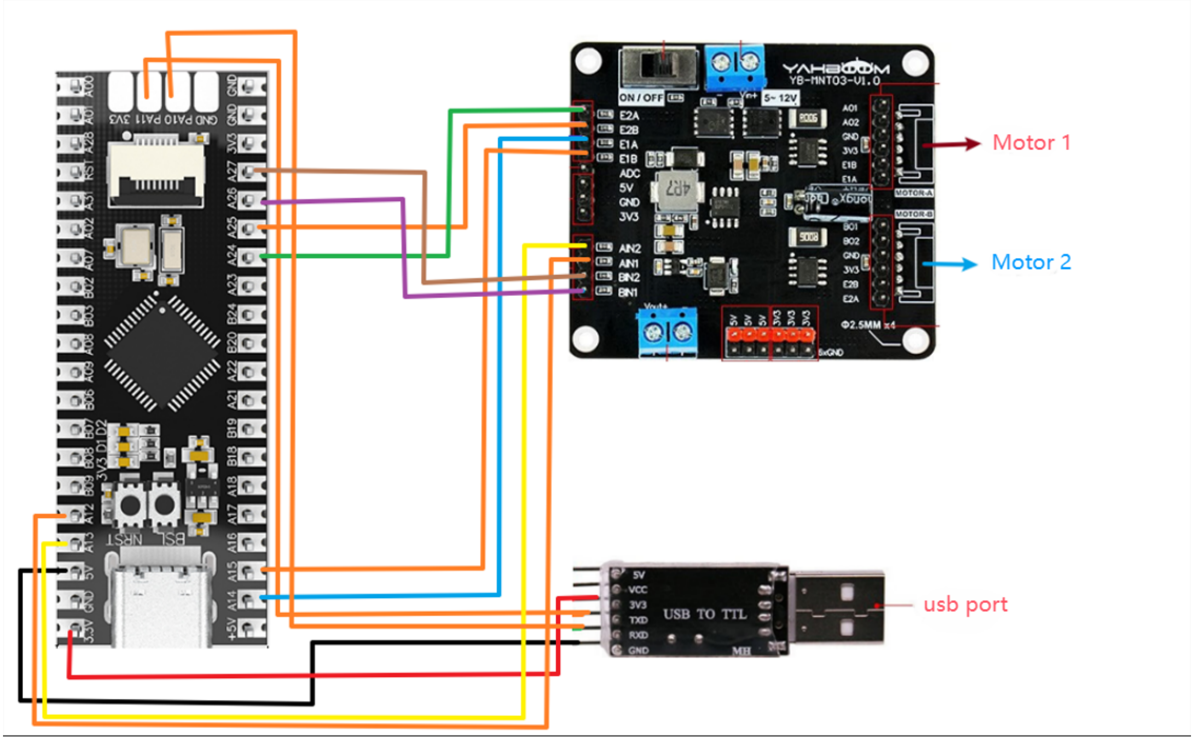
Encoder data acquisition

1. Learning objectives

Read the motor encoder data.

2. Hardware connection

MSPM0G3507 and AT8236 pin connection



L1 motor:

