SPC1168 PWM 使用指南

概述

PWM 在电力电子系统中有着至关重要的作用，广泛的应用在电机控制、开关电源及 UPS 中。SPC1168 中带有 6 个独立的 PWM 单元，每个 PWM 单元有两路输出。同时各个 PWM 单元可以连接在一起，产生同步的 PWM 组。

目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-14	CanChai	Outdated	首次发布。
C/0	2024-03-26	Jiali Zhou	Released	修改排版格式。

术语或缩写

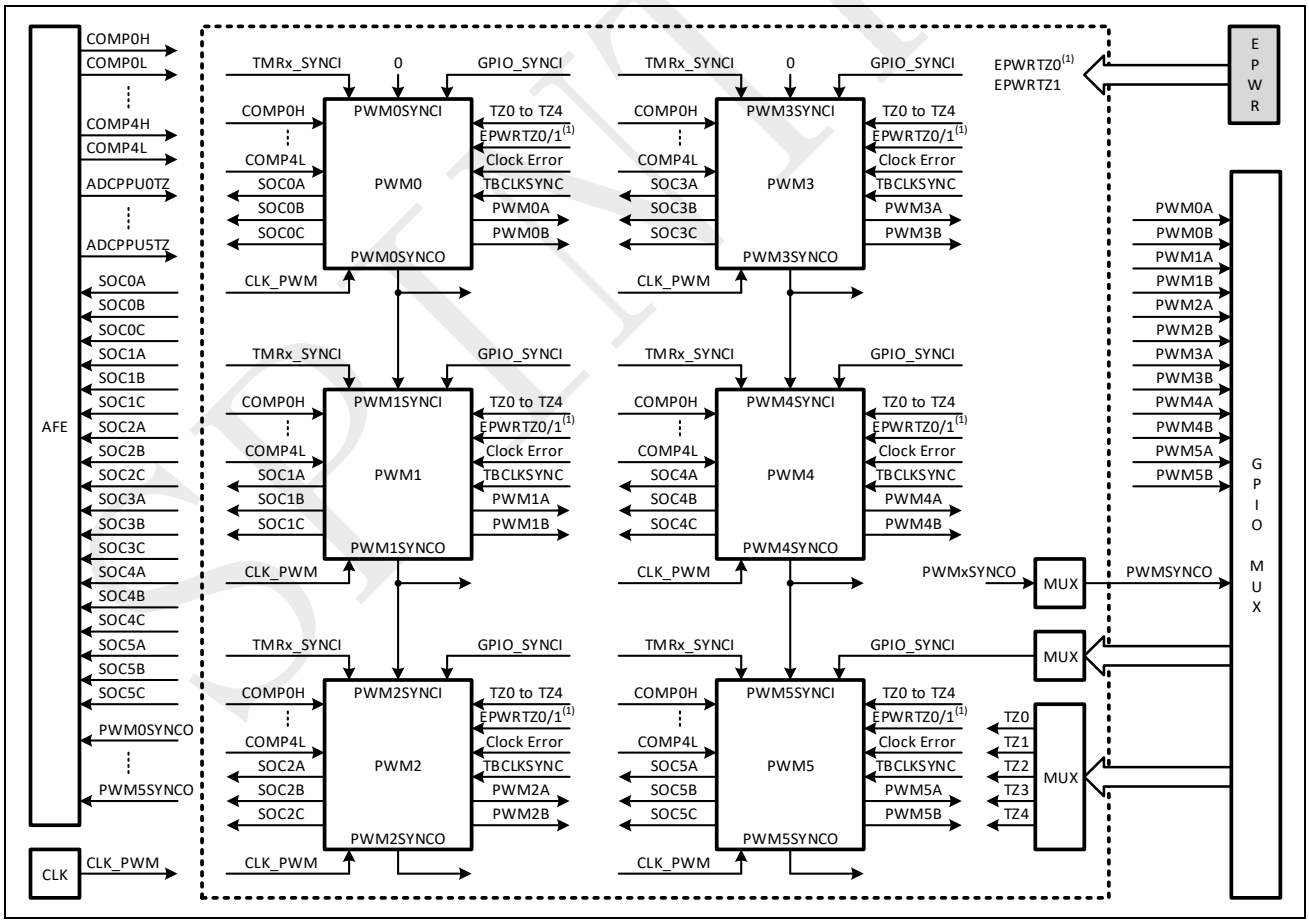
术语或缩写	描述
/	/

1 PWM 特性

SPC1168 的 PWM 单元有以下特点：

- 周期可配置的 16-bit 时基计数
- 软件直接强制改变 PWM 输出
- 独立可配的相位控制
- 周期性的相位同步
- 上升沿延迟和下降沿延迟独立可配置的死区控制
- 异常事件下可配的周期性或者一次性封锁
- 封锁时可以配置 PWM 输出为高、低或高阻
- 数字比较或者封锁信号输入可产生原始封锁事件或者过滤后的封锁事件
- 所有事件可以用于产生 CPU 中断或者 ADC SOC（开始采样）信号

图 1-1：PWM 单元功能框图

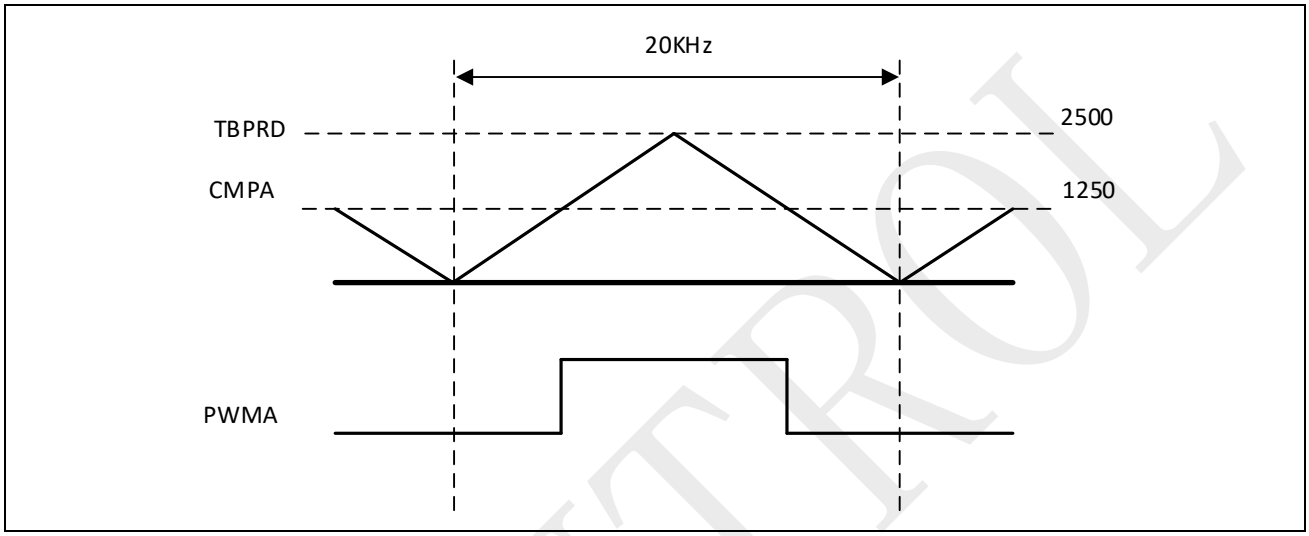


2 PWM 实例

2.1 单通道输出：固定频率

如果有产生任意占空比单通道 PWM 的需求，如图 2-1 所示。

图 2-1：单通道独立 PWM



PWM_InitSingleChannel ()配置重点如下：

- PWM 波形预设为中央对称型，中央对称型 PWM counter 之周期计算如下：

  Period = PWM_Clock_Freq / PWM_Freq/2

  以上述为例，PWM 的 Clock 与 HCLK 同频，因此为 100MHz，设定的 PWM 载波率为 20KHz，因此周期为 50us。故此时 Period = 2500；
- 预设当 counter 在往上数遇到 CMP 时，将波形拉高，往下数遇到 CMP 时，将波形拉低；
- 务必先执行 CLOCK_InitWithRCO()或是 CLOCK_Init()，否则 PWM clock 可能会有错误；
- 本函数不配置 PWM 波形的 dead time，用户若有控制需求，可参考死区配置示例。

Example Code

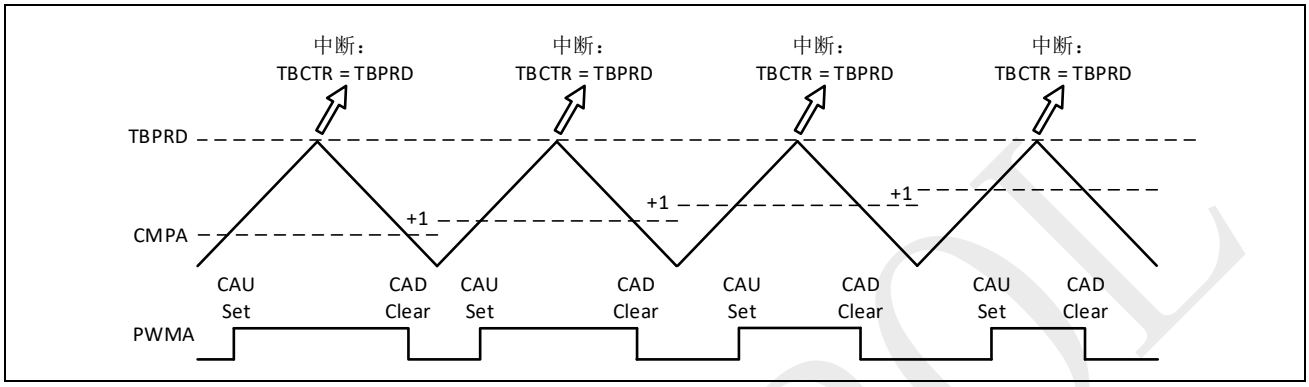
```
int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_100MHZ);

    Delay_Init();

    /*
     * Init the UART
```

```
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* Init PWM_Sel and output 20kHz waveform on channel B */
    PWM_SingleChannelInit(PWM_Sel, PWM_CHB, PWM_Frequency);

    /*
     * 'PWM_SingleChannelInit' had init the count mode as up and down, so,
     * there is no need to call 'PWM_SetCounterMode()'.
     */

    /* Set PWM_SelB output 50% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_Frequency);
    PWM_SetCMPB(PWM_Sel, u32PWMPeriod / 2);

    /* Select GPIO37 as the channel B output of PWM_Sel */
    GPIO_SetPinChannel(PIN_PWMB, PIN_PWMB_CH);

    /* Start PWM_Sel */
    PWM_RunCounter(PWM_Sel);

    while (1)
    {

    }
}
```

