定时器-Tim

STM32定时器可以分为3类：

- 高级控制定时器
  高级定时器适合多种用途，包含输入捕获、输出比较、PWM、**带死区控制**的PWM等，所以可以用来做电机控制。
- 通用定时器
  通用定时器就是基本的定时器，包含输入捕获、输出比较、PWM
- 基本定时器
  基本定时器可以为通用定时提供时间基准，可以为DAC提供时钟。

◆三种（4）STM32定时器区别

定时器种类	位数	计数器模式	产生DMA请求	捕获/比较通道	互补输出	特殊应用场景
高级定时器（TIM1,TIM8）	16	向上，向下，向上/下	可以	4	有	带可编程死区的互补输出
通用定时器（TIM2,TIM5）	32	向上，向下，向上/下	可以	4	无	通用。定时计数，PWM输出，输入捕获，输出比较
通用定时器（TIM3,TIM4）	16	向上，向下，向上/下	可以	4	无	通用。定时计数，PWM输出，输入捕获，输出比较
通用定时器（TIM9~TIM14）	16	向上	没有	2	无	通用。定时计数，PWM输出，输入捕获，输出比较
基本定时器（TIM6,TIM7）	16	向上，向下，向上/下	可以	0	无	主要应用于驱动DAC

STM32的通用定时器

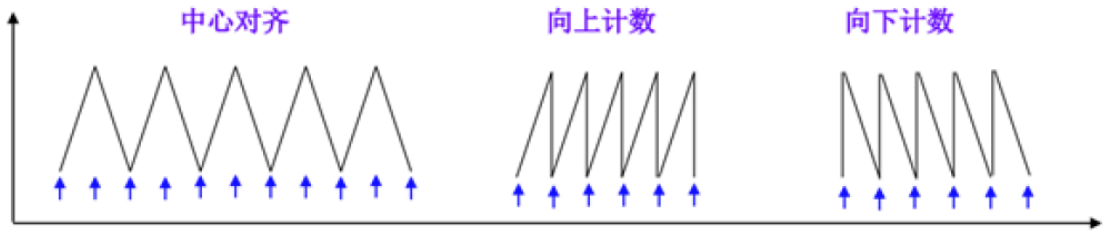
STM32的通用定时器是由一个可编程预分频器（PSC）驱动的16位自动重装载计数器（CNT）构成，可用于测量输入
脉冲长度（输入捕获）或者产生输出波形（输出比较和PWM）等。

- 位于低速的APB1总线上（注意：高级定时器是在高速的APB2总线上）；
- 16位向上、向下、向上/向下（中心对齐）计数模式，自动装载计数器（TIMx_CNT）；
- 16位可编程（可以实时修改）预分频器（TIMx_PSC），计数器时钟频率的分频系数 为 1～65535之间的任意
  数值；
- 4 个独立通道（TIMx_CH1~4），这些通道可以用来作为：
  1. 输入捕获
  2. *输出比较
  3. PWM生成(边缘或中间对齐模式)
  4. *单脉冲模式输出
- STM32每个通用定时器都是完全独立的，没有互相共享的任何资源

计数器模式

通用定时器可以向上计数、向下计数、向上向下双向计数模式。
向上计数模式：计数器从0计数到自动加载值（TIMx_ARR），然后重新从0开始计数并且产生一个计数器溢出事件。
向下计数模式：计数器从自动装入的值（TIMx_ARR）开始向下计数到0，然后从自动装入的值重新开始，并产生一
个计数器向下溢出事件。 中央对齐模式（向上/向下计数）：计数器从0开始计数到自动装入的值-1，产生一个计数器
溢出事件，然后向下计数到1并且产生一个计数器溢出事件；然后再从0开始重新计数。

可以借助于这个图来理解这三种模式：



三个主要的寄存器：

- 计数器（TIMx_CNT）：存放计数器的当前值。
- 预分频器(TIMx_PSC)：对CK_PSC进行预分频。此时需要注意：CK_CNT计算的时候，预分频系数要+1。
- 自动重装载寄存器(TIMx_ARR)：包含将要被传送至实际的自动重装载寄存器的数值。(有点绕口，后面慢慢理
  解其含义)

通用定时器超时时间

Tout=（ARR+1)(PSC+1)/TIMxCLK
其中：Tout的单位为us，TIMxCLK的单位为MHz。

通用定时器相关配置库函数

- 1个初始化函数

```
HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim);
```

作用：用于对**预分频系数、计数方式、自动重装载计数值**、时钟分频因子等参数的设置。

- 2个使能函数

```
__HAL_TIM_ENABLE(__HANDLE__) ; //使能定时器
__HAL_TIM_ENABLE_IT(__HANDLE__, __INTERRUPT__); //使能定时器中断
```

