**基本定时器**

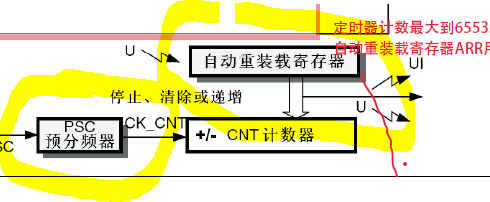
**基本定是器除了定时再什么功能都没有只有两个TIM6/7**

1. **概念**
2. **定时器分类**

定时器分三种：基本定时器，通用定时器和高级定时器，区别在于功能的多少。**基本<通用<高级.**



1. **影子寄存器**



看此处框图下面有一个灰色的阴影，这就代表有影子寄存器

影子寄存器是起到一个缓冲作用，执行流程如下：

用户值->寄存器->影子寄存器->起作用  
那么这样用户写进去的值就会在**这次计数结束后才会起作用**  
如果没影子寄存器就直接起作用

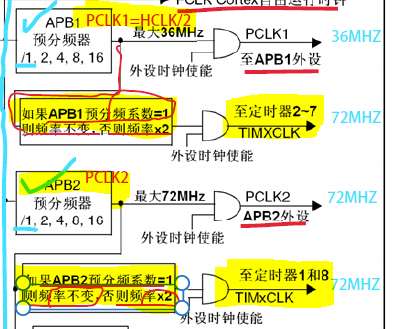
影子寄存器就是我们所说的**“数据缓存”**

1. **PCLK1和PCLK2时钟问题。**

此处的“如果预分配系数=1则频率不变，否则频率x2”指的**是APB1和APB2的分频**而不是PSC的分频，所以可得

APB1二分频=36M，所以TIM2~7=36M\*2=72M

APB1未分频=72M，所以TIM1/8=72M



1. T=PCLK/(PSC+1)\* (ARR+1)算出来的**值是秒**

F= (PSC+1)\* (ARR+1)/ PCLK算出来的是HZ

1. **编程问题**
2. **定时500ms呢？定时器计数周期根本没这么大，用一个中断(中断问题)**

定义一个全局变量time，定时器时基为1ms。

每过1ms定时器产生一次中断，中断函数里全局变量time++。

主函数判断time是否等于500，等于则计时500ms

定时器定义一个时基L，每计时L产生一次中断。定义两个全局变量value和时间time，每产生一次中断。读取一次传感器值value。

再到主函数里处理value，连续中断20次，20个value相加求平均。再到主函数根据value的变化判断姿态。

1. **编程中断 必须进行中断初始化NVIC\_INT()**和分组NVIC\_PriorityGroupConfig()
2. **各种中断**的区别(等待高级定时器的学习)
   1. TIM\_IT\_Update 更新中断，当计时器计时到设定的ARR值时，就会产生事件更新，从而产生中断。(就是计时周期到了就中断了)

更行中断 @arg TIM\_FLAG\_Update: TIM update Flag

* 1. **捕获/比较中断**
  2. @arg TIM\_FLAG\_CC1: TIM Capture Compare 1 Flag

\* @arg TIM\_FLAG\_CC2: TIM Capture Compare 2 Flag

\* @arg TIM\_FLAG\_CC3: TIM Capture Compare 3 Flag

\* @arg TIM\_FLAG\_CC4: TIM Capture Compare 4 Flag

③ 通讯中断

TIM\_FLAG\_COM: TIM Commutation Flag

④ 触发中断

TIM\_FLAG\_Trigger: TIM Trigger Flag

⑤xxx

TIM\_FLAG\_Break: TIM Break Flag

**④ 捕获/比较重复捕获标记**

TIM\_FLAG\_CC1OF: TIM Capture Compare 1 overcapture Flag

TIM\_FLAG\_CC2OF: TIM Capture Compare 2 overcapture Flag

TIM\_FLAG\_CC3OF: TIM Capture Compare 3 overcapture Flag

TIM\_FLAG\_CC4OF: TIM Capture Compare 4 overcapture Flag

TIM6 and TIM7 can have only one update flag.