2.2 单通道输出：动态频率

如果有产生任意占空比单通道 PWM 的需求，可以通过动态改变比较值的方式实现，如图 2-2 所示。

图 2-2：周期性变化 PWM



周期性变化 PWM 配置重点如下：

- 配置 CMPA 到 CMPAA 的时机；
- 配置 PWM 中断；
- 务必先执行 CLOCK_InitWithRCO()或是 CLOCK_Init()，否则 PWM clock 可能会有错误；
- 本函数不配置 PWM 波形的 dead time，用户若有控制需求，可参考死区配置示例。

Example Code

```
void main()
{
    FLASH_WALLOW();
    FLASH_SetTiming(200000000);
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
    * Init the UART
    *
    * 1.Set the GPIO44/45 as UART FUNC
    *
    * 2.Enable the UART CLK
    *
    * 3.Set the rest para
    */
    GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART,38400);

    /*** PWM initial on PWM0A ***/
    PWM_SingleChannelInit(PWM0,PWM_CHA,20000);
    /*** Load CMPA when TBCNT=ZERO ***/
```

```
    PWM_SetCMPALoadTiming(PWM0,ON_ZERO);
    /*** Run counter ***/
    PWM_RunCounter(PWM0);
    /*** Initialize 50% duty cycle ***/
    PWM_SetCMPA(PWM0,2500);
    /*** Set GPIO18 as PWM0A ***/
    GPIO_SetPinChannel(GPIO_18,GPIO18_PWM0A);
    /*** Generate interrupt whenever TBCNT=TBPRD ***/
    PWM_SetTimeEventIntTiming(PWM0,EQU_PERIOD);
    PWM_SetTimeEventIntPeriod(PWM0,ON_1ST_EVENT);
    PWM_EnableTimeEventInt(PWM0);
    /*** Enable PWM0 Interrupt in CPU side ***/
    NVIC_EnableIRQ(PWM0_IRQn);

    while(1)
    {

    }
}

void PWM0_IRQHandler(void)
{
    uint16_t u16CMPANextVal = PWM_GetCMPA(PWM0) + 1;
    /*** Update CMPA ***/
    if(u16CMPANextVal>PWM_GetPeriod(PWM0))
        PWM_SetCMPA(PWM0, 0);
    else
        PWM_SetCMPA(PWM0, u16CMPANextVal);
    /*** Clear interrupt flag ***/
    PWM_ClearTimeEventInt(PWM0);
}
```

2.3 双通道输出：带死区的互补波形

在电力电子、开关电源以及电机控制中，往往要应对半桥电路、全桥电路；这种电路的一个基本控制方法，就是用互补的 PWM 波去分别驱动半桥、全桥的上下桥臂开关管，同时互补的 PWM 波还需要带有一定的死区时间。

PWM 单元非常灵活，可以覆盖到半桥、全桥中的各种应用，这里先讲解一个简单的半桥带有死区的 PWM 波。下图是 PWM 波单元中的死区控制部分，可以看出它本身是具有复杂灵活的死区配置方式。对于半桥控制中的死区应用，一般是通过将一个 PWM 波增加死区时间后，取反得到带死区的互补 PWM 波。

如图 2-3 和图 2-4 中展示了使用 PWM 的死区控制单元实现带死区互补波形的过程。首先将 PWM 的某一路波形送入到死区控制单元，如图 2-3 所示，橙色和亮绿色信号均来自同一路 PWM 波，随后将对橙色波形的上升沿延时一段时间，形成深红色波形，深红色波形最后直接作为死区控制子模块的 A 通道口的波形输出到下一个 PWM 子模块；而亮绿色波形经过下降沿延时后，形成深绿色波形，深绿色波形将进一步翻转后作为死区控制子模块的 B 通道口的波形输出到下一个 PWM 子模块。详细的波形图如图 2-4 所示。

图 2-3：死区控制单元框图

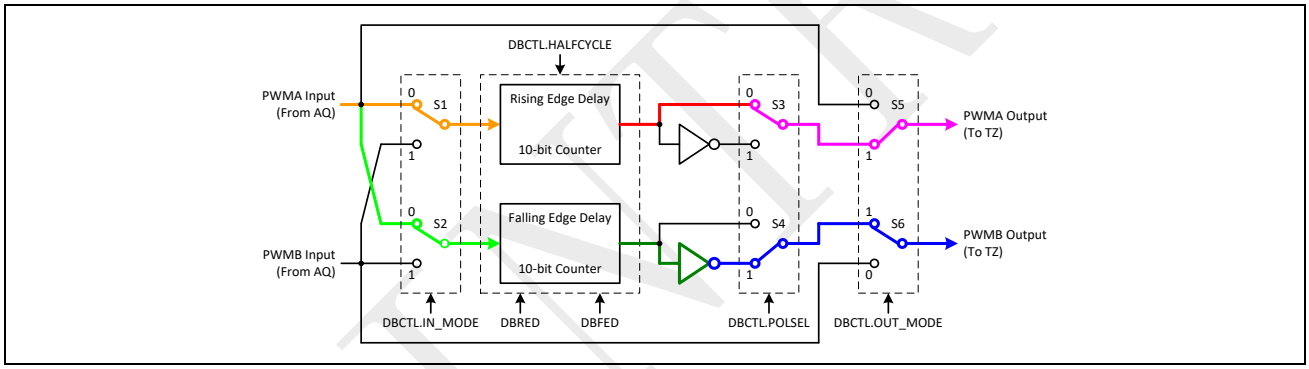
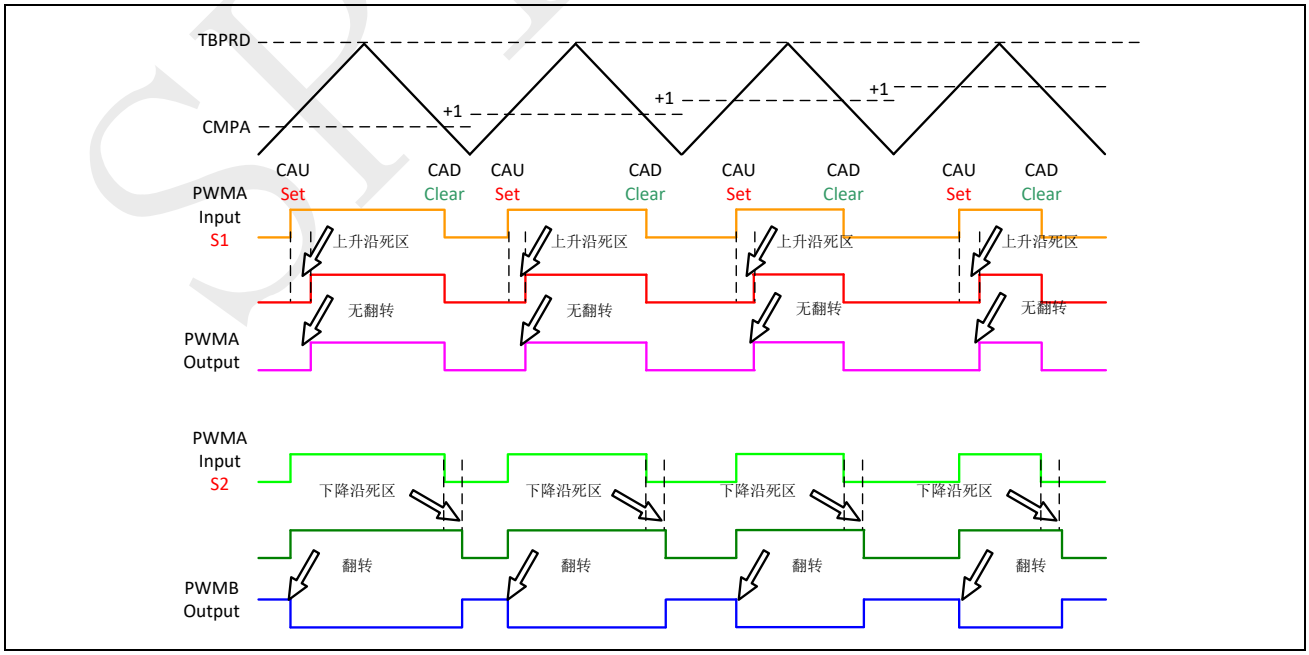


