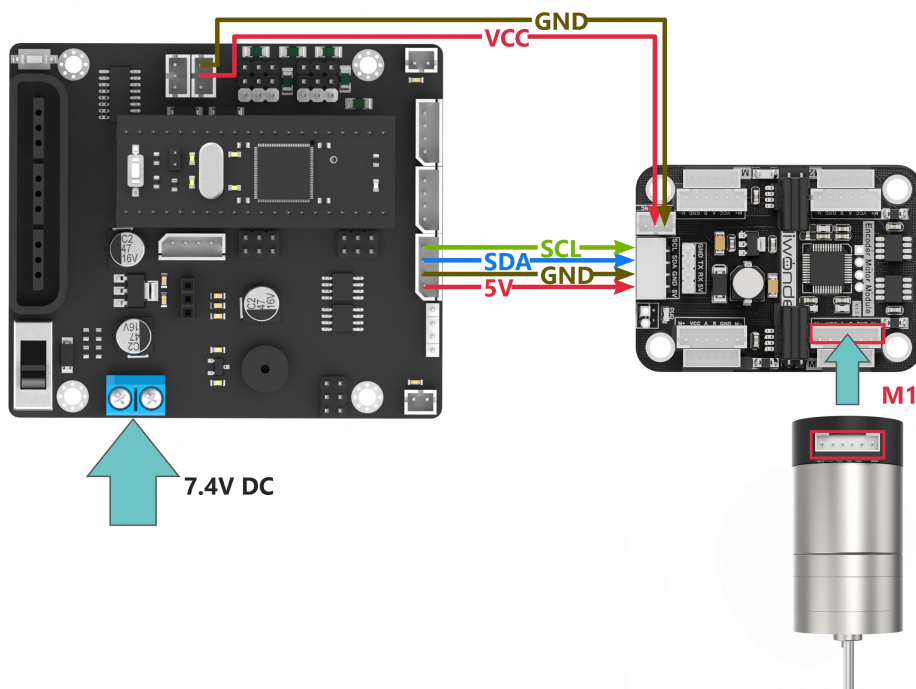


STM32 Development

1. Hardware Connection

This lesson employs the STM32F103 development board for control demonstration. The specific connection method is shown below:



Software Preparation:

According to the instruction in "Set Development Environment" in this folder, install and debug the Keil tool.

Note:

1. Please solder the pins on the power interface. Connect the power supply via a male-to-female connector, with 5V to 5V and GND to GND.

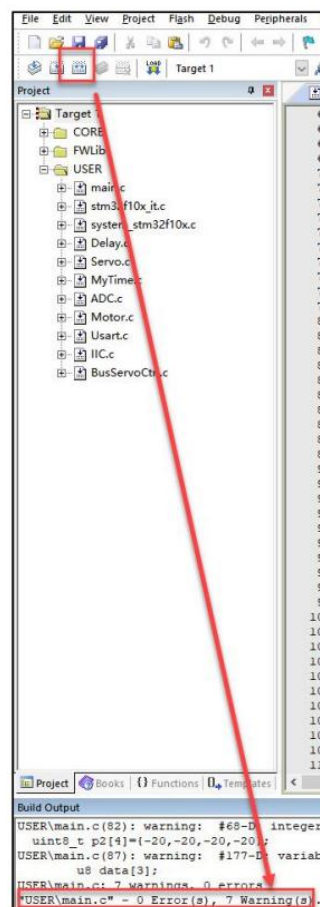
2. This diagram only shows the connection of one motor. If you want to connect more motors, you can refer to this wiring method.

