

第9章 定时器系统

9.1 定时器系统简介

9.2 基本定时功能

9.3 捕抓/比较功能

9.4 定时器典型应用步骤及常用库函数

9.5 应用实例

9.1 定时器系统简介

9.1 定时器系统简介

定时器在检测、控制领域有广泛应用，可作为应用系统运行的控制节拍，实现信号检测、控制、输入信号周期测量或电机驱动等功能。在很多的场合，都会用到定时器，因此定时器系统是现在微控制器中的一个不可缺少的组成部分。

定时器有很多用途，包括**基本定时功能**、**生成输出波形**（输出比较、PWM和带死区插入的互补PWM）和**测量输入信号的脉冲宽度**（输入捕抓）等。

9.1 定时器系统简介

9.1.1 定时器概述

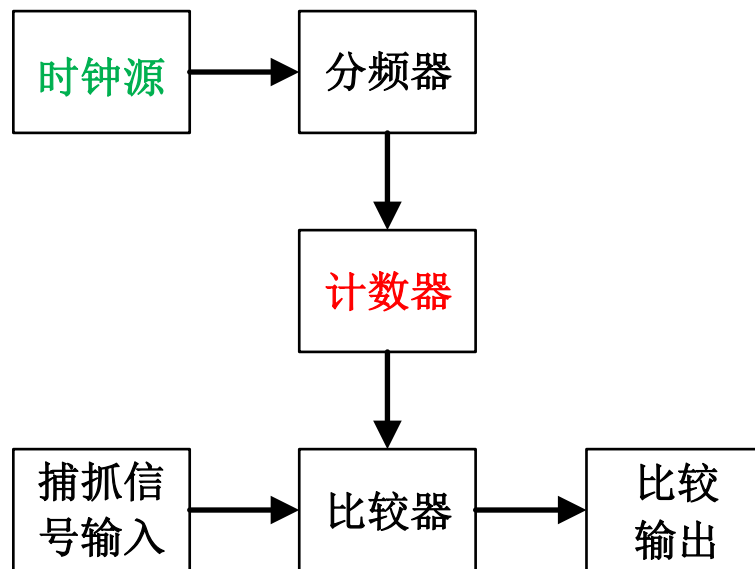
STM32F429系列总共**最多**有14个定时器：

定时器	类 型	计数器长度	预分频系数	计数方向	捕获/比较通道	总线	最大定时器时钟 (MHz)	互补输出	DMA 请求
TIM1 和 TIM8	高级控制	16 位	1~65536 (整 数)	递增、递 减、递增/ 递减	4	APB2	180	有	有
TIM2, TIM5	通用	32 位	1~65536 (整 数)	递增、递 减、递增/ 递减	4	APB1	90/180	无	有
TIM3, TIM4		16 位	1~65536 (整 数)	递增、递 减、递增/ 递减	4			无	有
TIM9		16 位	1~65536 (整 数)	递增	2	APB2	180	无	无
TIM10, TIM11		16 位	1~65536 (整 数)	递增	1			无	无
TIM12		16 位	1~65536 (整 数)	递增	2	APB1	90/180	无	无
TIM13, TIM14		16 位	1~65536 (整 数)	递增	1			无	无
TIM6 和 TIM7	基本	16 位	1~65536 (整 数)	递增	0			无	有

9.1 定时器系统简介

9.1.2 定时器结构

定时器可用于各种用途，包括基本定时功能、生成输出波形（输出比较、PWM 和带死区插入的互补 PWM）和测量输入信号的脉冲宽度（输入捕获）等。



定时器基本结构

9.1 定时器系统简介

9.1.2 定时器结构

- 1)、16/32位递增、递减、递增/递减自动重载计数器。
- 2)、16 位可编程预分频器，用于对计数器时钟频率进行分频（即运行时修改），分频系数介于1 到 65536 之间。
- 3)、多达 4 个独立通道，可用于：
 - 输入捕获
 - 输出比较
 - PWM 生成（边沿和中心对齐模式）
 - 单脉冲模式输出
- 4)、带可编程死区的互补输出。
- 5)、使用外部信号控制定时器且可实现多个定时器互连的同步电路。
- 6)、重复计数器，用于仅在给定数目的计数器周期后更新定时器寄存器。