作用：前者使能定时器，后者使能定时器中断。

- 4个状态标志位获取函数

```
__HAL_TIM_GET_FLAG(__HANDLE__, __FLAG__); //check中断标志位（返回0/1）
__HAL_TIM_CLEAR_FLAG(__HANDLE__, __FLAG__); //clear中断标志位（返回0/1）
__HAL_TIM_GET_IT_SOURCE(__HANDLE__, __INTERRUPT__); //check中断是否发生（返回SET or RESET）
__HAL_TIM_CLEAR_IT(__HANDLE__, __INTERRUPT__);//clear 中断标志位（无返回值）
```

作用：前两者获取（或清除）状态标志位，后两者为获取（或清除）中断状态标志位。

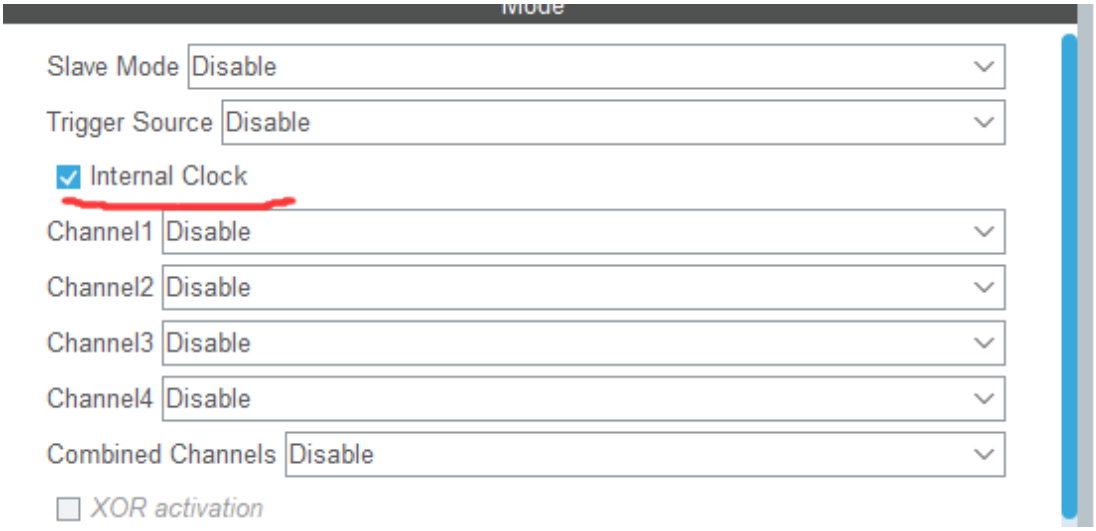
- ```
  __HAL_TIM_SET_COMPARE(__HANDLE__, __CHANNEL__, __COMPARE__);//设置CC寄存器的值
  __HAL_TIM_GET_COMPARE(__HANDLE__, __CHANNEL__); //获取CC寄存器的值
  __HAL_TIM_SET_COUNTER(__HANDLE__, __COUNTER__); //设置计数寄存器的值
  __HAL_TIM_GET_COUNTER(__HANDLE__); //获取计数寄存器当前值
  __HAL_TIM_SET_AUTORELOAD(__HANDLE__, __AUTORELOAD__); //设置自动重装载寄存器的值
  __HAL_TIM_GET_AUTORELOAD(__HANDLE__); //获取自动重装载寄存器的值
  __HAL_TIM_SET_CAPTUREPOLARITY(__HANDLE__, __CHANNEL__, __POLARITY__);//设置捕获模式（上升沿/下降沿/边沿）
  ```

- IT/DMA回调函数

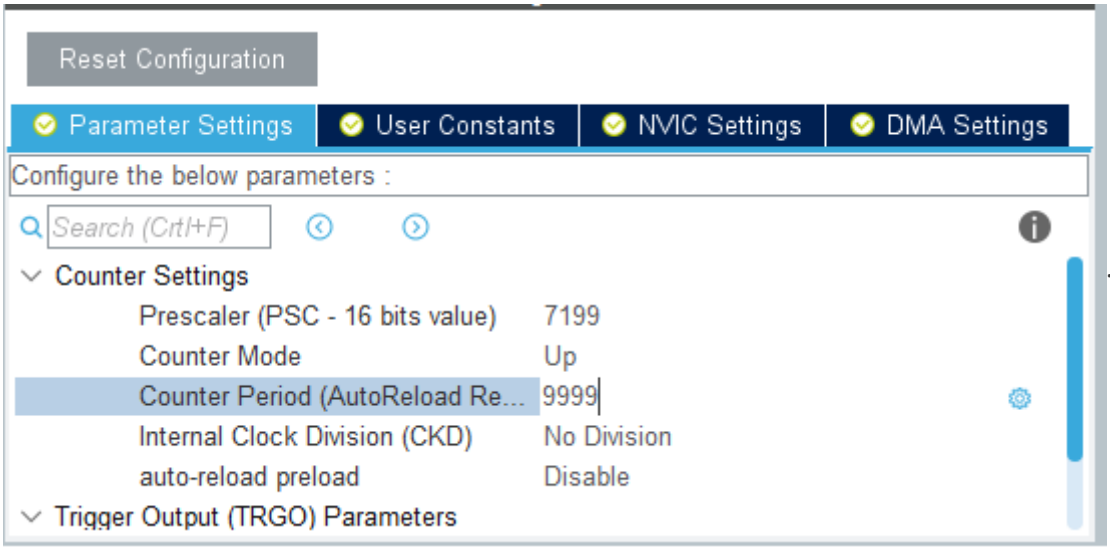
```
/* Callback in non blocking modes (Interrupt and DMA) *************************/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim);
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);
void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim);
void HAL_TIM_TriggerCallback(TIM_HandleTypeDef *htim);
void HAL_TIM_ErrorCallback(TIM_HandleTypeDef *htim);
/**
```