2. Program Download

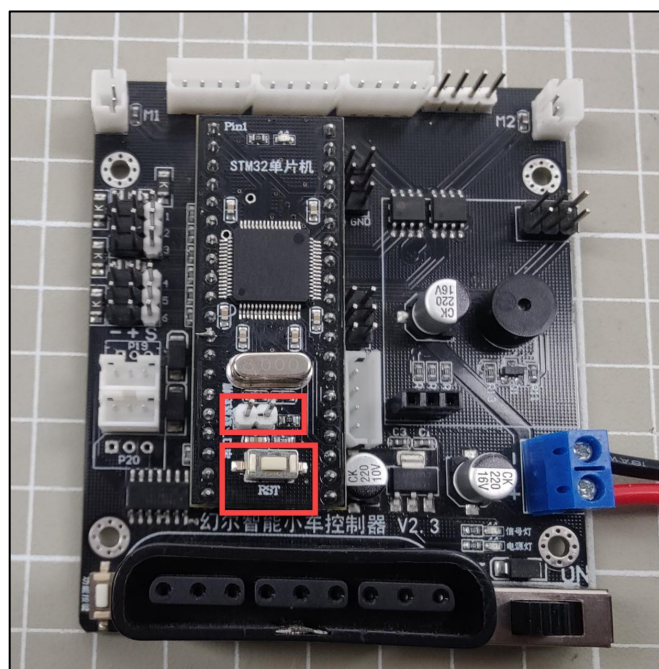
- 1) Connect the control board to any USB interface of your computer with a USB to TTL tool.
- 2) Open the STM32 case program in this folder.


名称	修改日期	类型	大小
CORE	2022/7/20 16:57	文件夹	
Obj	2022/7/20 18:25	文件夹	
STM32F10x_FWLib	2022/7/20 16:57	文件夹	
USER	2022/7/20 18:25	文件夹	
OpenArmSTM32.uvgui.1234	2022/7/20 16:56	1234 文件	70 KB
OpenArmSTM32.uvgui.98569	2022/7/20 18:25	98569 文件	69 KB
OpenArmSTM32.uvgui.Administrator	2020/5/8 11:29	ADMINISTRATO...	71 KB
OpenArmSTM32.uvgui.cp	2018/6/8 17:03	CP 文件	70 KB
OpenArmSTM32.uvgui.pc	2018/6/6 17:46	PC 文件	70 KB
OpenArmSTM32.uvgui.Xia	2018/4/12 20:16	XIA 文件	143 KB
OpenArmSTM32.uvgui.Zheng	2018/4/12 20:16	ZHENG 文件	73 KB
OpenArmSTM32.uvgui_1234.bak	2022/7/20 11:42	BAK 文件	70 KB
OpenArmSTM32.uvgui_98569.bak	2022/7/20 18:25	BAK 文件	69 KB
OpenArmSTM32.uvgui_Administrator....	2019/11/11 17:07	BAK 文件	70 KB
OpenArmSTM32.uvgui_cp.bak	2018/6/8 16:59	BAK 文件	68 KB
OpenArmSTM32.uvgui_pc.bak	2018/6/6 16:22	BAK 文件	70 KB
OpenArmSTM32.uvgui_Xia.bak	2018/4/12 20:16	BAK 文件	143 KB
OpenArmSTM32.uvopt	2022/7/20 18:25	UVOPT 文件	25 KB
OpenArmSTM32.uvproj	2022/7/20 16:58	碘ision4 Project	22 KB
OpenArmSTM32_target_1.dep	2022/7/21 10:10	DEP 文件	100 KB
OpenArmSTM32_uvopt.bak	类型: 碘ision4 Project 大小: 21.4 KB	BAK 文件	25 KB
OpenArmSTM32_uvproj.bak	修改日期: 2022/7/20 16:58	BAK 文件	22 KB