9.1 定时器系统简介

9.1.2 定时器结构

◆ 定时器功能特点描述

7)、用于将定时器的输出信号置于复位状态或已知状态的断路输入。

8)、发生如下事件时生成中断/DMA 请求：

(1)更新：计数器上溢/下溢、计数器初始化（通过软件或内部/外部触发）

(2)触发事件（计数器启动、停止、初始化或通过内部/外部触发计数）

(3)输入捕获

(4)输出比较

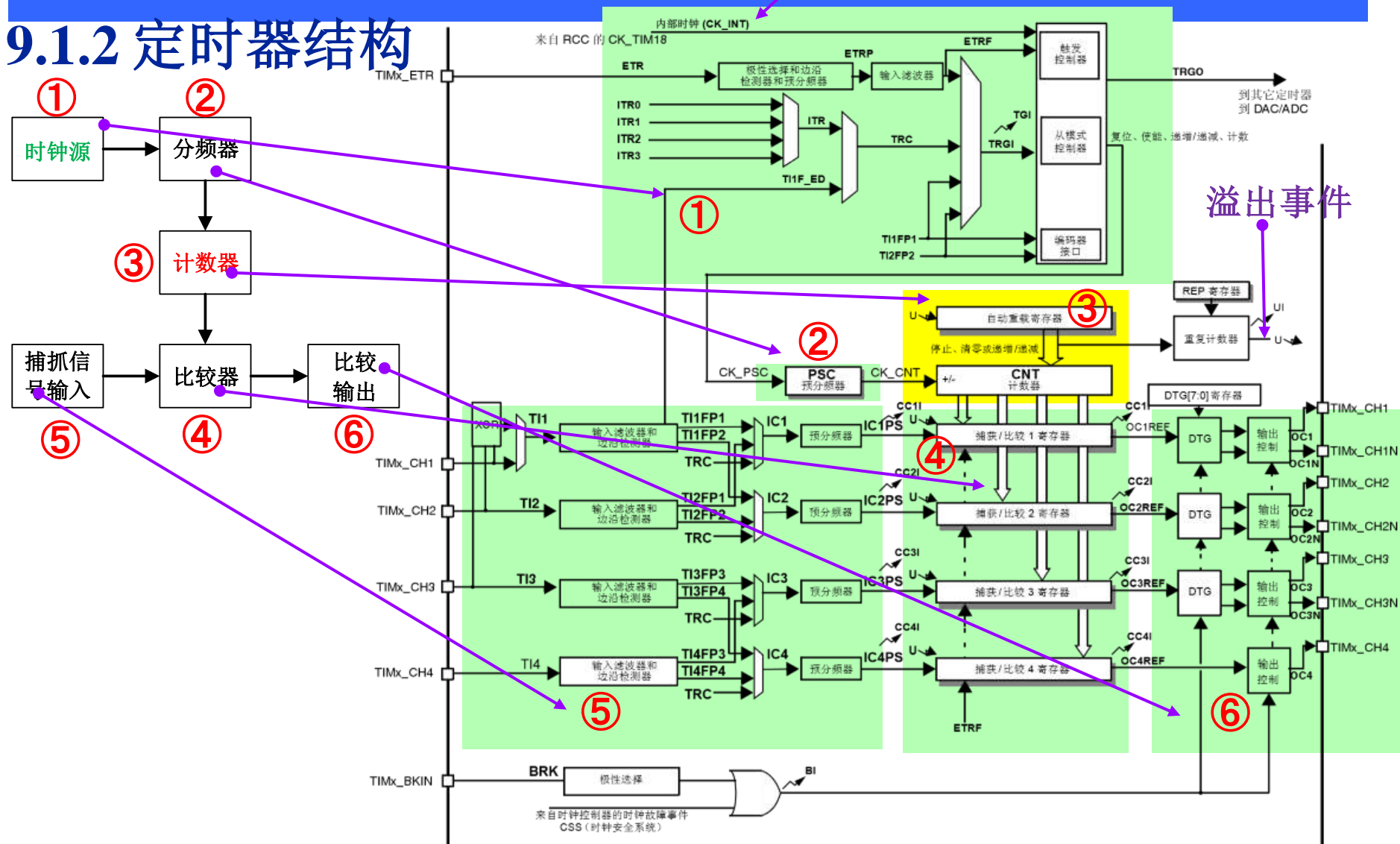
(5)断路输入

9)、支持定位用增量（正交）编码器和霍尔传感器电路。

9.1 定时器系统简介

内部时钟

9.1.2 定时器结构

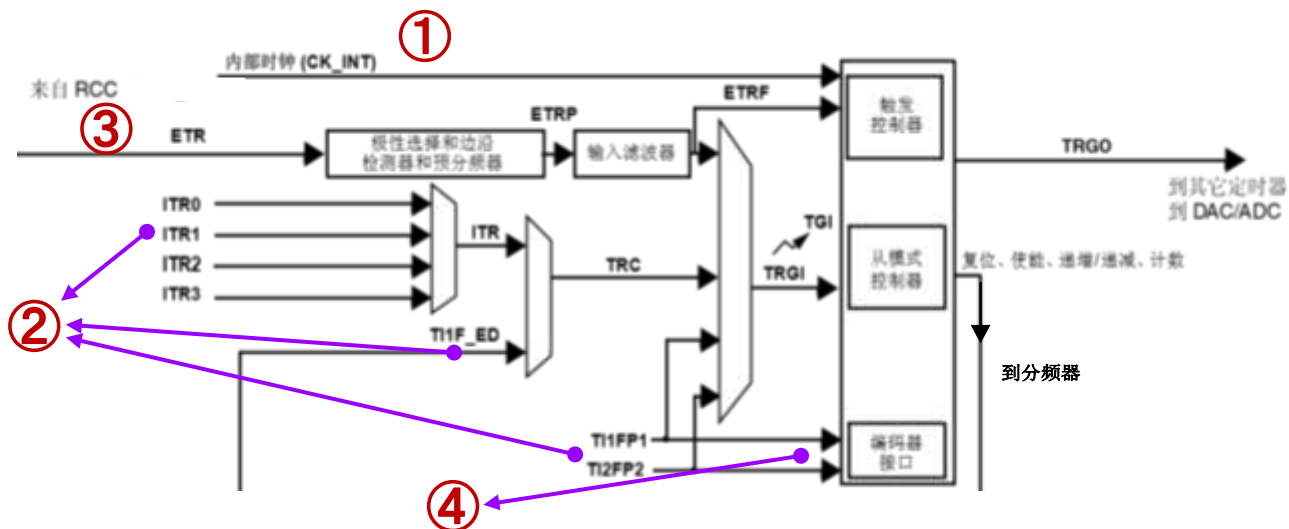


9.1 定时器系统简介

9.1.3 时钟源

定时器计数需要计数器时钟，可由下列时钟源提供：

- ① 内部时钟（CK_INT）源模式。
- ② 外部时钟源模式1：外部输入。
- ③ 外部时钟源模式2：外部触发输入ETR。
- ④ 编码器模式。



9.1 定时器系统简介

9.1.3 时钟源

1. 内部时钟源模式

根据RCC_DKCFGR的TIMPRE位的状态，定时器使用的内部时钟源频率不同，见第8章相关内容。例如，在 $HCLK=180MHz$ ， $PCLK2=90MHz$ （APB2总线时钟预分频系数=2）的情况下，定时器TIM1的内部时钟源 $=2 \times PCLK2=180MHz$ 。

