Motor control

This tutorial demonstrates how to control 4 310 motors using **timers** and print the encoder values via the serial port, where the KEY button controls the rotation mode of the motors.

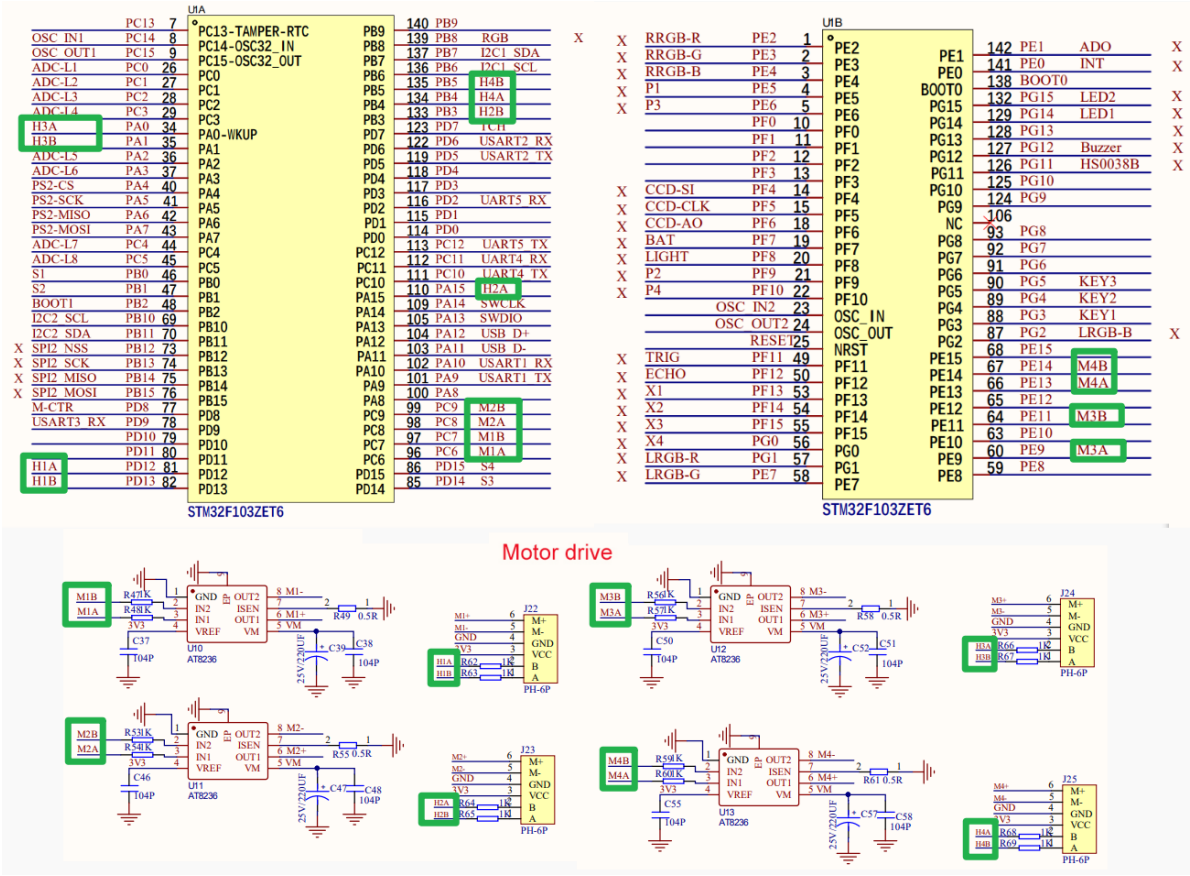
1、software-hardware

- **STM32F103CubeIDE**
- **STM32 robot expansion board**
- **310 motor * 4**
- **Type-C cable or ST-Link**