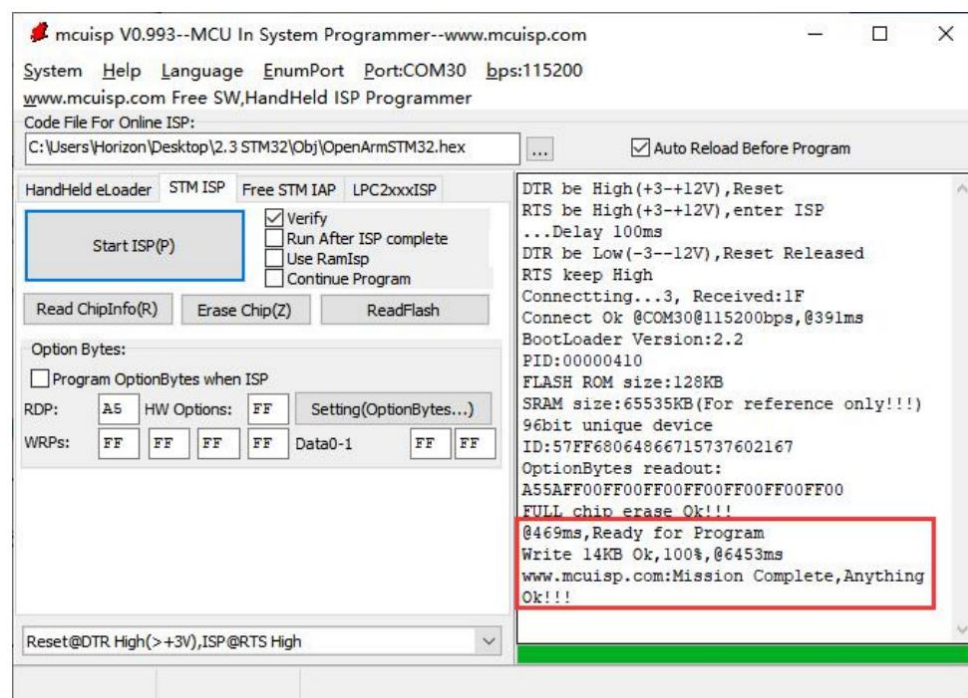
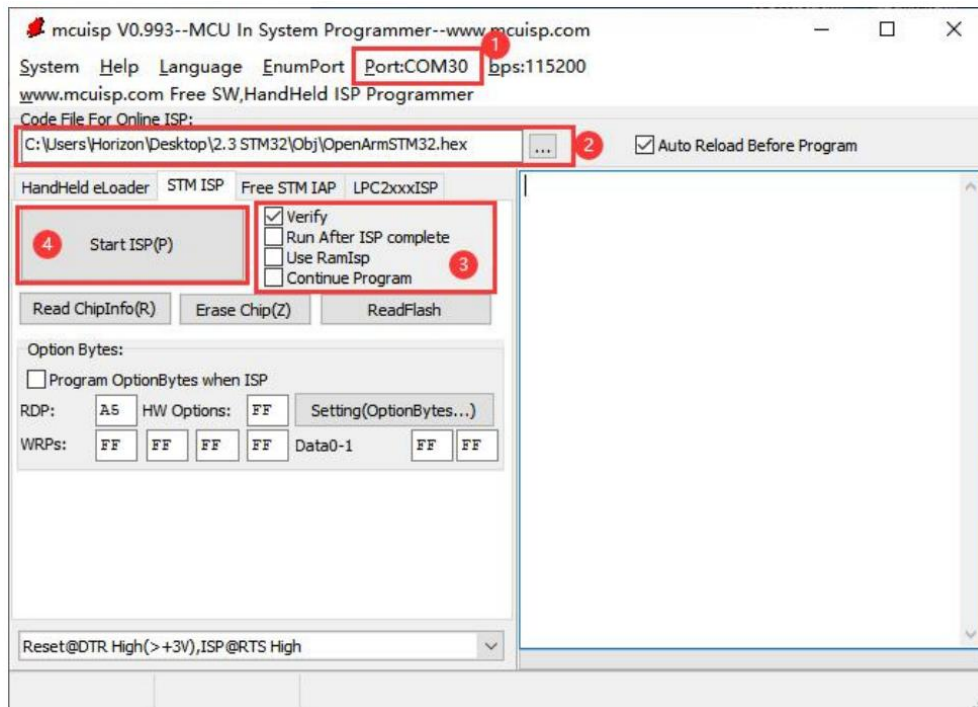
- 3) Generate an executable file for all the code.




- 4) Remove the jumper cap on the STM32 development board, and press the RST button.

