# 13. Timer Capture Encoder Data



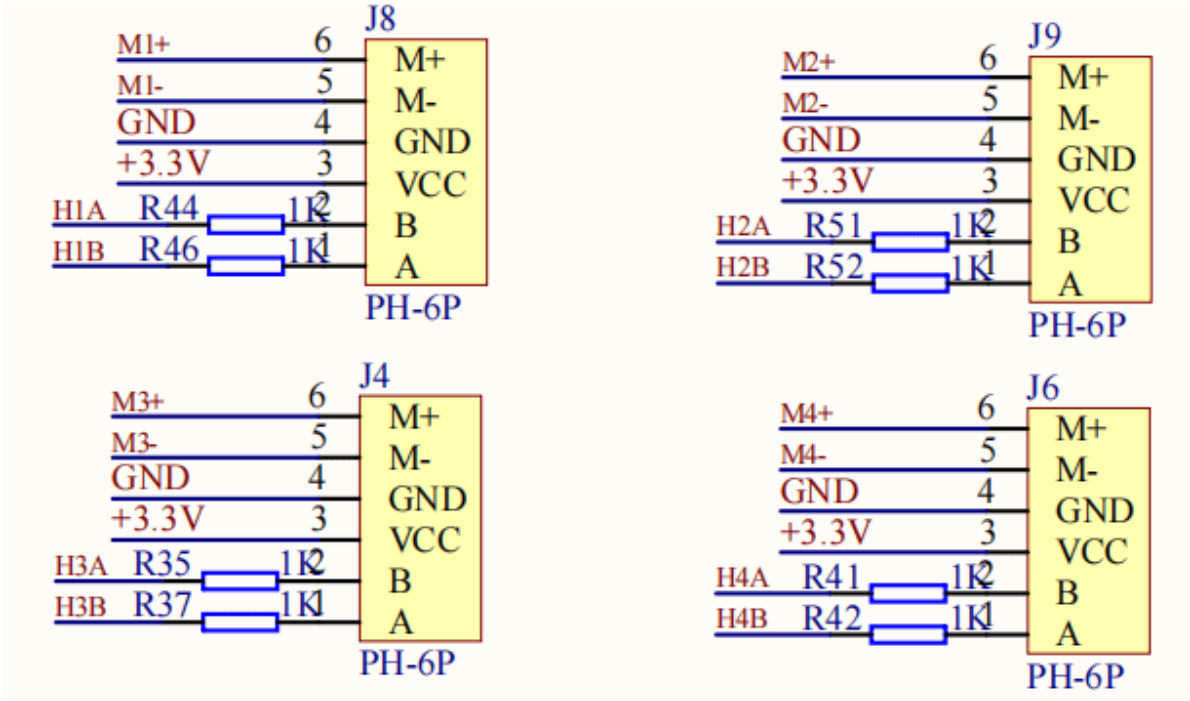
## 13.1. Purpose of the experiment

Capture encoder data using the STM32's timer encoder mode function.