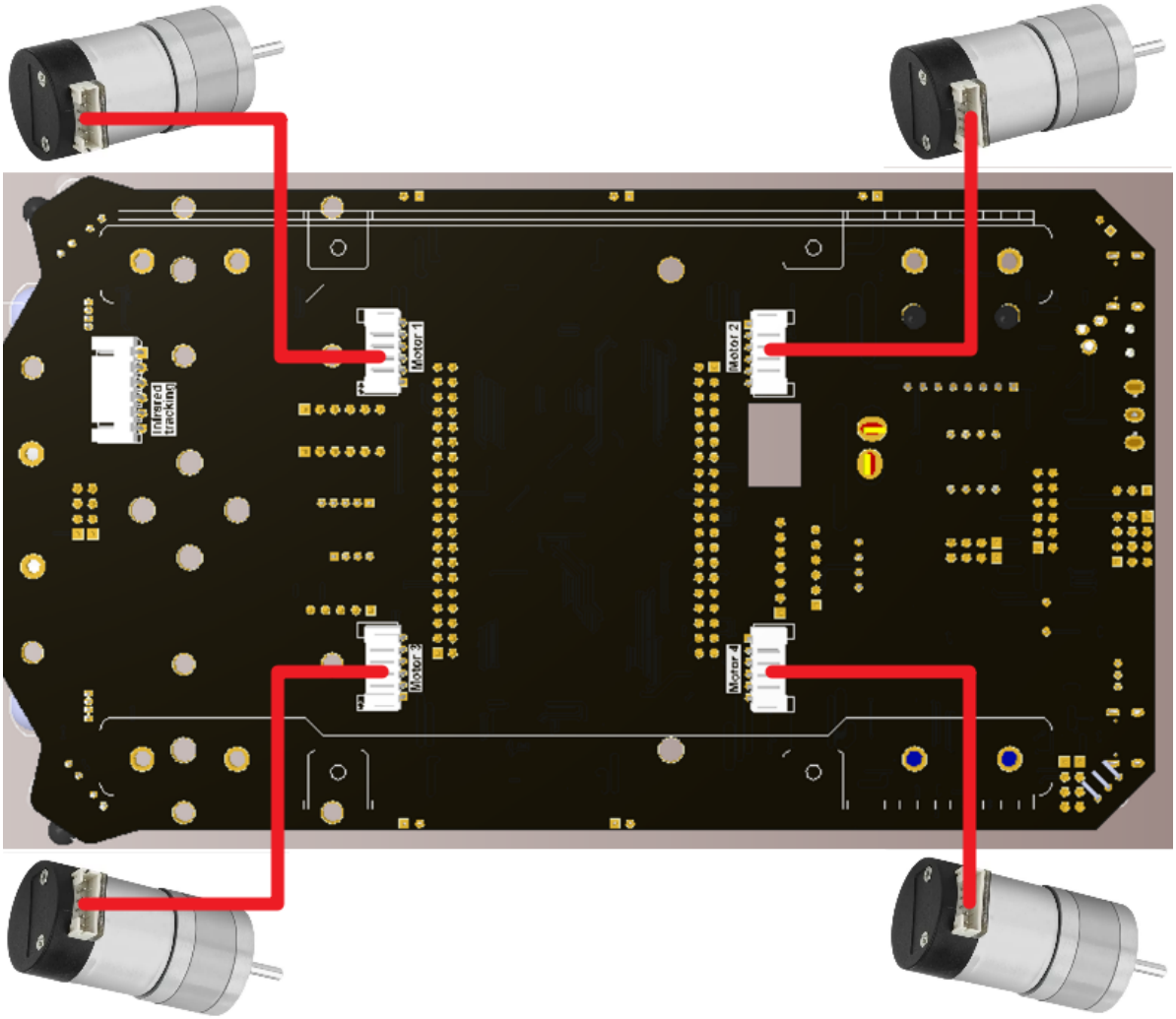
  Download or simulate the program of the development board

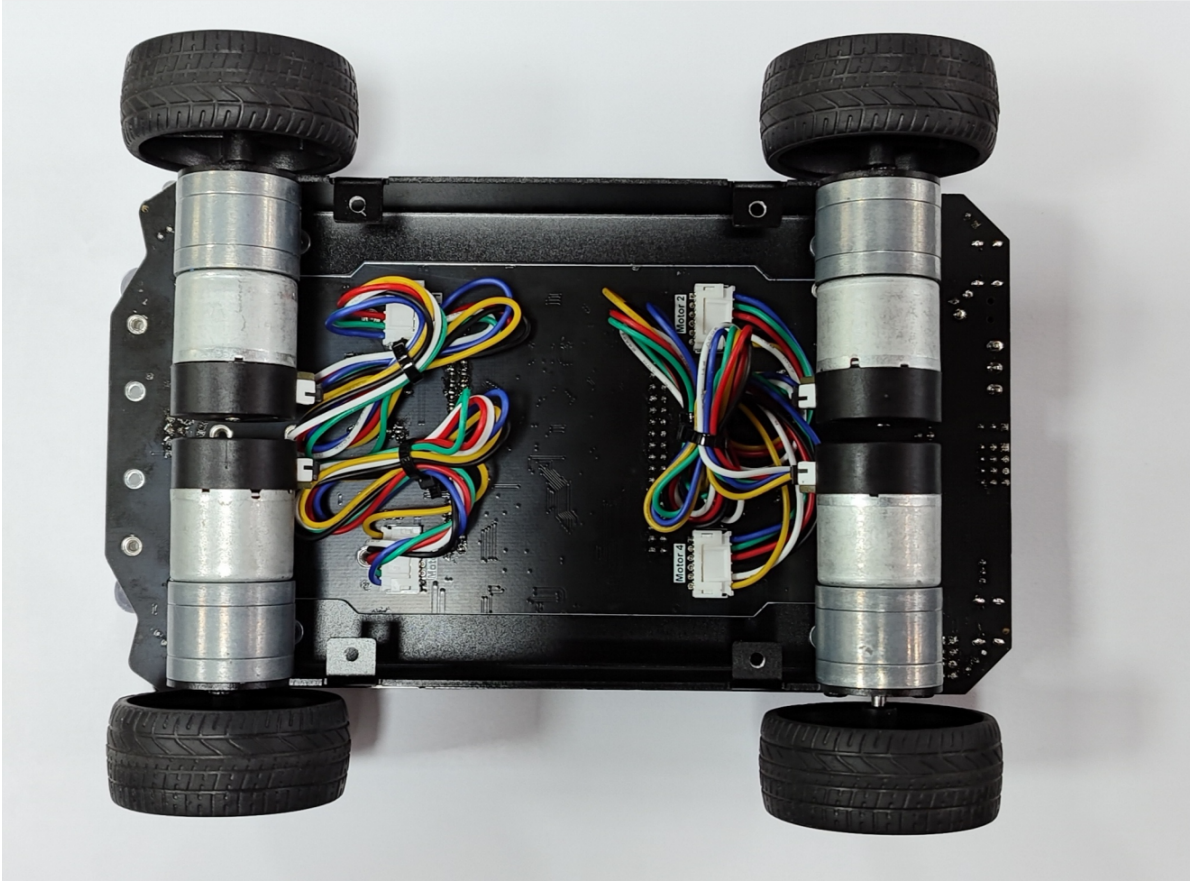
2、Brief principle

2.1、Hardware schematic diagram



2.2、Physical connection diagram





2.3、Principle of control

Motor speed and steering control: configure timer (TIM1 and TIM8) PWM output mode, change the duty cycle of PWM signal;

Motor encoder reading: Configure the encoder interface mode of timers (TIM2, TIM3, TIM4 and TIM5) to count the motor encoder signal.