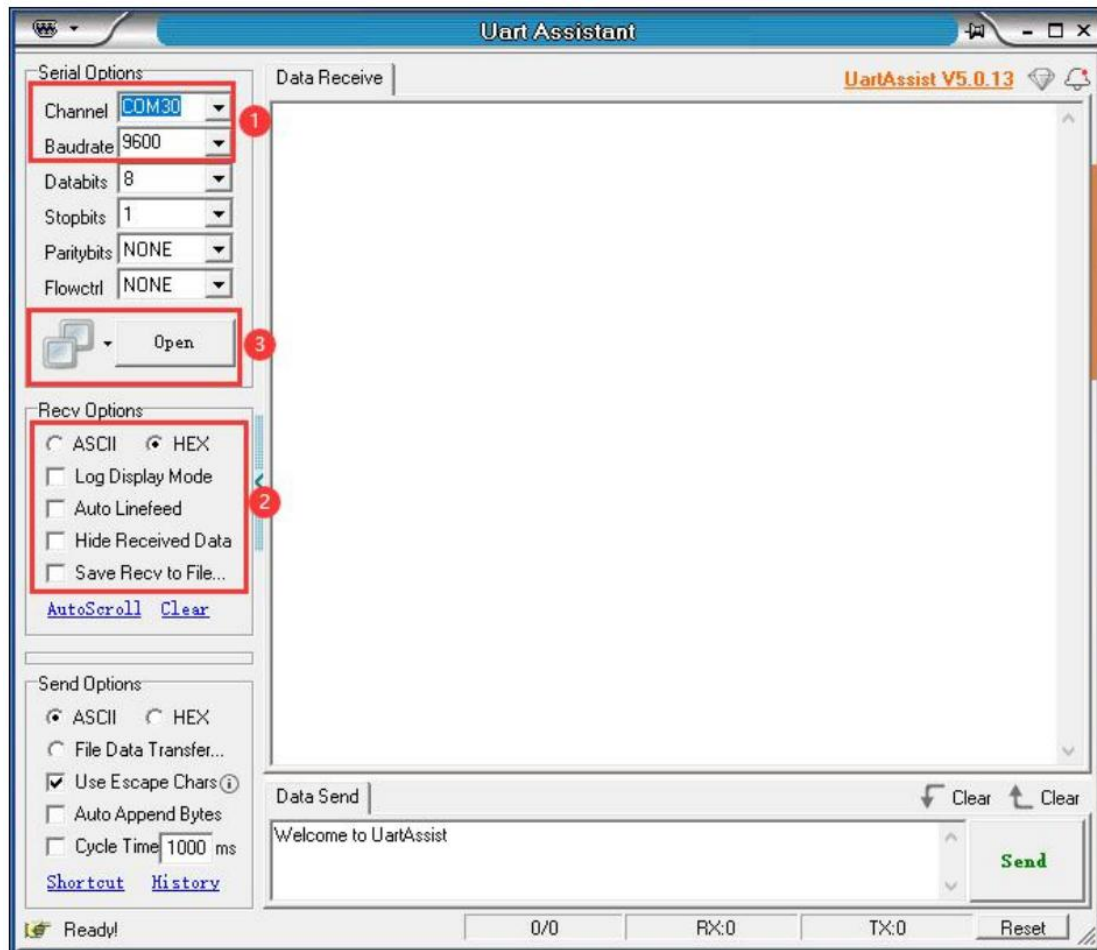


- 5) Open the program flashing tool . Select the correct port. Keep the baud rate unchanged. Flash the program into the development board. For specific instructions, please refer to the following figure:





- 6) After the flashing is completed, double-click the executable file  in this folder to open the serial port utility. Select CH340 as the serial port number and 9600 as the baud rate. Adjust the parameters as shown in the following figure.



- 7) Insert the jumper cap, and press the RST button to run the program.

3. Program Outcome

After powering on, STM32 will print the power voltage of the 4-ch encoder motor driver on the serial port. The motors will be controlled to rotate clockwise for 3 seconds and counterclockwise for 3 seconds. Then, the total

accumulated pulse value of the four encoder motors will be printed on the serial port. Next, the motors will stop. The total pulse value of all encoder motors will be reset.