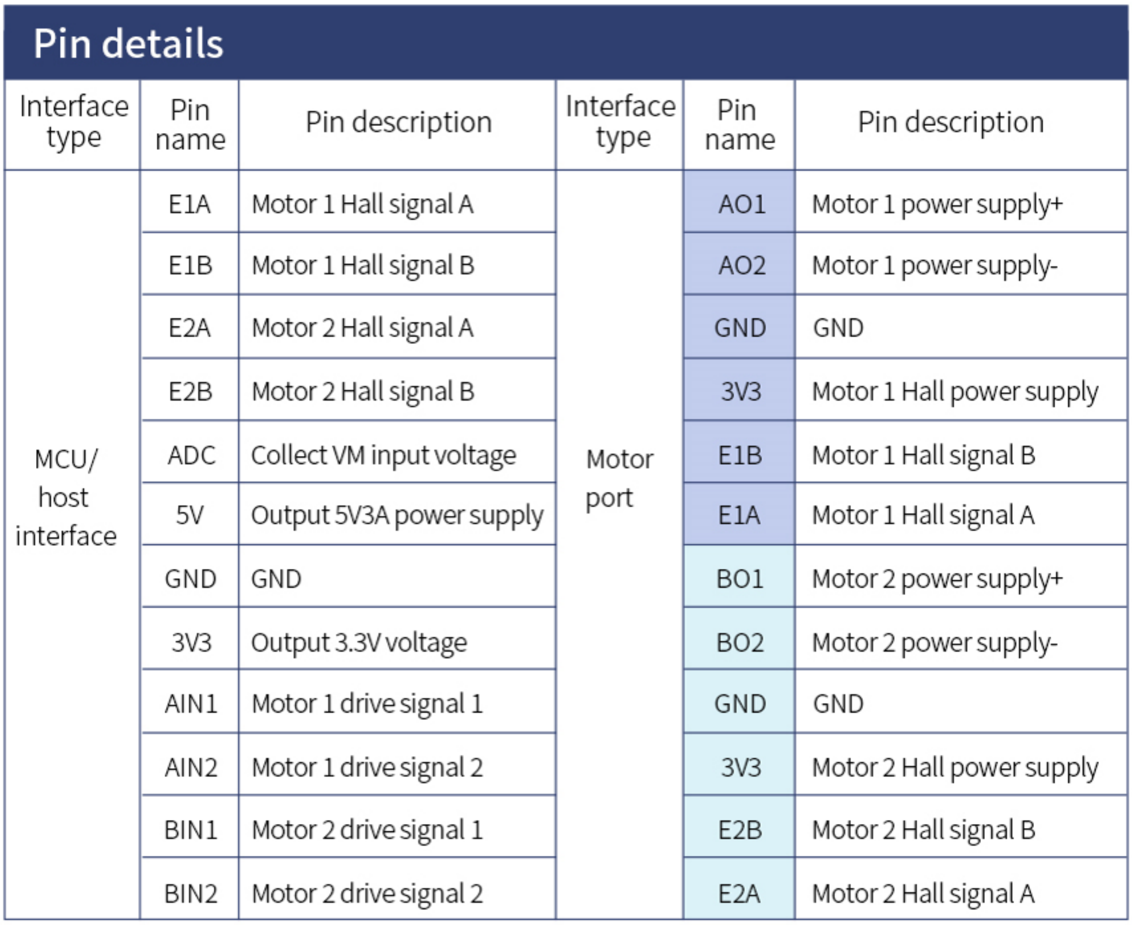
MSPM0G3507	AT8236
PA12	AIN1
PA13	AIN2
PA14	E1A
PA15	E1B

L2 motor:

MSPM0G3507	AT8236
PA26	BIN1
PA27	BIN2
PA24	E2A
PA25	E2B

Note: You can use the ttl module, or just use the Type-C port of the MSPM0G3507 to connect to the computer.

AT8236 motor driver module pin description:

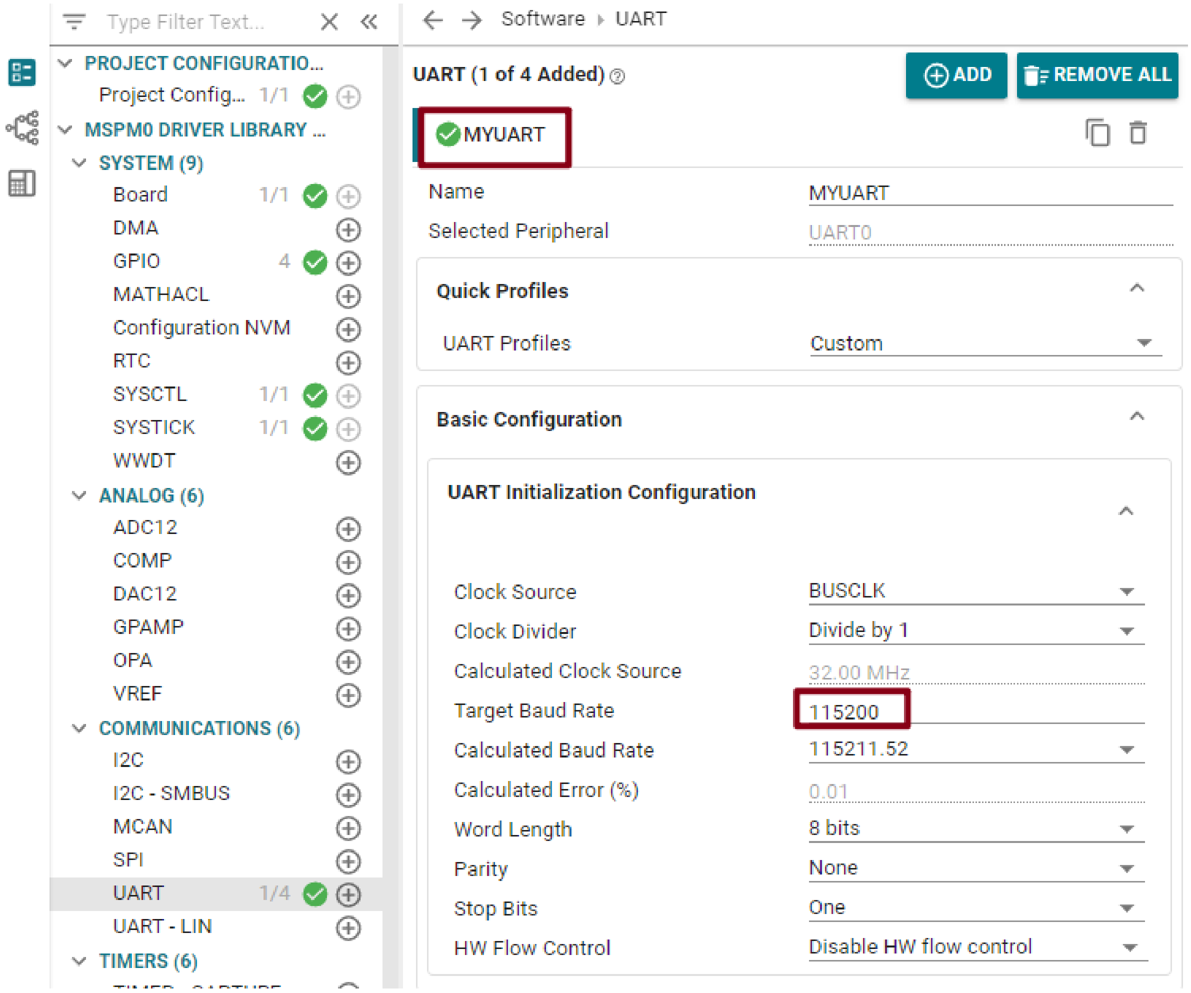


Pin details

Interface type	Pin name	Pin description	Interface type	Pin name	Pin description
MCU/ host interface	E1A	Motor 1 Hall signal A	Motor port	AO1	Motor 1 power supply+
	E1B	Motor 1 Hall signal B		AO2	Motor 1 power supply-
	E2A	Motor 2 Hall signal A		GND	GND
	E2B	Motor 2 Hall signal B		3V3	Motor 1 Hall power supply
	ADC	Collect VM input voltage		E1B	Motor 1 Hall signal B
	5V	Output 5V3A power supply		E1A	Motor 1 Hall signal A
	GND	GND		BO1	Motor 2 power supply+
	3V3	Output 3.3V voltage		BO2	Motor 2 power supply-
	AIN1	Motor 1 drive signal 1		GND	GND
	AIN2	Motor 1 drive signal 2		3V3	Motor 2 Hall power supply
	BIN1	Motor 2 drive signal 1		E2B	Motor 2 Hall signal B
	BIN2	Motor 2 drive signal 2		E2A	Motor 2 Hall signal A

3. Program Description

This example sets the baud rate of the serial port printing to 115200 bps.



