_Set_Torque(uint8_t enable);
void UartServo_Set_ID(uint8_t id);


void UartServo_Get_Angle(uint8_t id);
void UartServo_Revice(uint8_t Rx_Temp);
uint8_t UartServo_Rx_Parse(void);
```

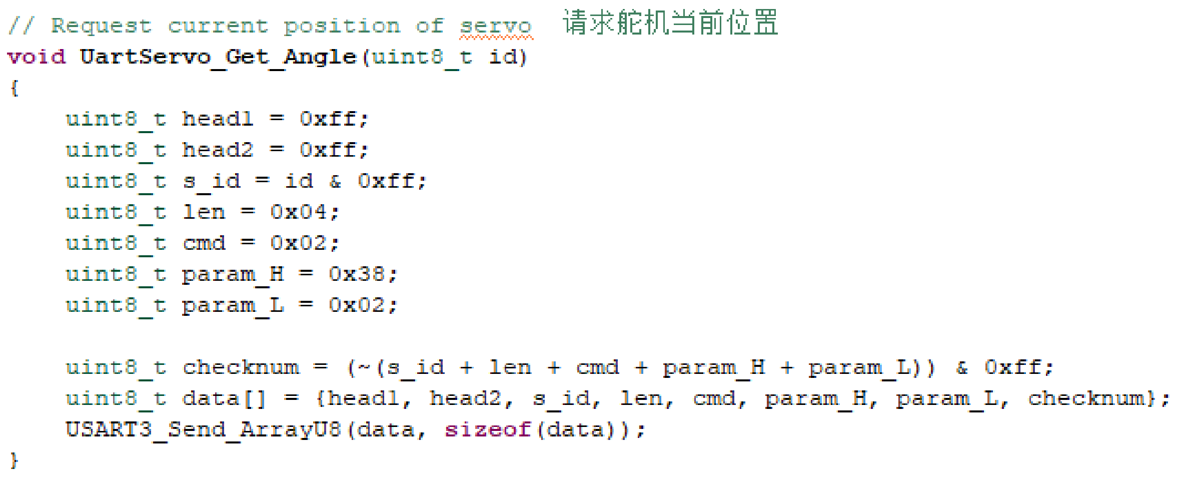
2. Create the following content in the bsp_uart_servo.c file:

According to the communication protocol of the serial servo, create a new UartServo_Ctrl(id, value, time) to control the servo. The id corresponds to the ID of the servo to be controlled. If id=0xFE(254), all servos will be controlled. value indicates the position to which the control servo moves, time indicates the running time, before reaching the maximum speed, the shorter the time, the faster the running.



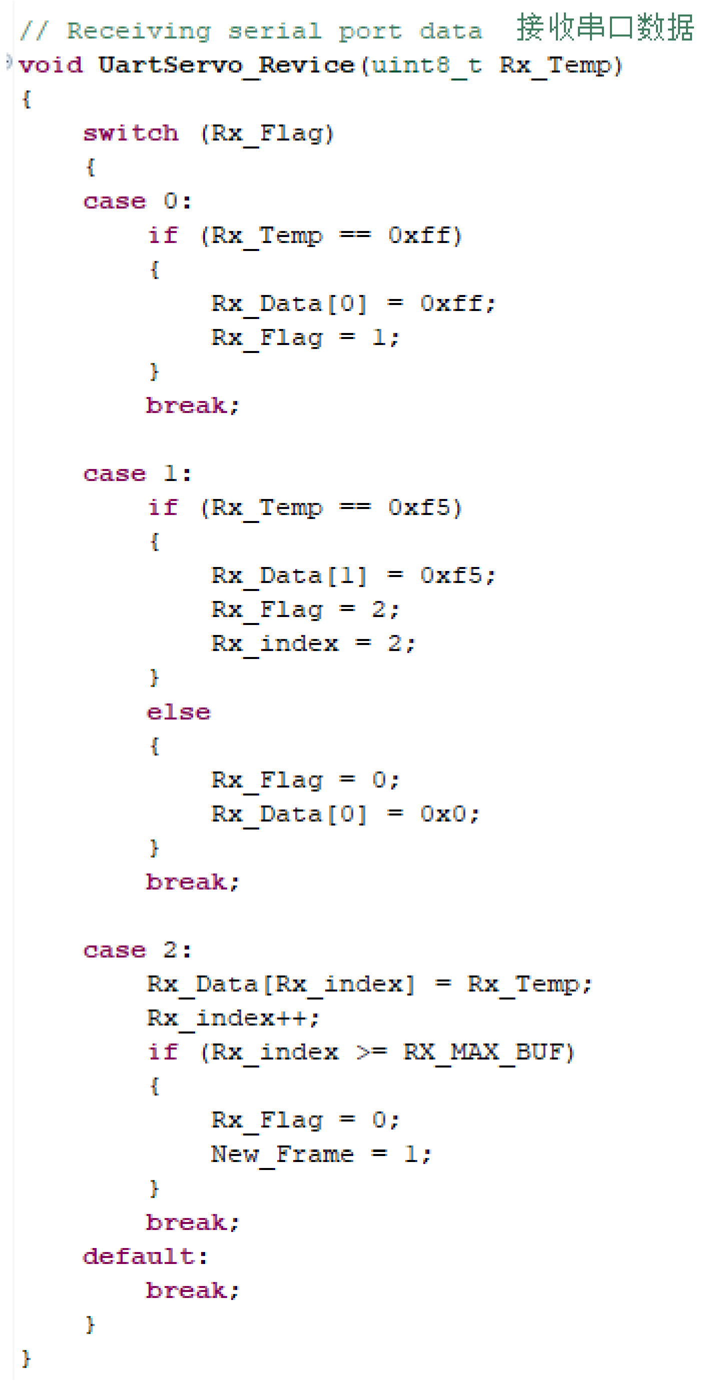
```c
// Control Servo 控制舵机, id=[1-254], value=[MIN_PULSE, MAX_PULSE], time=[0, 2000]
void UartServo_Ctrl(uint8_t id, uint16_t value, uint16_t time)
{
    uint8_t head1 = 0xff;
    uint8_t head2 = 0xff;
    uint8_t s_id = id & 0xff;
    uint8_t len = 0x07;
    uint8_t cmd = 0x03;
    uint8_t addr = 0x2a;