TIM定时器中断实验

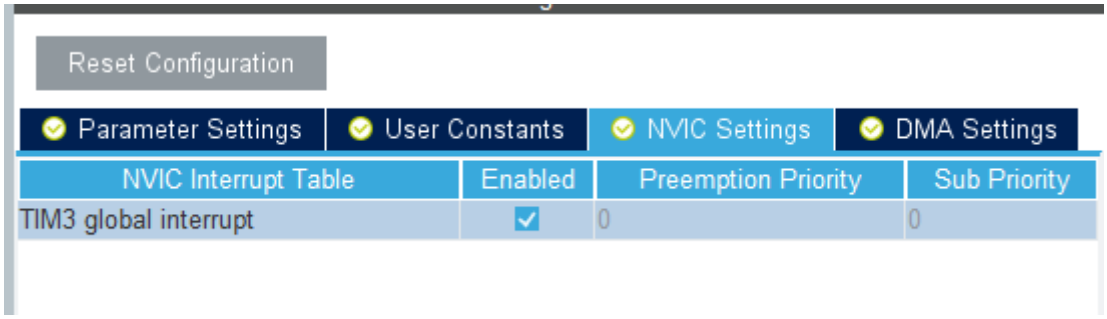
1. 配置RCC,SYS,配置时钟
2. 开启TIM3定时器
```



3. 设置参数psc、arrr



4. 开启定时器中断



5. 生成代码

6. 在主函数内加入

```
187   HAL_TIM_Base_Start_IT(&htim3); //开启定时器中断
188
```

7. 在主函数上方user code区域加入

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

  if( htim->Instance == TIM3 ){
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
  }

}
```