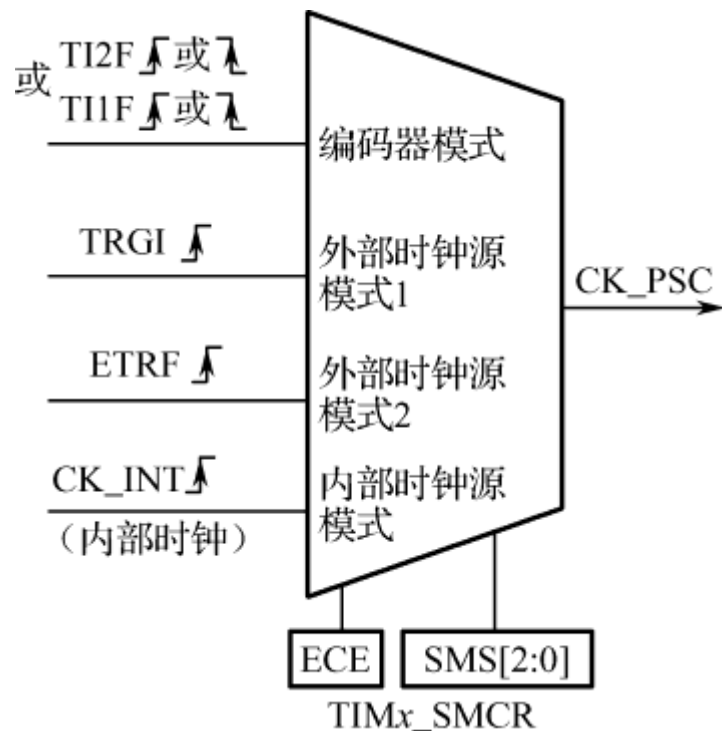


图9-3 定时器计数时钟源

9.1 定时器系统简介

9.1.3 时钟源

2. 外部时钟源模式1

计数器可在选定的输入信号上出现上升沿或下降沿时计数。外部时钟源模式1结构如图9-4所示。此时，对于TIMx_SMCR的中位段TS的设置，可选择的外部时钟源如下。

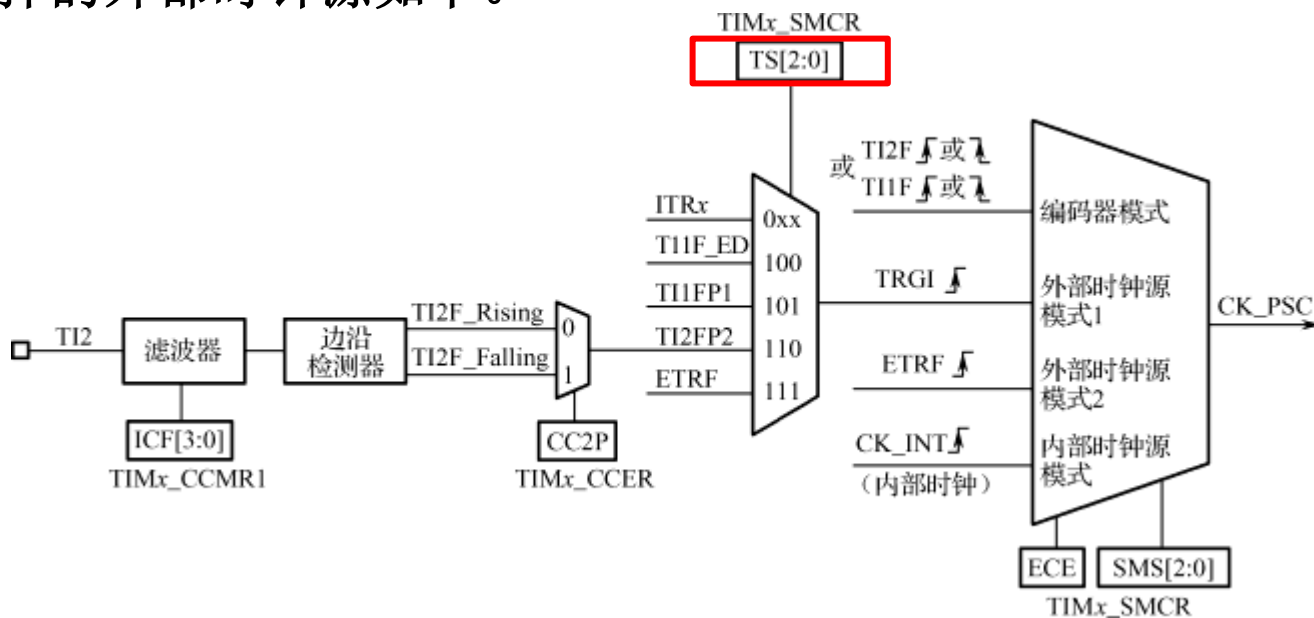


图9-4 外部时钟源模式1结构图

9.1 定时器系统简介

9.1.3 时钟源

3. 外部时钟源模式2

以**ETR**引脚上输入信号作为定时器的信号源。

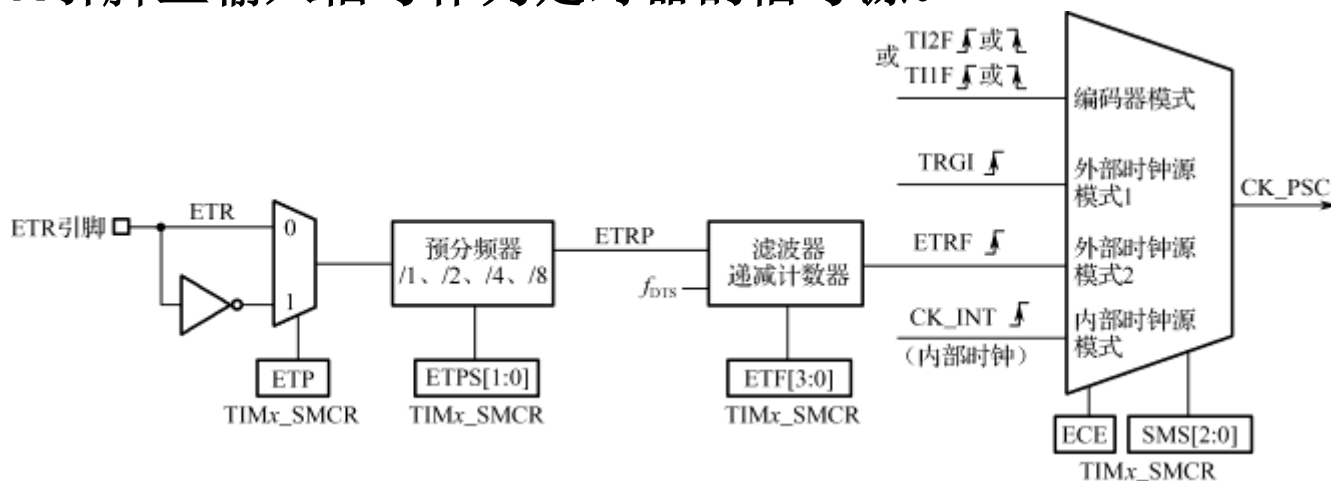


图9-5 外部时钟源模式2结构图

4. 编码器模式

将**TI1FP1**和**TI2FP2**信号的电平状态的变化作为定时器时钟源。这一模式主要用于测量光电正交编码器输出脉冲数，以及测量电机转速和方向。

9.2 基本定时功能

9.2 基本定时功能

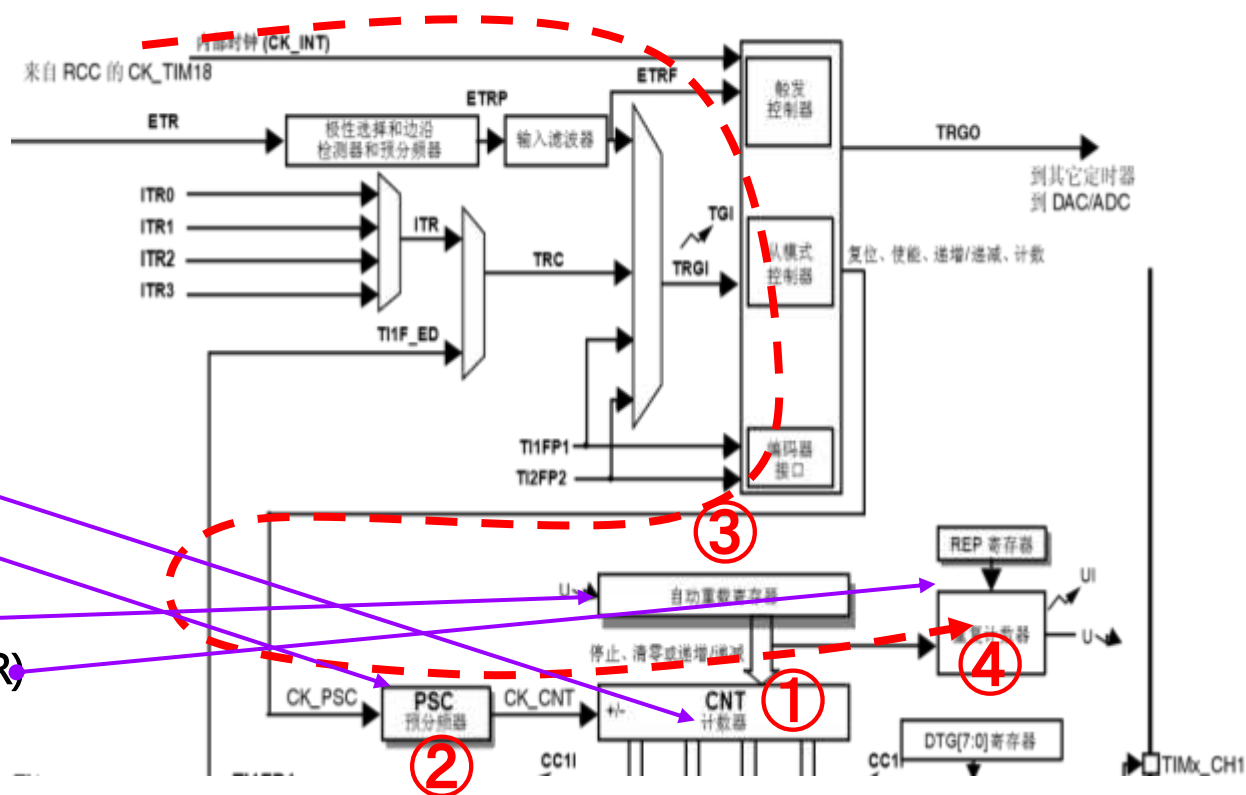
9.2.1 时基单元

定时器的主要模块是一个 16 位**计数器**及其相关的**自动重载寄存器**。计数器的时钟可通过**预分频器**进行分频。

时基单元包括：

- ① 计数器寄存器 (TIMx_CNT)
- ② 预分频器寄存器 (TIMx_PSC)
- ③ 自动重载寄存器 (TIMx_ARR)
- ④ 重复计数器寄存器 (TIMx_RCR)

以内部时钟作为计数基准



9.2 基本定时功能

9.2.1 定时器时基单元

计数器、自动重载寄存器和预分频器寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

①计数器寄存器 (TIMx_CNT): 计数器, 可递增计数、递减计数或交替进行递增和递减计数;

②预分频器寄存器 (TIMx_PSC): 可将计数时间基准在1-65536之间任意分频;

③自动重载寄存器 (TIMx_ARR): 存储计数器的溢出值;

④重复计数器寄存器 (TIMx_RCR): 高级定时器TIM1、8才有。控制重复计数的次数。

9.2 基本定时功能

9.2.2 计数模式

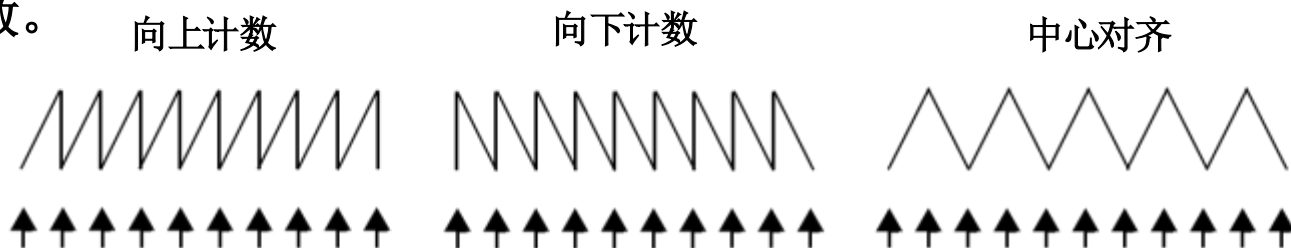
定时器以一定的**时钟基准**进行计数，计数对象是计数寄存器 **TIMx_CNT**。

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。

①**递增计数模式**：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**。

②**递减计数模式**：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**。

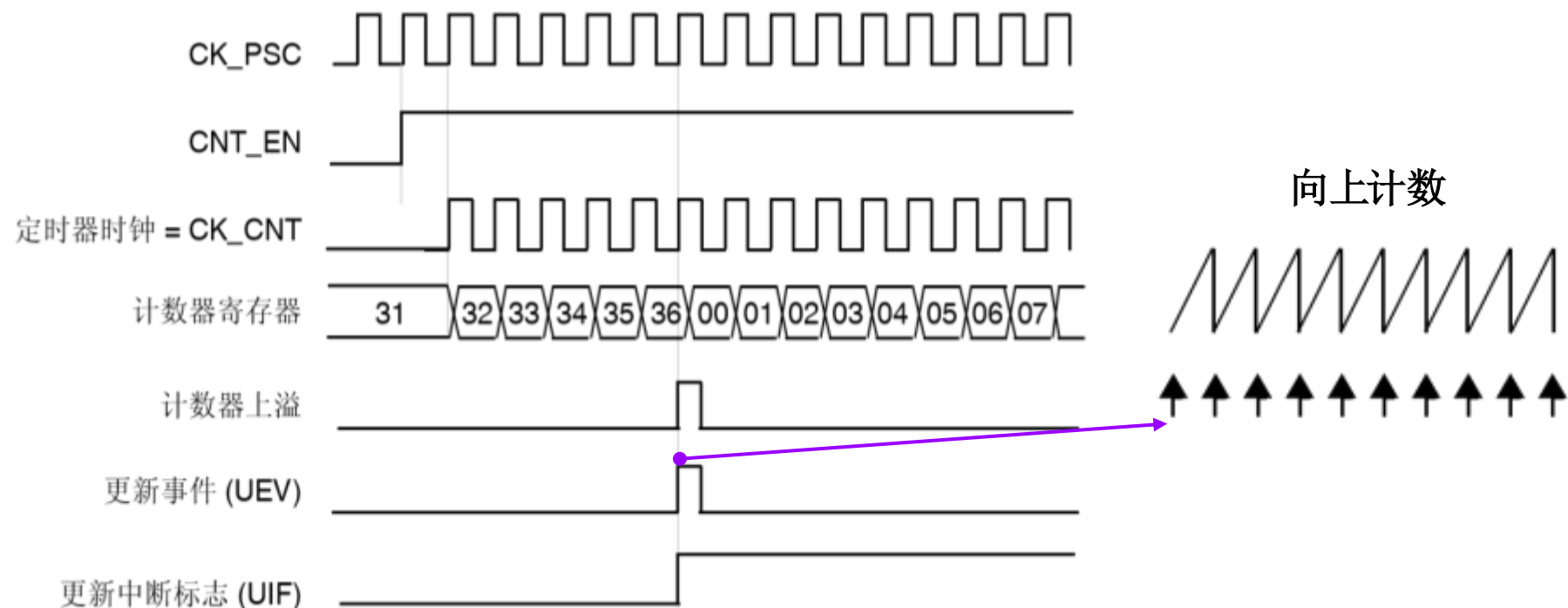
③**中央对齐模式（向上/向下计数）**：计数器从0开始计数到自动装入的值-1，产生一个计数器**溢出事件**，然后向下计数到1并且产生一个计数器**溢出事件**；然后再从0开始重新计数。