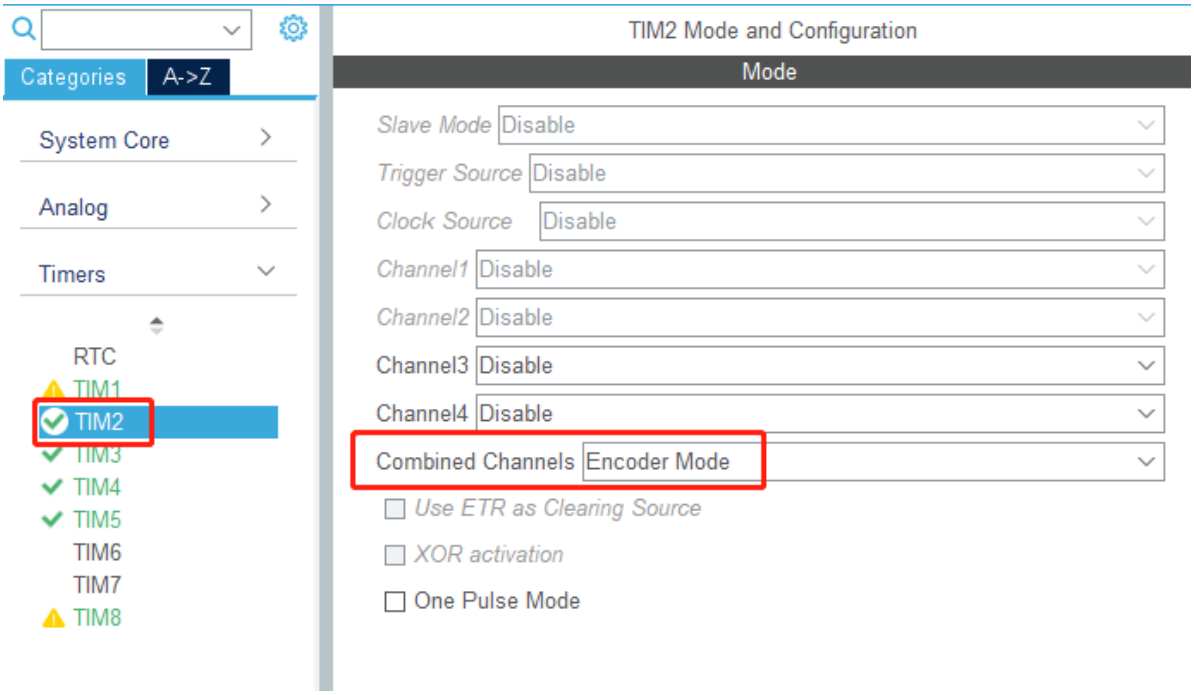
## 13.2 Configuring pin information

1. Import the ioc file from Motor's project and name it Encoder, then serial port 1's related driver.

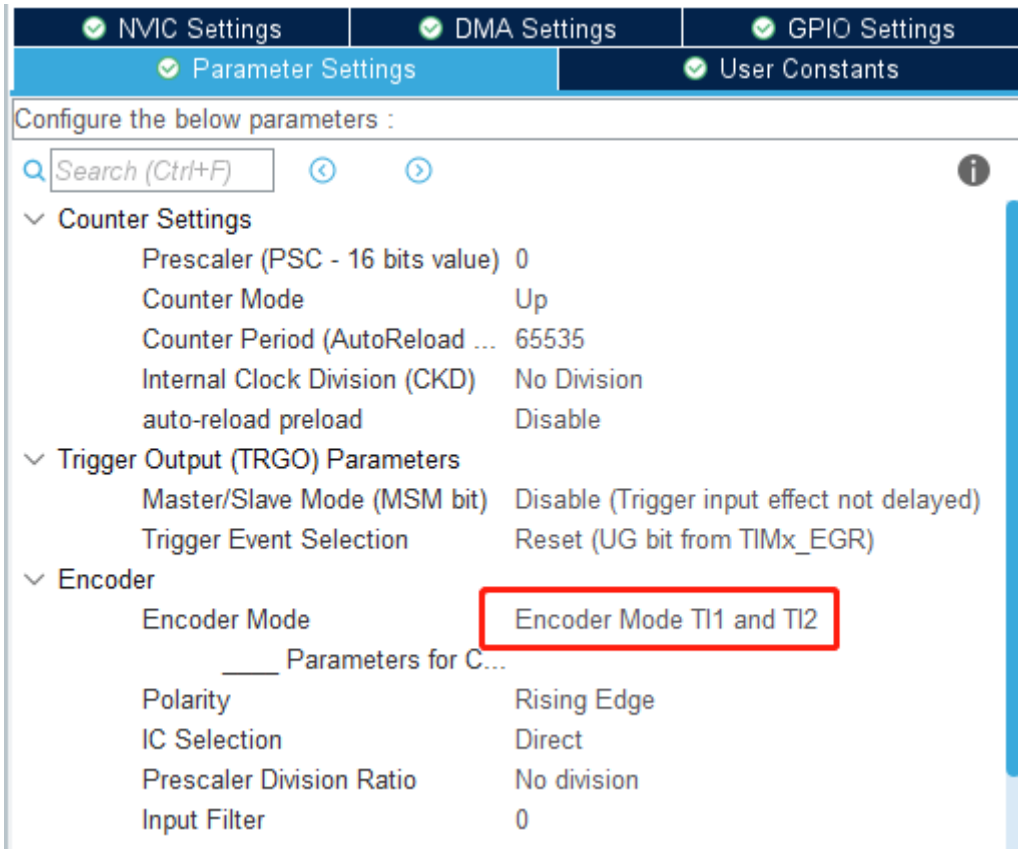
According to the schematic diagram, it can be seen that the two encoders of the four motors are connected to channel 1 and channel 2 of timer 2 3 4 5 respectively. Where motor M1 corresponds to timer TIM2, M2 corresponds to TIM4, M3 corresponds to corresponds to TIM5 and M4 corresponds to TIM3.