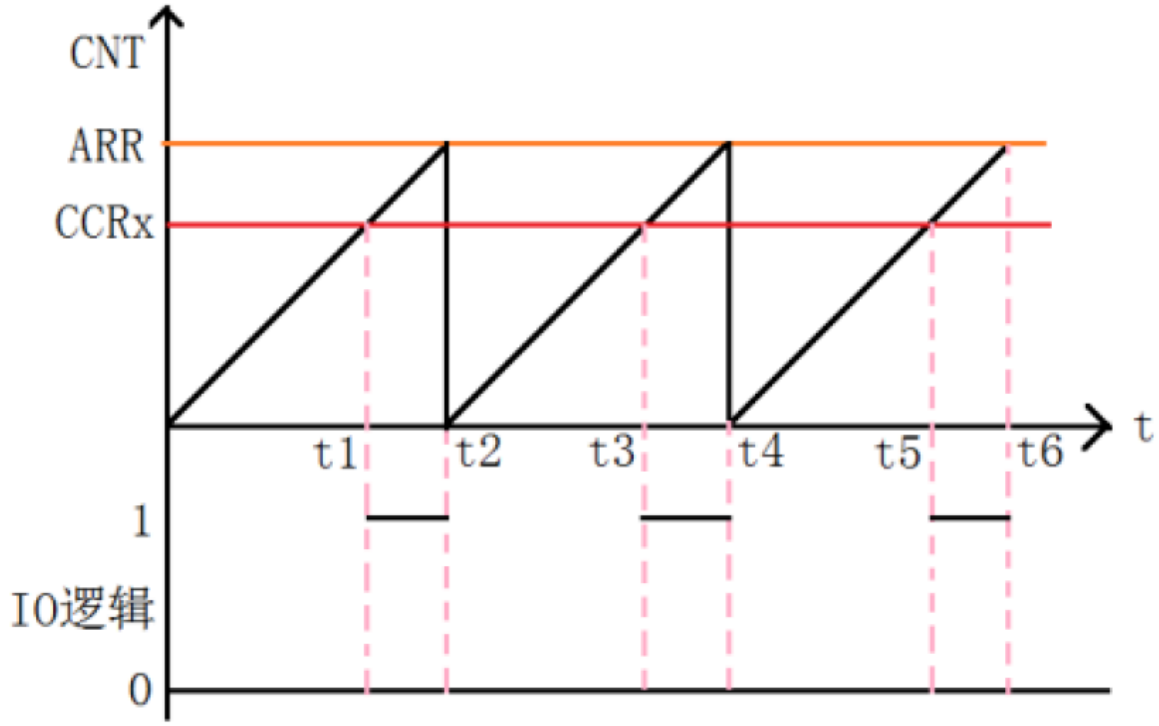
这样就能看到PC13以1s频率闪烁了

TIM定时器实现PWM输出

STM32定时器除了基本定时器以外，其他的定时器都可以产生PWM输出

TIM产生PWM波的原理

下面以向上计数为例，简单地讲述一下PWM的工作原理：



- 在PWM输出模式下，除了CNT（计数器当前值）、ARR（自动重装载值）之外，还多了一个值CCRx（捕获/
  比较寄存器值）。
- 当CNT小于CCRx时，TIMx_CHx通道输出低电平；
- 当CNT等于或大于CCRx时，TIMx_CHx通道输出高电平。

这个时候就可以对其下一个准确的定义了：**所谓脉冲宽度调制模式（PWM模式），就是可以产生一个由TIMx_ARR
寄存器确定频率，由TIMx_CCRx寄存器确定占空比的信号。它是利用微处理器的数字输出来对模拟电路进行控制的
一种非常有效的技术。**

占空比计算公式为: $q = \frac{TIMx->CCRx}{TIMx->ARR}$

与PWM产生有关的寄存器

CCRx寄存器：捕获 /比较值寄存器：设置比较值；

CCMRx寄存器：OCxM[2:0]位：对于PWM方式下，用于设置PWM模式1或者PWM模式2；

CCER寄存器：CCxP位：输出极性。0：高电平有效，1：低电平有效。

CCER寄存器：CCxE位：输出使能。0：关闭，1：打开。

PWM输出的模式区别

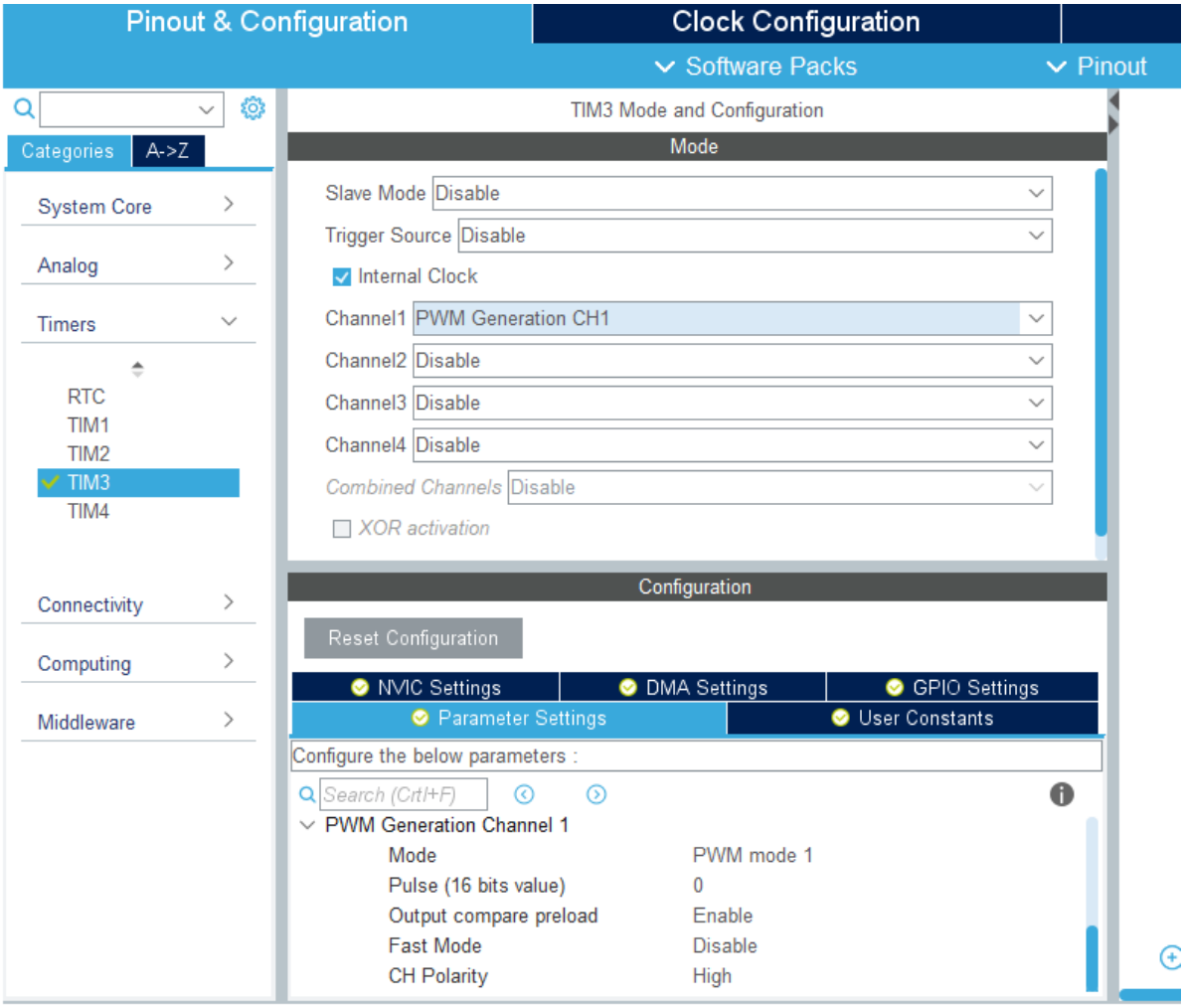
通过设置寄存器TIMx_CCMR1的OC1M[2:0]位来确定PWM的输出模式：
PWM模式1：在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。