9.2 基本定时功能

9.2.2 计数模式

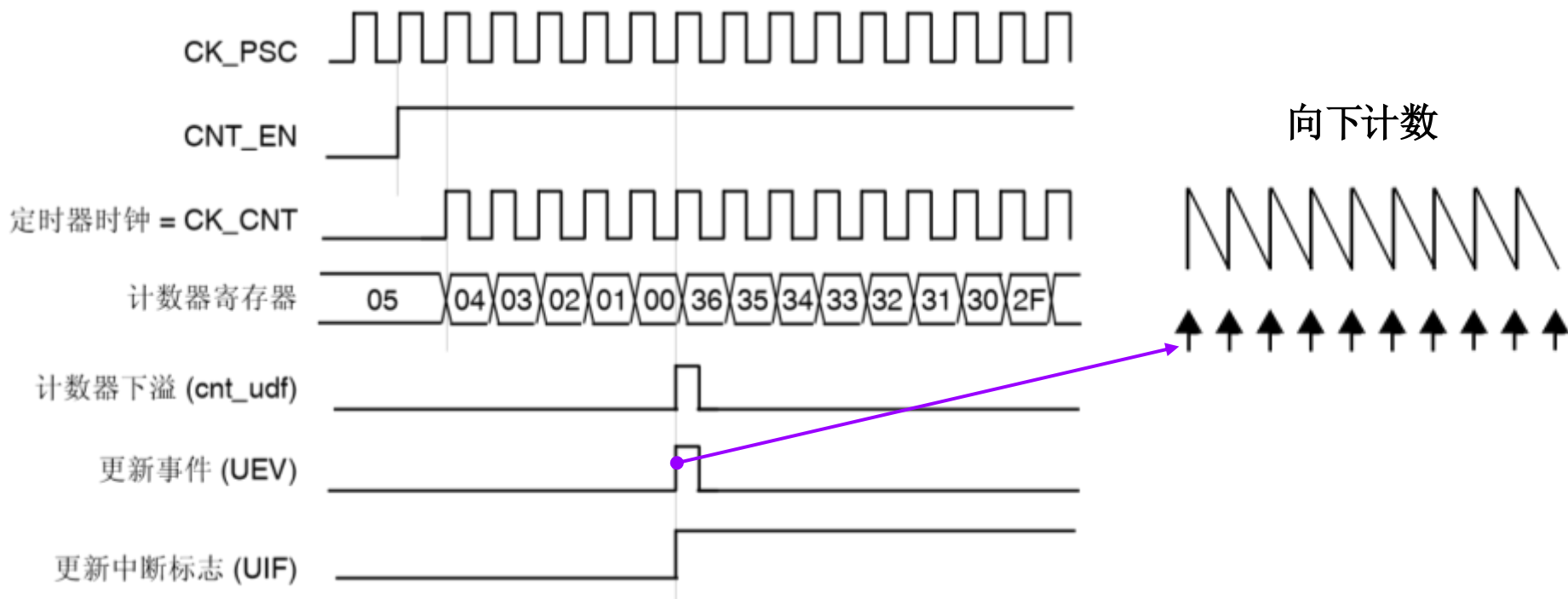
递增计数模式（时钟分频因子=1）



9.2 基本定时功能

9.2.2 计数模式

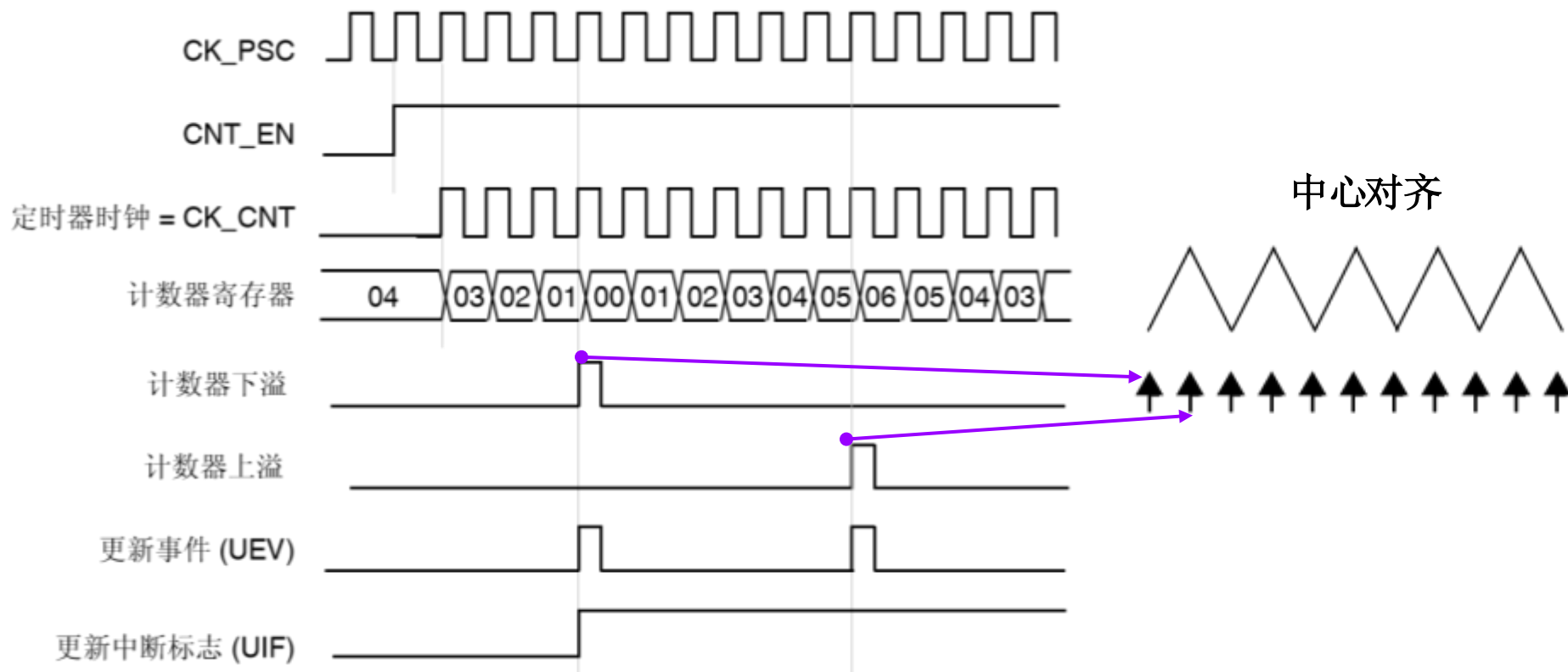
递减计数模式（时钟分频因子=1）



9.2 基本定时功能

9.2.2 计数模式

中央对齐计数模式（时钟分频因子=1 **ARR=6**）



9.3 捕抓/比较功能

9.3 捕抓/比较功能

STM32F429微控制器的高级定时器和通用定时器中有输入捕抓通道和输出比较通道。

1、输入捕抓通道：

- 1)、频率测量、PWM信号周期、占空比测量，以及霍尔传感器输出信号测量等。
- 2)、测量光电正交编码器输出信号，实现电机转速的测量。

2、输出比较通道：

实现PWM信号输出、6步PWM信号生成，用于电机控制。

9.3 捕获/比较功能

9.3.1 输入捕获/比较输出通道

1、输入捕获通道：

输入捕获通道由输入滤波器（防止去除输入信号电平切换产生的抖动，防止误判）、边沿检测器、边沿检测方式选择开关（2选1）、输入捕获信号选择多路开关（3选1）及预分频器组成。