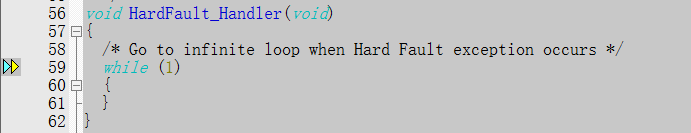
TIM9, TIM12 and TIM15 can have only TIM\_FLAG\_Update, TIM\_FLAG\_CC1, TIM\_FLAG\_CC2 or TIM\_FLAG\_Trigger.

TIM10, TIM11, TIM13, TIM14, TIM16 and TIM17 can have TIM\_FLAG\_Update or TIM\_FLAG\_CC1.

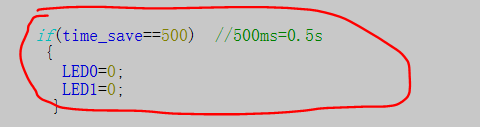
TIM\_FLAG\_Break is used only with TIM1, TIM8 and TIM15.

TIM\_FLAG\_COM is used only with TIM1, TIM8, TIM15, TIM16 and TIM17.

1. **基本定时器编程出现的错误**
2. debug程序跑到HardFault里



原因如下：



主函数没有写while(1)，所以程序刚跑起来，定时器还没计时呢，程序就跑完了。那必然出错。加上while(1)程序就一直跑，所以TIM6可以一直定时不会出错

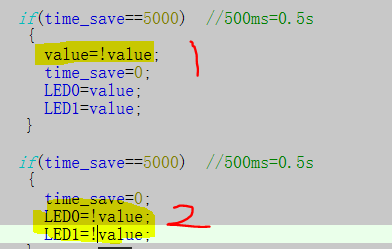
1. 为啥按下复位键后，**看着定时时间感觉不准确呢？**

如果按下复位键，程序开始从头从0x08000开始运行，先把初始化读取运行完了，在执行灯亮灯灭。所以时间不准。但在第一次闪烁之后，初始化结束了，就只执行定时函数了，这时候再计时时间就准确了。

1. 错误的表达

1 先取反再赋值 这样的写法是对的

2 取反赋值写一起 这种表达是错的，value不会改变值



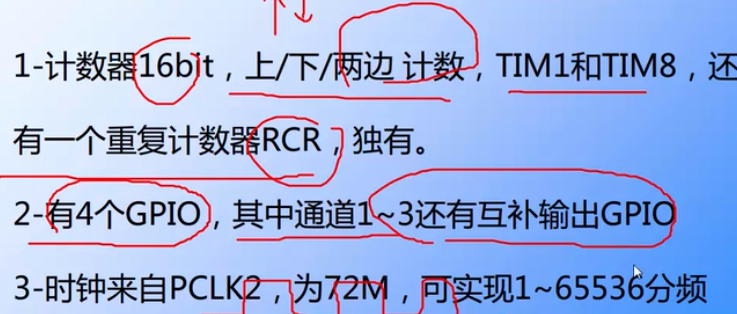
**高级/通用定时器**

高级/通用定时器：定时/输入捕获/输出比较/ PWM生成(边缘或中间对齐模式) /单模式模式输出

高级定时器多：重复计数/互补输出死区/刹车功能 (这些是电机控制中需要的)

* + 1. **概念**

1. **基本特性**



注意高级定时器来自PLCK2

1. 通用定时器和高级定时器**引脚分布(datasheet里有)**

牵扯到端口复用(使能RCC 使能复用功能即可打开)



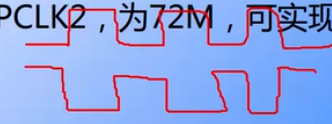
**CHx捕获的是脉冲频率和脉冲宽度**

**ETR捕获的是脉冲个数，每来一个上升沿计数一次**

1. 高级定时器TIM1和TIM2的**三个主要功能**：**定时，PWM波，输入捕获(测量脉冲宽度和PWM输入)**
2. 通用定时器也具有这三个功能，高级定时器比通用定时器多一个**互补输出**的功能和刹车功能。