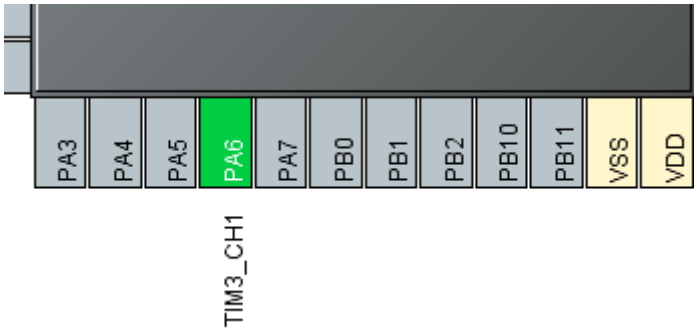
PWM模式2：在向上计数时， 一旦TIMx_CNT<TIMx_CCR1 时通道1 为无效电平， 否则为有效电平； 在向下计数时， 一旦
TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。

注意：PWM的模式只是区别什么时候是有效电平，但并没有确定是高电平有效还是低电平有效。这需要结合CCER
寄存器的CCxP位的值来确定。

例如：若PWM模式1，且CCER寄存器的CCxP位为0，则当TIMx_CNT<TIMx_CCR1时，输出高电平；同样的，若PWM
模式1，且CCER寄存器的CCxP位为2，则当TIMx_CNT<TIMx_CCR1时，输出低电平。

TIM定时器产生PWM波实验





生成工程模板，编写代码：

```
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1); //开启PWM通道
  __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,500);//调整CCR
  __HAL_TIM_SET_AUTORELOAD(&htim3,999);//调整ARR
  /* USER CODE END 2 */
```

TIM定时器实现输入捕获

输入捕获模式可以用来测量脉冲宽度或者测量频率。STM32的定时器，除了TIM6、TIM7，其他的定时器都有输入捕
获的功能。下面以一个简单的脉冲输入为例，简单地讲述一下输入捕获用于测量脉冲宽度的工作原理：

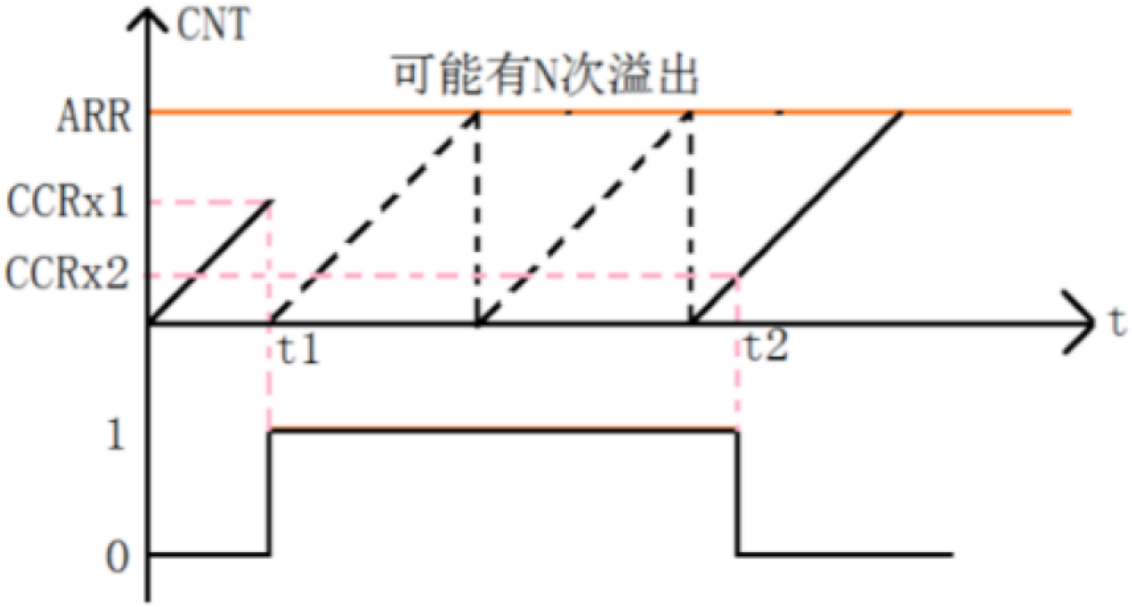
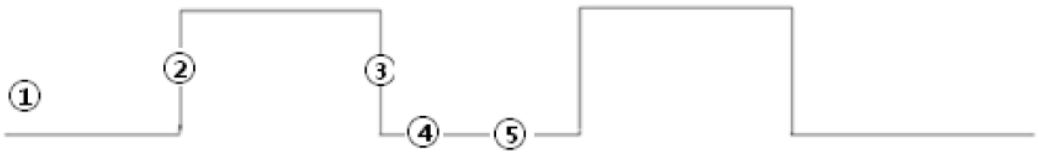