    if (value > MAX_PULSE)
        value = MEDIAN_VALUE;
    else if (value < MIN_PULSE)
        value = MEDIAN_VALUE;

    uint8_t pos_H = (value >> 8) & 0xff;
    uint8_t pos_L = value & 0xff;

    uint8_t time_H = (time >> 8) & 0xff;
    uint8_t time_L = time & 0xff;

    uint8_t checknum = (~(s_id + len + cmd + addr +
                          pos_H + pos_L + time_H + time_L)) & 0xff;
    uint8_t data[] = {head1, head2, s_id, len, cmd, addr,
                      pos_H, pos_L, time_H, time_L, checknum};

    USART3_Send_ArrayU8(data, sizeof(data));
}
```

3. The UartServo_Get_Angle() function requests the current position of the servo.



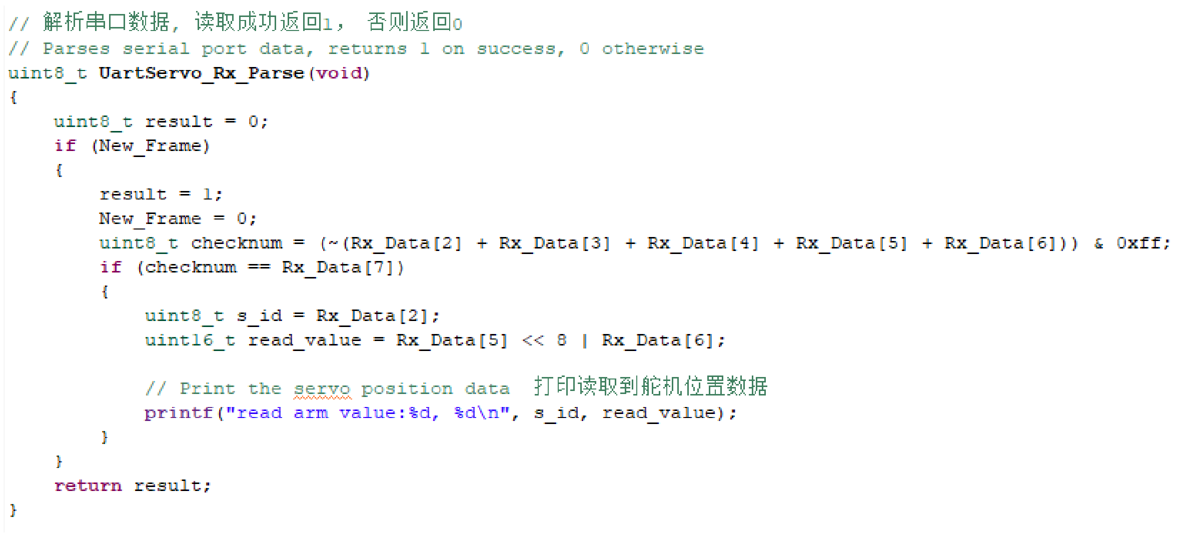
```
// Request current position of servo  请求舵机当前位置
void UartServo_Get_Angle(uint8_t id)
{
    uint8_t head1 = 0xff;
    uint8_t head2 = 0xff;
    uint8_t s_id = id & 0xff;