**互补输出：**如下图就是反着来

PWM输出010101，互补输出就输出101010



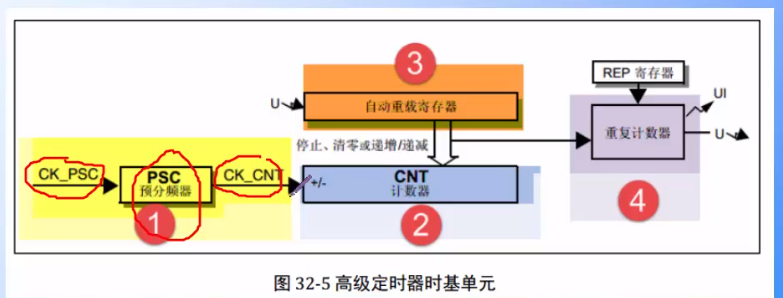
**高级定时器分为7大部分：1.时钟源2.控制器3.时基4.输入捕获部分5.输出比较部分6.断路部分**

1. 高级时钟有**三个时钟源+给别人提供时钟**
   1. 内部时钟CK\_INT:从时钟树可知=72M
   2. 外部的引脚输入的时钟(外部时钟源1)：GPIO\_TIx对应CH1/2/3/4
   3. ETR(外部时钟源2)：查表知对应的GPIO对应着ETR
   4. 内部触发输入：高级定时器的预分频时钟给普通定时器**提供时钟(不常用)**
2. **控制器CR1 CR2 SMCR CER 寄存器**

运用这几个寄存器对TIM1/8高级定时器进行配置

1. **时基**

何谓时基，就是定时器计数一次所需要的时间，一般是一个很小的值。ARR\*时基=计数周期



* 1. CK\_PSC是时钟源给的时钟(具体用哪个时钟源由配置决定)
  2. 经过PSC预分频器把CK\_PSC的时钟进行2/4/6分频得到CK\_CNT
  3. CNT计数器开始计数(CNT计数的最大值由自动重装在寄存器决定)，计数到设置值，产生事件更新中断(TIM\_IT\_Update),然后计数器清0再重新计数。

向下计数：从ARR开始计数到0产生TIM\_IT\_Update，再重复

向上计数：从0开始计数到ARR产生TIM\_IT\_Update，再重复

两端计数：计数器从0 计数到(ARR-1)发生上溢事件产生TIM\_IT\_Update，然后再从ARR减数到1发生下溢事件产生TIM\_IT\_Update。如此往复循环

**以上是所有定时器(高级/通用/基本)通用的部分**

* 1. REP寄存器和重复计数器

向上计数到ARR时不会产生TIM\_IT\_Update，而是REP-1，当REP减到0后才产生TIM\_IT\_Update

向下计数到0时不会产生TIM\_IT\_Update，而是REP-1，当REP减到0后才产生TIM\_IT\_Update

当产生(N+1)个上/下溢时间才会产生TIM\_IT\_Update

此功能就是**延长计时周期**

1. **输入捕获(！！！！！！捕获个555的波形啊测湿度什么的)**

可以这样配置：

* 1. 捕捉脉冲宽度测量：设置捕捉到高电平，CNT开始计数，又检测到下降沿产生中断。然后把CNT计数的值锁存到CCR寄存器里。CCR的值\*计数一次时间=高电平时间=脉冲宽度时间
  2. 检测PWM波的配置：检测PWM波只能用IC1和IC2 两个输入捕获寄存器，因为检测PWM输要用到触发控制器而触发控制器在框图中只有TIFP1和TIFP2与触发寄存器相连(见高级定时器框图1部分)。一个用于捕获高电平时间，一个用于捕获PWM周期

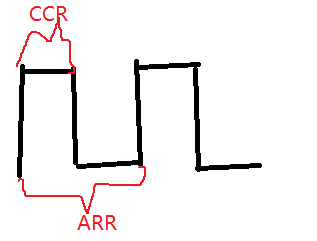
1. **输出比较部分(输出个PWM波)**

三个值CNT CCR ARR

CNT开始计时，计时到CCR寄存器的值时高电平跳变为低，然后计时到ARR寄存器的值时低电平再跳变到高

CCR控制PWM高电平时间 ARR控制PWM周期

CCR/ARR=PWM波占空比



输出PWM波，每个定时器能输出三路PWM，CH1/2/3/4.

