Universal timer

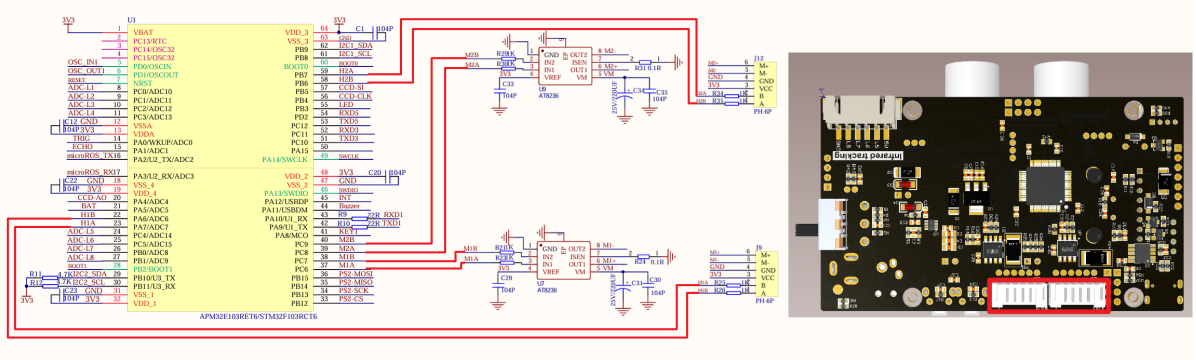
Universal timer

- Hardware connection
- Control principle
  - Pin definition
  - Software code
    - Control function
- Experimental phenomenon

The tutorial demonstrates the general timer (TIM3, TIM4) encoder interface mode to obtain the motor speed and print the motor speed through the serial port.

```
The tutorial only introduces the standard library project code
```

Hardware connection



Since we have configured a special connection line, we only need to install it to the corresponding interface:

Peripherals	Development Board	Description
Motor 1: M+ (M1A)	PC6	The PC6 pin of the development board controls the input pin of the motor driver chip, and the output pin of the driver chip controls motor 1: M+
Motor 1: M- (M1B)	PC7	The PC7 pin of the development board controls the input pin of the motor driver chip, and the output pin of the driver chip controls motor 1: M-
Motor 1: Encoder VCC	3.3V	Connect the development board 3.3V to power the encoder
Motor 1: Encoder GND	GND	Connect the development board GND to the encoder common ground
Motor 1: Encoder A phase (H1A)	PA7	Encoder A of motor 1 is connected to pin PA7 of the development board

Peripherals	Development Board	Description
Motor 1: Encoder B phase (H1B)	PA6	Encoder B of motor 1 is connected to pin PA6 of the development board
Motor 2: M+ (M2A)	PC8	The PC8 pin of the development board controls the input pin of the motor driver chip, and the output pin of the driver chip controls motor 2: M+
Motor 2: M- (M2B)	PC9	The PC9 pin of the development board controls the input pin of the motor driver chip, and the output pin of the driver chip controls motor 2: M-
Motor 2: Encoder VCC	3.3V	Connect the development board 3.3V to power the encoder
Motor 2: Encoder GND	GND	Connect the development board GNDV to the encoder common ground
Motor 2: Encoder A phase (H2A)	PB7	Encoder A of motor 2 is connected to pin PB7 of the development board
Motor 2: Encoder B phase (H2B)	PB6	Encoder B of motor 2 is connected to pin PB6 of the development board

Control principle

Use the encoding interface mode of TIM3 and TIM4 of the general timer on the STM32F103RCT6 development board to count the encoder signal of the motor.