    uint8_t len = 0x04;
    uint8_t cmd = 0x02;
    uint8_t param_H = 0x38;
    uint8_t param_L = 0x02;

    uint8_t checknum = (~(s_id + len + cmd + param_H + param_L)) & 0xff;
    uint8_t data[] = {head1, head2, s_id, len, cmd, param_H, param_L, checknum};
    USART3_Send_ArrayU8(data, sizeof(data));
}
```

4. The UartServo_Revice(Rx_Temp) function receives the serial port 3 data to determine whether it conforms to the serial port servo communication protocol. If it conforms to a frame of data, update the Rx_Data array and set New_Frame to 1.



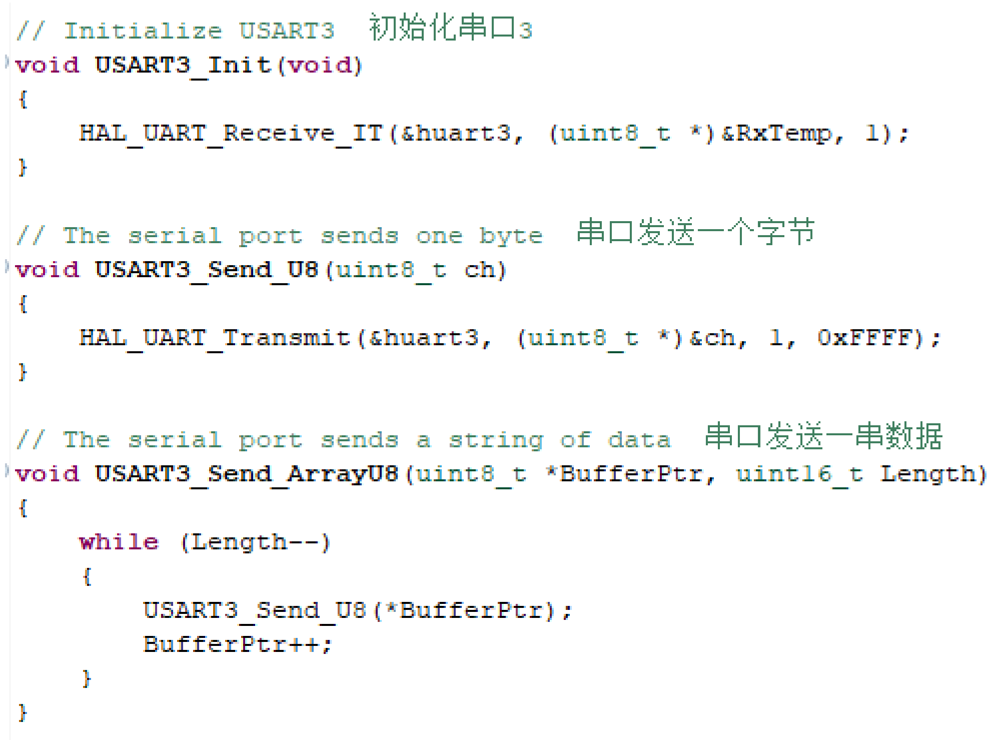
```
// Receiving serial port data  接收串口数据
void UartServo_Revice(uint8_t Rx_Temp)
{
    switch (Rx_Flag)
    {
    case 0:
        if (Rx_Temp == 0xff)
        {
            Rx_Data[0] = 0xff;
            Rx_Flag = 1;
        }
        break;

