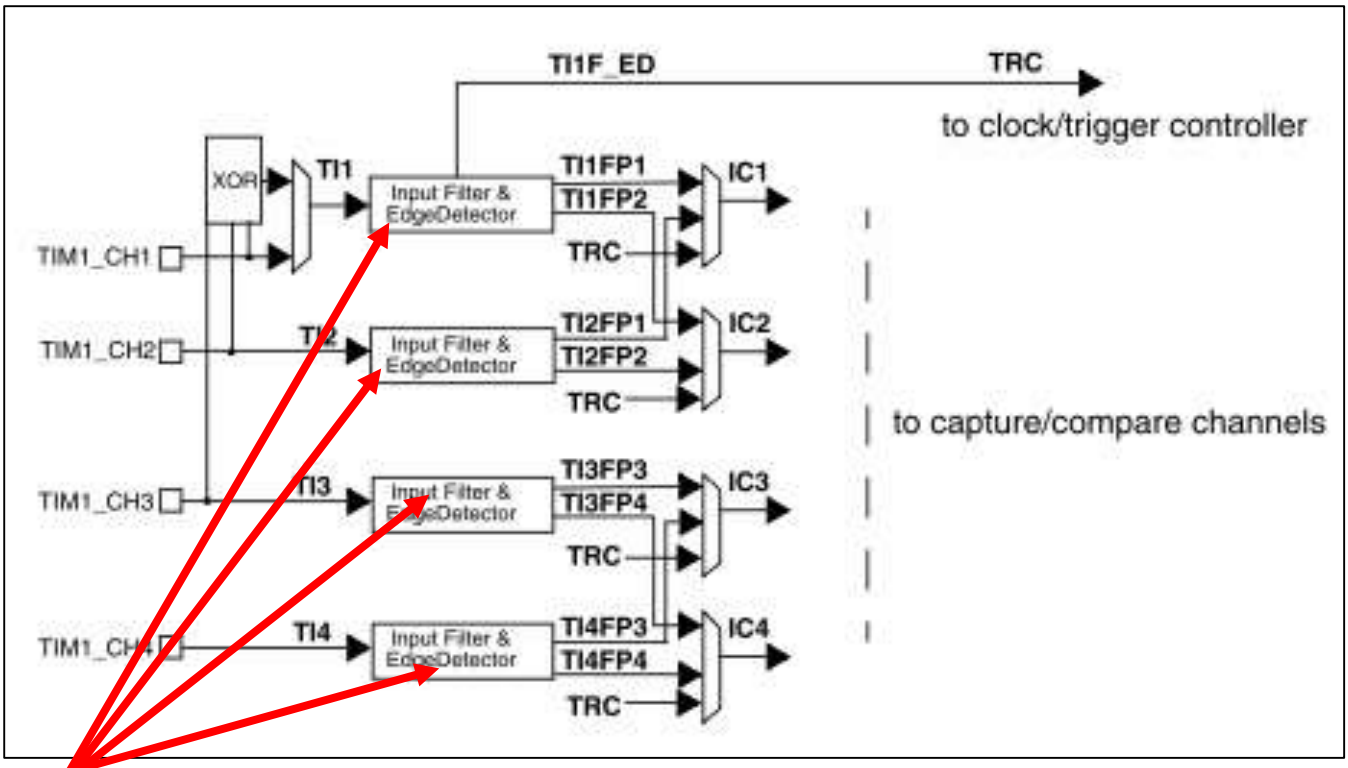

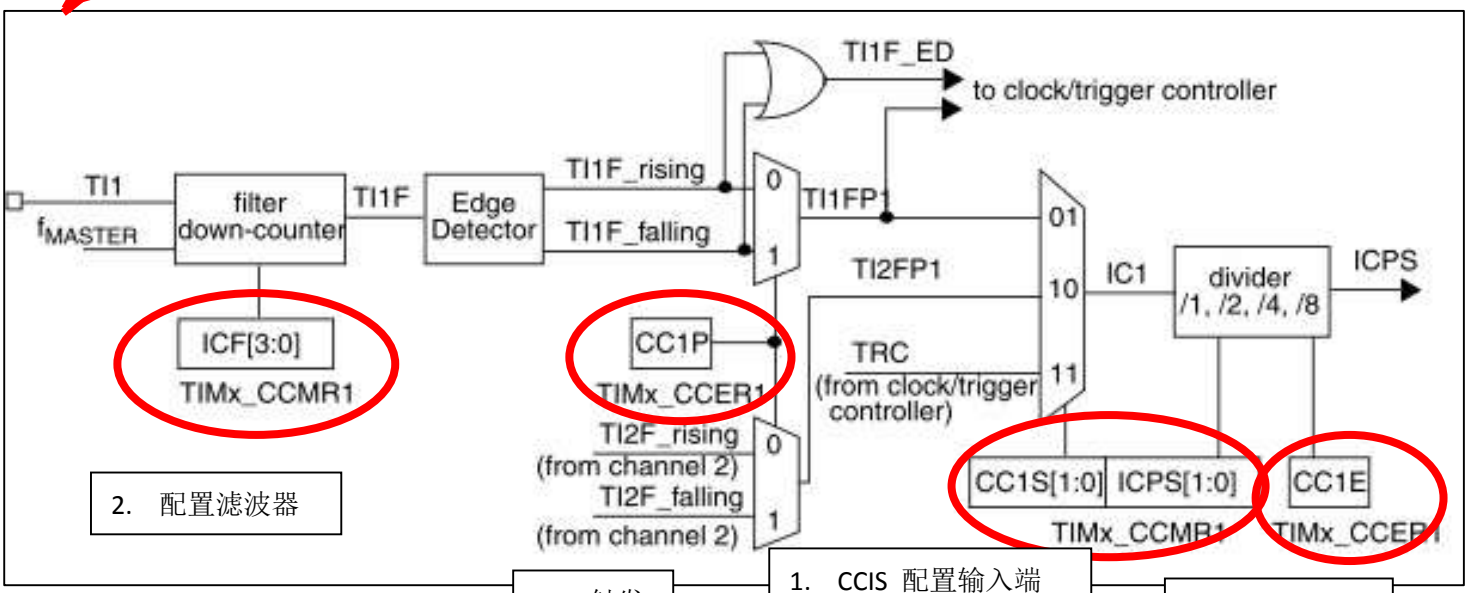

2. 配置滤波器

3. 触发

1. CCIS 配置输入端

4. ICPS 配置预分频

5. 允许通道

普通定时器捕获——捕获周期：

1. 选择有效输入端：例如TIM1_CCR1连接到TI1输入，所以写入TIM1_CCR1寄存器中的CC1S=01，此时通道被配置为输入，并且TIM1_CCR1寄存器变为只读。
2. 根据输入信号TIi的特点，可通过配置TIM1_CCMRi寄存器中的ICiF位来设置相应的输入滤波器的滤波时间。假设输入信号在最多5个时钟周期的时间内抖动，我们须配置滤波器的带宽长于5个时钟周期；因此我们可以连续采样8次，以确认在TI1上一次真实的边沿变换，即在TIMi_CCMR1寄存器中写入IC1F=0011，此时，只有连续采样到8个相同的TI1信号，信号才为有效(采样频率为f_{MASTER})。
3. 选择TI1通道的有效转换边沿，在TIM1_CCER1寄存器中写入CC1P=0(上升沿)。
4. 配置输入预分频器。在本例中，我们希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止(写TIM1_CCMR1寄存器的IC1PS=00)。
5. 设置TIM1_CCER1寄存器的CC1E=1，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置TIM1_IER寄存器中的CC1IE位允许相关中断请求。

- 当产生有效的电平转换时，计数器的值被传送到TIM1_CCR1寄存器。
- CC1IF标志被设置(中断标志)。当发生至少2个连续的捕获时，而CC1IF未曾被清除时，CC1OF也被置1。
- 如设置了CC1IE位，则会产生一个中断。