- **General purpose timer**

Timer type	General purpose timer
Timer name	TIM2, TIM3, TIM4, TIM5
Number of counter bits	16
Counting mode	Incremental/decremental/center-aligned
Prescaler factor	1-65536
Generate DMA request	Yes
Capture/compare channels	4
Complementary output	None
Clock frequency	72MHz (maximum)
Mounting bus	APB1

- **520 motor**

Motor model	520 motor
Motor rated voltage	12V
Encoder supply voltage	3.3-5V
Gear set reduction ratio	1:30
Number of magnetic ring lines	11 lines
Encoder type	AB phase incremental Hall encoder

> **Count value**

$$Maximumcountvalue = reductionratio * encoderlinenumber * 4 = 30 * 11 * 4 = 1320$$

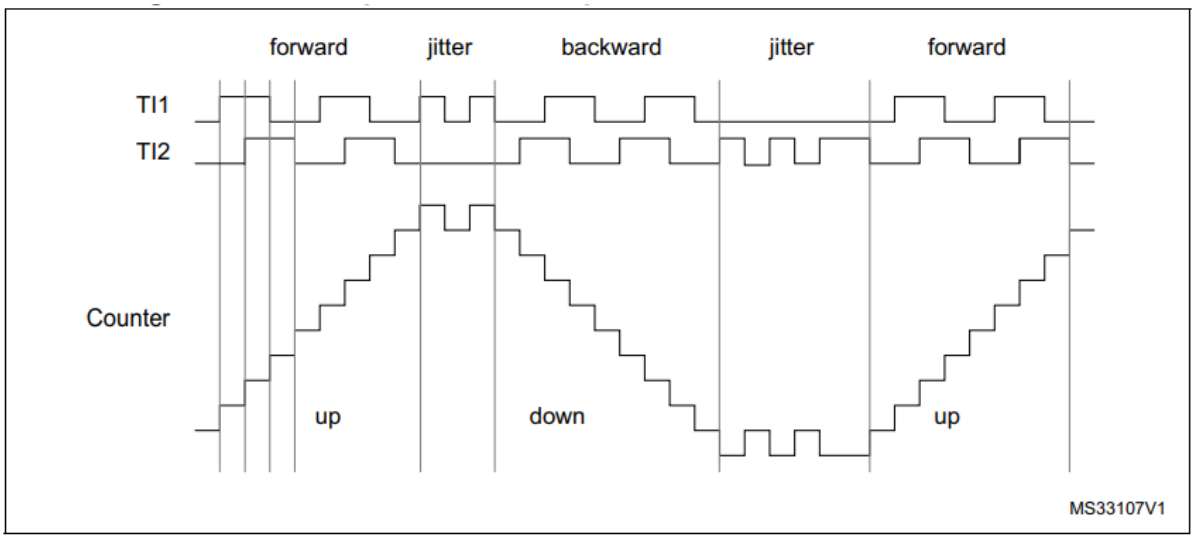
```
4: represents encoder frequency multiplication
According to the count value of one rotation, the speed of the car can be
calculated. This tutorial does not involve speed conversion
```

- **Encoder reading**

Get the count value by directly reading the timer CNT register value.

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 sets **counting on TI1 and TI2**, sets the encoder mode to 4 times frequency, and the input signal is not reversed;




```
Whether the counter counts up or down depends on the AB phase signal
```

Pin definition

Main control chip	Pin	Main function (after reset)	Default multiplexing function	Redefine function
STM32F103RCT6	PA6	PA6	SPI1_MISO/ ADC12_IN6/TIM3_CH1	TIM1_BKIN
STM32F103RCT6	PA7	PA7	SPI1_MOSI/ ADC12_IN7/TIM3_CH2	TIM1_CH1N
STM32F103RCT6	PB6	PB6	I2C1_SCL/TIM4_CH1	USART1_TX
STM32F103RCT6	PB7	PB7	I2C1_SDA/TIM4_CH2	USART1_RX

Software code

Since the default function of the pin is the normal IO pin function, we need to use the multiplexing function.

```
Product supporting materials source code path: Attachment → Source code summary → 1.Base_Course → 8.General_Timer
```

Control function

The tutorial only briefly introduces the code, you can open the project source code to read the details.