**高级定时器输出捕获配置PWM**

**CHx引脚**

1. **输出捕获PWM编程出现的问题**
2. TIM1的TIN1\_CH1N对应着PB13,而PB13又连接了W25QXX(Flash)的SPI2-CSK，**需要考虑干扰问题么**？

引脚选择手册是这么写的：



那应该先把W25Q128禁止了再使用吗？其实完全没必要或者说如果同时用W25Q128和TIM1时才需要禁止。假设我们同时使用W25Q128和TIM1，如果不禁止W25Q128的片选，那么TIM1输出PWM波势必会影响到W25Q128的运行。但如果只单独使用TIM1的话。虽然PB13即连接了PWM1的互补输出，又连接了W25Q128的SPI2-CSK，但我又不用W25Q128，因此PB13输出再怎么影响W25Q128都和我没关系，因为我又不用。但由于W25Q128芯片的供电PWM波输出会产生些毛刺。

如果同时用W25Q128和TIM1那么就要考虑这个问题了，即当用TIM1的时候就把W25Q128片选禁止。当用W25Q128就别用TIM1输出PWM。分开使用，考虑时间配合。这样用A不影响B，用B不打扰A.

PA8的P11(板子右下角)可以把PWM波输出给喇叭

1. 计算PWM**频率**，定时器**频率**

定时器周期：



所以定时器频率：



简单的计算频率是错误的！！！！

**必须先算出周期，再从周期算出频率**，从PSC直接算频率数错误的！！！！

1. TIM\_OC1Init() TIM\_OC1PreloadConfig()这两个函数有啥区别

查1.4固件库使用手册去

TIM\_OC1Init()函数用于配置TIM为PWM模式

TIM\_OC1PreloadConfig()预加载是否使能，使能则改变占空比在计数完成产生更新后下一次生效

1. **输出通道空闲电平极性配置**TIM\_OCInitStruct.TIM\_OCIdleState

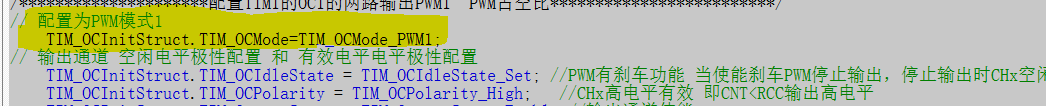
PWM有刹车功能，当使能刹车功能则PWM波就不再输出，此时的输出引脚就成了空闲状态，成了空闲状态时该引脚输出电平的设置

**互补输出通道电平极性配置，**配置到底有效电平是高还是低

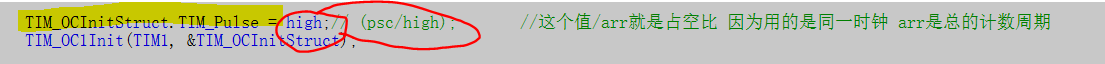
TIM\_OCInitStruct.TIM\_OCNPolarity = TIM\_OCNPolarity\_High;

即CNT<RCC输出有效电平，而CHxN高电平有效

1. 刹车BKIN---PB12 **刹车引脚也要配置成输出**呢(刹车也要配置成输出)
2. 出现的两个错误:
   1. 操作示波器的时候，不小心删除了这一样，导致没有波形输出。



* 1. 这个占空比的输入直接输入值就行，比如高点平时间是4，直接输入高电平时间4即可。不用输入占空比4/9 这样由于高电平时间太短，导致输出的PWM占空比及其小

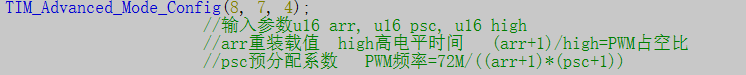


* 1. **为示波器啥看不到死区时间呢？**

我们先计算出死区时间，由BDTR寄存器的位DTG[7:5]可知。11d=00010001b，所以死区时间=Tdtg\*11=(1/72M)\*11=11/72M=0.152us=150ns



在算出PWM波的时间，T=(arr+1\*psc+1)/72M=1us 那么死区时间相比较与PWM一个周期只占1/10，所以很明显能看到死区时间



TIM\_Advanced\_Mode\_Config(89, 79, 4);Tpwm=(89+1)\*(79+1)/72M=100us 那么死区时间相比较与PWM一个周期只占1/100，所以不容易观察到，除非调大波形