图 2-4：生成带死区互补 PWM 示意图



PWM_ComplementaryPairChannelInit()配置重点如下：

- PWM 波形预设为中央对称型，中央对称型 PWM counter 之周期计算如下：

  Period = PWM_Clock_Freq / PWM_Freq/2

  以上述为例，PWM 的 Clock 与 HCLK 同频，因此为 200MHz，设定的 PWM 频率为 20KHz，因此周期为 50us。设置到周期寄存器的数值本应该为：Period = 200000000 / 20000 = 10000，但由于是上下计数模式，所以实际周期寄存器的数值只需要设置为 Period = Period / 2 = 5000 即可；

- 预设当 counter 在往上数遇到 CMPA 时，将波形拉高，往下数遇到 CMPA 时，将波形拉低。

注意： 当用这种波形配置时，CMPA 越高，代表最终输出的 PWM 波形 Duty 越低，反之亦然。

- CMPB 的配置在本配置中不影响波形输出；
- 务必先执行 CLOCK_InitWithRCO()或是 CLOCK_Init()，否则 PWM clock 可能会有错误。

Example Code

```
int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /*
     * 1. Init PWM_Sel and output 20kHz waveform on channel A
     *
     * 2. Add 1us dead band at falling and rising edge
     *
     * 3. Channel B output the channel A waveform which added DB
     */
    PWM_ComplementaryPairChannelInit(PWM_Sel, PWM_FREQ, PWM_DB_NS);
```

```
    /* Set PWM_SelA output 75% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM_Sel, u32PWMPeriod / 4);

    /* Select the channel A/B output of PWM_Sel respectively */
    GPIO_SetPinChannel(PIN_PWMA, PIN_PWMA_CH);
    GPIO_SetPinChannel(PIN_PWMB, PIN_PWMB_CH);

    /* Start PWM_Sel */
    PWM_RunCounter(PWM_Sel);

    while (1)
    {

    }
}
```

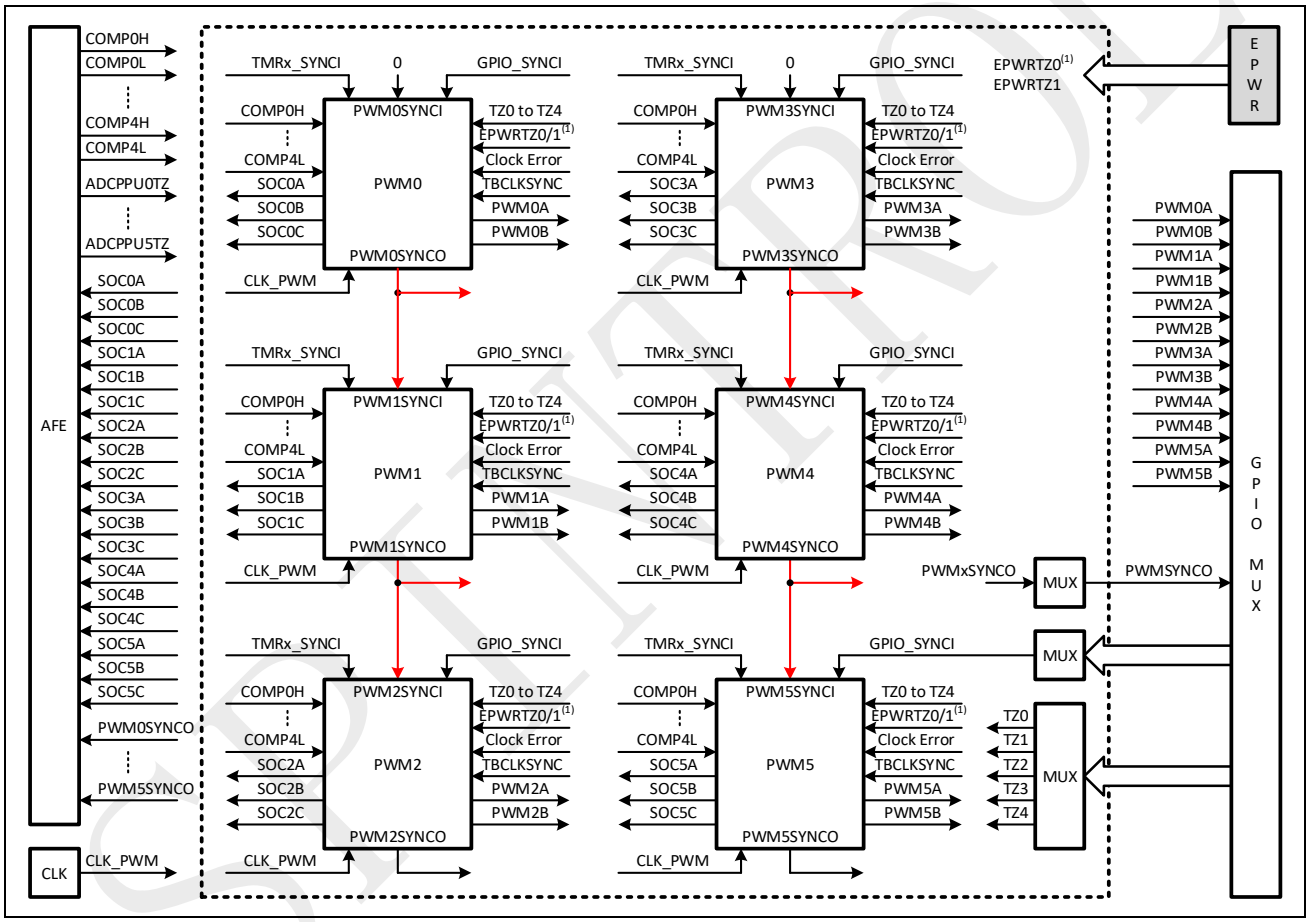
2.4 PWM 同步

在使用 PWM 过程中，有时需要 PWM 间进行相位同步，这时可以用 PWM 级联或非级联方式实现同步功能。

2.4.1 级联同步

级联同步指的是同步信号在不同 PWM 模块中依次传导，SPC1168 可以建立级联同步关系的链路为 PWM0->PWM1->PWM2 以及 PWM3->PWM4->PWM5，如图 2-5 所示。

图 2-5：级联关系



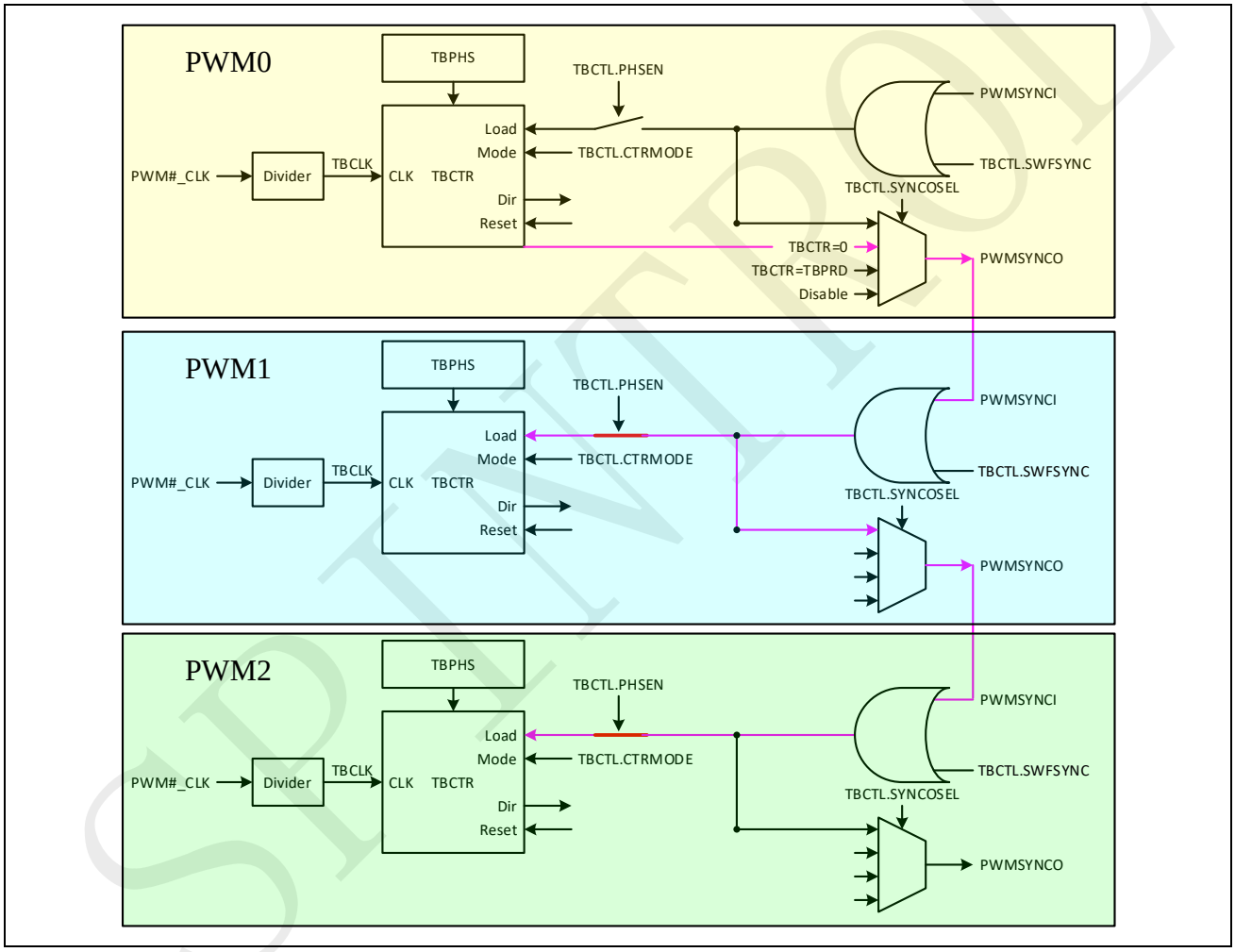
注意： PWMSYNCI 除了来自 PWMSYNCO，还可以来自 Timer 或 GPIO。