> **Encoder_Init_TIM3**

```
void Encoder_Init_TIM3(void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_ICInitTypeDef TIM_ICInitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE); //Enable timer 3 clock
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //Enable PA port clock

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7; //Port configuration
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //Floating input
    GPIO_Init(GPIOA, &GPIO_InitStructure); //Initialize GPIOA according to the set parameters

    TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
    TIM_TimeBaseStructure.TIM_Prescaler = 0; // Prescaler
    TIM_TimeBaseStructure.TIM_Period = ENCODER_TIM_PERIOD; //Set the counter auto-reload value
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; //Select clock division: no division
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; /////TIM counts up
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig(TIM3, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);//Use encoder mode 3
```

```
    TIM_ICStructInit(&TIM_ICInitStructure);
    TIM_ICInitStructure.TIM_ICFilter = 10; //Filter 10
    TIM_ICInit(TIM3, &TIM_ICInitStructure);
    TIM_ClearFlag(TIM3, TIM_FLAG_Update);//Clear TIM update flag
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
    //Reset counter
    TIM_SetCounter(TIM3,0);
    TIM_Cmd(TIM3, ENABLE);
}
```

Encoder_Init_TIM4

```
void Encoder_Init_TIM4(void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_ICInitTypeDef TIM_ICInitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE); //Enable the clock of
timer 4
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //Enable the clock of
PB port

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7; //Port configuration
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //Floating input
    GPIO_Init(GPIOB, &GPIO_InitStructure); //Initialize GPIOB according to the
set parameters
    TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
    TIM_TimeBaseStructure.TIM_Prescaler = 0x0; // Prescaler
    TIM_TimeBaseStructure.TIM_Period = ENCODER_TIM_PERIOD; // Set counter auto-
reload value
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1; // Select clock
division: no division
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; /////TIM counts
up
    TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig(TIM4, TIM_EncoderMode_TI12, TIM_ICPolarity_Rising,
TIM_ICPolarity_Rising); // Use encoder mode 3
    TIM_ICStructInit(&TIM_ICInitStructure); TIM_ICInitStructure.TIM_ICFilter =
10; TIM_ICInit(TIM4, &TIM_ICInitStructure);
    TIM_ClearFlag(TIM4, TIM_FLAG_Update);//Clear the TIM update flag
TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE); //Re set counter
TIM_SetCounter(TIM4,0); TIM_Cmd(TIM4, ENABLE);
}
```

Read_Encoder

```
int Read_Encoder(Motor_ID MYTIMX)
{
   int Encoder_TIM;
   switch(MYTIMX)
     {
         case MOTOR_ID_ML:  Encoder_TIM= (short)TIM3 -> CNT;  TIM3 ->
CNT=0;break;
         case MOTOR_ID_MR:  Encoder_TIM= (short)TIM4 -> CNT;  TIM4 ->
CNT=0;break;
         default: Encoder_TIM=0;
     }
        return Encoder_TIM;
}
```

> **TIM3_IRQHandler**

```
void TIM3_IRQHandler(void)
{
    if(TIM3->SR&0X0001)// The overflow is interrupted
    {
    }
    TIM3->SR&=~(1<<0); // Clears the interrupt flag bit
}
```