1. TIM\_Pulse写满了(**占空比满了)，为什么蜂鸣器还有声音**

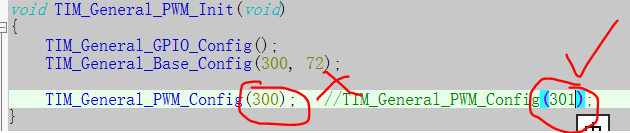
理论上占空比为0/满了时蜂鸣器都没有声音。总之没有具有频率的电平蜂鸣器是不会发声的。

占空比=ARR/RCC 但实际计算中应该是ARR/(写的数字+1).

所以当TIM\_Pulse=300时，占空比=300/301所以蜂鸣器仍有声。

当写入301时，占空比=301/301=1 这时就没声了.

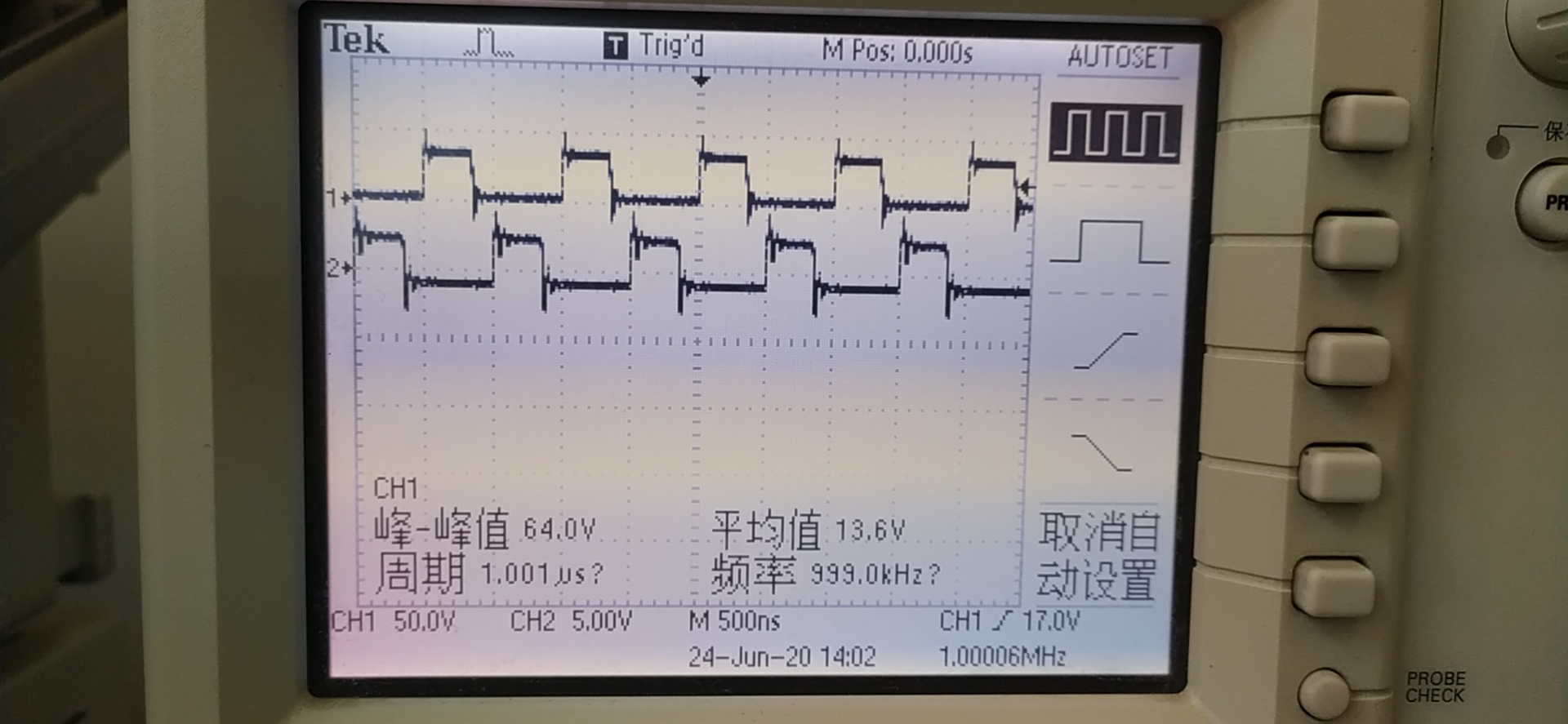
所以要让占空比为1，**TIM\_Pulse应写入(ARR+1)**



1. **TIM\_Pulse大小改变的是蜂鸣器的声音大小，**并不改变蜂鸣器频率！！

**ARR和RCC才改变蜂鸣器的振动频率**

* + 1. **输出捕获PWM观测到的波形**



可以清楚地看到PA8(TIM1\_CH1)的输出，和PB13(TIM1\_CHIN)的互补输出，也看到了死去时间。

* + 1. **PWM的一些思考应用**

PWM控制电机

死区防止MOS关断不及时

刹车关闭PWM输出。

搞一个小车,PWM驱动小车轮子动，刹车搞成个开关，一置1，PWM输出停止，小车就停止了

**高级定时器输入捕获脉宽**

**一，编程前的一些概念**

**1.原理问题：**

①捕捉脉冲宽度测量：GPIO口连接到信号

②设置捕捉到高电平，进入中断服务函数，CNT清0，因为CNT的值会锁到CRR寄存器，所以此时CRR的值为0

③又改成下降沿产生中断(捕获)。当产生中断时，CNT的值又会锁存到CRR里。则此时CRR值就是脉宽时间