本小节的示例代码演示了级联同步功能，如图 2-6 所示：

- PWM0 使能 TBCTR=0 信号通过 PWMSYNCO 的输出；
- PWM0 PWMSYNCO 输出通过 PWM1 PWMSYNCI 改变 PWM1 相位；
- 由于 PWM1 使能 PWMSYNCI 信号通过 PWMSYNCO 输出，这使得 PWM1 PWMSYNCO 信号可以达到 PWM2，并改变 PWM2 相位；

经过如上配置，将会在每个 PWM0 TBCTR=0 的时刻，同步 PWM0，PWM1，PWM2 之间的相位差。

图 2-6：级联式同步



代码实现如下所示：

Example Code

```
int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* Select GPIO18/19 as the channel A/B output of PWM0 respectively */
    GPIO_SetPinChannel(GPIO_18, GPIO18_PWM0A);
    GPIO_SetPinChannel(GPIO_19, GPIO19_PWM0B);
    /* Select GPIO20/21 as the channel A/B output of PWM1 respectively */
    GPIO_SetPinChannel(GPIO_20, GPIO20_PWM1A);
    GPIO_SetPinChannel(GPIO_21, GPIO21_PWM1B);
    /* Select GPIO22/23 as the channel A/B output of PWM2 respectively */
    GPIO_SetPinChannel(GPIO_22, GPIO22_PWM2A);
    GPIO_SetPinChannel(GPIO_23, GPIO23_PWM2B);

    /* Init PWM on both channel A and channel B */
    PWM_ComplementaryPairChannelInit(PWM0, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM1, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM2, PWM_FREQ, PWM_DB_NS);
    PWM_ActionQualifierCHA(PWM2, CAU_SET_LOW|CAD_SET_HIGH);

    /* Set PWMxA output duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM0, (u32PWMPeriod * 2) / 3);
    PWM_SetCMPA(PWM1, (u32PWMPeriod * 2) / 3);
    PWM_SetCMPA(PWM2, (u32PWMPeriod * 1) / 3);

    PWM_SetSyncOutEvent(PWM0, TBCTL_BIT_SYNCOSEL_TBCNT_EQU_ZERO);
    PWM_SetSyncOutEvent(PWM1, TBCTL_BIT_SYNCOSEL_SYNCI_AND_FRCSYNC);

    /* Enable PWM synchronization */
    PWM_EnableSync(PWM0);
    PWM_EnableSync(PWM1);
    PWM_EnableSync(PWM2);

    /* Set PWMx counting up after SYNC */
    PWM_SetCounterDirAfterSync(PWM0, COUNT_UP);
```

```
    PWM_SetCounterDirAfterSync(PWM1, COUNT_UP);
    PWM_SetCounterDirAfterSync(PWM2, COUNT_UP);

    /* Set the re-load value when SYNC signal happened */
    PWM_SetSyncReloadValue(PWM1, (PWM1)->CMPA.all);
    PWM_SetSyncReloadValue(PWM2, ((PWM2)->CMPA.all));

    /* Start counting */
    PWM_RunCounter(PWM2);
    PWM_RunCounter(PWM1);
    PWM_RunCounter(PWM0);

    while (1)
    {

    }
}
```

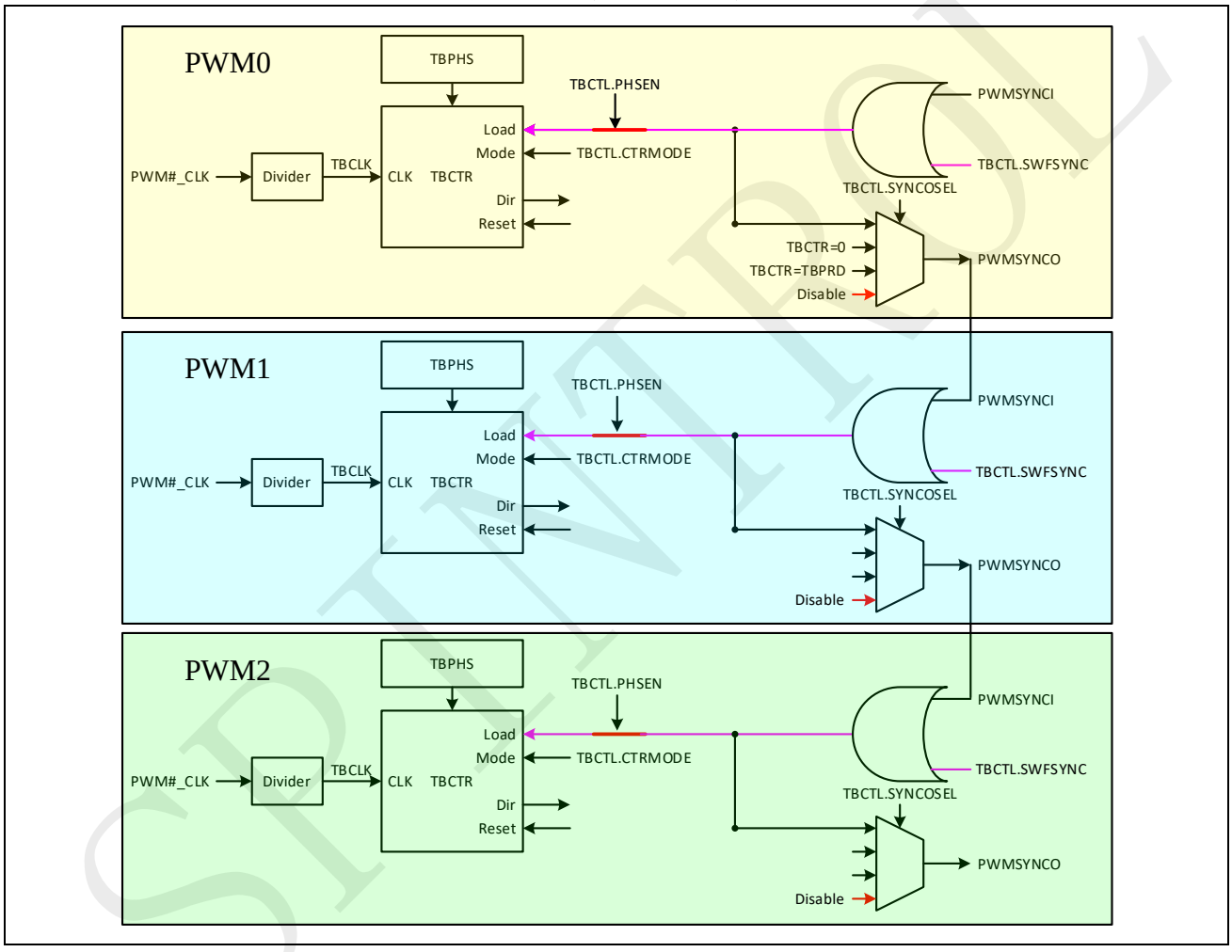
2.4.2 非级联同步

非级联同步指的是同步信号同时加在不同 PWM 模块上，其信号链路如图 2-7：非级联式同步中红色信号线路所示。

示例代码配置动作可描述为：

- 关闭各 PWM 模块的 PWMSYNCO 功能；
- 采用软件同时改变 PWM0，PWM1，PWM2 的相位。

图 2-7：非级联式同步



代码实现如下所示：

Example Code

```
int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
```

```
    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    GPIO_SetPinChannel(PIN_PWM2A, PIN_PWM2A_CH);
    GPIO_SetPinChannel(PIN_PWM2B, PIN_PWM2B_CH);

    GPIO_SetPinChannel(PIN_PWM1A, PIN_PWM1A_CH);
    GPIO_SetPinChannel(PIN_PWM1B, PIN_PWM1B_CH);

    GPIO_SetPinChannel(PIN_PWM0A, PIN_PWM0A_CH);
    GPIO_SetPinChannel(PIN_PWM0B, PIN_PWM0B_CH);

    /* Init PWM on both channel A and channel B */
    PWM_ComplementaryPairChannelInit(PWM_0, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM_1, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM_2, PWM_FREQ, PWM_DB_NS);
    PWM_ActionQualifierCHA(PWM_2, CAU_SET_LOW|CAD_SET_HIGH);

    /* Disable all SYNCO */
    PWM_SetSyncOutEvent(PWM_0, TBCTL_BIT_SYNCOSEL_DISABLE);
    PWM_SetSyncOutEvent(PWM_1, TBCTL_BIT_SYNCOSEL_DISABLE);
    PWM_SetSyncOutEvent(PWM_2, TBCTL_BIT_SYNCOSEL_DISABLE);

    /* Set PWM0A output duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM_0, (u32PWMPeriod * 2) / 3);
    PWM_SetCMPA(PWM_1, (u32PWMPeriod * 2) / 3);
    PWM_SetCMPA(PWM_2, (u32PWMPeriod * 1) / 3);

    /* Enable PWM synchronization */
    PWM_EnableSync(PWM_0);
    PWM_EnableSync(PWM_1);
    PWM_EnableSync(PWM_2);

    /* Set PWMx counting up after SYNC */
    PWM_SetCounterDirAfterSync(PWM_0, COUNT_UP);
    PWM_SetCounterDirAfterSync(PWM_1, COUNT_UP);
    PWM_SetCounterDirAfterSync(PWM_2, COUNT_UP);

    /*
     * Set the value will be loaded to TBCNT after SYNCI signal.
     */
    PWM_SetSyncReloadValue(PWM_0, (PWM0)->CMPA.all);
    PWM_SetSyncReloadValue(PWM_1, 0);
    PWM_SetSyncReloadValue(PWM_2, (PWM2)->CMPA.all);

    /*Start counting*/
    PWM_RunCounter(PWM_2);
    PWM_RunCounter(PWM_1);
```

```
    PWM_RunCounter(PWM_0);

    /* Assert software SYNC on PWM1/2/0 */
    PWM_ForceSync(INC_PWM0 | INC_PWM1 | INC_PWM2);

    while (1)
    {

    }
}
```