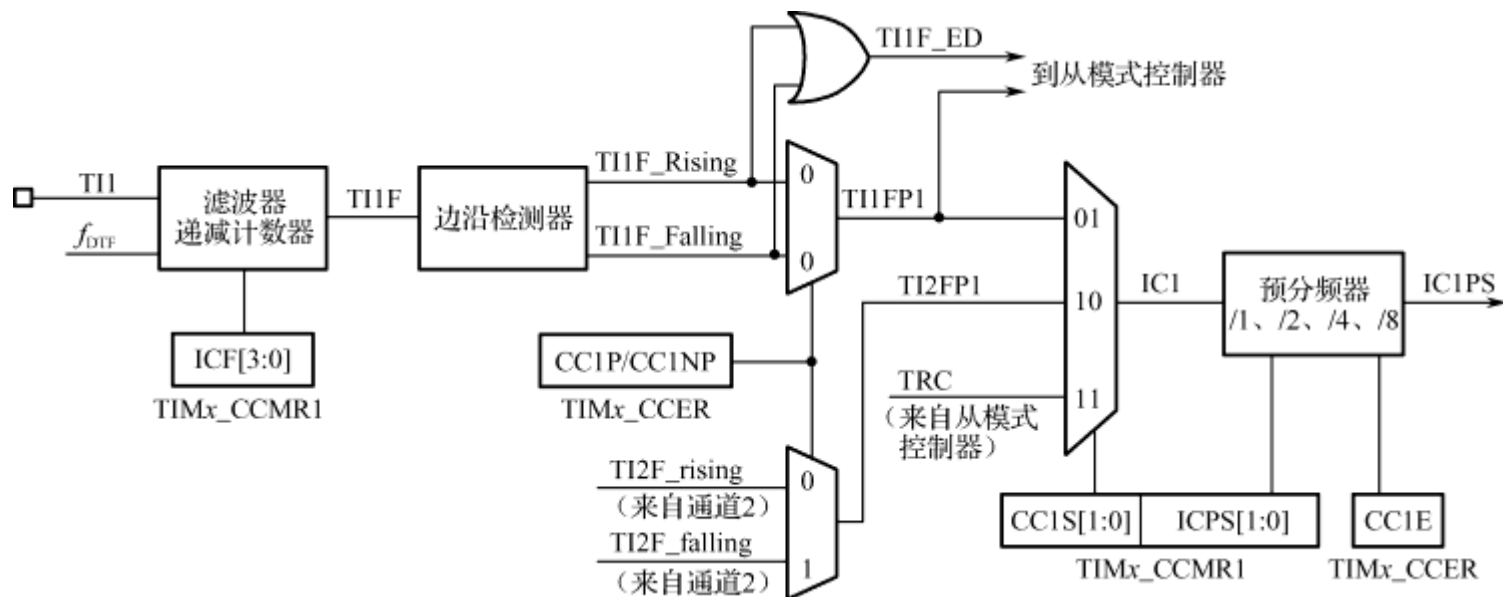


图9-12 输入捕获通道1结构示意图

9.3 捕获/比较功能

9.3.1 输入捕获/比较输出通道

2、输出比较通道

比较输出通道由输出模式控制器、死区发生器（防止电动机驱动上下桥臂控制的开关状态切换时出现同时导通）和输出使能控制电路组成。

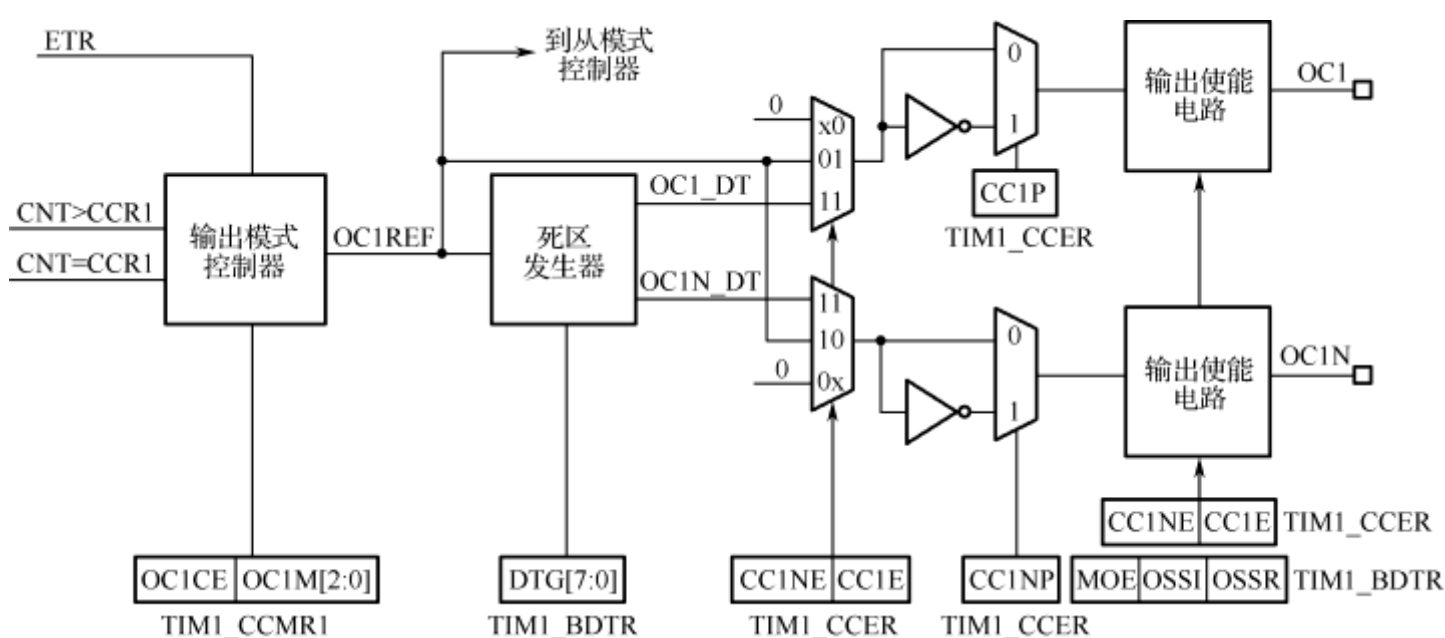
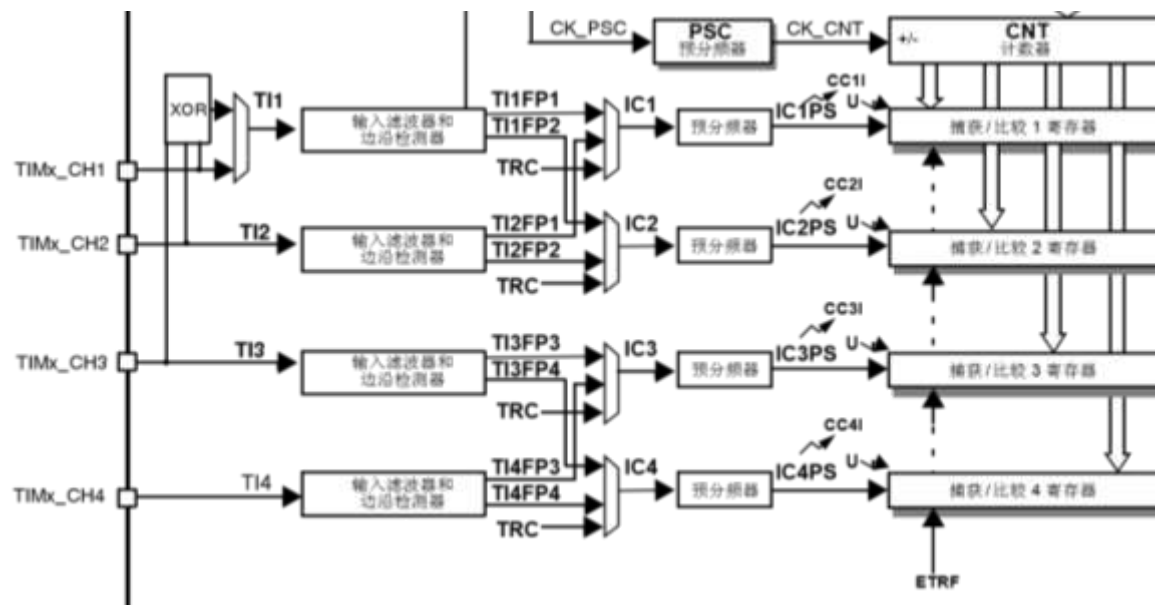


图9-13 比较输出通道结构示意图

9.3 捕获/比较功能

9.3.2 输入捕获模式

在输入捕获模式下，当检测到捕获输入通道信号的跳变后，**将使用捕获/比较寄存器 (TIMx_CCRx) 来锁存计数器的值**。发生捕获事件时，会将相应的 CCxIF 标志 (TIMx_SR 寄存器) 置 1，并可发送中断或 DMA 请求（如果已使能）。

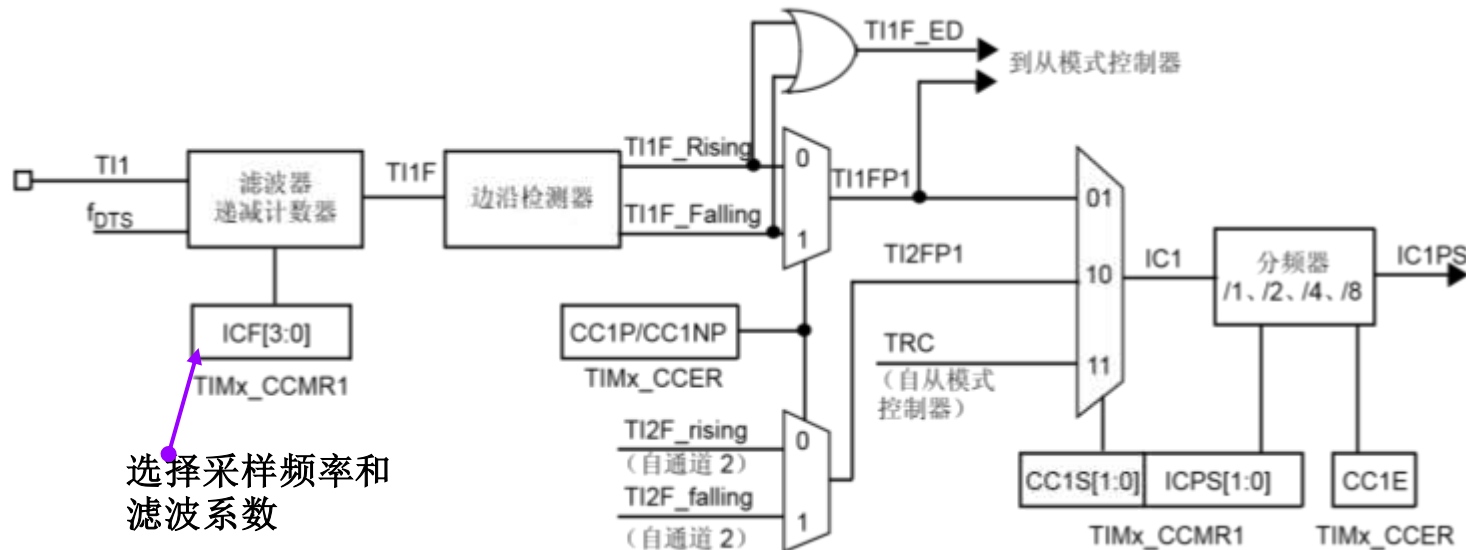


输入捕获引脚

12.4 定时器输入捕抓功能

9.3.2 输入捕抓模式

在输入捕获模式下，当相应的 IC_x 信号检测到跳变沿后，**将使用捕获/比较寄存器 (TIM_x_CCR_x) 来锁存计数器的值**。发生捕获事件时，会将相应的 CC_xIF 标志（TIM_x_SR 寄存器）置 1，并可发送中断或 DMA 请求（如果已使能）。



9.3 捕抓/比较功能

9.3.2 输入捕抓模式

在使用输入捕抓通道1，当检测到TI1引脚上输入的信号出现上升沿时，将计数器的值捕抓到TIMx_CCR1中。具体操作步骤如下：

1. 选择输入捕抓模式，IC1映射到TI1上
2. 设定输入信号边沿检测的滤波功能（防抖动）
3. 选择边沿触发方式
4. 对输入预分频器进行编程
5. 使能输入捕抓功能
6. 设置捕抓中断和DMA请求

当连续两次捕抓同一输出信号的连续两个边沿跳变，两次得到的计数寄存器值分别为C1和C2（假设C1和C2计数期间没有溢出事件），那么这一输入信号的周期：

$$\left((C2 - C1) / CK_CNT \right) / \text{输入捕抓通道预分频系数}。$$

9.3 捕获/比较功能

9.3.2 输入捕获模式

PWM波的周期测量:

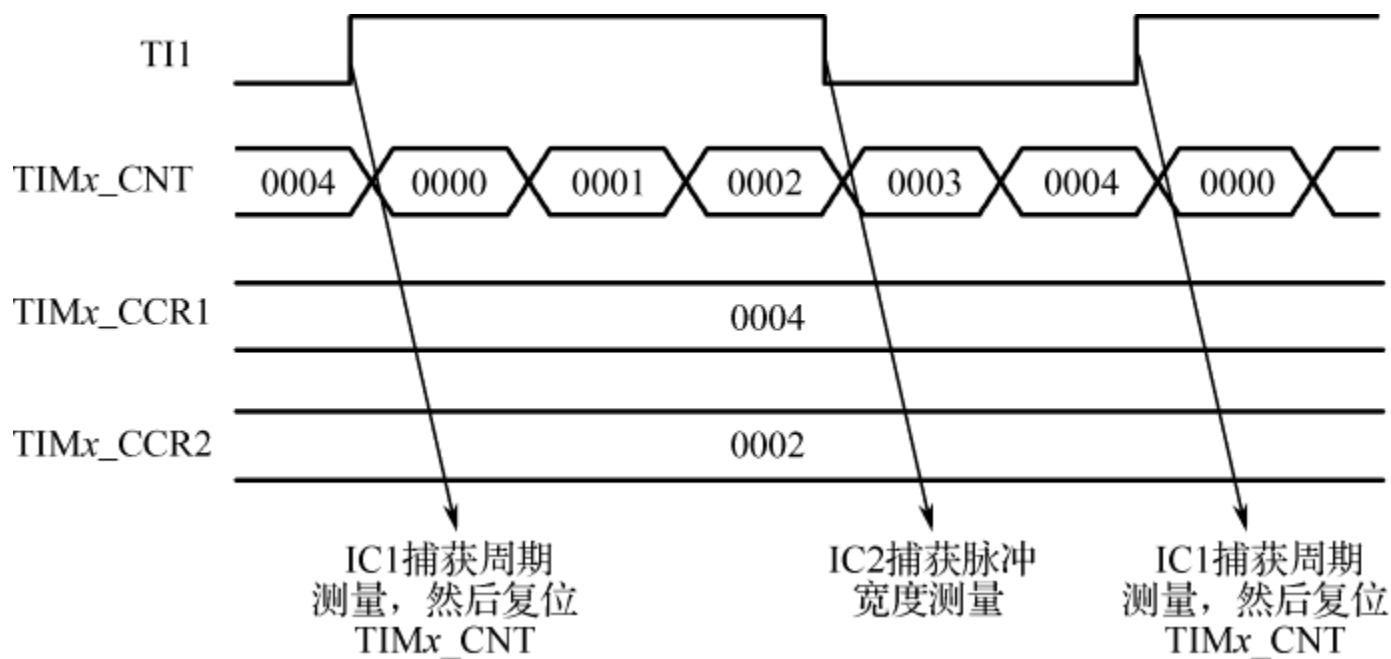


图9-14 PWM波的周期测量示意图

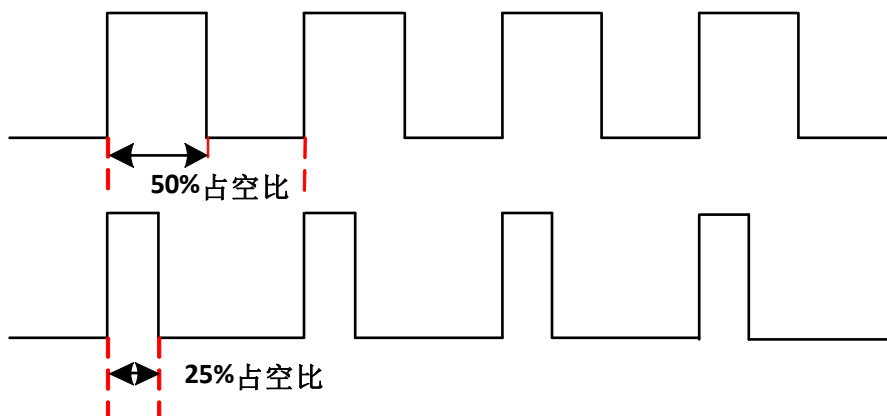
9.3 捕获/比较功能

9.3.3 PWM输出模式

PWM (Pulse Width Modulation)：脉冲宽度调制，简称脉宽调制。

PWM信号：周期内高电平占空比可调的信号。

占空比：一个周期内高电平持续时间与一个周期时间的比值。

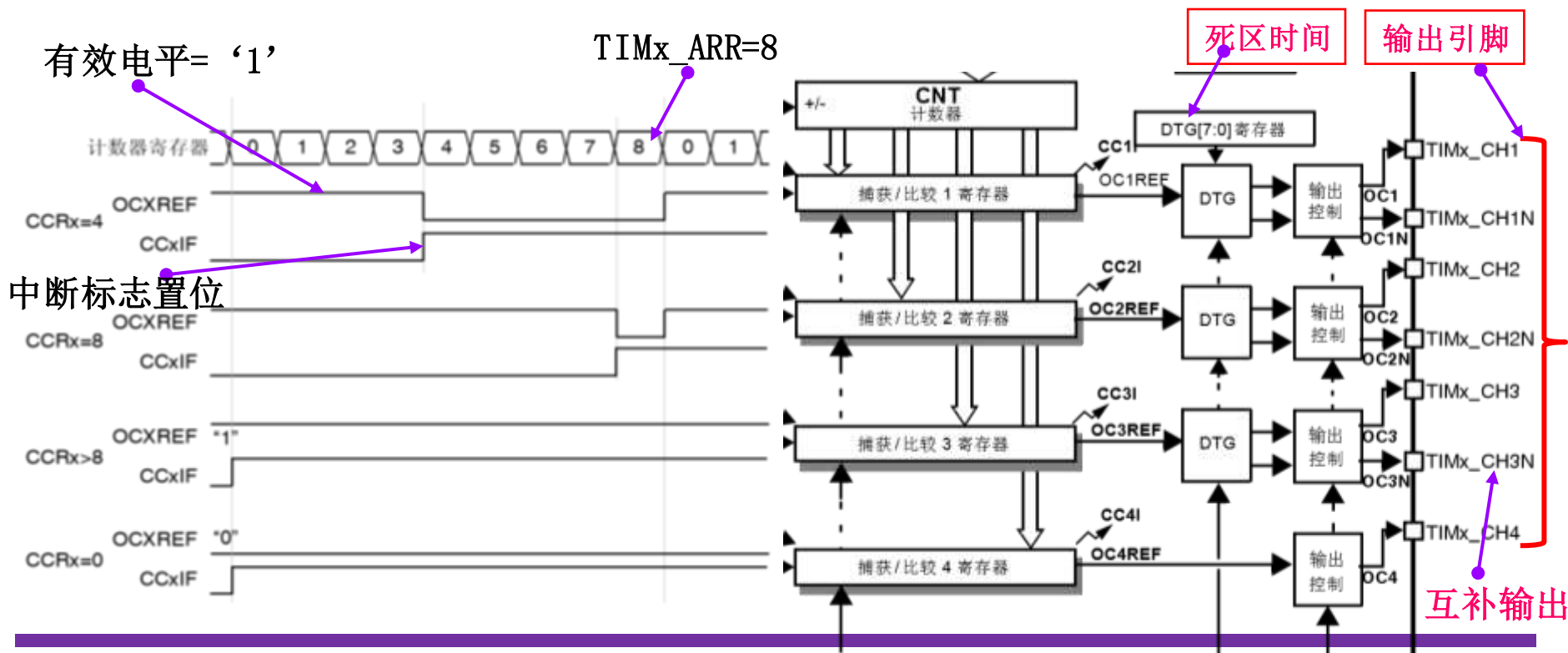


9.3 捕获/比较功能

9.3.3 PWM输出模式

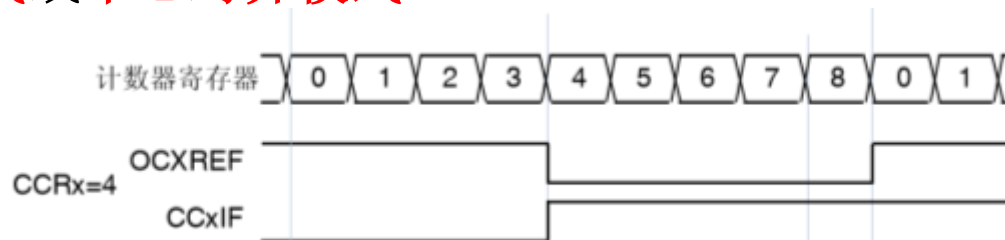
设定当使能比较输出功能（**TIMx_CCMR2x**）时，可以通过相关比较输出引脚输出PWM波。

自动重载寄存器 (**TIMx_ARR**) 设定PWM波的**周期**，捕获/比较寄存器 (**TIMx_CCRx**) 设定**占空比**。**计数器寄存器 (TIMx_CNT)**计数值 < **TIMx_CCRx** 的值时，输出**有效电平**。

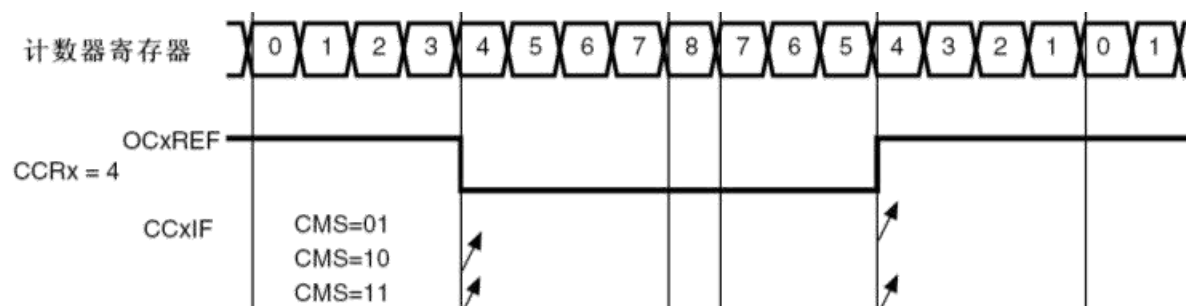


9.3 捕抓/比较功能

比较模式可以是**边沿比较模式**或**中心对齐模式**。



边沿比较模式



中心对齐模式

9.3 捕获/比较功能

9.3.3 PWM输出模式

例如：使用TIM1产生40KHz占空比=25%的PWM波，使用边沿比较模式，内部时钟源=180MHz

。

预分频器寄存器 (TIMx_PSC)=0

自动重载寄存器 (TIMx_ARR)=4499

重复计数器寄存器 (TIMx_RCR)=0

捕获/比较寄存器 (TIMx_CCRx)=1125

9.3 捕抓/比较功能

9.3.4 编码器接口模式

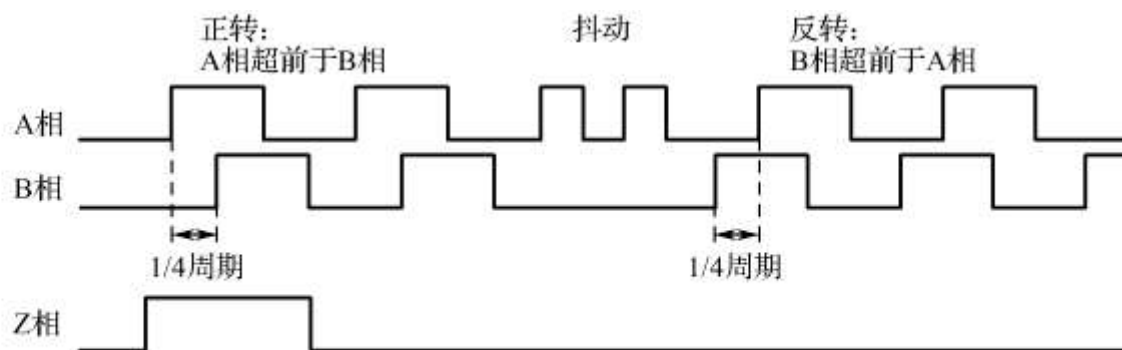
编码器接口主要用于连接正交编码器及**测量电机的转速和转向**。

光电编码器内部的LED发射的光，通过光栅到达光敏管，引起A相和B相电平变化。**如果正转**，则A相输出超前B相 90° ；**如果反转**，则A相滞后B相 90° 。每转一周，Z相经过LED一次，输出一个脉冲，可作为编码器的机械零位。

