### 定时器实现PWM控制

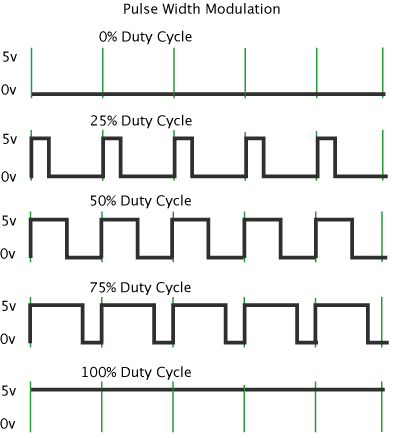
1. 认识PWM

物联网的硬件设备，无外乎就是将采集到的外界数据上传服务器或者根据服务器下发的指令控制设备，因此，采集、控制和联网是物联网硬件设备的三大特性。

对于控制有两种，第一种就是二进制信号量的控制，即控制设备的开或者关，比如通过GPIO口输出高电平或低电平来控制LED的点亮与熄灭。另外一种就是线性控制，比如控制灯光的亮暗，电机的转速。在线性控制中采用最多的方式就是使用 PWM 波进行控制。

PWM（Pulse Width Modulation，脉冲宽度调制）是一种周期和占空比可变的方波。

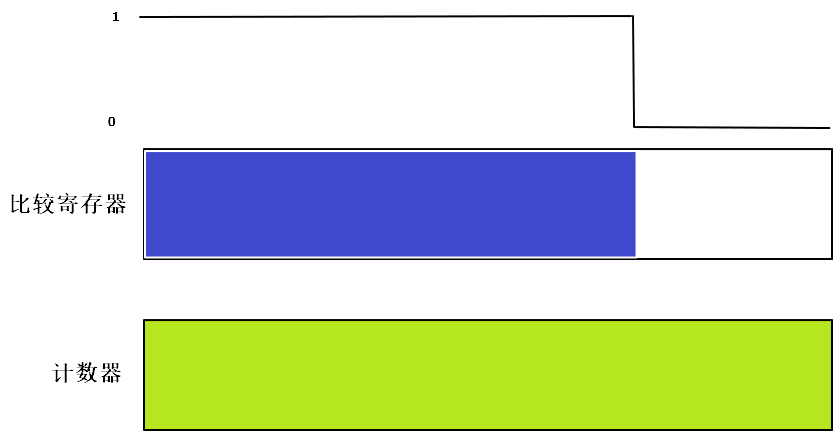
占空比就是同一周期的高电平持续时间和周期的比值，比如周期 100ms，高电平持续时间为 75ms，低电平持续时间为 25ms，则占空比为 75%。如下图所示：



占空比越高对应的等效电压就越高，反之占空比越低，对应的等效电压越低。因此通过调节占空比的大小可以控制灯光的亮度和电机的转速等。

1. 定时器生成PWM的过程

通常使用定时器的输出比较功能来产生 PWM。



这里最核心的就是两个寄存器，一个是计数器TIMx\_CNT，一个是比较寄存器TIMx\_CCR。

若配置脉冲计数器TIMx\_CNT为向上计数，而重载寄存器TIMx\_ARR被配置为N，即TIMx\_CNT的当前计数值数值X在TIMxCLK时钟源的驱动下不断累加，当TIMx\_CNT的数值X大于N时，会重置TIMx\_CNT数值为0重新计数。

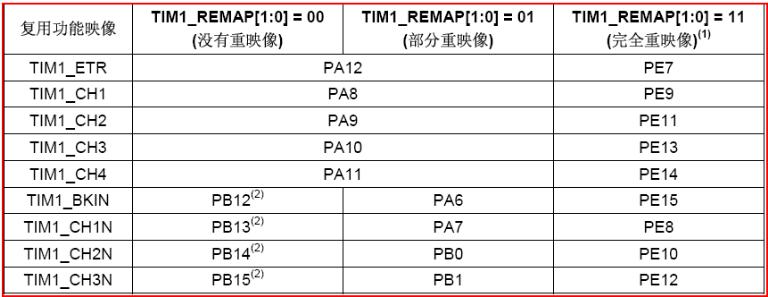
而在TIMxCNT计数的同时，TIMxCNT的计数值X会与比较寄存器TIMx\_CCR预先存储了的数值A进行比较，当脉冲计数器TIMx\_CNT的数值X小于比较寄存器TIMx\_CCR的值A时，输出高电平（或低电平），相反地，当脉冲计数器的数值X大于或等于比较寄存器的值A时，输出低电平（或高电平）。