2.5 PWM 封波

在很多应用场景里头，存在类似紧急停车信号，当这个信号来到时候，需要立即或者安全的停止一切动作。如果把 PWM 的输出看成这种动作或者动作的驱动信号，同样有着立即或者安全停止的需求。

- 周期性封锁（Cycle-by-Cycle，CBC）PWM 输出，会在下一个 PWM 周期重新启动（适用于电源控制中的定电流控制，或是步进电机中的恒流细分控制；
- 一次性封锁（One Shot，OST）PWM 输出，One-shot 则是会直接停止 PWM 输出，直到使用者介入，清除 one-shot flag 之后才重新输出波形；

图 2-8：周期性封锁信号

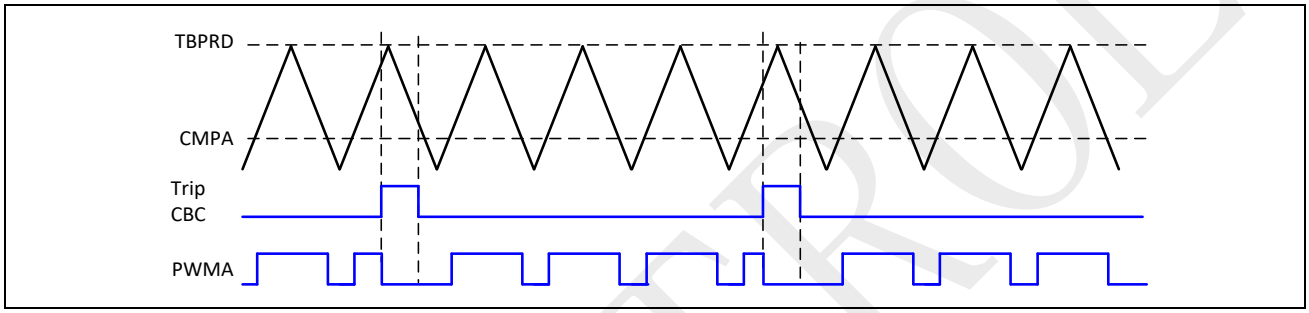
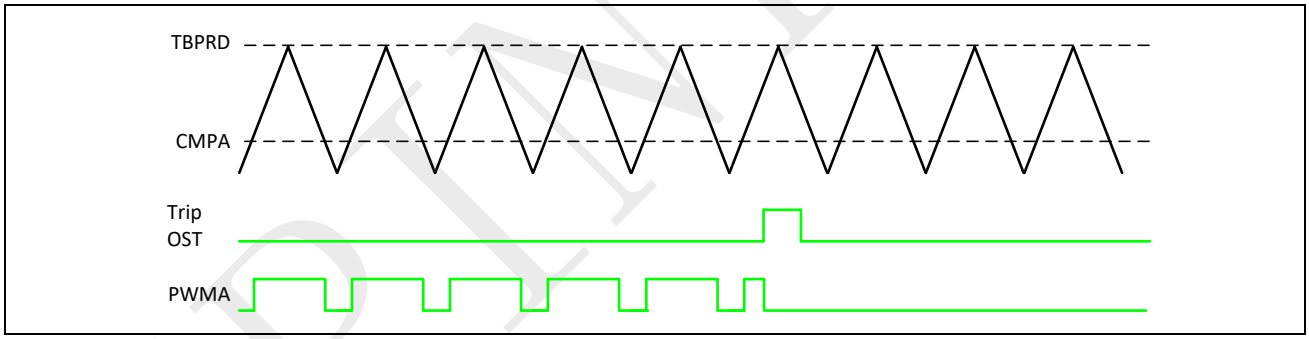


图 2-9：一次性封锁信号



PWM 封锁输入信号源有：COMP，ADCPPU，GPIO(TZ0~TZ4)。

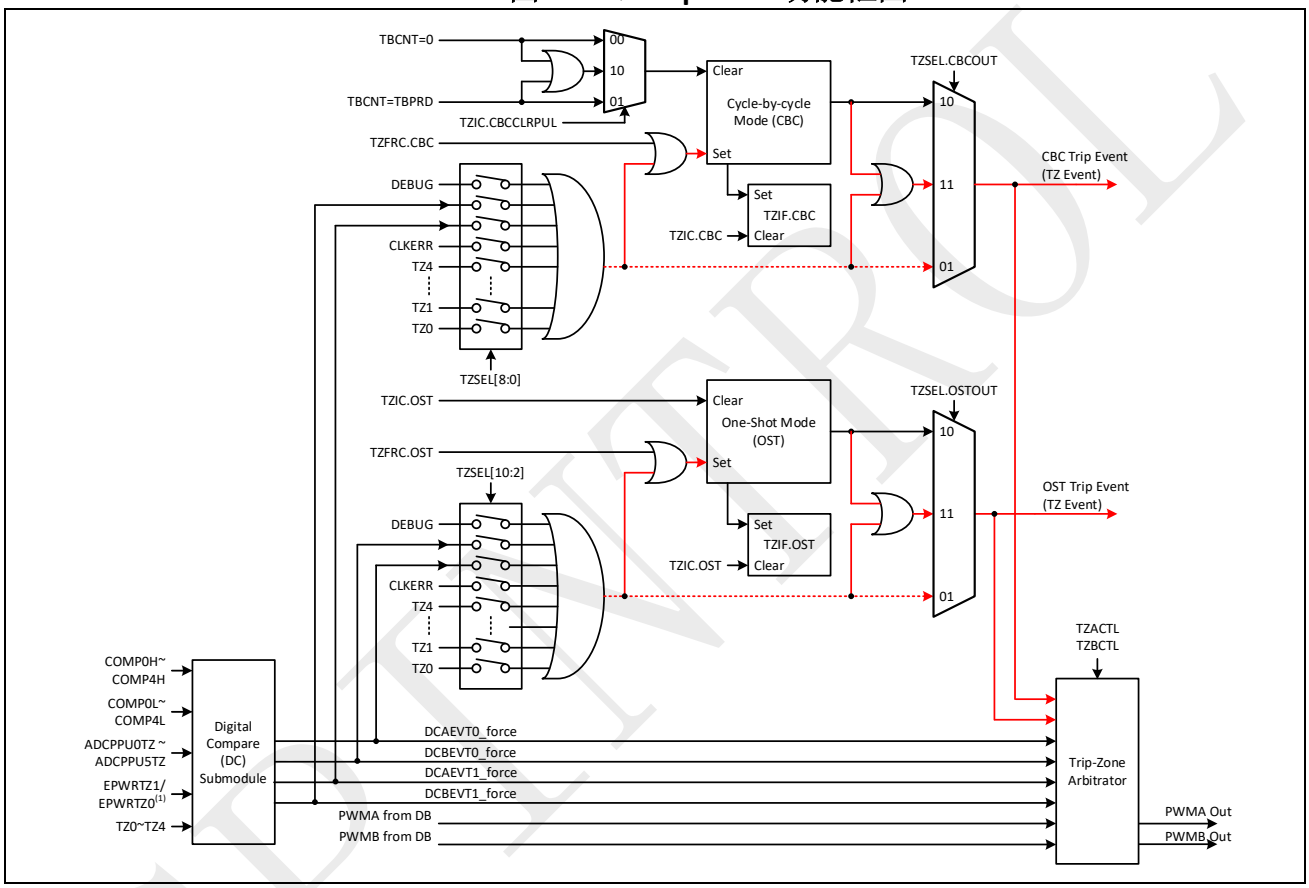
周期性和一次性封锁功能可以同时使能，如果两种信号同时产生，则一次性封锁的信号将覆盖周期性封锁信号，即一次性封锁事件优先级高于周期性封锁事件。

2.5.1 GPIO 触发封波事件

TZ0~TZ4 为来自 GPIO 的触发信号，如图 2-10 中绿色信号部分所示，既可以将 TZ0~TZ4 送到 CBC 或 OST 通路，从而产生 TZ 事件，也可以如图 2-10 蓝色信号部分所示，将 TZ0~TZ4 送到 PWM 的 DC 子模块，将其转换为 DCAEVT0_force、DCBEVT0_force、DCAEVT1_force、DCBEVT1_force 事件之后，再送到 CBC 或 OST 通路产生 TZ 事件。

