Control car and read encoder-IIC

1.1 Explanation

Please read 《0. Motor introduction and usage》first to understand the motor parameters, wiring method, and power supply voltage you are currently using. To avoid improper operation and damage to the driver board or motor.

I2C and serial communication cannot be shared, only one can be selected.

The 4 motor interfaces on the module correspond to motors on robot car, as shown below

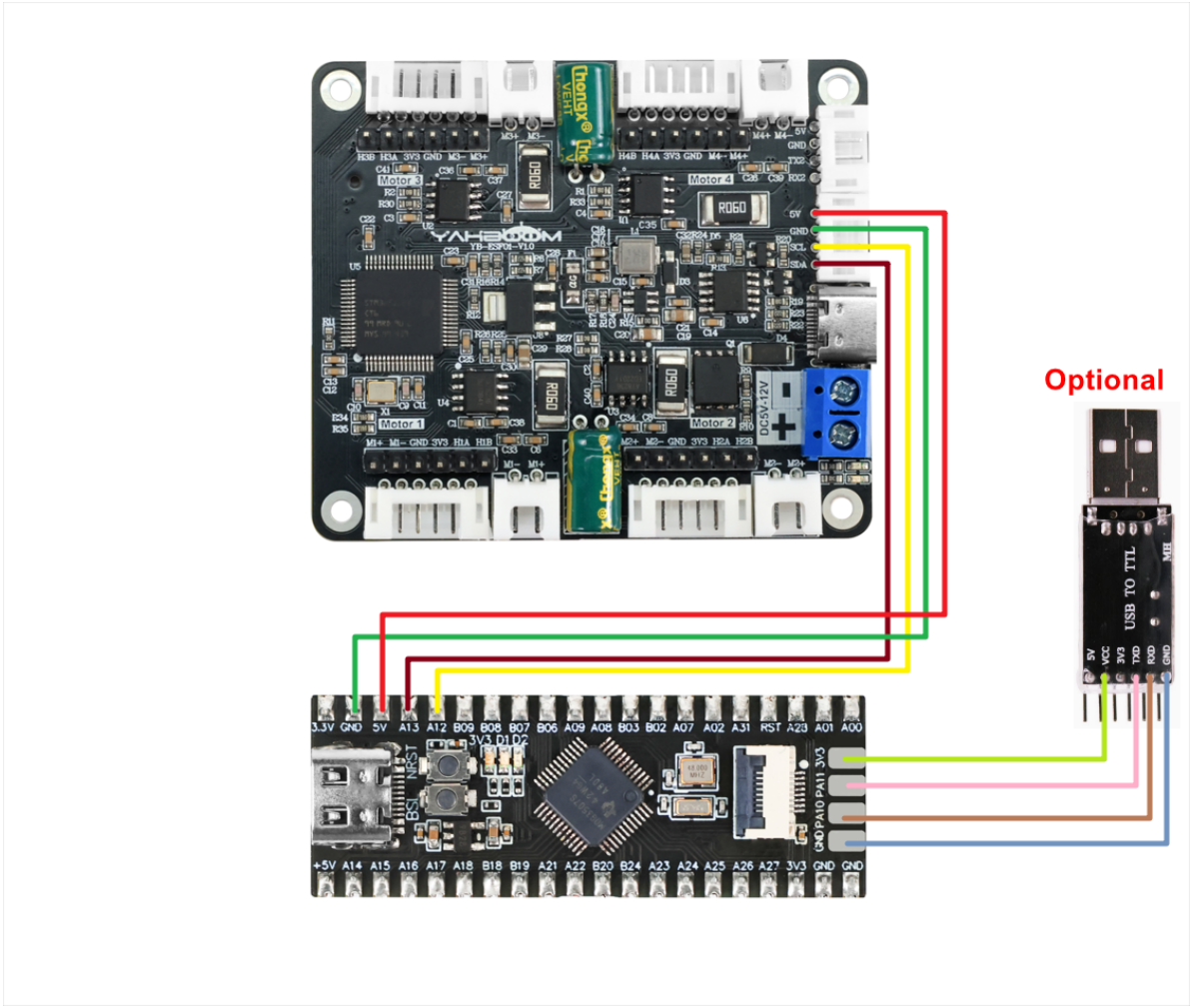
M1 -> Upper left motor (left front wheel of the car)

M2 -> Lower left motor (left rear wheel of the car)

M3 -> Upper right motor (right front wheel of the car)

M4 -> Lower right motor (right rear wheel of the car)

Hardware wiring:



4-channel motor drive board	MSPM0G3507
SDA	PA13
SCL	PA12
GND	GND
5V	5V

Motor	4-channel motor drive board(Motor)
M2	M-
V	3V3
A	H1B
B	H1A
G	GND
M1	M+

USB to TTL serial port module needs to be connected to print the processed encoder data.