UART (1 of 4 Added)

MYUART

Name: MYUART
Selected Peripheral: UART0

UART Profiles: Custom

Clock Source: BUSCLK
Clock Divider: Divide by 1
Calculated Clock Source: 32.00 MHz
Target Baud Rate: 115200
Calculated Baud Rate: 115211.52
Calculated Error (%): 0.01
Word Length: 8 bits
Parity: None
Stop Bits: One
HW Flow Control: Disable HW flow control

- bsp_at8236.h

```
#ifndef __BSP_TB6612_H_
#define __BSP_TB6612_H_

#include "ti_msp_dl_config.h"


void init_motor(void);

void L1_control(uint16_t motor_speed,uint8_t dir);
void L2_control(uint16_t motor_speed,uint8_t dir);
void R1_control(uint16_t motor_speed,uint8_t dir);
void R2_control(uint16_t motor_speed,uint8_t dir);

#endif
```

Define four motor control functions.

- bsp_at8236.c

```
void L1_control(uint16_t motor_speed,uint8_t dir)
{
    if(dir)
    {
            DL_TimerA_setCaptureCompareValue(PWM_L1_INST, motor_speed,
DL_TIMER_CC_0_INDEX);
            DL_TimerA_setCaptureCompareValue(PWM_L1_INST, 0,
DL_TIMER_CC_1_INDEX);
```

```
    }
    else
    {
            DL_TimerA_setCaptureCompareValue(PWM_L1_INST, 0,
DL_TIMER_CC_0_INDEX);
            DL_TimerA_setCaptureCompareValue(PWM_L1_INST, motor_speed,
DL_TIMER_CC_1_INDEX);
    }

}
```

The L1_control function is used to control the speed and direction of the L1 motor by adjusting the duty cycle of the PWM signal.

motor_speed and dir represent the motor speed and motor direction of the motor respectively.

- bsp_delay.h

```
#ifndef __BSP_DELAY_H_
#define __BSP_DELAY_H_


#include "ti_msp_dl_config.h"

void delay_ms(unsigned int ms);

#endif
```

Define a delay function delay_ms

- bsp_delay.c

```
#include "bsp_delay.h"
#include "ti_msp_dl_config.h"

volatile unsigned int delay_times = 0;

volatile unsigned int getspeed = 10;//Get speed flag 10ms once

extern volatile int32_t gEncoderCount_L1,gEncoderCount_L2;

extern int speed,speed2;

//Precise ms delay with tick timer

void delay_ms(unsigned int ms)//1ms timing
{
delay_times = ms;
while( delay_times != 0 );
}

//Tick timer interrupt service function
void SysTick_Handler(void)
{
if( delay_times != 0 )
{
delay_times--;
}

if( getspeed != 0 )
{
getspeed--; } else { getspeed = 10; speed = gEncoderCount_L1; speed2 =
gEncoderCount_L2; gEncoderCount_L1 = 0;//Clear 0 gEncoderCount_L2 = 0;//Clear 0
}

 }
```

It processes the counts of two encoders (L1 and L2) and updates their speed measurements every 10 milliseconds.