**1.编程难点**

①所以要不断在中断中更改捕获配置，高电平捕获中断里改成低电平捕获中断，低电平捕获中断里再改成低电平捕获中断。来为下次捕获做准备。

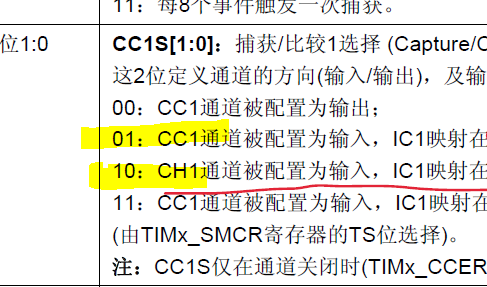
* 1. 脉冲时间过长怎么办？CRR只有16位计数最大65535.

一旦CRR计时到65535，就会**产生更新中断**，然后再重新计数。我们可以中断服务函数套中断，一旦更新中断服务函数里值就+1。然后二者时间相加

* 1. PA0连接的wkup，连接的3.3.又是TIM2的CH1，配置这个，作为实验

**二，编程出现的问题**

1. **CC1通道是什么？和CH1关系**

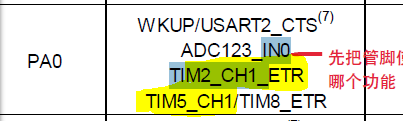


碰见这种英文看不懂，看看上下文的英语全拼。看手册要仔细！！！！



CC1通道:Capture/Compare 1 即捕获/比较1通道。就是CH1配置为输入时的叫法，**CHx输入=CCx=Tix**

1. **配置错误？**



**TIM2\_CH1\_ETR 则PA0即对应TIM2的CH1又对应ETR**

1. **输入捕获引脚的配置**

手册查询PA0配置成GPIO\_Mode\_IN\_FLOATING结果根本不行，配置成GPIO\_Mode\_IPD下拉/GPIO\_Mode\_IPU上拉都可以 **浮空却不行？**

正点的板子引脚外部没有上拉/下拉电阻，所以设置成浮空不能用。只能设置成内部上拉/下拉，即上拉/下拉输入(**一般都设置成下拉来读脉冲信号**)