4. Program Analysis

The overall process of the program is to prepare the necessary libraries and the relevant parameters for motor initialization. Then, control the motor through logical methods.

● Import Necessary Libraries

The include head file is used for I2C communication.

```
#include "include.h"
```

● Initialize communication address

Define the I2C communication address and the address of each type of encoder motor for easy calling later. The I2C communication address connected to the driver board is 0x34, which depends on your own hardware device. The type of the encoder motor is set to 0x14. Its direction polarity is set to 0x15. The above two numbers represent the address position of the written parameters, not the actual parameter values. Their values depend on your hardware device. Keep them default.

```
#define CAM_DEFAULT_I2C_ADDRESS    (0x34)    //I2C地址(I2C address)
#define MOTOR_TYPE_ADDR            0x14      //编码电机类型设置寄存器地址(Encoder motor type setting register address)
#define MOTOR_FIXED_SPEED_ADDR     0x33      //速度寄存器地址。属于闭环控制(Speed register address; belongs to closed-loop control)
#define MOTOR_ENCODER_POLARITY_ADDR 0x15      //电机编码方向极性地址(Motor encoder direction polarity address)
#define MOTOR_FIXED_PWM_ADDR       0x1F      //固定PWM控制地址。属于开环控制(Fixed PWM control address; belongs to open-loop control)
#define MOTOR_ENCODER_TOTAL_ADDR   0x3C      //4个编码电机各自的总脉冲值(Total pulse value of each of the 4 encoding motors)
#define ADC_BAT_ADDR               0x00      //电压地址(Voltage address)
```

● Module initialization

The motor type is initialized. The 4-ch encoder motor driver supports various types of motors, including TTL, N20, and JGB motors. All these specific types are defined. In this case, JGB motors are used. Therefore, the motor model is set to 3. The motor polarity is set to 0 simultaneously. If it is set to 1, the motors will rotate clockwise continuously. This makes it impossible to control via the code.

```
int main(void)
{
    u16 v; //用于暂存电压值(Used to temporarily store voltage)
    SystemInit(); //系统时钟初始化(Initialize system clock)
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //设置NVIC中断分组2:2位抢占优先级,2位响应优先级(Set NVIC interrupt group 2: 2-bit preemption priority, 2-bit response priority)
    InitDelay(72); //初始化延时函数(Initialize delay function)
    InitTimer2(); //定时器2初始化(Timer 2 initialization)
    IIC_Init(); //IIC初始化(IIC initialization)
    Usart1_Init(); //串口初始化(Serial port initialization)
    InitLED(); //初始化LED指示灯(Initialize LED indicator)
    DelayMs(200); //在电机类型地址中写入电机类型编号(Write the motor type number to the motor type address)
    I2C_Write_Len(MOTOR_TYPE_ADDR, &MotorType, 1);
    DelayMs(5);
    I2C_Write_Len(MOTOR_ENCODER_POLARITY_ADDR, &MotorEncoderPolarity, 1); //设置电机极性(Set motor polarity)
    DelayMs(5);
}
```

● Variable and array definition

In this case, the I2C communication is mainly called to send data collection. The motor driver module can control the rotation of four motors simultaneously. Encapsulate the rotation speed of the four motors in an array and send it to the motor driver module. After receiving the data, the driver module will control the motor to rotate based on the set speed. Note: If it is PWM control, continuous sending is required. Any delay in the program will affect the motor rotation speed!

```
/*用数组传递电机速度, 正数为设置前进速度, 负数为设置后退速度(Use an array to pass motor speed, positive number is set to forward speed, negative number is set to reverse speed)
   p1: p2为电机速度, p1=4个电机以50的速度前进 p2=4个电机以20的速度后退 (Take p1 and p2 as examples: p1 = 4 motors with a speed of 50 forward p2 = 4 motors with a speed of 20 backward)*/
uint8_t p1[4]={50,50,50,50}; //电机速度设置(set motor speed)
uint8_t p2[4]={-20,-20,-20,-20}; //电机速度设置(set motor speed)
uint8_t stop[4]={0,0,0,0}; //用于重置脉冲宽度(reset the pulse width)
uint8_t EncoderReset[16]={0}; //电机极性控制变量(motor polarity control variable)
uint8_t MotorEncoderPolarity = 0; //用于暂存电机累积转动量的值, 正转递增, 反转递减(Used to temporarily store the accumulated rotation of the motor, increasing when rotating forward, decreasing when rotating backward)
uint32_t EncoderTotal[4]; //设置电机类型(set motor type)
int8_t MotorType = MOTOR_TYPE_JGB37_520_12V_110RPM; //用于暂存电压ADC数据(used to temporarily store voltage ADC data)
uint6_t data[3];
```