通常使用第一种，因为其路径较短，如图 2-10 中红色信号线所示。

图 2-10：Trip Zone 功能框图



代码实现如下所示：

Example Code

```
/* This macro is used to get the PWM period with a specified frequency */
#define             PWMPeriod(u32PWMFreqHz)
((CLOCK_GetModuleClock(PWM_MODULE))/u32PWMFreqHz)/2;

#define             PWM_FREQ                        20000
/* 20kHz PWM */

#define             PWM_Sel             PWM2
#define             PIN_PWMB            GPIO_39
#define             PIN_PWMB_CH          GPIO39_PWM2B
#define             PWMTZ_IRQn_Sel        PWM2TZ_IRQn
#define             PIN_TZ0             GPIO_0
#define             PIN_TZ1             GPIO_1
#define             PIN_TZ2             GPIO_4

uint32_t            u32PWMPeriod;

int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* Init PWM_Sel and output 20kHz waveform on channel B */
    PWM_SingleChannelInit(PWM_Sel, PWM_CHB, PWM_FREQ);

    /* Configure PWM counter as up counting mode */
    PWM_SetCounterMode(PWM_Sel, COUNT_DOWN);

    /*
     * The PWM counting mode is set to be up-down by default, as a result, we
need
     * to re-config the AQCTLA register, which is used for controlling the PWM
output
     * action when ZRO/CAU/PRD/CAD (for more information, please read the PWM
application
     * note) event had happened.
     *
```

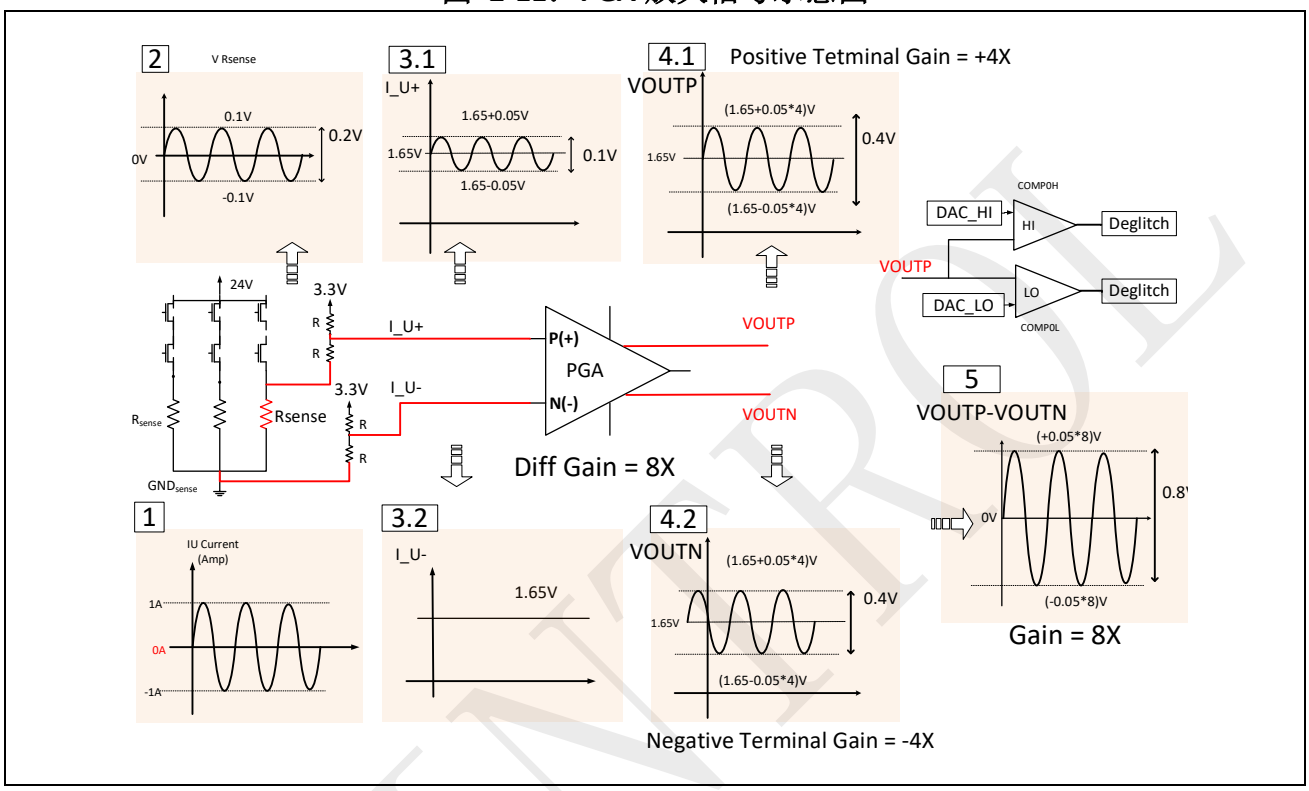
```
     * Key Point: the PWM freq will be double because of the ZRO/CAU/PRD/CAD
event.
     */
    PWM_ActionQualifierCHB(PWM_Sel, AQCTLB_ALL_ZRO_SET_HIGH
                         | AQCTLB_ALL_CAU_DO_NOTHING
                         | AQCTLB_ALL_PRD_DO_NOTHING
                         | AQCTLB_ALL_CAD_SET_LOW);

    /* Set PWM_SelB output 50% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM_Sel, u32PWMPeriod / 2);

    /* Select the channel B output of PWM_Sel */
    GPIO_SetPinChannel(PIN_PWMB, PIN_PWMB_CH);

    /* Start PWM_Sel */
    PWM_RunCounter(PWM_Sel);

    /* Trigger TZ0 event when GPIO is low */
    PWM_SetTZ0FromGPIO(PIN_TZ0, GPIO_LEVEL_LOW);

    /* Trigger TZ1 event when GPIO is low */
    PWM_SetTZ1FromGPIO(PIN_TZ1, GPIO_LEVEL_LOW);

    /* Trigger TZ2 event when GPIO is low */
    PWM_SetTZ2FromGPIO(PIN_TZ2, GPIO_LEVEL_LOW);

    /*
     * Set TZ0 as one-shot trip event. The one-shot mode means once the
     * corresponding event has happened, the PWM waveform will stop, and
     * will not start until the flag is cleared.
     *
     * The key poiny there is 'TZEVT_ASYNC_ONLY_FOR_DEBUG' can not be
     * set in practical engineering, it can only be used in DEBUG mode.
     */
    PWM_SetOneShotTripEvent(PWM_Sel, TRIP_EVENT_TZ0,
TRIP_OUTPUT_ASYNC_OR_LATCH);

    /*
     * Set TZ1 and TZ2 as CBC trip event. The symbol 'CBC' means once the
     * corresponding event has happened, the PWM waveform will stop, but
     * start again at the next PWM CLK period with do nothing manually.
     *
     * The key point there is the same as 'PWM_SetOneShotTripEvent()'
     */
    PWM_SetCBCTripEvent(PWM_Sel, TRIP_EVENT_TZ1 | TRIP_EVENT_TZ2,
TRIP_OUTPUT_ASYNC_OR_LATCH);

    /*
     * Set PWM0 output as tristate upon one-shot and CBC trip event,
     * need to explicitly specify actions for all 6 trip scenarios
     */
    PWM_SetCHAOutputWhenTrip(PWM_Sel, TZU_TRIP_AS_TRI_STATE |
                           TZD_TRIP_AS_TRI_STATE |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHBOutputWhenTrip(PWM_Sel, TZU_TRIP_AS_TRI_STATE |
                           TZD_TRIP_AS_TRI_STATE |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
```

```
                            DCEVT1U_TRIP_DO_NOTHING |
                            DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableCBCTripInt(PWM_Sel);
    PWM_EnableOneShotTripInt(PWM_Sel);

    NVIC_EnableIRQ(PWMTZ_IRQn_Sel);

    while (1)
    {
        Delay_Ms(500);

        /* Restore the output of wave in oneshot mode*/
        PWM_ClearOneShotTripInt(PWM_Sel);
    }
}

void PWM2TZ_IRQHandler(void)
{
    uint32_t u32TZStatus = PWM2->TZSTS.all;

    if (u32TZStatus & TZSTS_ALL_TZ0OST_OCCUR)
    {
        printf("TZ0 one-shot trip event occurred\n");

        PWM2->TZSTSCLR.bit.TZ0OST = 1;
    }

    if (u32TZStatus & TZSTS_ALL_TZ1CBC_OCCUR)
    {
        printf("TZ1 cycle-by-cycle trip event occurred\n");

        PWM2->TZSTSCLR.bit.TZ1CBC = 1;
    }

    if (u32TZStatus & TZSTS_ALL_TZ2CBC_OCCUR)
    {
        printf("TZ2 cycle-by-cycle trip event occurred\n");

        PWM2->TZSTSCLR.bit.TZ2CBC = 1;
    }

    PWM_ClearTripGlobalInt(PWM2);
}
```

2.5.2 COMP 触发封波事件

COMP 可作为电流防护使用，以下代码以 PWMU 电流为例，当 PGAP 输出超过 DAC_HI 时，COMP0H 触动 PWM 输出停止，当 PGAP 输出电压小于 DAC_LO，由 COMP0L 触发 PWM 停止。