If you are using Yahboom MSPM0G3507, you can directly connect the TYPE-C interface on the MSPM0G3507 development board to the computer USB port, and you can also achieve serial port printing, so you don't need to connect a USB to TTL serial port module.

USB TO TTL	MSPM0G3507
VCC	3V3
GND	GND
RXD	PA10
TXD	PA11

Serial port configuration: **Baud rate 115200, no parity check, no hardware flow control, 1 stop bit**

Note: The serial port here is used to print data on the serial port assistant, not for communication with the driver board

1.2 Code analysis

```
#define UPLOAD_DATA 2  // 1:接收总的编码器数据 2:接收实时的编码器 1: Receive total encoder data 2: Receive real-time encoder data

#define MOTOR_TYPE 1   //1:520电机 2:310电机 3:测速码盘TT电机 4:TT直流减速电机 5:L型520电机
                       //1:520 motor 2:310 motor 3:speed code disc TT motor 4:TT DC reduction motor 5:L type 520 motor
```

- UPLOAD_DATA: used to set the data of the motor encoder. Set 1 to the total number of encoder pulses and 2 to the real-time pulse data of 10ms.
- MOTOR_TYPE: used to set the type of motor used. Just modify the corresponding numbers according to the comments according to the motor you are currently using. You don't need to modify the rest of the code.

If you need to drive the motor and observe the data, just modify the two numbers at the beginning of the program. No changes are required to the rest of the code.

```
    #if MOTOR_TYPE == 1
    Set_motor_type(1);//配置电机类型  Configure motor type
    delay_ms(100);
    Set_Pluse_Phase(30);//配置减速比 查电机手册得出 Configure the reduction ratio.
Check the motor manual to find out
    delay_ms(100);
    Set_Pluse_line(11);//配置磁环线 查电机手册得出  Configure the magnetic ring wire.
Check the motor manual to get the result.
    delay_ms(100);
    Set_Wheel_dis(67.00);//配置轮子直径,测量得出      Configure the wheel diameter
and measure it
    delay_ms(100);
    Set_motor_deadzone(1900);//配置电机死区,实验得出  Configure the motor dead zone,
and the experiment shows
    delay_ms(100);

    #elif MOTOR_TYPE == 2
    Set_motor_type(2);
    delay_ms(100);
    Set_Pluse_Phase(20);
    delay_ms(100);
    Set_Pluse_line(13);
    delay_ms(100);
    Set_Wheel_dis(48.00);
    delay_ms(100);
    Set_motor_deadzone(1600);
    delay_ms(100);

    #elif MOTOR_TYPE == 3
    Set_motor_type(3);
    delay_ms(100);
    Set_Pluse_Phase(45);
    delay_ms(100);
    Set_Pluse_line(13);
    delay_ms(100);
    Set_Wheel_dis(68.00);
    delay_ms(100);
    Set_motor_deadzone(1250);
    delay_ms(100);

    #elif MOTOR_TYPE == 4
    Set_motor_type(4);
    delay_ms(100);
    Set_Pluse_Phase(48);
    delay_ms(100);
    Set_motor_deadzone(1000);
```

```
    delay_ms(100);

    #elif MOTOR_TYPE == 5
    Set_motor_type(1);
    delay_ms(100);
    Set_Pluse_Phase(40);
    delay_ms(100);
    Set_Pluse_line(11);
    delay_ms(100);
    Set_Wheel_dis(67.00);
    delay_ms(100);
    Set_motor_deadzone(1900);
    delay_ms(100);
    #endif
```

This is used to store the parameters of the Yahboom motor. By modifying the MOTOR_TYPE parameter above, one-click configuration can be achieved.

In normally, do not modify the code here when using the Yahboom motor.