- bsp_enconder.c

```
void GROUP1_IRQHandler(void)
{
    //Get interrupt signal
    gpioA = DL_GPIO_getEnabledInterruptStatus(GPIOA,
    L1_Enconder_A_pin_14_PIN | L1_Enconder_B_pin_15_PIN|
        L2_Enconder_A_pin_24_PIN|L2_Enconder_B_pin_25_PIN);

    //If the interrupt is generated by GPIO_EncoderA_PIN_0_PIN
    if((gpioA & L1_Enconder_A_pin_14_PIN) == L1_Enconder_A_pin_14_PIN)
    {
        //Pin14 rising edge, look at the level of Pin15, if it is low level, it
is judged as reverse rotation, and high level is judged as forward rotation
        if(!DL_GPIO_readPins(GPIOA,L1_Enconder_B_pin_15_PIN))//P15 is low level
        {
            gEncoderCount_L1--;
        }
        else//P15 is high level
        {
            gEncoderCount_L1++;
        }
    }

    else if((gpioA & L1_Enconder_B_pin_15_PIN) == L1_Enconder_B_pin_15_PIN)
    {
        //Pin15 rising edge
        if(!DL_GPIO_readPins(GPIOA,L1_Enconder_A_pin_14_PIN))//P14为低电平
        {
            gEncoderCount_L1++;
        }
        else//P14 is high level
        {
            gEncoderCount_L1--;
        }
    }
```

Used to handle encoder interrupts, read and update encoder data.

- bsp_usart.h

```c
#ifndef __BSP_USART_H_
#define __BSP_USART_H_

#include "ti_msp_dl_config.h"
#include "bsp_delay.h"

void uart0_send_char(char ch);
void uart0_send_string(char* str);

#endif
```

A header file for serial communication is defined, which is mainly used to implement simple character and string sending functions.

- bsp_usart.c

```c
//Send a single character through the serial port
void uart0_send_char(char ch)
{
    //Wait when serial port 0 is busy, and send the incoming characters when it is not busy
    while( DL_UART_isBusy(MYUART_INST) == true );
    //Send a single character
    DL_UART_Main_transmitData(MYUART_INST, ch);
}
//Send string via serial port
void uart0_send_string(char* str)
{
    //The current string address is not at the end and the string's first address is not empty
    while(*str!=0&&str!=0)
    {
        //Send the characters in the first address of the string, and the first address will increment automatically after the sending is completed.
        uart0_send_char(*str++);
    }
}


//Serial port interrupt service function
void MYUART_INST_IRQHandler(void)
{
    //If a serial port interrupt occurs
    switch( DL_UART_getPendingInterrupt(MYUART_INST) )
    {
        case DL_UART_IIDX_RX://If it is a receive interrupt
            //The data received and sent is stored in the variable
            uart_data = DL_UART_Main_receiveData(MYUART_INST);
            //Send the saved data again
            uart0_send_char(uart_data);
            break;

        default://Other serial port interrupts
            break;
    }
}
```