- **310 motor**

Type of motor	**310 motor**
Rated voltage of moto	7.4V
Encoder supply voltage	3.3-5V
Gear set reduction ratio	1：20
Number of magnetic ring lines	13 line
encoder type	Incremental Hall encoder with phase AB

> **Count value**

```
Maximum count value={Reduction ratio*Number of encoder lines*4}=20*13*4=1040
```

```
4: Represents the encoder frequency doubling
The rotation speed of the car can be calculated according to the count of
rotation. This tutorial does not cover the conversion of speed
```

- **Motor drive**

The STM32F103ZET6 integrates four AT8236 single-channel brush DC motor driver chips.

The input pins IN1 and IN2 control the output state of the H-bridge. The following table is the logical relationship between the input and output:

IN1	IN2	OUT1	OUT2	feature
0	0	Z	Z	Glide and sleep
1	0	H	L	Forward direction
0	1	L	H	Reverse direction
1	1	L	L	brake

When PWM control is used to implement the speed regulation function, the H-bridge can be operated in two different states: fast decay or slow decay.

IN1	IN2	feature
PWM	0	Forward PWM, fast attenuation
1	PWM	Forward PWM, slow decay
0	PWM	Reverse PWM, fast decay
PWM	1	Reverse PWM, slow decay

PWM period：T = 50us → f = 20KHz

$$T(s) = \frac{(ARR+1)*(PSC+1)}{TIM\_CLK(Hz)} = \frac{(3599+1)*(0+1)}{72000000(Hz)} = 0.00005s = 50us$$

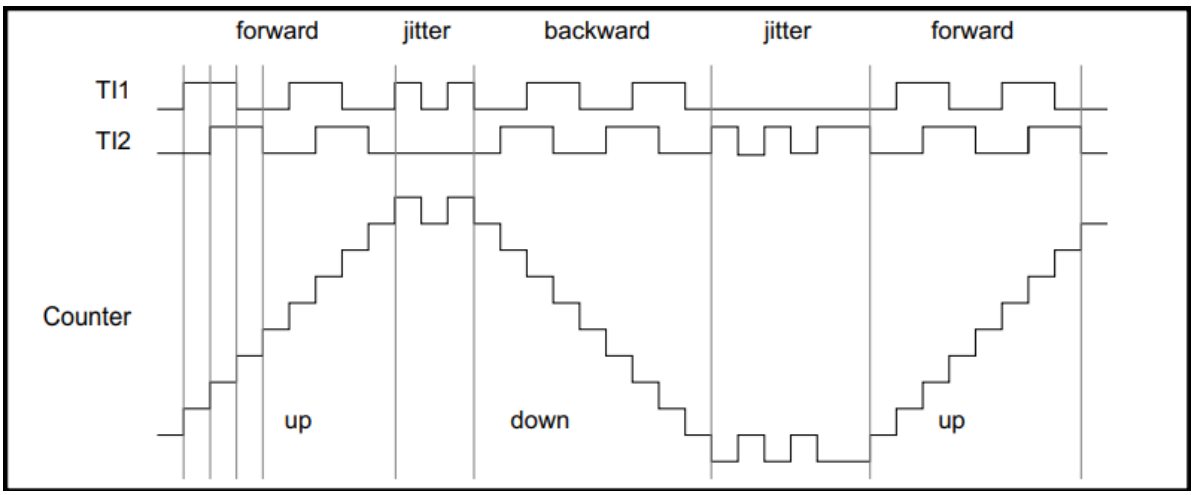
- **Encoder reading**

  The count value is obtained by reading the timer CNT register value directly.

Table 81. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The tutorial setup is **count on TI1 and TI2** , set the encoder mode to 4x frequency, and the input signal is not reversed;




```
Whether the counter counts up or down depends on the AB trust sign
```