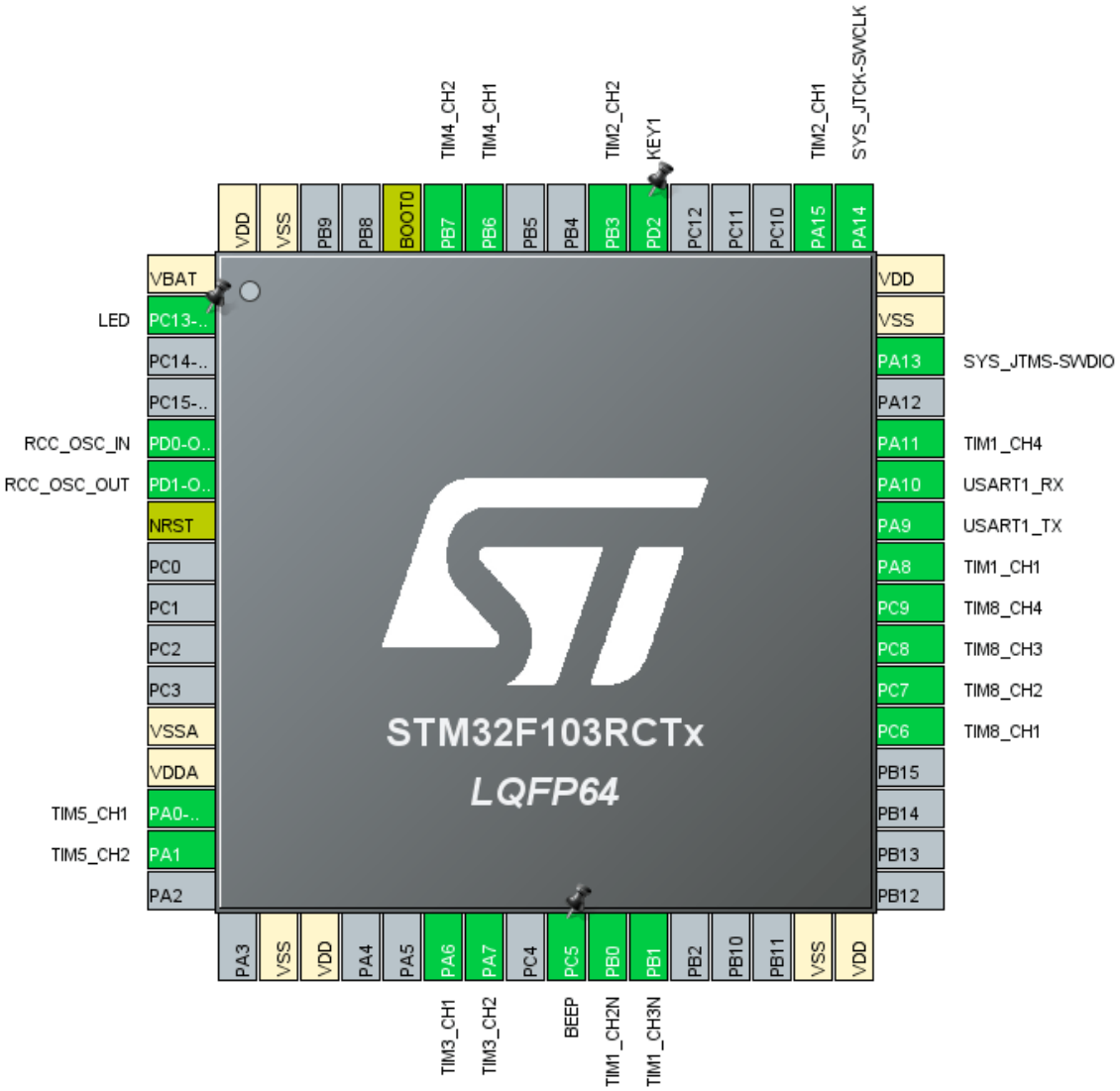

2. Take Timer TIM2 for example, TIM3 TIM4 TIM5 is set in the same way. Select Combined Channels mode as Encoder Mode.