为了处理捕获溢出(CC1OF位)，建议在读出重复捕获标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的重复捕获信息。

PWM 捕获——捕获周期和占空比：

同时捕获周期和占空比，所以需要两个通道同时捕获。

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个ICi信号被映射至同一个TIi输入。
- 这两个ICi信号的有效边沿的极性相反。
- 其中一个TIiFP信号被作为触发输入信号，而触发模式控制器被配置成复位触发模式。

例如，你可以用以下方式测量TI1上输入的PWM信号的周期(TIM1_CCR1寄存器)和占空比(TIM1_CCR2寄存器)。(具体取决于f_{MASTER}的频率和预分频器的值)

1. 选择TIM1_CCR1的有效输入：置TIM1_CCMR1寄存器的CC1S=01(选中TI1)。
2. 选择TI1FP1的有效极性(用来捕获数据到TIM1_CCR1中和清除计数器)：置CC1P=0(上升沿有效)。
3. 选择TIM1_CCR2的有效输入：置TIM1_CCMR2寄存器的CC2S=10(选中TI1FP2)。
4. 选择TI1FP2的有效极性(捕获数据到TIM1_CCR2)：置CC2P=1(下降沿有效)。
5. 选择有效的触发输入信号：置TIM1_SMCR寄存器中的TS=101(选择TI1FP1)。
6. 配置触发模式控制器为复位触发模式：置TIM1_SMCR中的SMS=100。
7. 使能捕获：置TIM1_CCER1寄存器中CC1E=1，CC2E=1。

CCMR1:

位7:4	**IC1F[3:0]**：输入捕获1滤波器 这几位定义了TI1输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，只有发生了N个事件后输出的跳变才被认为有效。 0000：无滤波器，$f_{SAMPLING}=f_{MASTER}$ 0001：采样频率$f_{SAMPLING}=f_{MASTER}$，N=2 0010：采样频率$f_{SAMPLING}=f_{MASTER}$，N=4 0011：采样频率$f_{SAMPLING}=f_{MASTER}$，N=8 0100：采样频率$f_{SAMPLING}=f_{MASTER}/2$，N=6 0101：采样频率$f_{SAMPLING}=f_{MASTER}/2$，N=8 0110：采样频率$f_{SAMPLING}=f_{MASTER}/4$，N=6 0111：采样频率$f_{SAMPLING}=f_{MASTER}/4$，N=8 1000：采样频率$f_{SAMPLING}=f_{MASTER}/8$，N=6 1001：采样频率$f_{SAMPLING}=f_{MASTER}/8$，N=8 1010：采样频率$f_{SAMPLING}=f_{MASTER}/16$，N=5 1011：采样频率$f_{SAMPLING}=f_{MASTER}/16$，N=6 1100：采样频率$f_{SAMPLING}=f_{MASTER}/16$，N=8 1101：采样频率$f_{SAMPLING}=f_{MASTER}/32$，N=5 1110：采样频率$f_{SAMPLING}=f_{MASTER}/32$，N=6 1111：采样频率$f_{SAMPLING}=f_{MASTER}/32$，N=8 **注**：即使对于带互补输出的通道，该位域也是非预装载的，并且不会考虑CCPC（TIM1_CR2寄存器）的值。
位3:2	**IC1PSC[1:0]**：输入/捕获1预分频器 这2位定义了CC1输入(IC1)的预分频系数。 一旦CC1E=0(TIM1_CCER寄存器中)，则预分频器复位。 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获； 01：每2个事件触发一次捕获； 10：每4个事件触发一次捕获； 11：每8个事件触发一次捕获。