因为:低电平比较准确就是0，而高电平说不准 多少为高

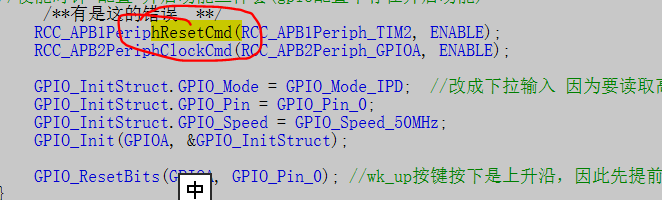
1. 本人编程缺少**使能中断**



定时器配置中断后，记得要使能中断

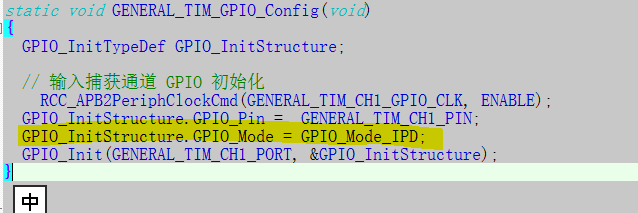
1. 正点的代码永远从后往前看 先看else后看if
2. 为什么进不去中断？？？

debug给中断处加一个串口打印，但并不能打印出数据。从而判断无法进入中断，筛查得出RCC使能有问题。



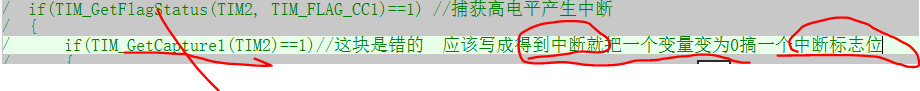
1. 野火的代码为什么不对呢？

野火的GPIO的配置，配置成浮空所以有些问题，改成上拉/下拉输入就好了。



1. 上升沿下降沿中断形式一样但却是不同的中断，因此要**搞出一个标志位。**

**中断进去if判断中断标志位**



1. TIM\_GetCounter(TIM2);

TIM\_GetCapture1(TIM2); 有何区别？

GetCapture1的到的CCRx的值，由上一次输入捕获1事件(IC1)传输的计数器值。

GetCounter得到的是CNT的值。

二者关系就是CNT把值通过ICI传入了CCRx，其实都一样的

1. 输入捕获**时间不对**？

原因一：这的时基配置的**计算要加1！！！**(65535+1)\*( 1/( 72M/(71+1) ) )

所以实际配置时，配置的值要-1 才能写入(因为寄存器从0开始)



原因二：寄存器是从0开始的 rcc产生中断时计数的个数是 配置的值RCC+1

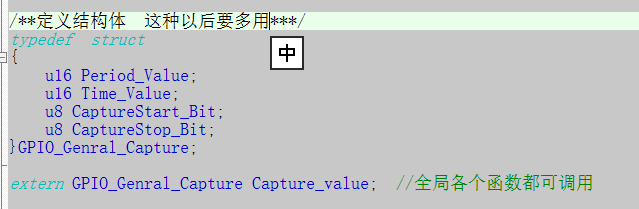
算出的采样次数也要+1 因为寄存器是从0开始计数的



原因三：手松开按键，机械案件有个回弹。

1. 同一组变量，多用结构体定义。方便。

先定义一个结构体，再用结构体名定义变量，声明变量为全局



1. 为啥位操作的这种判断不行呢？

1∈真 但 真只要是非0就行，不一定是1

所以当用式1判断时，if判断成立，只要TIM2CH1\_CAPTURE\_STA的高7位=1 则判断成立，这个值算出来等于128

当你用第二个判断必然是不行的，因为128不等于1





得出结论 如果是**位的操作，if判断不能用==1，只能&**

1. **输入捕获的一些思考应用**

测一个信号的高电平时间？？？

也可以配置成nm级的 更改psc时钟频率72M\*2。然后测量输入的信号的跳变。按键只是提供了一个脉冲信号，完全可以测其他设备的脉冲波

**高级定时器输入捕获PWM**

**CHx详见官方手册13.3.7 PWM输入模式**

* + 1. **概念**
       - 1. 同一个定时器虽然有多个CHx通道，**但不能配置成即是输出捕获又是输入捕获**
         2. 我们用通道定时器TIM2的CH1(PA0)输出PWM, 用高级定时器TIM1的CH1(PA8)进行捕获。(高级定时器和通用定时器只多个刹车功能和互补输出)
         3. PWM捕获原理