3、Engineering configuration

3.1、Notes

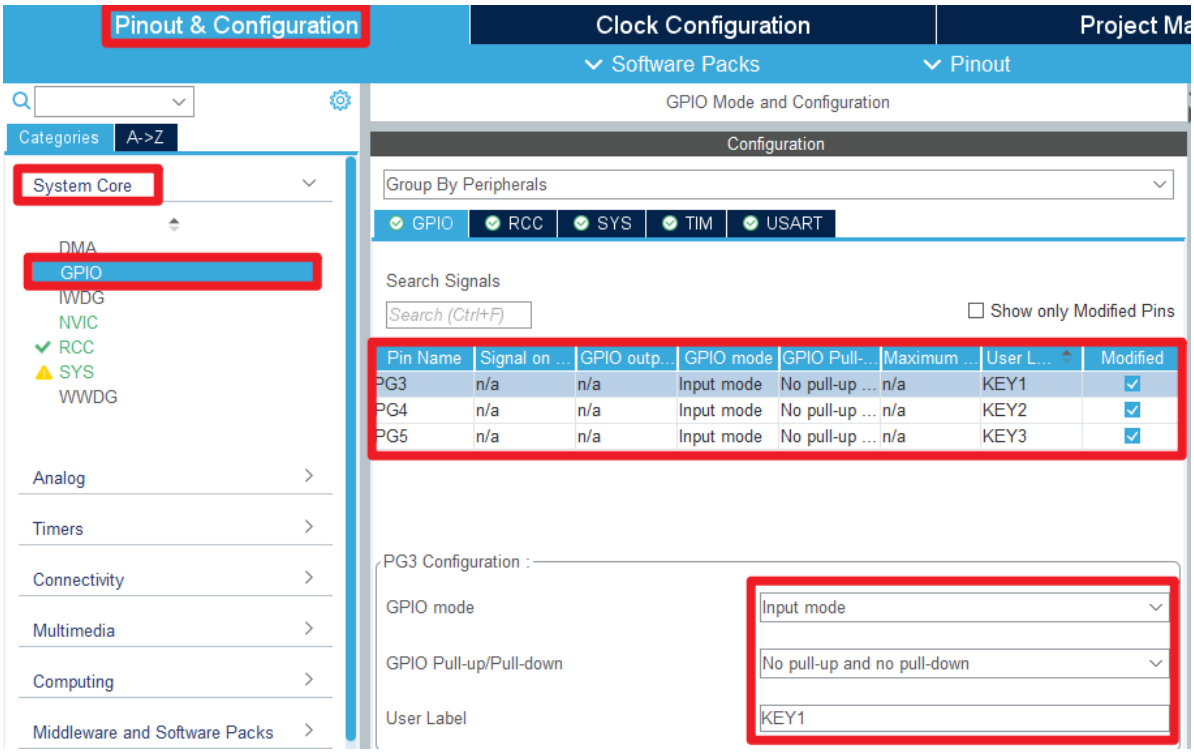
Omitted project configuration: **New project, chip selection, project configuration, SYS for pin configuration, RCC configuration, clock configuration, and project configuration** content

The project cofiguration part, which is not omitted, is the key point to configure in this tutorial.

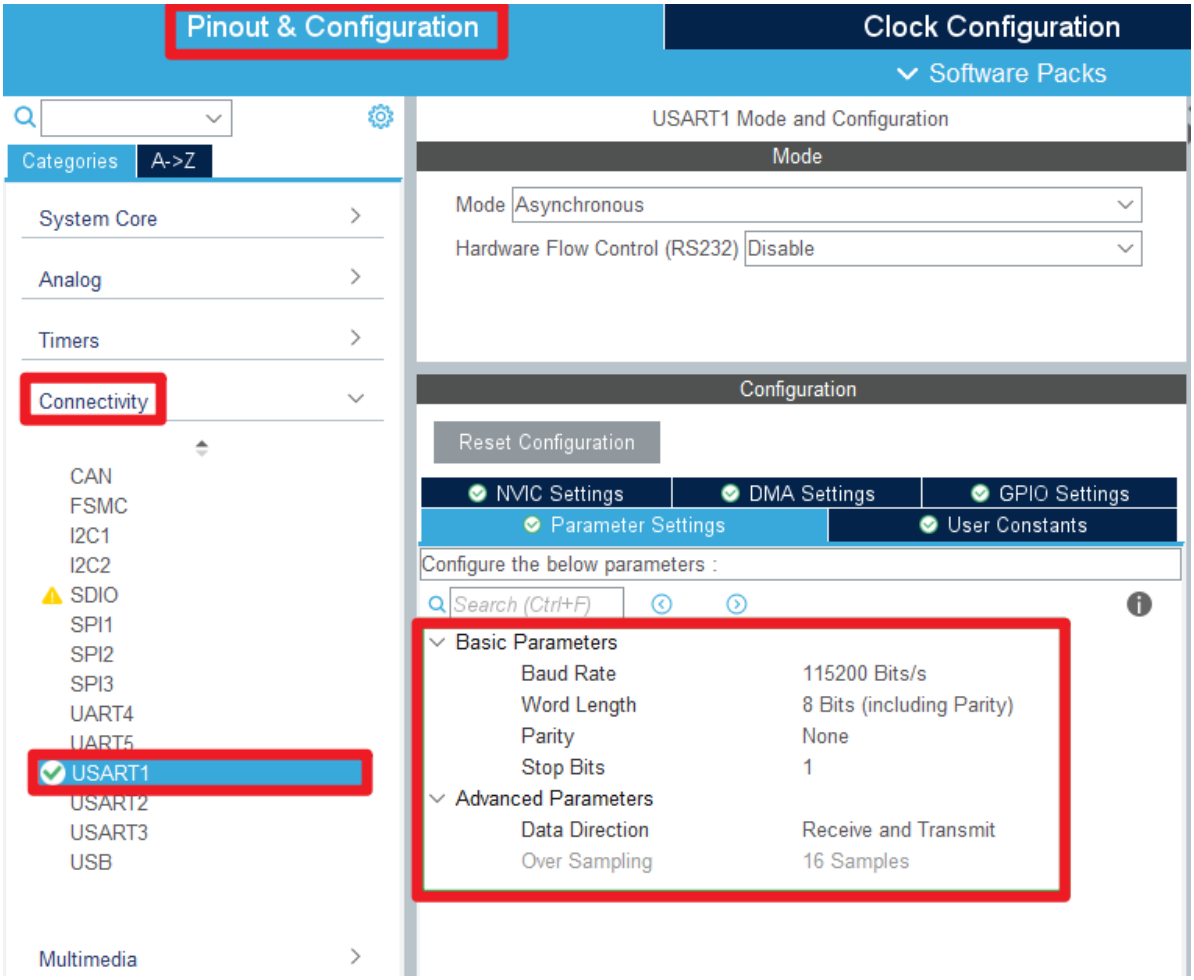
```
Please refer to [2, development environment construction and use: STM32CubeIDE
installation - Use] to understand how to configure the omitted part of the
project
```