    case 1:
        if (Rx_Temp == 0xf5)
        {
            Rx_Data[1] = 0xf5;
            Rx_Flag = 2;
            Rx_index = 2;
        }
        else
        {
            Rx_Flag = 0;
            Rx_Data[0] = 0x0;
        }
        break;

    case 2:
        Rx_Data[Rx_index] = Rx_Temp;
        Rx_index++;
        if (Rx_index >= RX_MAX_BUF)
        {
            Rx_Flag = 0;
            New_Frame = 1;
        }
        break;
    default:
        break;
    }
}
```

5. Parse the data returned by the serial port servo, return 1 if the read is successful, and print the data, otherwise return 0.

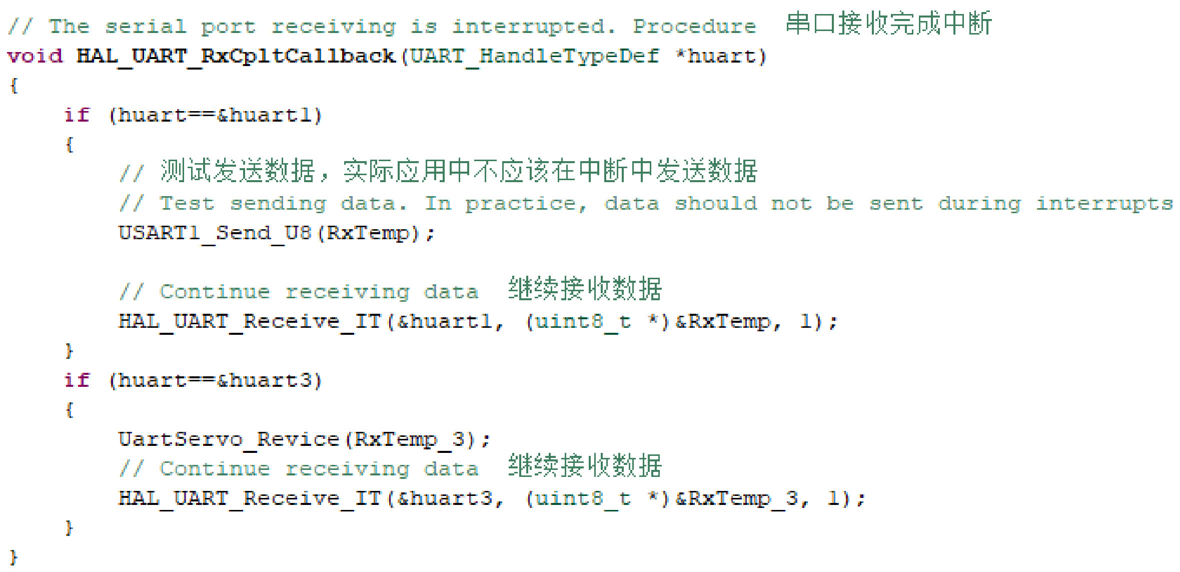


```
// 解析串口数据，读取成功返回1， 否则返回0
// Parses serial port data, returns 1 on success, 0 otherwise
uint8_t UartServo_Rx_Parse(void)
{
    uint8_t result = 0;
    if (New_Frame)
    {
        result = 1;
        New_Frame = 0;
        uint8_t checknum = (~(Rx_Data[2] + Rx_Data[3] + Rx_Data[4] + Rx_Data[5] + Rx_Data[6])) & 0xff;
        if (checknum == Rx_Data[7])
        {
            uint8_t s_id = Rx_Data[2];
            uint16_t read_value = Rx_Data[5] << 8 | Rx_Data[6];