图 14.1.1 输入捕获脉宽测量原理



如图 14.1.1 所示，就是输入捕获测量高电平脉宽的原理，假定定时器工作在向上计数模式， 图中 t1~t2 时间，就是我
们需要测量的高电平时间。

测量方法如下：首先设置定时器通道 x 为 上升沿捕获，这样，t1 时刻，就会捕获到当前的 CNT 值，然后立即清零
CNT，并设置通道 x 为下降沿捕获，这样到 t2 时刻，又会发生捕获事件，得到此时的 CNT 值，记为 CCRx2。这样，
根据定时器的计数频率，我们就可以算出 t1~t2 的时间，从而得到高电平脉宽。 在 t1~t2 之间，可能产生 N 次定时器
溢出，这就要求我们对定时器溢出做处理，防止高 电平太长，导致数据不准确。如图14.1.1所示，t1~t2 之间，CNT计
数的次数等于：**N*ARR+CCRx2**， 有了这个计数次数，再乘以 CNT 的计数周期，即可得到 t2-t1 的时间长度，即高
电平持续时间。 输入捕获的原理，我们就介绍到这。

同时还可以配置捕获时是否触发中断/DMA 等。

输入捕获相关寄存器

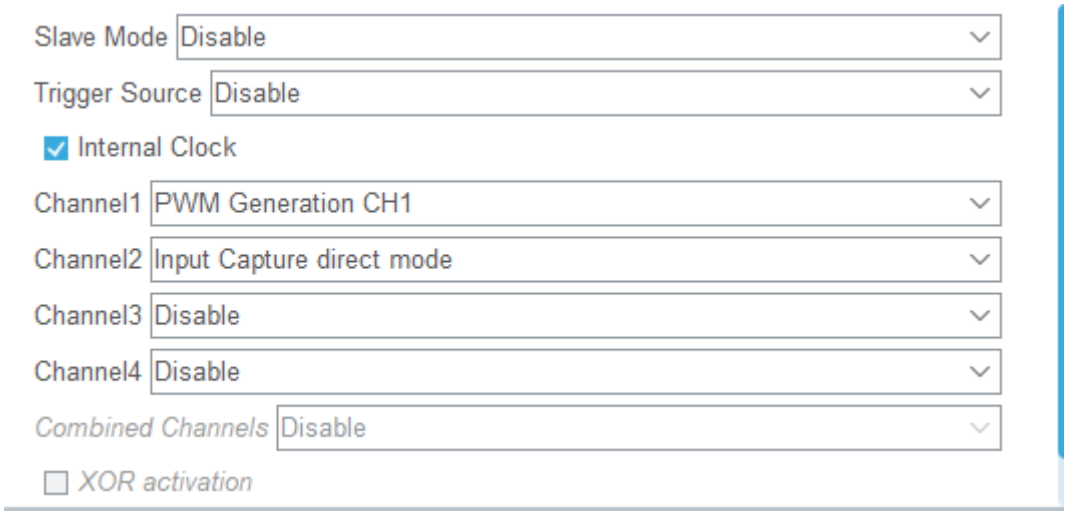
捕获/比较模式寄存器1（TIMx_CCMR1）
作用：在输入捕获模式下，确定数字滤波器、通道映射、预分频系数。
捕获/比较使能寄存器（TIMx_CCER）
作用：在输入捕获模式下，确定捕捉极性和捕捉使能。
捕获/比较寄存器1（TIMx_CCR1）
作用：在输入捕获模式下，确定上一次输入捕捉事件传输的计数值。