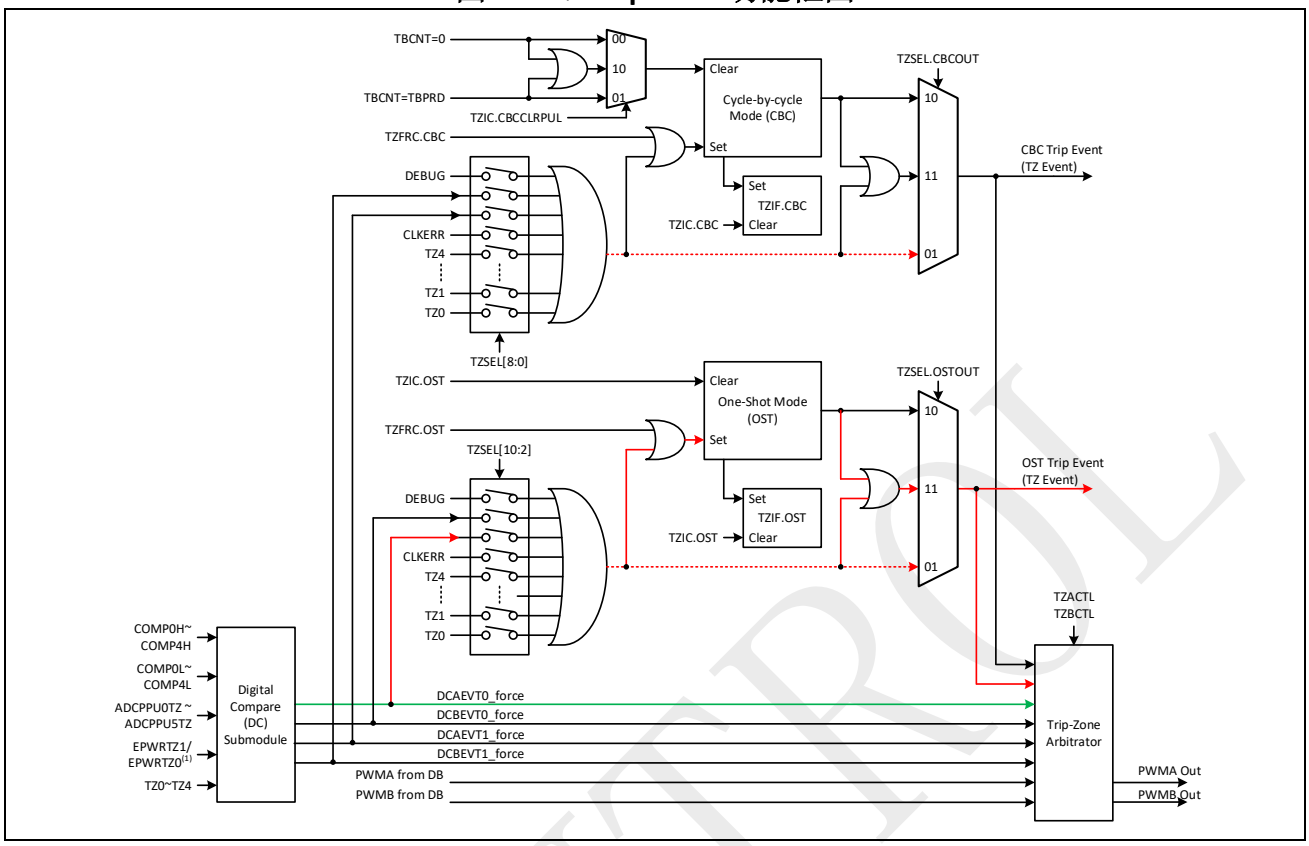
图 2-11：PGA 放大信号示意图



COMP0H 或 COMP0L 可以通过 DC 模块转换为 DCAEVT0_force，DCBEVT0_force，DCAEVT1_force，DCBEVT1_force，再通过 CBC 或 OST 产生 TZ 事件，如图 2-12 所示。

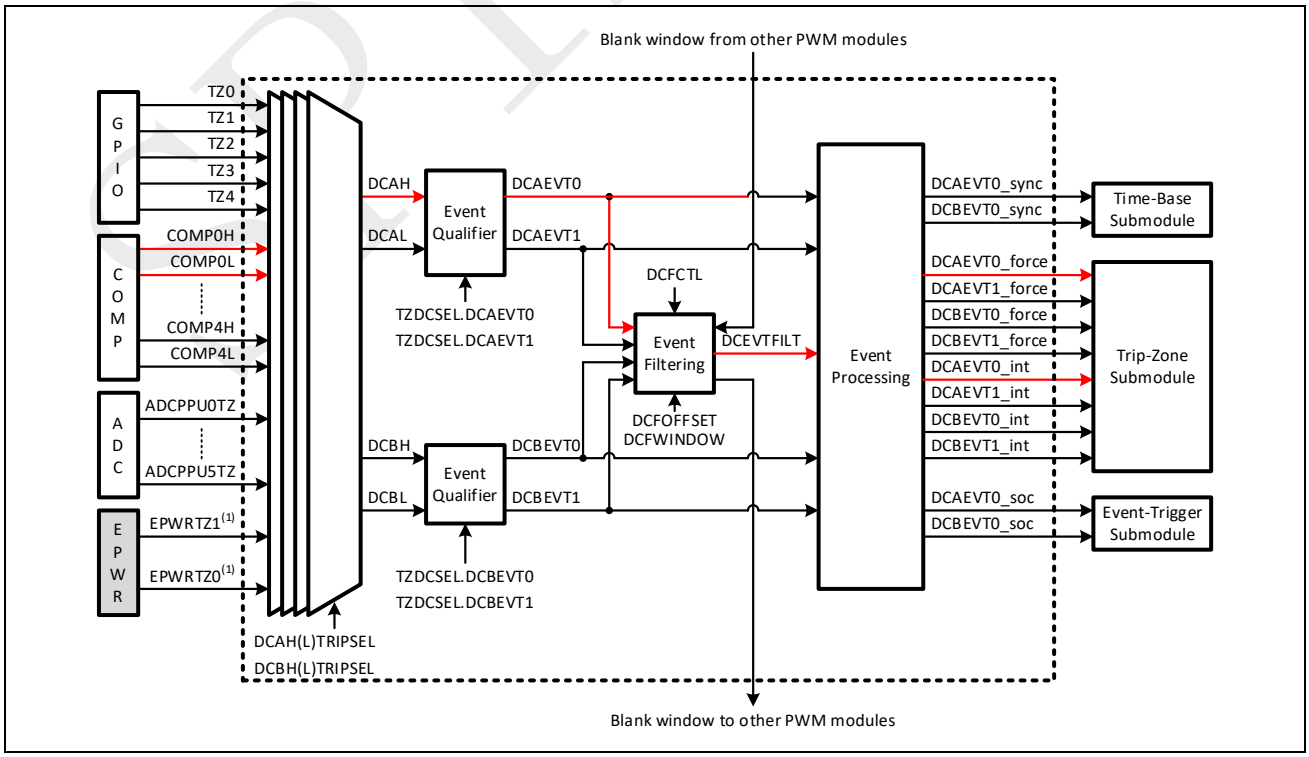
需要特别强调的是，在使用该路信号时，要将在 TZACTL 及 TZBCTL 寄存器中，将 DCAEVT0_force、DCBEVT0_force、DCAEVT1_force、DCBEVT1_force 事件的相关控制位设置为 DO_NOTHING，否则 DCAEVT0_force、DCBEVT0_force、DCAEVT1_force、DCBEVT1_force 事件信号将不经过 CBC 或 OST，直接送达 Trip-Zone Arbitrator 模块，如图 2-12 中绿色信号线所示。

图 2-12：Trip Zone 功能框图



其中在 DC 模块中的信号流如图 2-13 所示，示例代码通过 Event Filtering 模块对 DCAEVT0 信号进行过滤，避免了 Trip Zone 误触发生。

