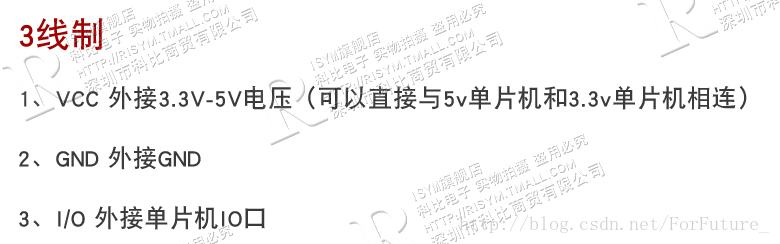
### 定时器实现无源蜂鸣器发声

1. 无源蜂鸣器介绍



**驱动这种无源蜂鸣器发声,需要一定频率的脉冲信号.**

我们可以用STM32的定时器**溢出中断**来产生**一定频率的脉冲信号.**

1. **电路连接与实验效果**

**STM32F103的PB0,PA7,PA6三个引脚分别产生**523Hz、588Hz、660Hz的脉冲信号，蜂鸣器模块的IO引脚分别连接到**PB0,PA7,PA6，**蜂鸣器将会发出不同音调的声音。

1. STM32 定时器简述

我们使用的 STM32F103C8T6 含有 4 个 16 位定时器，分别是一个高级定时器 TIM1 和 3 个通用定时器 TIM2-4。

STM32 的定时器非常强大，功能也异常丰富，但是我们平时用到的无外乎三种：

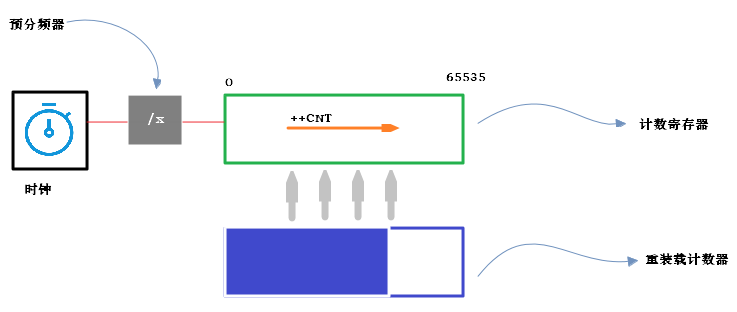
1. 一个是定时器**溢出中断**用于执行定时任务，比如每隔固定时间采集一次传感器数据 ；

2. 第二个是输入捕获，捕获输入信号的周期，对输入信号进行计数；

3. 第三个就是通过输出比较产生PWM ，PWM 应用非常广泛，比如调节灯光亮度，调节电机转速 ，调节声音 等等。

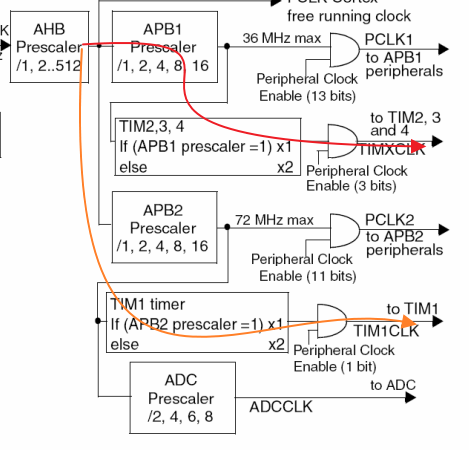
1. 通用定时器结构

通用定时器是一个通过可编程预分频器驱动的 16 位自动装载计数器构成。



1. 时钟源

这是任何一个定时计数器的基础部件，要计数就要有时钟。我们通常使用“内部时钟”(CK\_INT)