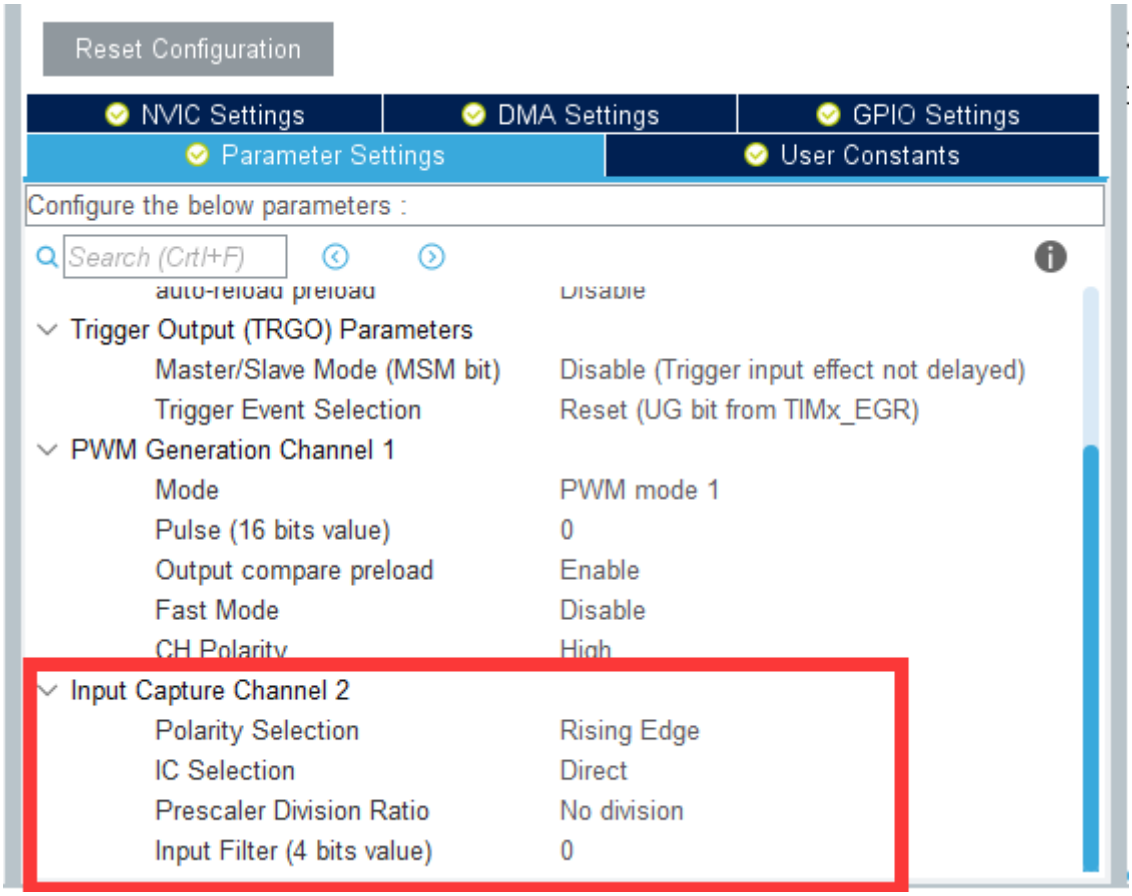
输入捕获相关配置库函数

```
/* Timer Input Capture functions
**********************************************/
HAL_StatusTypeDef HAL_TIM_IC_Init(TIM_HandleTypeDef *htim);
HAL_StatusTypeDef HAL_TIM_IC_DeInit(TIM_HandleTypeDef *htim);
void HAL_TIM_IC_MspInit(TIM_HandleTypeDef *htim);
void HAL_TIM_IC_MspDeInit(TIM_HandleTypeDef *htim);
/* Blocking mode: Polling */
HAL_StatusTypeDef HAL_TIM_IC_Start(TIM_HandleTypeDef *htim, uint32_t Channel);
HAL_StatusTypeDef HAL_TIM_IC_Stop(TIM_HandleTypeDef *htim, uint32_t Channel);
/* Non-Blocking mode: Interrupt */
HAL_StatusTypeDef HAL_TIM_IC_Start_IT(TIM_HandleTypeDef *htim, uint32_t
Channel);
HAL_StatusTypeDef HAL_TIM_IC_Stop_IT(TIM_HandleTypeDef *htim, uint32_t Channel);
/* Non-Blocking mode: DMA */
HAL_StatusTypeDef HAL_TIM_IC_Start_DMA(TIM_HandleTypeDef *htim, uint32_t
Channel, uint32_t *pData,
uint16_t Length);
HAL_StatusTypeDef HAL_TIM_IC_Stop_DMA(TIM_HandleTypeDef *htim, uint32_t
Channel);
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);//输入捕获中断回调函数
```