位1:0	**CC1S[1:0]**：捕获/比较1 选择。 这2位定义通道的方向(输入/输出)，及输入脚的选择： 00：CC1通道被配置为输出； 01：CC1通道被配置为输入，IC1映射在TI1FP1上； 10：CC1通道被配置为输入，IC1映射在TI2FP1上； 11：CC1通道被配置为输入，IC1映射在TRC上。此模式仅工作在内部触发器输入被选中时(由TIM1_SMCR寄存器的TS位选择)。 注：CC1S仅在通道关闭时(TIM1_CCER1寄存器的CC1E=0)才是可写的。

CCER1:

位7	**CC2NP**：输入捕获/比较2互补输出极性。参考CC1NP的描述。
位6	**CC2NE**：输入捕获/比较2互补输出使能。参考CC1NE的描述。
位5	**CC2P**：输入捕获/比较2输出极性。参考CC1P的描述。
位4	**CC2E**：输入捕获/比较2输出使能。参考CC1E的描述。
位3	**CC1NP**：输入捕获/比较1互补输出极性 0：OC1N高电平有效； 1：OC1N低电平有效。 **注1**：一旦LOCK级别(TIM1_BKR寄存器中的LCCK位)设为3或2且CC1S=00(通道配置为输出)则该位不能被修改。 **注2**：对于有互补输出的通道，该位是预装载的。如果CCPC=1（TIM1_CR2寄存器），只有在COM事件发生时，CC1NP位才从预装载位中取新值。
位2	**CC1NE**：输入捕获/比较1互补输出使能 0：关闭－ OC1N禁止输出，因此OC1N的输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1E位的值。 1：开启－ OC1N信号输出到对应的输出引脚，其输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1E位的值。 **注**：对于有互补输出的通道，该位是预装载的。如果CCPC=1(TIM1_CR2寄存器)，只有在COM事件发生时，CC1NE位才从预装载位中取新值。
位1	**CC1P**：输入捕获/比较1输出极性 **CC1通道配置为输出**： 0：OC1高电平有效； 1：OC1低电平有效。 **CC1通道配置为触发(参考*图61*)**： 0：触发发生在TI1F的高电平或上升沿； 1：触发发生在TI1F的低电平或下降沿。 **CC1通道配置为输入(参考*图61*)**： 0：捕捉发生在TI1F的高电平或上升沿； 1：捕捉发生在TI1F的低电平或下降沿。 **注1**：一旦LOCK级别(TIM1_BKR寄存器中的LCCK位)设为3或2，则该位不能被修改。 **注2**：对于有互补输出的通道，该位是预装载的。如果CCPC=1（TIM1_CR2寄存器），只有在COM事件发生时，CC1P位才从预装载位中取新值。
位0	**CC1E**：输入捕获/比较1输出使能 **CC1通道配置为输出**： 0： 关闭－ OC1禁止输出，因此OC1的输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1NE位的值。 1： 开启－ OC1信号输出到对应的输出引脚，其输出电平依赖于MOE、OSSI、OSSR、OIS1、OIS1N和CC1NE位的值。 **CC1通道配置为输入**： 该位决定了计数器的值是否能捕获入TIM1_CCR1寄存器。 0：捕获禁止； 0：捕获使能。 注：对于有互补输出的通道，该位是预装载的。如果CCPC=1(TIM1_CR2寄存器)，只有在COM事件发生时，CC1E位才从预装载位中取新值。

IER:

位7	BIE：允许刹车中断 0：禁止刹车中断； 1：允许刹车中断。
位6	TIE：触发中断使能 0：禁止触发中断； 1：使能触发中断。
位5	COMIE：允许COM中断 0：禁止COM中断； 1：允许COM中断。
位4	CC4IE：允许捕获/比较4中断 0：禁止捕获/比较4中断； 1：允许捕获/比较4中断。
位3	CC3IE：允许捕获/比较3中断 0：禁止捕获/比较3中断； 1：允许捕获/比较3中断。
位2	CC2IE：允许捕获/比较2中断 0：禁止捕获/比较2中断； 1：允许捕获/比较2中断。
位1	CC1IE：允许捕获/比较1中断 0：禁止捕获/比较1中断； 1：允许捕获/比较1中断。
位0	UIE：允许更新中断 0：禁止更新中断； 1：允许更新中断。