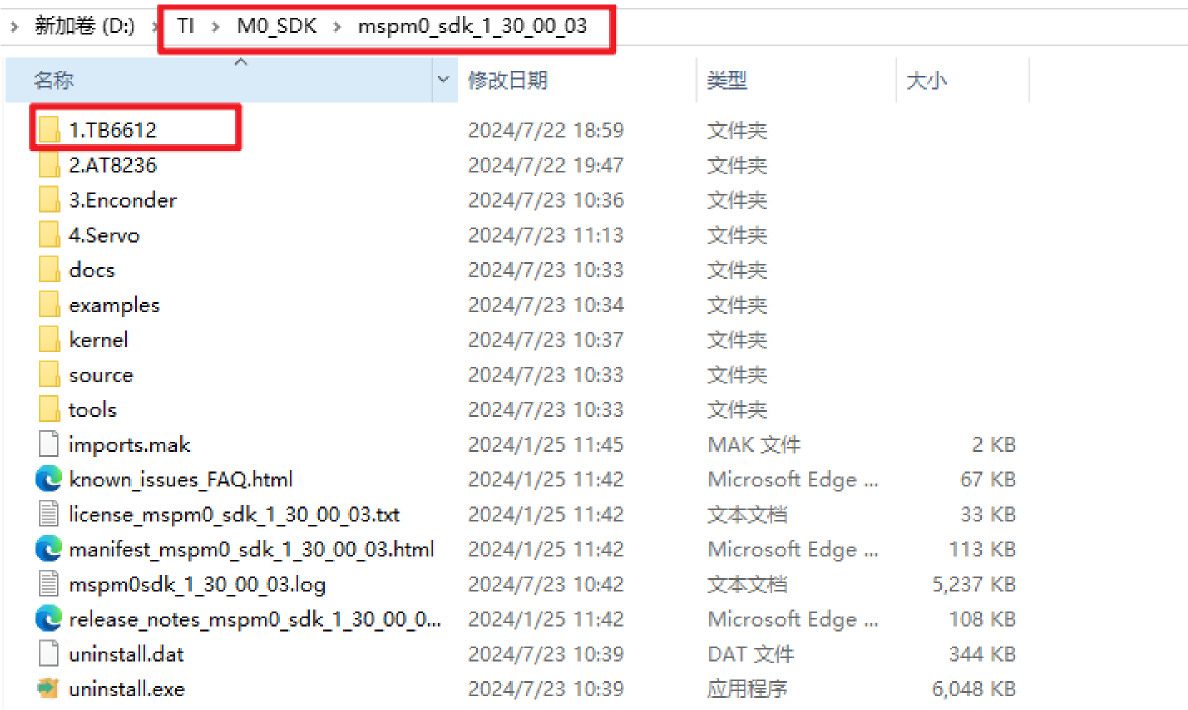
- Use USART for serial communication. Includes functions for sending single characters, sending strings, and a USART interrupt service routine.
    - main.c

```
int main(void)
{
    SYSCFG_DL_init();

     //Clear the serial port interrupt flag
    NVIC_ClearPendingIRQ(MYUART_INST_INT_IRQN);
    //Enable serial port interrupt
    NVIC_EnableIRQ(MYUART_INST_INT_IRQN);
//
        NVIC_EnableIRQ(GPIO_MULTIPLE_GPIOA_INT_IRQN);//Enable external interrupt
        init_motor();//Motor timer on

    while (1)
    {
            sprintf(buf,"speed1(10ms):%d\t speed2(10ms):%d\r\n",speed,speed2);
            uart0_send_string(buf);

            L1_control(600,0);//0-1000
            L2_control(400,1);//0-1000

            delay_ms(300);



    }
}
```

Continuously update the speed of the two motors, and set the speed and direction of motors L1 and L2 respectively through the L1_control and L2_control functions. Use the sprintf function to format a message that contains the current speed information of the two motors, and send this message through the uart0_send_string function to monitor or record the status of the motors.

Note: The project source code must be placed in the SDK path for compilation,

For example, the path: D:\TI\M0_SDK\mspm0_sdk_1_30_00_03\1.TB6612



› 新加卷 (D:) › TI › M0_SDK › mspm0_sdk_1_30_00_03

名称	修改日期	类型	大小
1.TB6612	2024/7/22 18:59	文件夹	
2.AT8236	2024/7/22 19:47	文件夹	
3.Enconder	2024/7/23 10:36	文件夹	
4.Servo	2024/7/23 11:13	文件夹	
docs	2024/7/23 10:33	文件夹	
examples	2024/7/23 10:34	文件夹	
kernel	2024/7/23 10:37	文件夹	
source	2024/7/23 10:33	文件夹	
tools	2024/7/23 10:33	文件夹	
imports.mak	2024/1/25 11:45	MAK 文件	2 KB
known_issues_FAQ.html	2024/1/25 11:42	Microsoft Edge ...	67 KB
license_mspm0_sdk_1_30_00_03.txt	2024/1/25 11:42	文本文档	33 KB
manifest_mspm0_sdk_1_30_00_03.html	2024/1/25 11:42	Microsoft Edge ...	113 KB
mspm0sdk_1_30_00_03.log	2024/7/23 10:42	文本文档	5,237 KB
release_notes_mspm0_sdk_1_30_00_0...	2024/1/25 11:42	Microsoft Edge ...	108 KB
uninstall.dat	2024/7/23 10:39	DAT 文件	344 KB
uninstall.exe	2024/7/23 10:39	应用程序	6,048 KB