            // Print the servo position data  打印读取到舵机位置数据
            printf("read arm value:%d, %d\n", s_id, read_value);
        }
    }
    return result;
}
```

6. Add the following functions related to writing and reading of serial port 3 in bsp_uart.c.

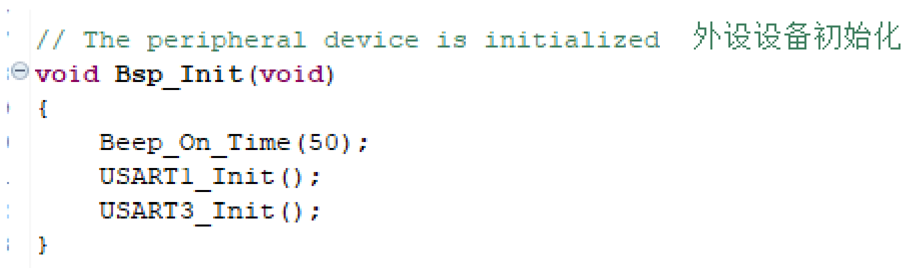


```
// Initialize USART3  初始化串口3
void USART3_Init(void)
{
    HAL_UART_Receive_IT(&huart3, (uint8_t *)&RxTemp, 1);
}

// The serial port sends one byte  串口发送一个字节
void USART3_Send_U8(uint8_t ch)
{
    HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);
}

// The serial port sends a string of data  串口发送一串数据
void USART3_Send_ArrayU8(uint8_t *BufferPtr, uint16_t Length)
{
    while (Length--)
    {
        USART3_Send_U8(*BufferPtr);
        BufferPtr++;
    }
}
```



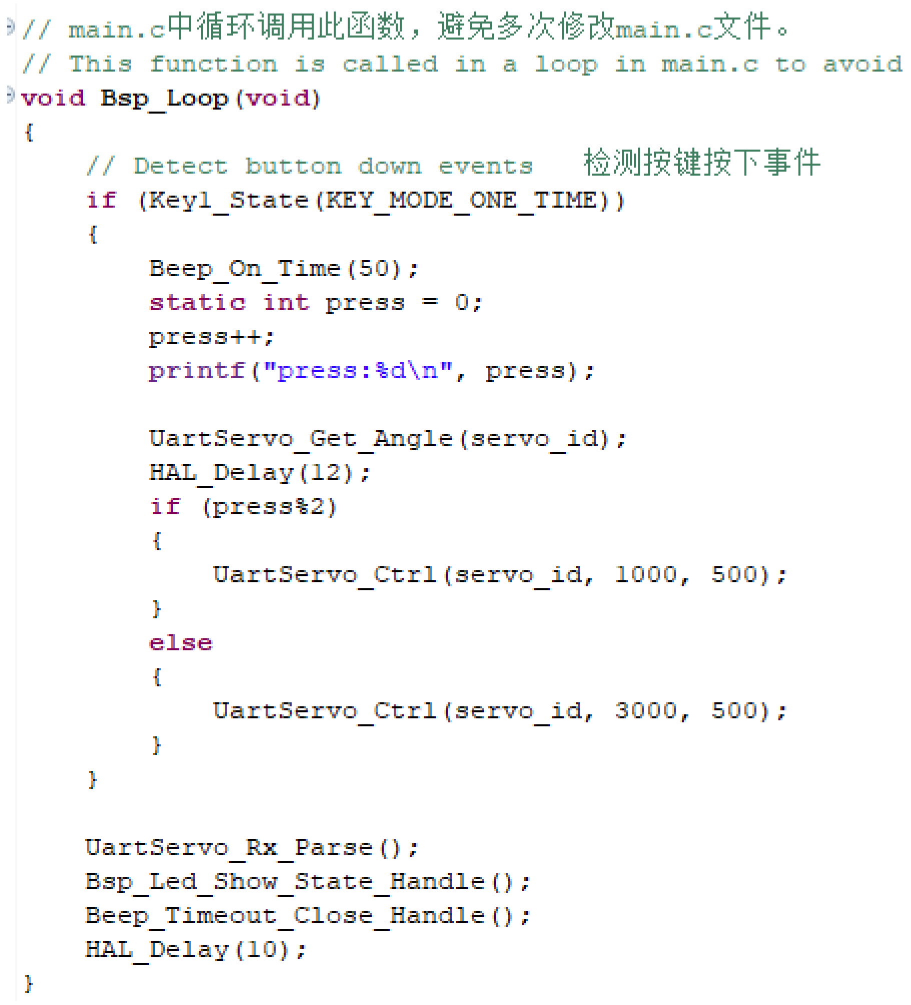
```
// The serial port receiving is interrupted. Procedure  串口接收完成中断
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart==&huart1)
    {
        // 测试发送数据，实际应用中不应该在中断中发送数据
        // Test sending data. In practice, data should not be sent during interrupts
        USART1_Send_U8(RxTemp);