3.2、Pin configuration

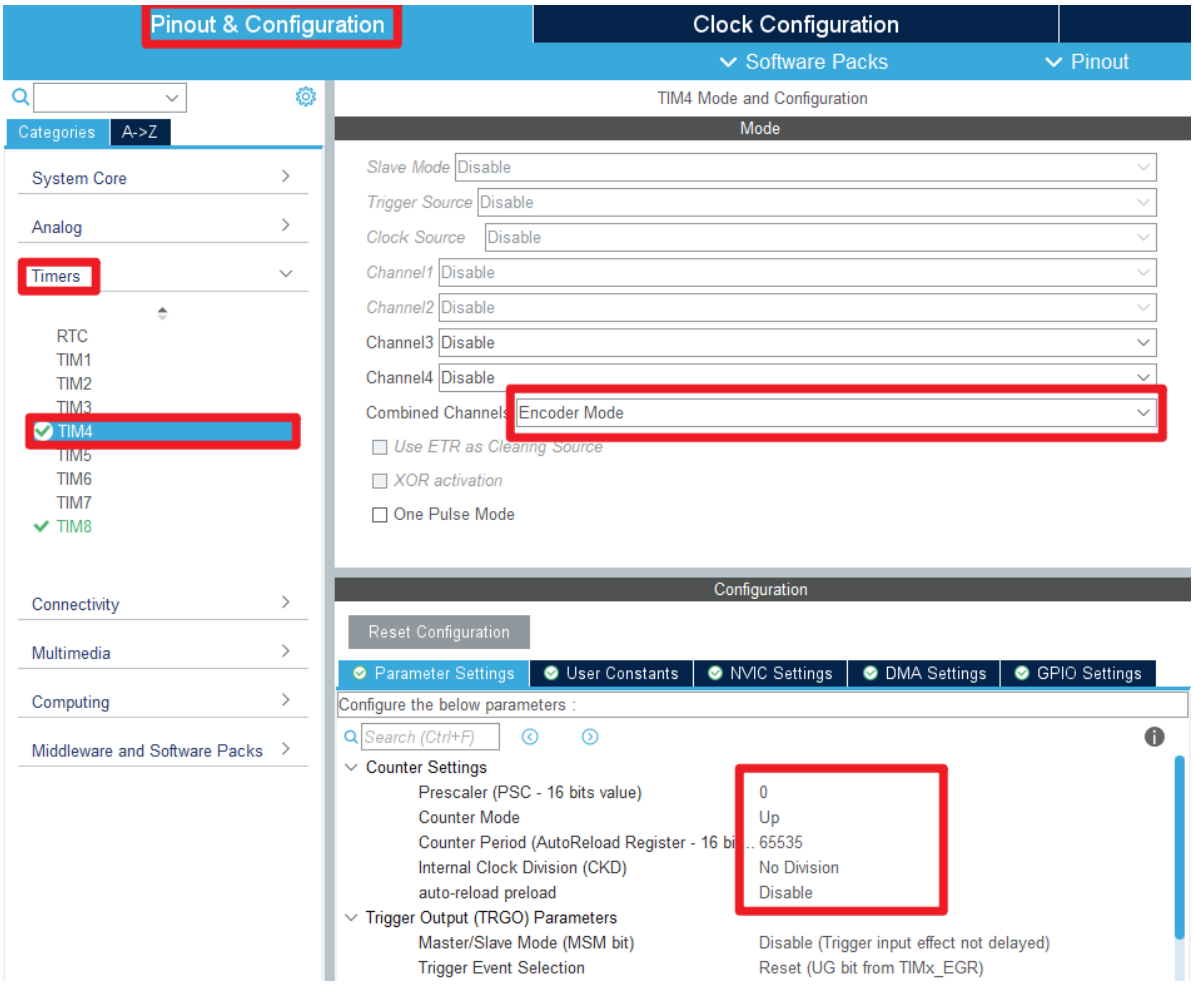
- **GPIO**

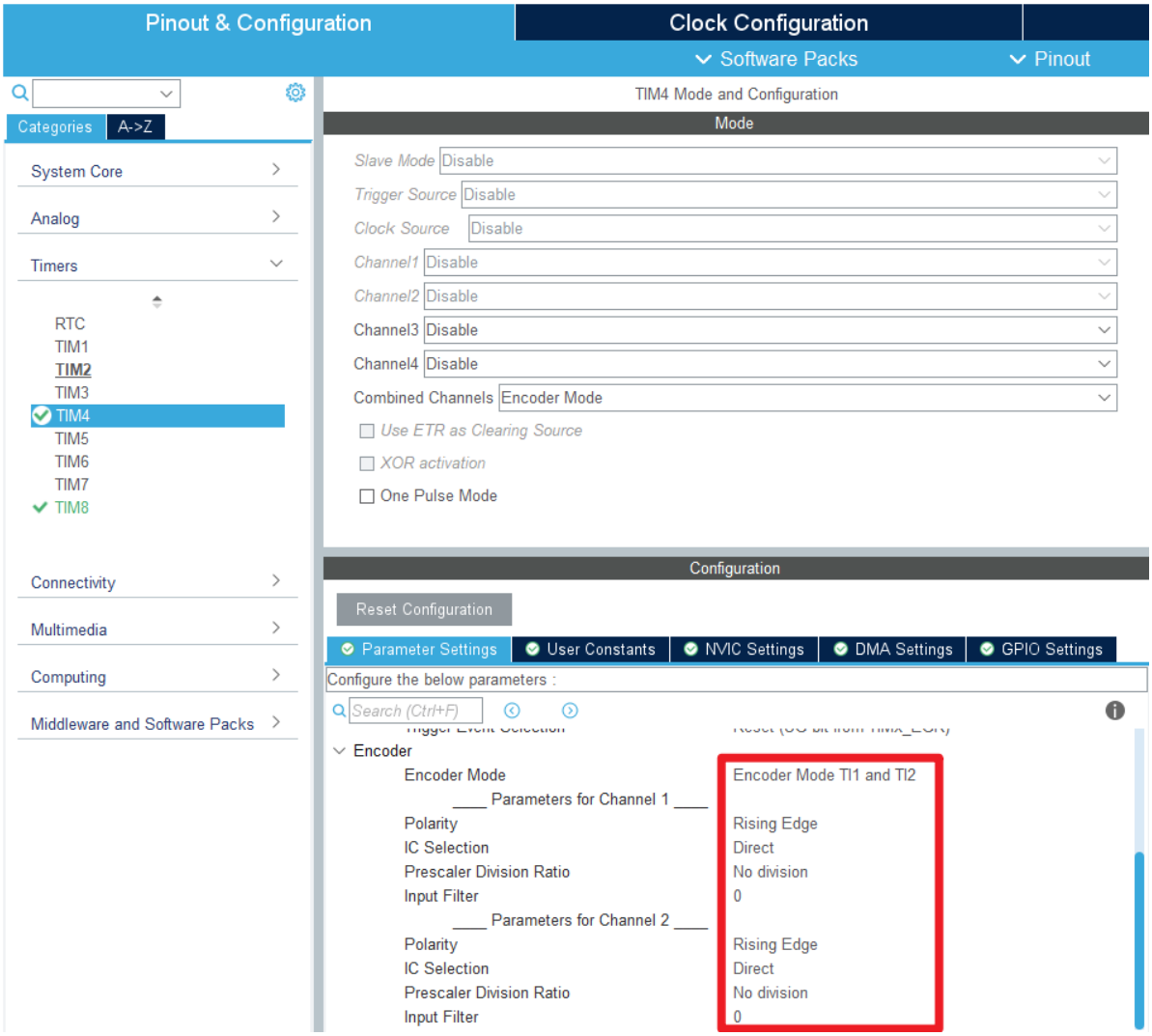


- **USART**

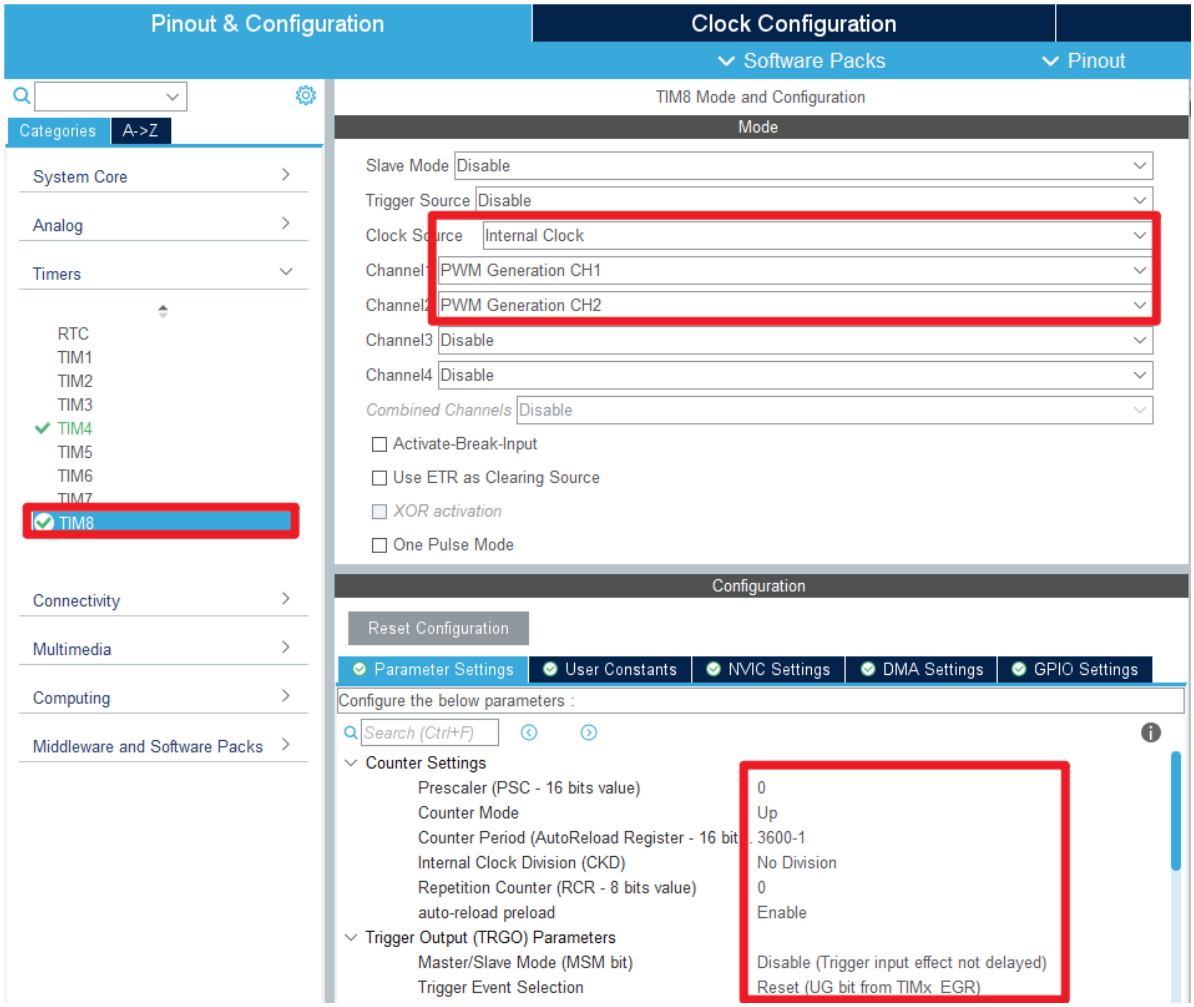


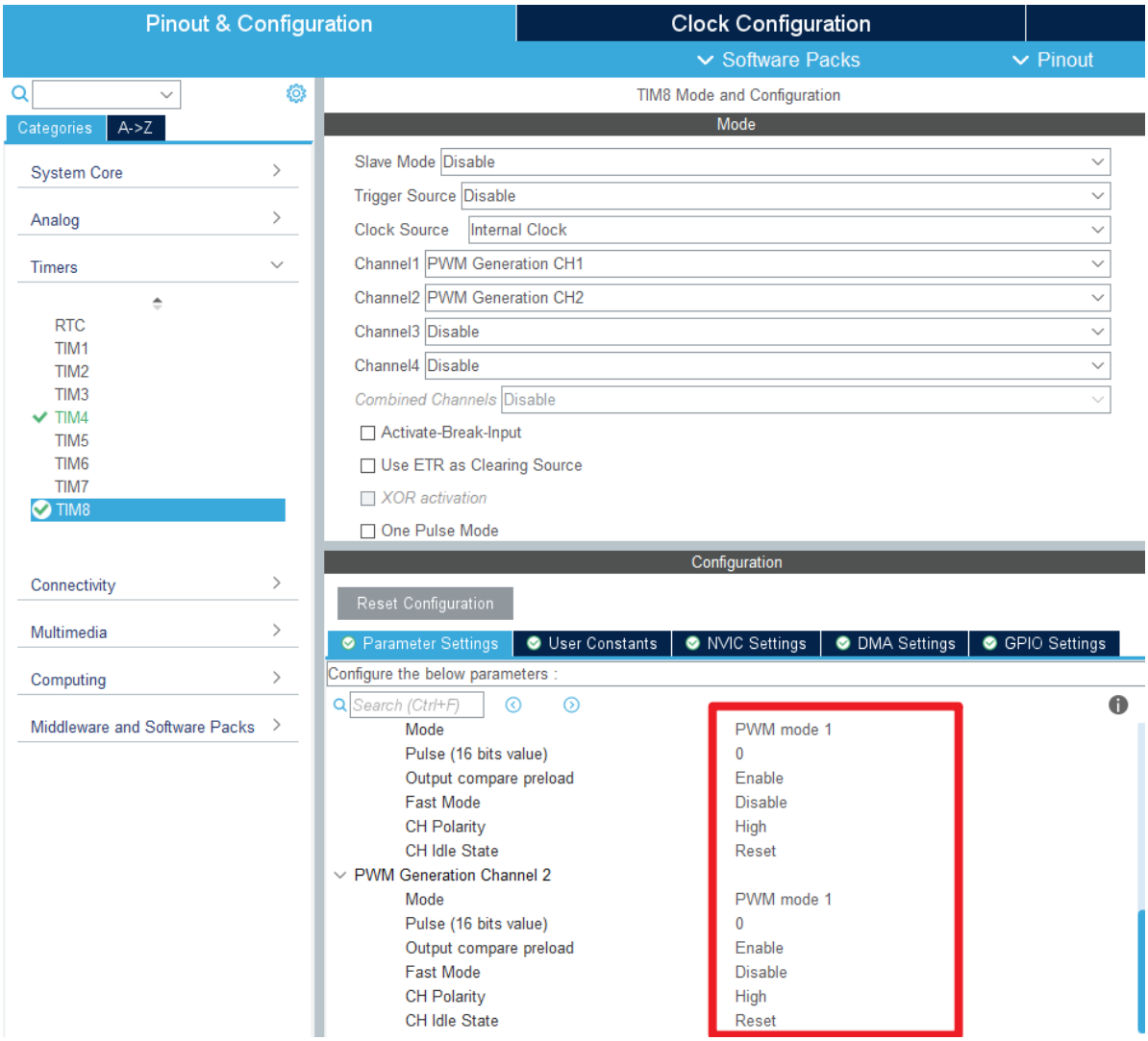
- **Same configuration for TIM2, TIM3, TIM4, TIM5**: TIM4 configuration is demonstrated here





- **TIM1 and TIM8 configuration are the same**: TIM8 configuration is demonstrated here


Pinout & Configuration
Clock Configuration
Software Packs
Pinout
Categories
A->Z
System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
Connectivity
Multimedia
Computing
Middleware and Software Packs
TIM8 Mode and Configuration
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 PWM Generation CH1
Channel2 PWM Generation CH2
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Activate-Break-Input
Use ETR as Clearing Source
XOR activation
One Pulse Mode
Configuration
Reset Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Configure the below parameters :
Search (Ctrl+F)
Counter Settings
Prescaler (PSC - 16 bits value) 0
Counter Mode Up
Counter Period (AutoReload Register - 16 bit 3600-1
Internal Clock Division (CKD) No Division
Repetition Counter (RCR - 8 bits value) 0
auto-reload preload Enable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)
Trigger Event Selection Reset (UG bit from TIMx_EGR)