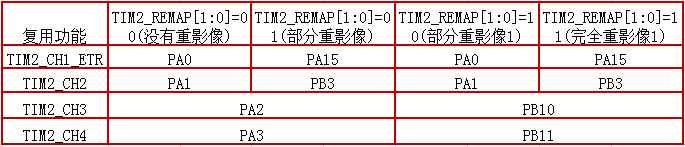
如此循环，得到的输出脉冲周期就为重载寄存器TIMx\_ARR存储的数值(N+1)乘以触发脉冲的时钟周期，其脉冲宽度则为比较寄存器TIMx\_CCR的值A乘以触发脉冲的时钟周期，输出PWM的占空比为 A/(N+1) 。

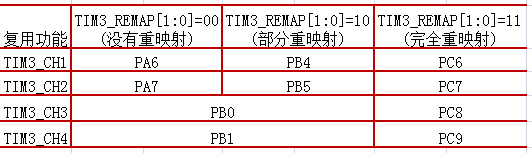
三、定时器输出比较生成 PWM 的软件配置过程：

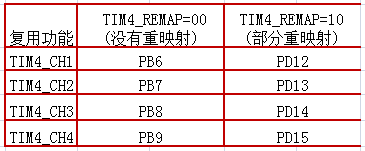
1、配置GPIO口：

配置IO口的时候无非就是开启时钟，然后选择引脚、模式、速率，最后就是用结构体初始化。不过在STM32上，不是每一个IO引脚都可以直接使用于PWM输出，因为在硬件上已经规定了用某些引脚来连接PWM的输出口。下面是定时器的引脚重映像，其实就是引脚的复用功能选择：









根据以上重映像表，我们使用定时器3的通道2作为PWM的输出引脚，所以需要对PA7引脚进行配置，对IO口操作代码：

GPIO\_InitTypeDef GPIO\_InitStructure;//定义结构体

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_AFIO, ENABLE);//使能GPIO外设和AFIO复用功能模块时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_7; //TIM\_CH2

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽功能

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化引脚

1. 初始化定时器：

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;//定义初始化结构体

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3, ENABLE); //使能定时器3时钟

//初始化TIM3

TIM\_TimeBaseStructure.TIM\_Period = arr -1 ; //自动重装载寄存器的值

TIM\_TimeBaseStructure.TIM\_Prescaler =psc - 1; //TIMX预分频的值

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1; //时钟不分割

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; //向上计数

TIM\_TimeBaseInit(TIM3, &TIM\_TimeBaseStructure); //根据以上功能对定时器进行初始化

TIM\_Cmd(TIM3, ENABLE); //使能定时器TIM3

1. 设置TIM3\_CH2的PWM模式，使能TIM3的CH2输出：

TIM\_OCInitTypeDef TIM\_OCInitStructure;//定义结构体

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;//选择TIM脉冲宽度调制模式1

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;//比较输出使能

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_Low;//起始波形为低电平

TIM\_OCInitStructure.TIM\_Pulse = arr/2 ; //比较寄存器值,则占空比为50%

TIM\_OC2Init(TIM3, &TIM\_OCInitStructure);//根据结构体信息进行初始化

TIM\_OC2PreloadConfig(TIM3, TIM\_OCPreload\_Enable); //使能定时器TIM2在CCR2上的预装载值

编写一个函数：void TIM3\_PWM\_Init(u16 arr,u16 psc)将以上所有的代码都加进这个函数中.

void TIM3\_PWM\_Init(u16 arr,u16 psc)

{

}

主函数

int main(void)

{

TIM3\_PWM\_Init(10000,7200);//频率为：72000 000/10000/7200=1Hz

TIM\_SetCompare2(TIM3,5000);//得到占空比为5000/10000=50%的pwm波形

while(1);

}