STM32F429微控制器的高级定时器**TIM1和TIM8**，通用定时器**TIM2~TIM5**集成了**编码器接口功能**。



(a) 光电编码器外观



(b) 正交光电编码器输出波形

图9-18 常用光电编码器

9.3 捕抓/比较功能

9.3.4 编码器接口模式

在编码器模式下，计数器会根据增量编码器的速度和方向自动进行修改，因此计数器内容始终表示编码器的位置。计数方向对应于定时器所连传感器的轴旋转方向。

表9-3 不同编码器模式下的计数方式

有效边沿	相反信号的电平（TI1FP1对应TI2， TI2FP2对应TI1）	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1处计数	高	递减	递增	不计数	不计数
	低	递增	递减	不计数	不计数
仅在TI2处计数	高	不计数	不计数	递增	递减
	低	不计数	不计数	递减	递增
在TI1和TI2处均计数	高	递减	递增	递增	递减
	低	递增	递减	递减	递增

9.3 捕获/比较功能

9.3.4 编码器接口模式

(1) 正转时，计数器递增计数。

TI1高电平时，TI2上升沿处，计数器递增计数。

TI1低电平时，TI1下降沿处，计数器递增计数。

TI2高电平时，TI2下降沿处，计数器递增计数。

TI2低电平时，TI1上升沿处，计数器递增计数。

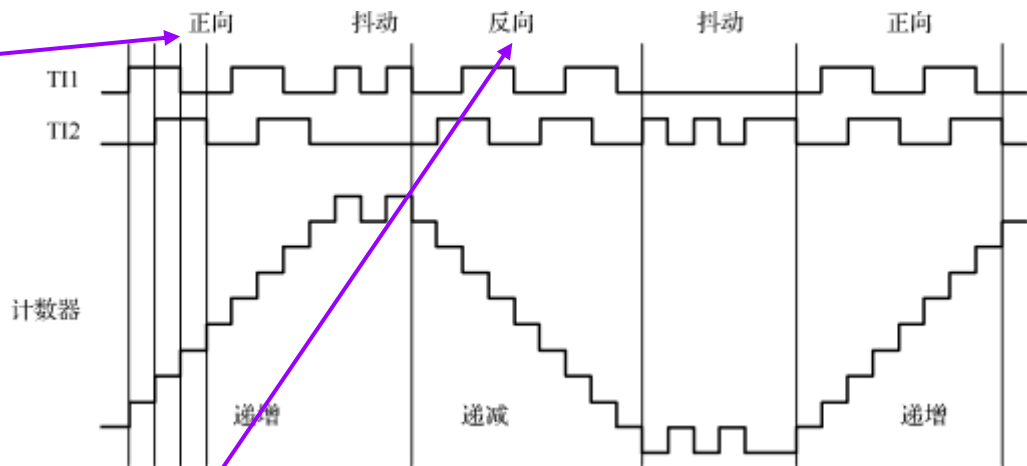


图9-19 编码器接口模式，在TI1和TI2处均计数示意图

(2) 反转时，计数器递减计数。

TI1高电平时，TI2下降沿处，计数器递减计数。

TI1低电平时，TI1上升沿处，计数器递减计数。

TI2高电平时，TI2上升沿处，计数器递减计数。

TI2低电平时，TI1下降沿处，计数器递减计数。

9.4 定时器典型应用步骤及常用库函数

9.4 定时器典型应用步骤及常用库函数

9.4.1 基本定时功能应用步骤

①使能定时器时钟。

```
RCC_APB1PeriphClockCmd();
```

②初始化定时器，配置ARR,PSC。

```
TIM_TimeBaseInit();
```

③开启定时器中断，选择中断请求事件并配置NVIC。

```
void TIM_ITConfig(TIM_TypeDef* TIMx, uint16_t TIM_IT,  
FunctionalState NewState);
```

```
NVIC_Init();
```

④使能定时器。

```
TIM_Cmd();
```

⑤编写中断服务函数。

```
TIMx_IRQHandler();
```

9.4 定时器典型应用步骤及常用库函数

9.4.2 输入捕获模式应用步骤

- ①使能定时器时钟和GPIO时钟
- ②初始化定时器输入捕获通道引脚
- ③初始化定时器测量时钟
- ④设置输入捕获通道
- ⑤选择定时器复位触发源
- ⑥开启定时器中断
- ⑦使能定时器
- ⑧编写中断服务函数

9.4 定时器典型应用步骤及常用库函数

9.4.3 PWM输出应用步骤

- ①使能定时器时钟
- ②初始化定时器比较输出通道引脚
- ③定义PWM波的周期
- ④设置比较输出通道
- ⑤使能定时器
- ⑥开启定时器中断

9.4 定时器典型应用步骤及常用库函数

9.4.4 编码器接口应用步骤

- ①使能定时器时钟
- ②初始化定时器编码器接口输入通道引脚
- ③定义编码器接口的计数值溢出值
- ④设置定时器编码器接口模式
- ⑤开启定时器中断
- ⑥使能定时器
- ⑦编写测量速度应用程序

9.4 定时器典型应用步骤及常用库函数

9.4.5 常用库函数

头文件: `stm32f4xx_tim.h`
库函数: `stm32f4xx_tim.c`

1、定时器定时周期初始化函数:

```
void TIM_TimeBaseInit(TIM_TypeDef* TIMx,  
    TIM_TimeBaseInitTypeDef* TIM_TimeBaseInitStruct);  
  
#define TIM2                ((TIM_TypeDef *) TIM2_BASE)  
#define TIM3                ((TIM_TypeDef *) TIM3_BASE)  
#define TIM4                ((TIM_TypeDef *) TIM4_BASE)  
  
typedef struct  
{  
    uint16_t TIM_Prescaler;    //预分频系数  
    uint16_t TIM_CounterMode;  //计数模式  
    uint16_t TIM_Period;       //计数长度  
    uint16_t TIM_ClockDivision; //与死区长度及捕抓采样频率频率相关  
    uint8_t TIM_RepetitionCounter; //重复计数次数  
} TIM_TimeBaseInitTypeDef;  
  
TIM_TimeBaseStructure.TIM_Period = 4999;  
TIM_TimeBaseStructure.TIM_Prescaler = 7199;  
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;  
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;  
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
```


9.4 定时器典型应用步骤及常用库函数

9.4.5 常用库函数

2、定时器使能函数：

```
void TIM_Cmd(TIM_TypeDef* TIMx, FunctionalState NewState)
```

使能或禁止定时器计数。

3、定时器中断使能函数：

```
void TIM_ITConfig(TIM_TypeDef* TIMx, uint16_t TIM_IT,  
FunctionalState NewState);
```

使能或禁止定时器中断。

4、状态标志位获取和清除

//获取需要的定时器状态标志位

```
FlagStatus TIM_GetFlagStatus(TIM_TypeDef* TIMx, uint16_t TIM_FLAG);
```

```
Void TIM_ClearFlag(TIM_TypeDef* TIMx, uint16_t TIM_FLAG);
```

//获取需要的定时器状态标志位，并且判断对应中断是否被允许

```
ITStatus TIM_GetITStatus(TIM_TypeDef* TIMx, uint16_t TIM_IT);
```

```
void TIM_ClearITPendingBit(TIM_TypeDef* TIMx, uint16_t TIM_IT);
```

```
945 #define TIM_FLAG_Update      ((uint16_t)0x0001)  
946 #define TIM_FLAG_CC1        ((uint16_t)0x0002)  
947 #define TIM_FLAG_CC2        ((uint16_t)0x0004)  
948 #define TIM_FLAG_CC3        ((uint16_t)0x0008)  
949 #define TIM_FLAG_CC4        ((uint16_t)0x0010)  
950 #define TIM_FLAG_COM        ((uint16_t)0x0020)  
951 #define TIM_FLAG_Trigger     ((uint16_t)0x0040)
```

```
553 #define TIM_IT_Update        ((uint16_t)0x0001)  
554 #define TIM_IT_CC1          ((uint16_t)0x0002)  
555 #define TIM_IT_CC2          ((uint16_t)0x0004)  
556 #define TIM_IT_CC3          ((uint16_t)0x0008)  
557 #define TIM_IT_CC4          ((uint16_t)0x0010)  
558 #define TIM_IT_COM          ((uint16_t)0x0020)  
559 #define TIM_IT_Trigger       ((uint16_t)0x0040)  
560 #define TIM_IT_Break         ((uint16_t)0x0080)
```

9.5 应用实例

9.5 应用实例

9.5.1 定时器控制实现灯闪烁

通过定时器TIM2，实现LED灯的闪烁，闪烁周期为2s。

使用中断方式实现。LED灯电路图如图9-20所示。

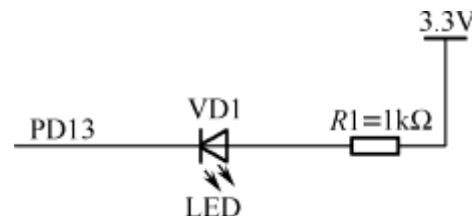


图9-20 LED灯电路图

1. 编程要点

(1) 计算时间

- a)、定时时间设定为1s，每1s产生一次中断，在中断服务程序中反转LED灯控制状态。
- b)、选择使用内部时钟源。TIM2挂在在APB1总线上。在本历程中HCLK=180MHz，APB1分频系数=4，APB1频率=45MHz。TIM2计数时钟源=90MHz。
- c)、 T_{out} （定时时间）=1s= $(ARR+1)(PSC+1)/90000000$ ，则可以取PSC=9000-1，ARR=10000-1。