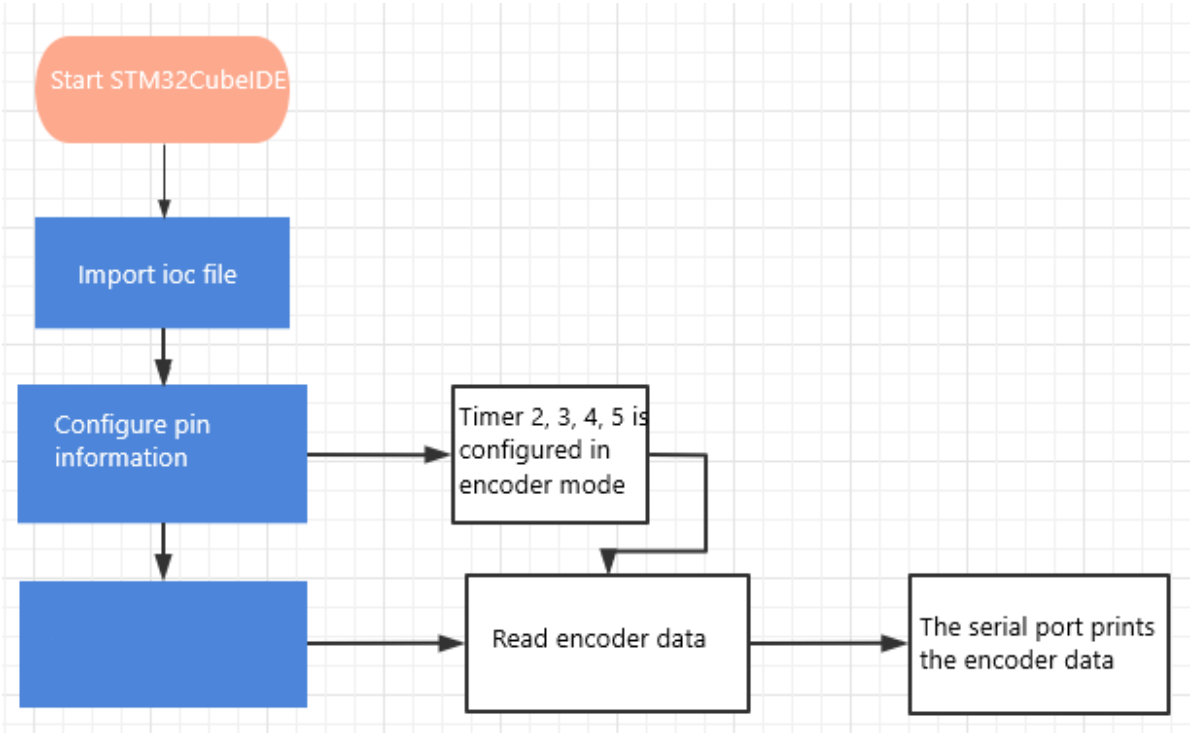

3. In the settings, change Encoder Mode to Encoder Mode TI1 and TI2, set it to Quadruple Frequency, and other parameters are shown in the figure below.