If you are using your own motor, or if a certain data needs to be modified according to your needs, you can check the course《1.2 Control command》 to understand the specific meaning of each command.

```
    while(1)
    {
        if(times>=25)//定时器每100ms累加一次，当达到25次，即2.5秒的时候转换一次小车的状态
The timer accumulates every 100ms, and when it reaches 25 times, that is, every
2.5 seconds, the state of the car is changed.
        {
            #if MOTOR_TYPE == 4
            Car_Move_PWM();
            #else
            Car_Move();
            #endif
            times = 0;
        }
        #if UPLOAD_DATA == 1
        Read_ALL_Enconder();
        printf("M1:%d\t M2:%d\t M3:%d\t M4:%d\t
\r\n",Encoder_Now[0],Encoder_Now[1],Encoder_Now[2],Encoder_Now[3]);
        #elif UPLOAD_DATA == 2
        Read_10_Enconder();
        printf("M1:%d\t M2:%d\t M3:%d\t M4:%d\t
\r\n",Encoder_Offset[0],Encoder_Offset[1],Encoder_Offset[2],Encoder_Offset[3]);
        #endif
    }

}

//定时器的中断服务函数,每100ms中断一次
//The timer's interrupt service function interrupts once every 100ms
void TIMER_0_INST_IRQHandler(void)
{
```

```
    //如果产生了定时器中断    If a timer interrupt occurs
    switch( DL_TimerG_getPendingInterrupt(TIMER_0_INST) )
    {
        case DL_TIMER_IIDX_ZERO://如果是0溢出中断  If it is 0 overflow interrupt
            times++;
            break;

        default:
            break;
    }
}
```

A 100ms timer is set in the program. Each time the timer interrupt is triggered, the times variable will be incremented by one. When it reaches 25 times, that is, 2.5 seconds, the motion state of the car will be changed.

If the motor type is 4, that is, the motor without encoder, then the pwm version of the car state switching function is used. At the same time, read the data sent by the driver board and print the data out at the same time.

```
//读取相对时间的编码器的数据 10ms的   Read the data of the encoder of relative time
10ms
void Read_10_Enconder(void)
{
    static int8_t buf[2];

    //M1电机编码器的数据    M1 motor encoder data
    i2cRead(Motor_model_ADDR, READ_TEN_M1Enconer_REG, 2, buf);
    Encoder_Offset[0] = buf[0]<<8|buf[1];

    //M2电机编码器的数据    M2 motor encoder data
    i2cRead(Motor_model_ADDR, READ_TEN_M2Enconer_REG, 2, buf);
    Encoder_Offset[1] = buf[0]<<8|buf[1];

    //M3电机编码器的数据    M3 motor encoder data
    i2cRead(Motor_model_ADDR, READ_TEN_M3Enconer_REG, 2, buf);
    Encoder_Offset[2] = buf[0]<<8|buf[1];

    //M4电机编码器的数据    M4 motor encoder data
    i2cRead(Motor_model_ADDR, READ_TEN_M4Enconer_REG, 2, buf);
    Encoder_Offset[3] = buf[0]<<8|buf[1];

}

//读取电机转动的编码器数据  Read the encoder data of the motor rotation
void Read_ALL_Enconder(void)
{
    static uint8_t buf[2];
    static uint8_t buf2[2];

    //M1电机编码器的数据    M1 motor encoder data
    i2cRead(Motor_model_ADDR, READ_ALLHigh_M1_REG, 2, buf);
    i2cRead(Motor_model_ADDR, READ_ALLLOW_M1_REG, 2, buf2);
```

```
    Encoder_Now[0] = buf[0]<<24|buf[1]<<16|buf2[0]<<8|buf2[1];

    //M2电机编码器的数据    M2 motor encoder data
    i2cRead(Motor_model_ADDR, READ_ALLHigh_M2_REG, 2, buf);
    i2cRead(Motor_model_ADDR, READ_ALLLOW_M2_REG, 2, buf2);
    Encoder_Now[1] = buf[0]<<24|buf[1]<<16|buf2[0]<<8|buf2[1];

    //M3电机编码器的数据    M3 motor encoder data
    i2cRead(Motor_model_ADDR, READ_ALLHigh_M3_REG, 2, buf);
    i2cRead(Motor_model_ADDR, READ_ALLLOW_M3_REG, 2, buf2);
    Encoder_Now[2] = buf[0]<<24|buf[1]<<16|buf2[0]<<8|buf2[1];

    //M4电机编码器的数据    M4 motor encoder data
    i2cRead(Motor_model_ADDR, READ_ALLHigh_M4_REG, 2, buf);
    i2cRead(Motor_model_ADDR, READ_ALLLOW_M4_REG, 2, buf2);
    Encoder_Now[3] = buf[0]<<24|buf[1]<<16|buf2[0]<<8|buf2[1];

}
```

After getting the data from the driver board, it is shifted, and the shift is required to get the correct data.

1.3 Experimental phenomenon

After the wiring is correct, write the program into the mainboard. After resetting, you can see that the car will move forward for 2.5S, move backward for 2.5S, rotate right for 2.5S, rotate left for 2.5S, then stop for 2.5S, and continue the above state switching actions.

At the same time, you can see that the printed motor values are constantly changing in the serial port assistant.

```
M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119

M1:1118,M2:1127,M3:1136,M4:1119
```