Pinout & Configuration
Clock Configuration
Software Packs
Pinout
Categories
A->Z
System Core
Analog
Timers
RTC
TIM1
TIM2
TIM3
TIM4
TIM5
TIM6
TIM7
TIM8
Connectivity
Multimedia
Computing
Middleware and Software Packs
TIM8 Mode and Configuration
Mode
Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 PWM Generation CH1
Channel2 PWM Generation CH2
Channel3 Disable
Channel4 Disable
Combined Channels Disable
Activate-Break-Input
Use ETR as Clearing Source
XOR activation
One Pulse Mode
Configuration
Reset Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Configure the below parameters :
Search (Ctrl+F)
Mode PWM mode 1
Pulse (16 bits value) 0
Output compare preload Enable
Fast Mode Disable
CH Polarity High
CH Idle State Reset
PWM Generation Channel 2
Mode PWM mode 1
Pulse (16 bits value) 0
Output compare preload Enable
Fast Mode Disable
CH Polarity High
CH Idle State Reset

4、Main Function

Detailed code can open the project file we provide, into the Bsp folder to view the source code.

Main Function

feature	Corresponding tutorial
key	3. Development board basic tutorial: Key control
Serial port	3, development board basic tutorial: serial communication

4.1、User function

> **function：Motor_Ignore_Dead_Zone**

Function prototypes	int16_t Motor_Ignore_Dead_Zone(int16_t pulse)
Functional Description	The PWM signal dead zone is ignored
Input parameters	**pulse**：Value of pulse
Return value	Value of pulse

> **function：Motor_Stop**

Function prototypes	void Motor_Stop(uint8_t brake)
Functional Description	All motors stopped
Input parameters	**brake**：The value is 0 or 1
Return value	None

> **function：Motor_Set_Pwm**

Function prototypes	void Motor_Set_Pwm(uint8_t id, int16_t speed)
Functional Description	Set motor speed
Input parameters1	**id**：Motor ID
Input parameters2	**speed**：Motor speed
Return value	None

> **function：Encoder_Read_CNT**

Function prototypes	int16_t Encoder_Read_CNT(uint8_t Motor_id)
Functional Description	Read the encoder count
Input parameters	**Motor_id**：Motor ID

Function prototypes	int16_t Encoder_Read_CNT(uint8_t Motor_id)
Return value	Count value

> function：Encoder_Get_Count_Now

Function prototypes	int Encoder_Get_Count_Now(uint8_t Motor_id)
Functional Description	Returns the total count of encoders counted since boot up (single way)
Input parameters	**Motor_id**：Motor ID
Return value	Count value

> function：Encoder_Get_ALL

Function prototypes	void Encoder_Get_ALL(int *Encoder_all)
Functional Description	Returns the total count of encoders counted since boot up (single way)
Input parameters	**Encoder_all**：Points to the head of the encoder array
Return value	None

> function：Encoder_Update_Count

Function prototypes	void Encoder_Update_Count(void)
Functional Description	Update the total count of the encoder
Input parameters	None
Return value	None

4.2、HAL library functions

The HAL library functions that were covered in the previous tutorial will not be covered

```
For information on how to parse the HAL and LL libraries that are covered throughout this tutorial, check out the documentation under the STM32 Manual: STM32F1_HAL and LL Library_ User Manual
```

> function：HAL_TIM_Encoder_Init

Function prototypes	HAL_StatusTypeDef HAL_TIM_Encoder_Init (TIM_HandleTypeDef *htim, const TIM_Encoder_InitTypeDef *sConfig)

Function prototypes	**HAL_StatusTypeDef HAL_TIM_Encoder_Init (TIM_HandleTypeDef *htim, const TIM_Encoder_InitTypeDef *sConfig)**
Functional Description	**Initialize the encoder mode for the timer**
Input parameters1	**htim**：Timer handle address
Input parameters2	**sConfig**：The encoder initializes the configuration structure
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

> **function：HAL_TIM_Encoder_MspInit**

Function prototypes	**void HAL_TIM_Encoder_MspInit(TIM_HandleTypeDef *htim)**
Functional Description	**Initialize the peripheral clock and GPIO for the timer encoder interface**
Input parameters	**htim**：Timer handle address
Return value	None

> **function：HAL_TIM_Encoder_MspDeInit**

Function prototypes	**void HAL_TIM_Encoder_MspDeInit(TIM_HandleTypeDef *htim)**
Functional Description	**Cancel peripheral clock and GPIO initialization for the timer encoder interface**
Input parameters	**htim**：Timer handle address
Return value	None

> **function：HAL_TIM_Encoder_Start**

Function prototypes	**HAL_StatusTypeDef HAL_TIM_Encoder_Start(TIM_HandleTypeDef *htim, uint32_t Channel)**
Functional Description	**Start timer encoder mode**
Input parameters1	**htim**：Timer handle address
Input parameters2	**Channel**：The timer channel
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

5、Experimental phenomenon

After downloading the program successfully, press the RESET button of the development board to open the serial debugging assistant to observe the phenomenon

```
Program download can refer to [2, development environment construction and use:
program download and simulation]
```

phenomenon:

Press KEY1: You can manually turn the four motors to observe the value of the encoder printed by the serial port;

Press KEY2: The motor turns forward and prints the encoder value on the serial port;

Press KEY3: The motor reverses and prints the encoder value on the serial port.

```
The experimental phenomenon can be seen [Motor control _ Experimental
Phenomenon.MP4]
```