The final chip configuration pins are shown below:




## 13.3. Experimental flowchart analysis



## 13.4. Core code explanation

1. Create a new bsp_encoder.h and bsp_encoder.c and add the following to bsp_encoder.h:

```
// 轮子转一整圈，编码器获得的脉冲数:30*11*2*2
// One full turn of the wheel, the number of pulses
#define ENCODER_CIRCLE            (1320)

void Encoder_Init(void);
void Encoder_Update_Count(void);
int Encoder_Get_Count_Now(uint8_t Motor_id);
void Encoder_Get_ALL(int* Encoder_all);
```

2. Create the following new contents in the bsp_encoder.c file:

Encoder timer initialization.

```
// Initializing timer  初始化定时器
void Encoder_Init(void)
{
    HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_1 | TIM_CHANNEL_2);
    HAL_TIM_Encoder_Start(&htim3, TIM_CHANNEL_1 | TIM_CHANNEL_2);
    HAL_TIM_Encoder_Start(&htim4, TIM_CHANNEL_1 | TIM_CHANNEL_2);
    HAL_TIM_Encoder_Start(&htim5, TIM_CHANNEL_1 | TIM_CHANNEL_2);
}
```

3. Read the data from the encoder.

```
/**
 * @Brief: To read the encoder count, call every 10 milliseconds  读取编码器计数，需每10毫秒调用一次
 * @Note:
 * @Parm: Motor id: 电机的ID号:MOTOR_ID_M1, MOTOR_ID_M2, MOTOR_ID_M3, MOTOR_ID_M4
 * @Retval: Returns encoder count data  返回编码器计数数据
 */
static int16_t Encoder_Read_CNT(uint8_t Motor_id)
{
    int16_t Encoder_TIM = 0;
    switch(Motor_id)
    {
    case MOTOR_ID_M1:  Encoder_TIM = (short)TIM2 -> CNT; TIM2 -> CNT = 0; break;
    case MOTOR_ID_M2:  Encoder_TIM = (short)TIM4 -> CNT; TIM4 -> CNT = 0; break;
    case MOTOR_ID_M3:  Encoder_TIM = (short)TIM5 -> CNT; TIM5 -> CNT = 0; break;
    case MOTOR_ID_M4:  Encoder_TIM = (short)TIM3 -> CNT; TIM3 -> CNT = 0; break;
    default:  break;
    }
    return Encoder_TIM;
}
```

4. Updates the total value of the encoder count. Needs to be called every 10 milliseconds.

```
// 更新编码器的计数总值。需每10毫秒调用一次
// Update the count value of the encoder. call every 10 milliseconds
void Encoder_Update_Count(void)
{
    // g_Encoder_M1_Now += Encoder_Read_CNT(MOTOR_ID_M1);
    g_Encoder_M1_Now -= Encoder_Read_CNT(MOTOR_ID_M1);

    g_Encoder_M2_Now += Encoder_Read_CNT(MOTOR_ID_M2);
    // g_Encoder_M2_Now -= Encoder_Read_CNT(MOTOR_ID_M2);

    g_Encoder_M3_Now += Encoder_Read_CNT(MOTOR_ID_M3);
    // g_Encoder_M3_Now -= Encoder_Read_CNT(MOTOR_ID_M3);

    // g_Encoder_M4_Now += Encoder_Read_CNT(MOTOR_ID_M4);
    g_Encoder_M4_Now -= Encoder_Read_CNT(MOTOR_ID_M4);
}
```