SMCR:

位7	**MSM**：主/从模式 0：无作用； 1：触发输入(TRGI)上的事件被延迟了，以允许定时器1与它的从定时器间的完美同步(通过TRGO)。
位6:4	**TS**：触发选择 这3位选择用于选择同步计数器的触发输入。 000：内部触发ITR0连接到TIM6 TRGO 001：保留 010：内部触发ITR2连接到TIM5 TRGO 011：保留 100：TI1的边沿检测器(TI1F_ED) 101：滤波后的定时器输入1(TI1FP1) 110：滤波后的定时器输入2(TI2FP2) 111：外部触发输入(ETRF) 注：这些位只能在未用到(如SMS=000)时被改变，以避免在改变时产生错误的边沿检测。
位3	保留，始终读为0。
位2:0	**SMS**：时钟/触发/从模式选择 当选择了外部信号，触发信号(TRGI)的有效边沿与选中的外部输入极性相关(见输入控制寄存器和控制寄存器的说明) 000：时钟/触发控制器禁止 – 如果CEN=1，则预分频器直接由内部时钟驱动。 001：编码器模式1 – 根据TI1FP1的电平，计数器在TI2FP2的边沿向上/下计数。 010：编码器模式2 – 根据TI2FP2的电平，计数器在TI1FP1的边沿向上/下计数。 011：编码器模式3 – 根据另一个输入的电平，计数器在TI1FP1和TI2FP2的边沿向上/下计数。 100：复位模式 – 在选中的触发输入(TRGI)的上升沿时重新初始化计数器，并且产生一个更新寄存器的信号。 101：门控模式 – 当触发输入(TRGI)为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止(但不复位)。计数器的启动和停止都是受控的。 110：触发模式 – 计数器在触发输入TRGI的上升沿启动(但不复位)，只有计数器的启动是受控的。 111：外部时钟模式1 – 选中的触发输入(TRGI)的上升沿驱动计数器。 注：如果TI1F_ED被选为触发输入(TS=100)时，不要使用门控模式。这是因为TI1F_ED在每次TI1F变化时只是输出一个脉冲，然而门控模式是要检查触发输入的电平。

CR1:

位7	**ARPE**：自动预装载允许位 0：TIM1_ARR寄存器没有缓冲，它可以被直接写入； 1：TIM1_ARR寄存器由预装载缓冲器缓冲。
位6:5	**CMS**：选择中央对齐模式 00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。 01：中央对齐模式1。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中*CCiS*=00)的输出比较中断标志位，只在计数器向下计数时被置1。 10：中央对齐模式2。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中*CCiS*=00)的输出比较中断标志位，只在计数器向上计数时被置1。 11：中央对齐模式3。计数器交替地向上和向下计数。配置为输出的通道(TIM1_CCMRx寄存器中*CCiS*=00)的输出比较中断标志位，在计数器向上和向下计数时均被置1。 注1：在计数器开启时(CEN=1)，不允许从边沿对齐模式转换到中央对齐模式。 注2：在中央对齐模式下，编码器模式（ GPT_SMCR寄存器中的SMS=001，010，011）必须被禁止。
位4	**DIR**：方向 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位3	**OPM**：单脉冲模式 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除CEN位)时，计数器停止。
位2	**URS**：更新请求源 0：如果UDIS允许产生更新事件，则下述任一事件产生一个更新中断： – 寄存器被更新(计数器上溢/下溢) – 软件设置UG位 – 时钟/触发控制器产生的更新 1：如果UDIS允许产生更新事件，则只有当下列事件发生时才产生更新中断，并UIF置1： – 寄存器被更新(计数器上溢/下溢)
位1	**UDIS**：禁止更新 0：一旦下列事件发生，产生更新(UEV)事件： – 计数器溢出/下溢 – 产生软件更新事件 – 时钟/触发模式控制器产生的硬件复位 被缓存的寄存器被装入它们的预装载值。 1：不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了UG位或时钟/触发控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位0	**CEN**：允许计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了CEN位后，外部时钟、门控模式和编码器模式才能工作。然而触发模式可以自动地通过硬件设置CEN位。

```
#include "type_def.h"
uint IrCycle = 0;                                   //红外周期
uint IrHigh = 0;                                    //占空比
void delay_1us(void){
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
}
void delay_ms(u32 nTime){
    u16 i;
    while(nTime--)
    for(i=900;i>0;i--)
    delay_1us();
}
void inline delay_us(unsigned int i){
    while(i--)
        delay_1us();
}

void Ir_Init(void){
    PC_DDR_DDR1 = 0;       //PC1 为输入，PC1 复用 TIM1_CH1
    PC_CR1_C11 = 1;
    PC_CR2_C21 = 0;         //中断
}
//使用 TIM1 的 PWM 输入捕获
void Ir_PWM_Init(void){
    TIM1_PSCRH = 0;
    TIM1_PSCRL = 0;
    TIM1_CCER1 &= 0xee;     //禁止输入捕获 IC1,IC2;
    TIM1_CCMR1 |= 0x31;     //设置滤波器 8 个事件周期，CC1 通道配置为输入并映射 IC1 到 TI1FP1;
    TIM1_CCER1 &= 0xec;     //IC1 上升沿触发;
    TIM1_CCMR2 |= 0x32;     //设置滤波器 8 个事件周期，CC2 通道配置为输入并映射 IC2 到 TI1FP2;
    TIM1_CCER1 |= 0x20;     //IC2 下降沿触发，IC1 上升沿触发;
    TIM1_SMCR |= 0x54;      //01010100 选源触发源为 TIFP1 和触发方式 复位模式;
    TIM1_CCER1 |= 0x11;     //使能输入捕获 1，2;
    TIM1_CR1 |= 0x05;       //允许更新请求源，允许计数;
}

void pwm_ch2_output_init(){     //设置 TIM2 为输出
    TIM2_PSCR = 0X01;
    TIM2_ARRH = 0X00;
    TIM2_ARRL = 100;
    TIM2_CR1 &= 0XFE;
    CLK_CKDIVR&= (uint8_t)(~0x18);/*使能内部时钟*/
    CLK_CKDIVR|= (uint8_t)0x00;/*设置时钟为内部 16M 高速时钟*/
    TIM2_CCER1 &= 0xFC;
    TIM2_CCER1 |= 0x01;
```

```
    TIM2_CCMR1 &= 0x8F;
    TIM2_CCMR1 |= 0X60;
    TIM2_CCR1H = 0X00;
    TIM2_CCR1L = 52;
    TIM2_CR1 |= 0X01;
}
void Ir_Receive(void){
    if ((TIM1_SR1_CC1IF == 1)&&(TIM1_SR1_CC2IF == 1)){
        IrHigh = (uint)(TIM1_CCR2H);
        IrHigh = (IrHigh << 8) + TIM1_CCR2L;  //IrHigh 占空比
        IrCycle = (uint)(TIM1_CCR1H);
        IrCycle = (IrCycle << 8) + TIM1_CCR1L;  //IrCycle 周期
    }
}
//初始化
void Devices_Init(void){
    delay_ms(200);
    CLK_CKDIVR&= (uint8_t)(~0x18);/*使能内部时钟*/
    CLK_CKDIVR|= (uint8_t)0x00;/*设置时钟为内部 16M 高速时钟*/
    Ir_Init();
    pwm_ch2_output_init();
    Ir_PWM_Init();

}
//主程序
void main( void ){
  Devices_Init();
  while(1){
    Ir_Receive();
    delay_ms(200); //这个 delay 是必须的，不知道为什么不 delay 一下可能 receive 函数中无法进入判断
  }
}
```