参数能不能有其他选择？

(2) 使能TIM2的工作时钟。

(3) 配置TIM2时基参数。

主要涉及计数周期、预分频系数、计数模式、采样时钟预分频系数。

9.5 应用实例

9.5.1 定时器控制实现灯闪烁

2. 主程序

```
int main(void)
{
    /* 初始化LED*/
    LED_Config();
    /* 初始化通用定时器定时，1s产生一次中断 */
    TIM_Config_Basic ();

    while(1)
    {
        /* 循环，等待定时器中断 */
    }
}
```

4、定时器配置

```
void TIMx_Configuration(void)
{
    /*NVIC中断初始化*/
    TIMx_NVIC_Configuration();
    /*定时器初始化*/
    TIM_Mode_Config();
}
```

12.6 定时器应用

9.5.1 定时器控制实现灯闪烁

3. 定时器初始化函数

```
void TIM_Config_Basic (void)
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    /*-----第1步-----*/
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE); //使能TIM2时钟
    /*-----第2步-----*/
    TIM_TimeBaseStructure.TIM_Period = 10000-1; //设定计数周期
    TIM_TimeBaseStructure.TIM_Prescaler = 9000-1; //设定预分频系数
    TIM_TimeBaseStructure.TIM_CounterMode=TIM_CounterMode_Up; //计数模式
    TIM_TimeBaseStructure.TIM_ClockDivision=TIM_CKD_DIV1; //采样时钟分频系数
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure); //初始化TIM2
}
```

12.6 定时器应用

9.5.1 定时器控制实现灯闪烁

3. 定时器初始化函数

续上

```
/*-----第3-1步-----*/
TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); //使能TIM2更新中断
/*-----第3-2步-----*/
//配置TIM2中断通道
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0); //设置中断组为0
NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn; //设置中断来源

NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //设置抢占优先级

NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3; //设置响应优先级

NVIC_Init(&NVIC_InitStructure);
/*-----第4步-----*/
TIM_Cmd(TIM2, ENABLE); //启动定时器计数
}
```

12.6 定时器应用

4、定时器中断服务函数

```
void TIM2_IRQHandler (void)
```

```
{
```

```
//检测是否是定时器溢出中断
```

```
if ( TIM_GetITStatus( TIM2, TIM_IT_Update) != RESET )
```

```
{
```

```
//请定时器中断标志位
```

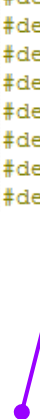
```
TIM_ClearITPendingBit(TIM2 , TIM_IT_Update);
```

```
GPIO_ToggleBits (GPIOD,GPIO_Pin_13); //反转LED灯状态
```

```
}
```

```
}
```

```
553 #define TIM_IT_Update      ((uint16_t)0x0001)
554 #define TIM_IT_CC1         ((uint16_t)0x0002)
555 #define TIM_IT_CC2         ((uint16_t)0x0004)
556 #define TIM_IT_CC3         ((uint16_t)0x0008)
557 #define TIM_IT_CC4         ((uint16_t)0x0010)
558 #define TIM_IT_COM         ((uint16_t)0x0020)
559 #define TIM_IT_Trigger     ((uint16_t)0x0040)
560 #define TIM_IT_Break       ((uint16_t)0x0080)
```



9.5 应用实例

9.5.2 直流电机调速控制

通过TIM3输出通道3（PB0引脚）输出控制电机的PWM波。使用图5-11中的两个按键控制转速，图5-11中按键S1提升转速，按键S2降低转速，每次按键动作调整1%的占空比。

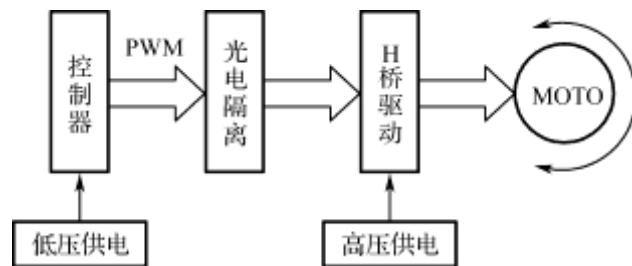


图9-21 直流电机控制框图

1. 编程要点

- (1) 使能TIM3和GPIO的时钟。
- (2) 配置定时器输入捕抓通道引脚。
- (3) 配置TIM3时基参数。

主要涉及计数周期、预分频系数、计数模式、采样时钟预分频系数。

初始化时基单元定义了PWM波的周期。

- (4) 比较输出通道配置。

将TIM3的比较输出通道3配置为PWM模式1，占空比为30%，选择高电平为输出有效电平，并使能输出。

- (5) 使能比较输出通道3重载功能。

使能了捕抓/比较值的缓冲功能。

- (6) 启动定时器。

9.5 应用实例

9.5.2 直流电机调速控制

2. 主程序

```
int main(void)
{
    uint16_t Duty_Ratio=300; //占空比, 初始化位30%
    delay_init(); //初始化SysTick, 用于延时
    Key_Config(); //初始化按键GPIO引脚
    TIM_Config(); //初始化定时器
    while(1)
    {
        if(Key_Scan(GPIOA,GPIO_Pin_0,1) == KEY_ON) //增加PWM占空比
        {
            if(Duty_Ratio<900) //上边界控制, 最大90%占空比
            {
                Duty_Ratio+=10;
                TIM_SetCompare3(TIM3,Duty_Ratio);
            }
        }
        if(Key_Scan(GPIOA,GPIO_Pin_1,0) == KEY_ON) //降低PWM占空比
        {
            if(Duty_Ratio>50) //下边界控制, 最小5%占空比
            {
                Duty_Ratio-=10;
                TIM_SetCompare3(TIM3, Duty_Ratio);
            }
        }
    }
}
```

9.5 应用实例

9.5.2 直流电机调速控制

3. 定时器输出PWM波初始化函数

```
void TIM_Config_PWM (void)                //定时器PWM输出模式初始化
{
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    TIM_OCInitTypeDef  TIM_OCInitStructure;
    /*-----第1步-----*/
    RCC_AHB1PeriphClockCmd (RCC_AHB1Periph_GPIOB, ENABLE);      //使能GPIOB时钟
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);         //使能TIM3时钟

    /*-----第2步-----*/
    //TIM3比较输出通道3复用引脚配置
    GPIO_PinAFConfig(GPIOB,GPIO_PinSource0,GPIO_AF_TIM3); //TIM3比较输出通道3引脚复用

    /*TIM3比较输出通道3引脚配置*/
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_OType=GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd=GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_25MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
```

9.5 应用实例

9.5.2 直流电机调速控制

3. 定时器输出PWM波初始化函数

续上

```
/*-----第3步-----*/
//定义PWM波的周期
TIM_TimeBaseStructure.TIM_Period = 1000-1; //计数周期
TIM_TimeBaseStructure.TIM_Prescaler = 900-1; //计数器预分频系数
TIM_TimeBaseStructure.TIM_ClockDivision=TIM_CKD_DIV1; //采样时钟分频系数
TIM_TimeBaseStructure.TIM_CounterMode=TIM_CounterMode_Up; //递增计数模式
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure); //初始化TIM3时基单元
/*-----第4步-----*/
//设置比较输出通道, PWM模式配置
//配置为PWM模式1, 当定时器计数值小于CCR1_Val时, 输出有效电平
TIM_OCInitStructure.TIM_OCMode=TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState=TIM_OutputState_Enable; //使能通道信号输出
TIM_OCInitStructure.TIM_Pulse=300; //初始化占空比 30%
TIM_OCInitStructure.TIM_OCPolarity=TIM_OCPolarity_High; //输出有效电平为高电平
TIM_OC3Init(TIM3, &TIM_OCInitStructure); //使能TIM3的比较输出通道3
TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Enable); //使能比较输出通道3重载功能
/*-----第5步-----*/
TIM_Cmd(TIM3, ENABLE); //启动定时器
}
```

9.5 应用实例

9.5.2 直流电机调速控制

4. 更改占空比

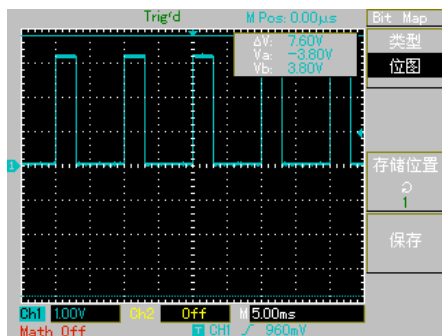
更改占空比是需要改变TIM3_CCR3值即可。

例如，依本例设置，将占空比改为40%， $TIM3_CCR3=40\% \times \text{计数周期}$
 $=40\% \times 1000=400$ 。两种方式：

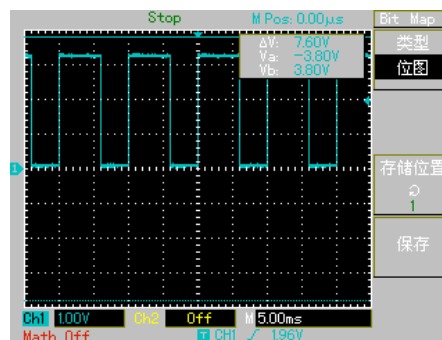
`TIM_SetCompare3(TIM3,400)`

或

`TIM3->CCR3=400;`



(a) 占空比为30% PWM波形



(b) 占空比为60% PWM波形

图9-22 周期为10ms，占空比分别为30%和60%的PWM波形