● Main Function

In the main function, use the “I2C_Read_Len()” function to read the power voltage of the 4-ch encoder motor driver. Call the “I2C_Write_Len()” function to write the motor control speed. The motors will be controlled to rotate clockwise for 3 seconds and counterclockwise for 3 seconds. Then, the total

accumulated pulse value of the four encoder motors will be printed on the serial port. The “MOTOR_FIXED_PWM_ADDR” represents the PWM speed control address. The “MOTOR_FIXED_SPEED_ADDR” represents the fixed speed control address. The “MOTOR_ENCODER_TOTAL_ADDR” is read to obtain the total pulse values of the motors. Then, write a stop array with four elements of 0 to control the motors to stop rotating. The p1, p2, and stop arrays are used to control the motor speed. The corresponding motor speed value in the array can be adjusted to achieve different motion effects. Next, write 0 to the “MOTOR_ENCODER_TOTAL_ADDR”. This can clear the pulse values of the motors.

```
while(1)
{
    I2C_Read_Len(ADC_BAT_ADDR,data,2);           //读取电机电压(Read motor voltage)
    v = data[0] + (data[1]<<8);                  //转换电压(Convert voltage)
    printf("V = "); printf("%d",v); printf("\n"); //打印电压(Print voltage)

    /*在电机转速地址中写入电机的转动方向和速度: WireWriteDataArray (转速控制地址, 电机转速数组, 电机个数)
    (Write the motor rotation direction and speed to the motor speed address: WireWriteDataArray (speed control address, motor speed array, and number of motors))*/
    I2C_Write_Len(MOTOR_FIXED_SPEED_ADDR,p1,4); //控制电机转动(Control motor rotation)
    DelayMs(3000);
    I2C_Write_Len(MOTOR_FIXED_SPEED_ADDR,p2,4); //控制电机转动(Control motor rotation)
    DelayMs(3000);

    //I2C_Write_Len(MOTOR_FIXED_PWM_ADDR,p1,4); //PWM控制(PWM控制程序中不能有延时, 否则会停止电机运行, 打印也有一定的延时) (PWM control
    //DelayMs(3000); //There should be no delay in the PWM control program; otherwise, it will interrupt motor operation, and printing also introduces some delays))

    I2C_Read_Len(MOTOR_ENCODER_TOTAL_ADDR,(uint8_t*)EncodeTotal,16);
    printf("Encode1 = %ld Encode2 = %ld Encode3 = %ld Encode4 = %ld \r\n", EncodeTotal[0], EncodeTotal[1], EncodeTotal[2], EncodeTotal[3]);

    I2C_Write_Len(MOTOR_FIXED_SPEED_ADDR,stop,4); //控制电机停止(control the motor to stop)
    DelayMs(1000);

    I2C_Write_Len(MOTOR_ENCODER_TOTAL_ADDR,EncodeReset,16); //重置脉冲值(reset pulse value)
    I2C_Read_Len(MOTOR_ENCODER_TOTAL_ADDR,(uint8_t*)EncodeTotal,16);
    printf("Encode1 = %ld Encode2 = %ld Encode3 = %ld Encode4 = %ld \r\n", EncodeTotal[0], EncodeTotal[1], EncodeTotal[2], EncodeTotal[3]);
    DelayMs(3000);
}
```