5. Return the total number of encoder counts from power on to now, Encoder_Get_Count_Now returns one way, Encoder_Get_ALL returns four ways.

```
// 返回开机到现在总共统计的编码器的计数（单路）。
// Returns the total count of encoders from boot up to now (single channel)
int Encoder_Get_Count_Now(uint8_t Motor_id)
{
    if (Motor_id == MOTOR_ID_M1) return g_Encoder_M1_Now;
    if (Motor_id == MOTOR_ID_M2) return g_Encoder_M2_Now;
    if (Motor_id == MOTOR_ID_M3) return g_Encoder_M3_Now;
    if (Motor_id == MOTOR_ID_M4) return g_Encoder_M4_Now;
    return 0;
}

// 获取开机到现在总共的四路编码器计数。
// Get the total four - way encoder count up to now
void Encoder_Get_ALL(int* Encoder_all)
{
    Encoder_all[0] = g_Encoder_M1_Now;
    Encoder_all[1] = g_Encoder_M2_Now;
    Encoder_all[2] = g_Encoder_M3_Now;
    Encoder_all[3] = g_Encoder_M4_Now;
}
```

6. Add the encoder initialization to the Bsp_Init() function.

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

7. The new encoder array is used to save the encoder data and show_encoder is used to print the encoder count.

```
int encoder[4] = {0};
int show_encoder = 0;
```

8. On the basis of the original control motor, add the function of printing encoder data every 100 milliseconds.

```
// 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 (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int state = 0;
        state++;
        int speed = 0;
        if (state == 1)
        {
            speed = 2000;
            Motor_Set_Pwm(MOTOR_ID_M1, speed);
            Motor_Set_Pwm(MOTOR_ID_M2, speed);
            Motor_Set_Pwm(MOTOR_ID_M3, speed);
            Motor_Set_Pwm(MOTOR_ID_M4, speed);
        }
        if (state == 2)
        {
            Motor_Stop(0);
        }
        if (state == 3)
        {
            speed = -2000;
            Motor_Set_Pwm(MOTOR_ID_M1, speed);
            Motor_Set_Pwm(MOTOR_ID_M2, speed);
            Motor_Set_Pwm(MOTOR_ID_M3, speed);
            Motor_Set_Pwm(MOTOR_ID_M4, speed);
        }
        if (state == 4)
        {
            state = 0;
            Motor_Stop(1);
        }
    }

    show_encoder++;
    if (show_encoder > 10)
    {
        show_encoder = 0;
        Encoder_Get_ALL(encoder);
        printf("Encoder:%d, %d, %d, %d\n", encoder[0], encoder[1], encoder[2], encoder[3]);
    }

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

## 13.5 Hardware Connection

The motor connecting wires need to be connected to the corresponding motors as shown in the following figure, otherwise it may cause the problem that the program does not match the phenomenon. Motor 1 corresponds to the motor in the upper left corner of the body, Motor 2 corresponds to the motor in the lower left corner, Motor 3 corresponds to the motor in the upper right corner, and Motor 4 corresponds to the motor in the lower right corner.


Due to the relatively high power of the motor, the expansion board should not be powered directly by USB 5V, but must be powered by DC 12V.

Then connect the micro-USB cable to the expansion board and computer.

## 13.6. Experimental Effect

As the motor will start to turn, please set up the cart before the experiment, the motor wheels are suspended, to avoid running across the road.

After burning the program, the LED flashes every 200 milliseconds. Open the serial assistant, you can see the encoder data. Press the first forward, the second free stop, the third backward, the fourth brake stop.


串口调试助手
串口设置
串口号 COM39 #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
[2022-03-11 16:19:34.760]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:34.888]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.017]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.126]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.254]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.381]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.504]# RECV ASCII>
Encoder:1030, 264, 766, 464
[2022-03-11 16:19:35.630]# RECV ASCII>
Encoder:1030, 264, 766, 464
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!
178/0
RX:4239
TX:0
复位计数