TIM2-4使用APB1时钟

TIM1使用APB2时钟

1. 计数器（计数寄存器）  
    是16位计数器，计数值从0到65535。

计数方向可以“向上计数”，“向下计数”，“双向计数”。

向上计数：计数值从0增加到设置值，产生一个向上溢出中断。

向下计数：计数值从设置值减小到0，产生一个向下溢出中断。

双向计数：即从 0 增加到设置值后产生一个向上溢出中断，然后自动从设置值向下计数到 0,再产生一个向下溢出中断，以此往复。

3. 重装载寄存器

我们设置计数器的计数值是保存在重装载寄存器中。每当计数器溢出后，重装载寄存器会自动将其内部的数据复制到计数器中。

1. 通用定时器溢出中断实现定时任务配置步骤：

1.配置定时器初始化结构，主要内容包括：计数方向，分频值，重装载值；

/\*\*

\* 功能：初始化定时器

\* 参数：TIMx：指定待设置的定时器，TIM1-TIM4

\*/

void initTIMx(TIM\_TypeDef\* TIMx , u16 psc)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

switch( (u32)TIMx )

{

case (u32)TIM1: RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM1, ENABLE);break; //开启定时器1时钟

case (u32)TIM2: RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);break; //开启定时器2时钟

case (u32)TIM3: RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3, ENABLE);break; //开启定时器3时钟

case (u32)TIM4: RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE);break; //开启定时器4时钟

/\*其他密度的单片机对case进行删减即可兼容，本程序针对中密度单片机\*/

default : break;

}

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

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;

TIM\_TimeBaseStructure.TIM\_Period =10-1;

//设置计数值

TIM\_TimeBaseStructure.TIM\_Prescaler = psc - 1;

//设置预分频值7200，则每计数1次，时间为0.1ms

TIM\_TimeBaseInit(TIMx, &TIM\_TimeBaseStructure); //生效更改

TIM\_Cmd(TIMx, ENABLE); //开启计数器

TIM\_ITConfig(TIMx,TIM\_IT\_Update,ENABLE);

//使能定时器更新中断

}

2. 配置定时器中断函数

void TIM2\_IRQHandler(void)

{

if(TIM\_GetITStatus(TIM3, TIM\_IT\_Update) == SET) //由于定时器中断源很多，因此要判断是哪个中断源触发的中断

{

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update); //软件清除中断挂起位 R\_LED\_FZ;

}

}

3. 配置 NVIC，开启定时器中断，并为其分配抢占优先级和子优先级；

/\*\*

\*

\*功能: 设置中断抢占优先级

\*

\*/

void setNVIC(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设置中断优先级分组

NVIC\_InitTypeDef NVIC\_InitStructure; //定义NVIC初始化结构体

NVIC\_InitStructure.NVIC\_IRQChannel = TIM2\_IRQn; //设置中断号

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0; //设置抢占优先级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1; //设置子优先级

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //开启中断

NVIC\_Init(&NVIC\_InitStructure); //设置生效

NVIC\_InitStructure.NVIC\_IRQChannel = TIM3\_IRQn;

NVIC\_Init(&NVIC\_InitStructure);

NVIC\_InitStructure.NVIC\_IRQChannel = TIM4\_IRQn;

NVIC\_Init(&NVIC\_InitStructure);

}

/\*\*

\* 功能：设置定时器周期，即设置重装载寄存器的值

\* 参数：

\* TIMx：指定待设置的定时器，TIM1-TIM4

\* period：设置中断周期，即设置重装载寄存器的值 0-65535

\* 返回值：None

\*/

void setPeriod(TIM\_TypeDef\* TIMx,u16 period)

{

TIM\_SetAutoreload(TIMx, period-1); //设置重装载值

}

//主函数,

int main(void)

{

initLED();

initTIMx(TIM2, 72);//预分频7200(0.1ms),计数10

setPeriod(TIM2,1912); //设置计数值为5000 --523Hz

initTIMx(TIM3, 72);//预分频7200(0.1ms),计数10

setPeriod(TIM3,1700); //设置计数值为5000 --588Hz

initTIMx(TIM4, 72);//预分频7200(0.1ms),计数10

setPeriod(TIM4,1515); //设置计数值为5000 --660Hz

//psc\*f = 72 000 000Hz

//分频后的时钟频率f = 72 000 000Hz / psc = 72 000 000Hz /72 = 1000 000Hz

setNVIC();

// 定时器的频率f' = f /period = 1000 000Hz /1000 = 523Hz

// 定时器的周期t' = 1 / f' =1/2Hz =0.5秒

}