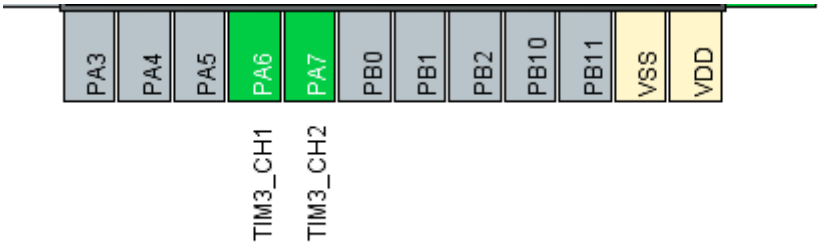
用Cube生成工程

本实验我们利用TIM3的CH1和CH2，CHI设置为PWM模式，CH2设置为IC输出模式。并开启中断。





我们开启TIM3的CH1和CH2通道的同时，Cube帮我们映射到了管脚PA6\PA7上，如图。



Cube主要设置就这么多。

代码部分我们将用输出捕获测量输出的pwm脉宽

主函数前加入

```
/* USER CODE BEGIN PV */
int pwmVal = 0;
int high_time, IC_flag, IC_buf[2];
/* USER CODE END PV */
```

初始化函数后加入代码开启PWM输出，输入捕获

```
  /* USER CODE BEGIN 2 */
HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_2); //开启输入捕获中断
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1); //开始pwm输出
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    while(pwmVal < 9999){
      pwmVal += 1000;
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,pwmVal);//修改ccr 调节占空比和脉宽
      HAL_Delay(2000);
    }
    while(pwmVal){
      pwmVal -= 1000;
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,pwmVal);
      HAL_Delay(2000);
    }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
```

编写输入捕获中断回调函数

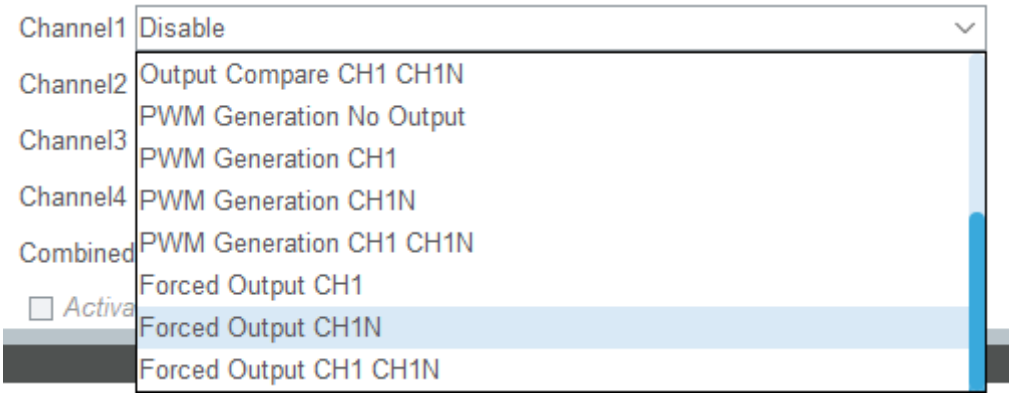
```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
  if(htim->Instance == htim3.Instance){

    if(IC_flag == 0){
        IC_buf[0] = HAL_TIM_ReadCapturedValue(&htim3, TIM_CHANNEL_2);
        __HAL_TIM_SET_CAPTUREPOLARITY(&htim3, TIM_CHANNEL_2, TIM_INPUTCHANNELPOLARITY_FALLING);
        IC_flag = 1;
    }
    else{
      IC_buf[1] = HAL_TIM_ReadCapturedValue(&htim3, TIM_CHANNEL_2);
      if(IC_buf[1]>IC_buf[0]){
        high_time = IC_buf[1]-IC_buf[0];
      }
      else{
        high_time = IC_buf[1]-IC_buf[0] + __HAL_TIM_GET_AUTORELOAD(&htim3);
      }
    }

  }
}
/* USER CODE END 0 */
```

Homework

- 了解这些中断的意义并举出使用场景的例子

NVIC Interrupt Table	Enab...	Preemption Pri...	Sub Pri...
TIM1 break interrupt	☐	0	0
TIM1 update interrupt	☐	0	0
TIM1 trigger and commutation interrupts	☐	0	0
TIM1 capture compare interrupt	☐	0	0

- 百度了解课上没有说明的定时器通道的其他功能 写出他们的意义(不仅局限于图中内容)



- 用定时器自带的功能读编码器
- 完成腾讯文档的任务二
- 预习串口通信，IIC通信，DMA是什么