> **TIM4_IRQHandler**

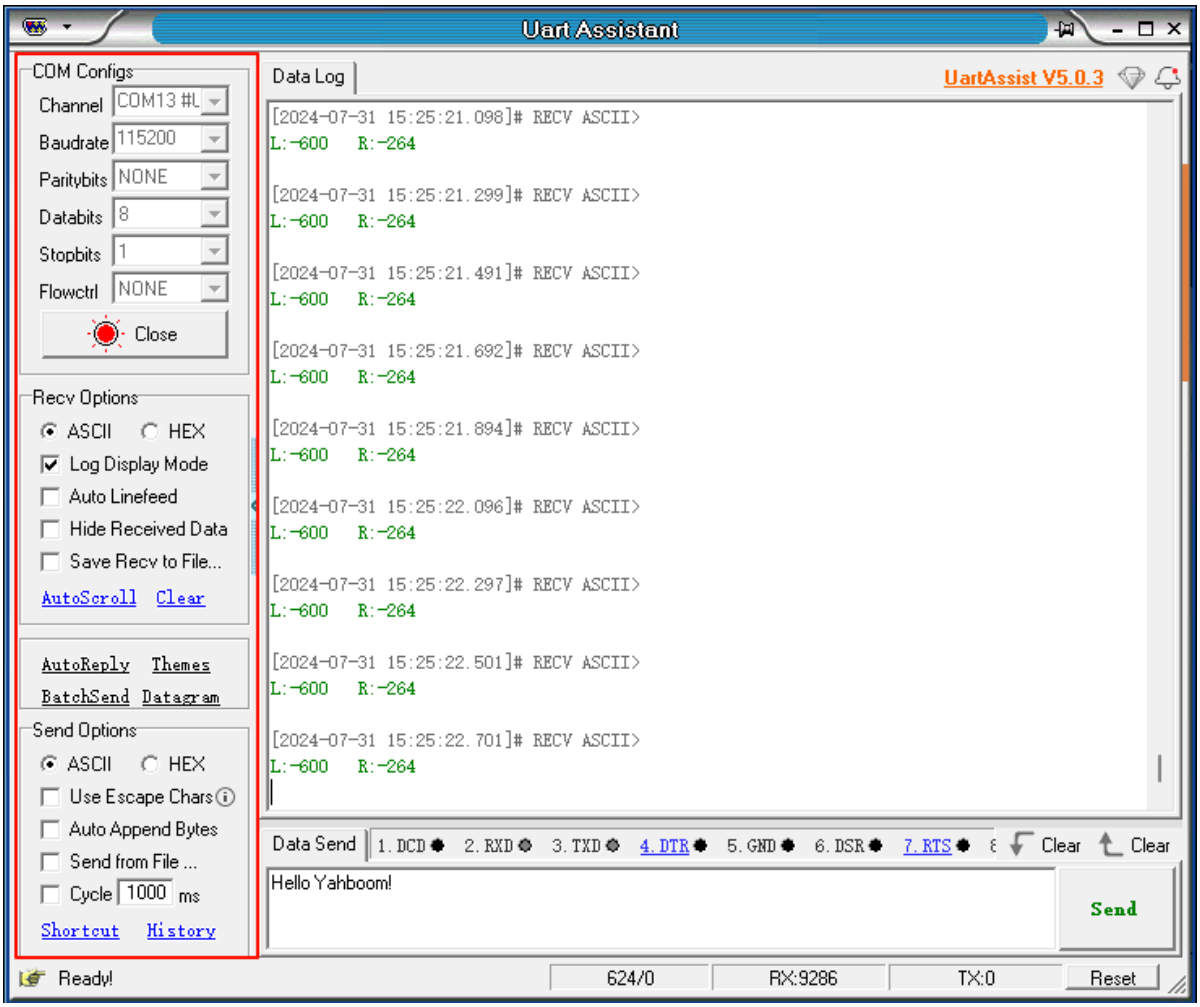
```
void TIM4_IRQHandler(void)
{
    if(TIM4->SR&0X0001)//Overflow interrupt
    {
    }
    TIM4->SR&=~(1<<0);//Clear interrupt flag
}
```

Experimental phenomenon

The General_Timer.hex file generated by the project compilation is located in the OBJ folder of the General_Timer project. Find the General_Timer.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

After the program is successfully downloaded: Use the serial port debugging assistant to see the encoder count information printed by the serial port.

```
When using the serial port debugging assistant, you need to pay attention to the
serial port settings. If the settings are wrong, the phenomenon may be
inconsistent.
```


Uart Assistant
COM Configs
Channel
COM13 #L
Baudrate
115200
Paritybits
NONE
Databits
8
Stopbits
1
Flowctrl
NONE
Close
Recv Options
ASCII
HEX
Log Display Mode
Auto Linefeed
Hide Received Data
Save Recv to File...
AutoScroll
Clear
AutoReply
Themes
BatchSend
Datagram
Send Options
ASCII
HEX
Use Escape Chars
Auto Append Bytes
Send from File ...
Cycle
1000
ms
Shortcut
History
Data Log
UartAssist V5.0.3
[2024-07-31 15:25:21.098]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:21.299]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:21.491]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:21.692]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:21.894]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:22.096]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:22.297]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:22.501]# RECV ASCII>
L:-600 R:-264
[2024-07-31 15:25:22.701]# RECV ASCII>
L:-600 R:-264
Data Send
1. DCD
2. RXD
3. TXD
4. DTR
5. GND
6. DSR
7. RTS
Clear
Clear
Hello Yahboom!
Send
Ready!
624/0
RX:9286
TX:0
Reset