        // Continue receiving data  继续接收数据
        HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
    }
    if (huart==&huart3)
    {
        UartServo_Revice(RxTemp_3);
        // Continue receiving data  继续接收数据
        HAL_UART_Receive_IT(&huart3, (uint8_t *)&RxTemp_3, 1);
    }
}
```

7. Add the content of serial port 3 initialization in the Bsp_Init() function.



```c
// The peripheral device is initialized  外设设备初始化
void Bsp_Init(void)
{
    Beep_On_Time(50);
    USART1_Init();
    USART3_Init();
}
```

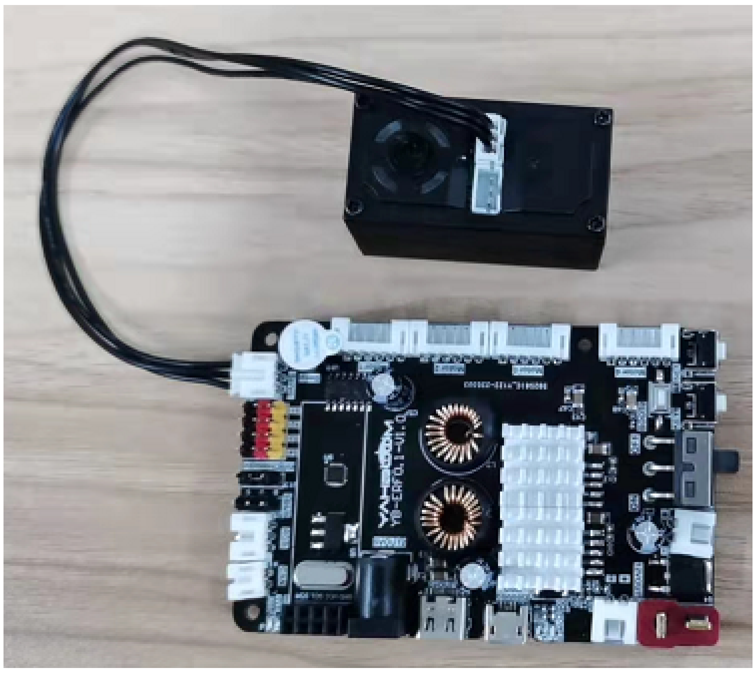
8. In the Bsp_Loop() function, add the function of key reading and controlling the serial port servo.



```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
void Bsp_Loop(void)
{
    // Detect button down events   检测按键按下事件
    if (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int press = 0;
        press++;
        printf("press:%d\n", press);

        UartServo_Get_Angle(servo_id);
        HAL_Delay(12);
        if (press%2)
        {
            UartServo_Ctrl(servo_id, 1000, 500);
        }
        else
        {
            UartServo_Ctrl(servo_id, 3000, 500);
        }
    }

    UartServo_Rx_Parse();
    Bsp_Led_Show_State_Handle();
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

10.5. Hardware connection

The serial port servo needs to be connected to the serial port servo port on the expansion board. The serial port servo port has the function of preventing reverse connection, and it can be inserted in the direction. Multiple serial servos can be cascaded. Due to the limited power supply current of the expansion board, do not connect too many servos. At present, six servos can be used normally.



Because the power of the serial port servo is relatively large, the expansion board should not be powered by USB 5V directly, it needs to be powered by DC 12V.

10.6. Experimental effect

After the program is programmed, the LED light flashes every 200 milliseconds. Press the button multiple times, the serial servo will go back and forth between the 1000 position and the 3000 position, and return to the position data before the movement.