①测pwm频率：CH1映射到IC1。

第一次上升沿捕获中断 记录value1 清除捕获中断

第二次再发生上升沿捕获中断再记录一次值value3

清除捕获中断

(value3-value2)/1=频率值

②测pwm占空比：CH1第二路再映射到IC2

测高电平时间PW1捕获脉宽，计算占空比。不用在中断里配置高/低捕获 因为PWM的捕获模式，**IC2会自动捕获脉冲宽度不用配置**

4. 直连与非直连的区别TIM\_ICSelection\_DirectTI

CH1配置成直连，则IC1上升沿触发捕获周期 IC2捕获高电平

非直连，则IC1上升沿触发捕获周期 IC2捕获高电平

5. 注意：捕获输入的 捕获时钟的周期要>PWM输入的周期，这样才能捕获完一个PWM周期。反之来说，就是捕获时钟的频率<PWM频率

实际根据PWM输入频率来调节获时钟的频率，实际中电机的PWM频率为10K~25K 所以我们输入捕获时基一般配置为1K

6. PWM输出模式要用**到T1FP1/2触发信号来触发从模式控制器**因此只有CH1和CH2能配置成PWM捕获模式(触发信号是用来**触发从模式控制器**)

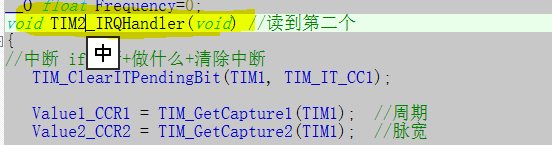
要配置成复位模式，同时设置一捕获到上升沿就自动CNT清0

* + 1. **编程中一些错误**

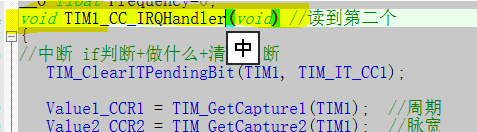
1.无法进入中断？？？

**中断函数名写错了**TIM1高级定时器要写成中断捕获函数TIM1\_CC\_IRQHandler()，如果用的是**基本定时器**(基本定时器没有捕获中断)则要在中断函数里**加一个if判断捕获标志位。**

错误的：



正确的：



2. GPIO配置

PWM捕获的GPIO可以配置成浮空输入(GPIO\_Mode\_IN\_FLOATING)没有问题，但输入捕获不行。为了方便**统一配置成下拉输入**(GPIO\_Mode\_IPD)

**3. PWM输入捕获配置**

当工作做PWM输入模式时,**只需要设置触发信号的那一路即可**（用于测量周期）**另外一路（用于测量占空比）会由硬件自带设置**，不需要再配置。

但记得**开启从模式控制器**(这玩意的作用就相当于，默认PWM模式+第一次捕获自动清零CNT)

**附录**

1. **滤波(香浓采样)**

香浓采样定理：采样的频率必须大于等于两倍的输入信号。

比如输入的信号为1M，又存在高频的信号干扰，那么此时就很有必要进行滤波，我们可以设置采样频率为2M，这样可以在保证采样到有效信号的基础上把高于2M 的高频干扰信号过滤掉。

**为什么能采样？能滤波？**

一般人们把连续信号到离散信号的过程叫采样，把测量值本身的离散化过程叫量化.

**为什么能滤波呢**，假设2M采样一次，而这个信号中有个3M的信号，2M采样一次采样时这个3M的信号都发生完了，因此根本不会采样到这个3M的信号。那么离散信号量化的时候自然就不会量化这个3M的信号，那不就被滤波滤掉了。

1. **带宽**

当通常的时域信号被变换到频域内时，它取值不为零的部分所跨越的频率范围就是这个**信号的带宽**. 定理中关于带宽的表述有时会被误用成“信号最高频率的两倍”。信号带宽是信号频谱的宽度，也就是**信号的最高频率分量与最低频率分量之差**。

假设一信号L:f =2kHz，其最高频率分量是其7次谐波频率，即7f =7×2=14kHz，因此该信号带宽为7f - f =14-2=12kHz。信道带宽则限定了允许通过该信道的信号下限频率和上限频率，也就是限定了一个频率通带。

如果比如一个信道允许的通带为1.5kHz至15kHz，其带宽为13.5kHz，则L的低频2kz>1.5kz 高频14k<15k且带宽12k<13.5k则这个信号就能被很好的传输。相反若低频过小则低频信号被滤掉，高频信号过大高频信号也被滤掉