图 2-13：DC 功能框图



代码实现如下所示：

Example Code

```
    #define             PWM_U                       PWM5
    #define             PWM_V                       PWM1
    #define             PWM_W                       PWM2
    #define             LINK_PWM_U                  SEL_PWM5
    #define             GPIO_PWM_U_H                GPIO_28
    #define             GPIO_PWM_V_H                GPIO_20
    #define             GPIO_PWM_W_H                GPIO_22
    #define             GPIO_PWM_U_H_SEL            GPIO28_PWM5A
    #define             GPIO_PWM_V_H_SEL            GPIO20_PWM1A
    #define             GPIO_PWM_W_H_SEL            GPIO22_PWM2A
    #define             GPIO_PWM_U_L                GPIO_29
    #define             GPIO_PWM_V_L                GPIO_21
    #define             GPIO_PWM_W_L                GPIO_23
    #define             GPIO_PWM_U_L_SEL            GPIO29_PWM5B
    #define             GPIO_PWM_V_L_SEL            GPIO21_PWM1B
    #define             GPIO_PWM_W_L_SEL            GPIO23_PWM2B
    #define             PGA0_CH_P                   PGA0_CH_P_ADC8
    #define             PGA0_P_PIN                  GPIO_8
    #define             PGA1_CH_P                   PGA1_CH_P_ADC10
    #define             PGA1_P_PIN                  GPIO_10
    #define             PGA2_CH_P                   PGA2_CH_P_ADC12
    #define             PGA2_P_PIN                  GPIO_12


#define             PWM_FREQ            20000                       /* 20kHz
PWM */
#define             PWM_DB_NS           1000                        /*
1000ns */
#define             PWM_CMPA            2500                        /* the
CPMA value for all three */

#define             PWM_BlankWIN_Size    100                         /* 100
TBCNT CLK */
#define             PWM_Blank_Offset     50                          /* 50
TBCNT CLK */
#define             COMP_FilterWIN       300                         /*
300ns */
#define             SampleRgisterNVol    1650                        /*
Sampler register negative voltage : 1650mV */
#define             VolOffsetmV          550                         /* COMP
monitor voltage range offset */

void PWMx_Set_Digiter_Compare(PWM_REGS *PWMx)
{
    /* Affected by results monitored from all three phases */
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0H |
                        DC_TRIP_COMP1H |
                        DC_TRIP_COMP2H |
                        DC_TRIP_COMP0L |
                        DC_TRIP_COMP1L |
                        DC_TRIP_COMP2L);

    PWM_SetRawDCAEVT0(PWMx, DCH_HIGH_DCL_X);

    /* Set the filtered event as the final DCAEVT0, and trigger TZ sb-model when
DCH is high */
    PWM_SetDCAEVT0(PWMx, DCEVT_FILTERED);
```

```
    /*
     * 1.Configure DC event filter
     *
     * 2.Input is DCAEVT0, apply blank window with original polarity
     *
     * 3.Set the blank window position relative to TBCNT=0
     */
    PWM_SetDCFilter(PWMx, DCF_FROM_RAW_DCAEVT0, DCF_ALIGN_ON_ZERO);

    /* Set blank window size as 100 TBCLK and offset as 50 TBCLK */
    PWM_SetDCFilterBlankWindow(PWMx, PWM_BlankWIN_Size, PWM_Blank_Offset);

    /* Enable PWM DC filter blank function */
    PWM_EnableDCFilterBlank(PWMx);

    PWM_SetOneShotTripEvent(PWMx, TRIP_EVENT_DCAEVT, TRIP_OUTPUT_LATCH);

    /* Set output to tri-state upon DCAEVT0 trip event */
    PWM_SetCHAOutputWhenTrip(PWMx, TZU_TRIP_AS_TRI_STATE |
                             TZD_TRIP_AS_TRI_STATE   |
                             DCEVT0U_TRIP_DO_NOTHING |
                             DCEVT0D_TRIP_DO_NOTHING |
                             DCEVT1U_TRIP_DO_NOTHING |
                             DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHBOutputWhenTrip(PWMx, TZU_TRIP_AS_TRI_STATE |
                             TZD_TRIP_AS_TRI_STATE   |
                             DCEVT0U_TRIP_DO_NOTHING |
                             DCEVT0D_TRIP_DO_NOTHING |
                             DCEVT1U_TRIP_DO_NOTHING |
                             DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableDCAEVT0TripInt(PWMx);
}

/*
 * This function output Trip-Phase waveform for BLDC(Brushless Derect Current)
 */
void PWM_Output_Trip_Phase(void)
{
    /*
     * Init PWM5/1/2 and output 20kHz waveform on both channel A and channel B.
     */
    PWM_ComplementaryPairChannelInit(PWM_U, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM_V, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM_W, PWM_FREQ, PWM_DB_NS);

    /* Start counting */
    PWM_RunCounter(PWM_U);
    PWM_RunCounter(PWM_V);
    PWM_RunCounter(PWM_W);

    /* PWM5 set CMPA and loading the same value to PWM1/2 synchronously */
    PWM_LinkCMPA(PWM_V, LINK_PWM_U);
    PWM_LinkCMPA(PWM_W, LINK_PWM_U);
    PWM_SetCMPA(PWM_U, PWM_CMPA);
}

int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
```

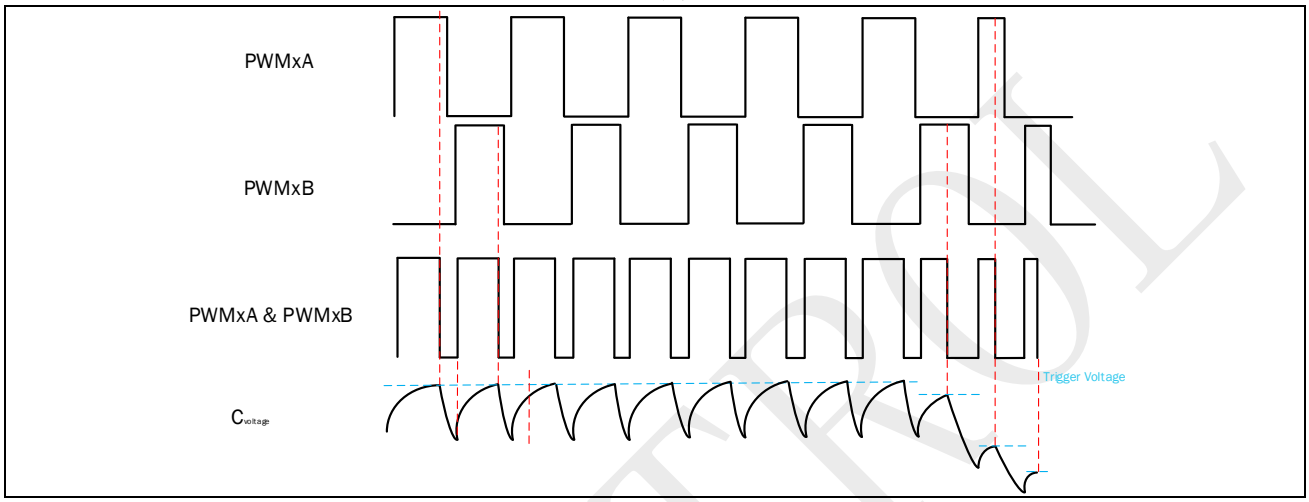
```
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /*
     * As the title described, we suppose GPIO8/10/12(ADC8/10/12) can be used to
sample the voltage
     * of current sampling resistor in the BLDC circuit. so, we need to set the
GPIOs as ADCs.
     */
    GPIO_SetPinAsAnalog(PGA0_P_PIN);
    GPIO_SetPinAsAnalog(PGA1_P_PIN);
    GPIO_SetPinAsAnalog(PGA2_P_PIN);

    /*
     * Set PGA in differential ended mode.
     *
     * 1.Set the GPIO8(ADC8)/GPIO10(ADC10)/GPIO12(ADC12) as the P-input of the
PGA0/PGA1/PGA2.
     *
     * 2.Set the DAC1 as the N-input of the PGA0/PGA1/PGA2.
     *
     * 3.Set the PGA0/PGA1/PGA2 amplitude scale as 4x.
     *
     * 4.Enable the PGA0/PGA1/PGA2.
     */
    PGA_DifferentialInit(PGA0, PGA0_CH_P, PGA0_CH_N_DAC1, PGA_SCALE_8X);
    PGA_DifferentialInit(PGA1, PGA1_CH_P, PGA1_CH_N_DAC1, PGA_SCALE_8X);
    PGA_DifferentialInit(PGA2, PGA2_CH_P, PGA2_CH_N_DAC1, PGA_SCALE_8X);

    /* Set DAC1 output 1.65V */
    COMP_SetDACVoltage(DAC1, SampleRgisterNVol);
    COMP_EnableDAC(DAC1);

    /*
     * 1. Initialize comparator with 300ns deglitch filtering window and set
over
     *    current value as 2.2V.
     * 2. Set DAC 2 & 3 as the comparator too-High trigger input and too-low
trigger input.
     * 3. Set DAC 2 (too-high trigger input) output 1.65V + 0.55V = 2.2V
     */
    COMP_Init(COMP_0_HI, COMP0_FROM_PGA0P_OUT, SampleRgisterNVol + VolOffsetmV,
COMP_FilterWIN);
    COMP_Init(COMP_0_LO, COMP0_FROM_PGA0P_OUT, SampleRgisterNVol - VolOffsetmV,
COMP_FilterWIN);
```

```
    COMP_Init(COMP_1_HI, COMP1_FROM_PGA1P_OUT, SampleRgisterNVol + VolOffsetmV,
COMP_FilterWIN);
    COMP_Init(COMP_1_LO, COMP1_FROM_PGA1P_OUT, SampleRgisterNVol - VolOffsetmV,
COMP_FilterWIN);
    COMP_Init(COMP_2_HI, COMP2_FROM_PGA2P_OUT, SampleRgisterNVol + VolOffsetmV,
COMP_FilterWIN);
    COMP_Init(COMP_2_LO, COMP2_FROM_PGA2P_OUT, SampleRgisterNVol - VolOffsetmV,
COMP_FilterWIN);

    /* Select GPIO28/29 as the channel A/B output of PWM5 respectively */
    GPIO_SetPinChannel(GPIO_PWM_U_H, GPIO_PWM_U_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_U_L, GPIO_PWM_U_L_SEL);

    /* Select GPIO20/21 as the channel A/B output of PWM1 respectively */
    GPIO_SetPinChannel(GPIO_PWM_V_H, GPIO_PWM_V_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_V_L, GPIO_PWM_V_L_SEL);

    /* Select GPIO22/23 as the channel A/B output of PWM2 respectively */
    GPIO_SetPinChannel(GPIO_PWM_W_H, GPIO_PWM_W_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_W_L, GPIO_PWM_W_L_SEL);

    /* Generate a tri-phase waveform */
    PWM_Output_Trip_Phase();

    /* Set the DC sub-module for PWM_U */
    PWMx_Set_Digiter_Compare(PWM_U);

    /* Set the DC sub-module for PWM_V */
    PWMx_Set_Digiter_Compare(PWM_V);

    /* Set the DC sub-module for PWM_W */
    PWMx_Set_Digiter_Compare(PWM_W);


    NVIC_EnableIRQ(PWM5TZ_IRQn);

    while (1)
    {
        Delay_Ms(500);

        /* Restore the output of wave in oneshot mode*/
        PWM_ClearOneShotTripInt(PWM_U);
        PWM_ClearOneShotTripInt(PWM_V);
        PWM_ClearOneShotTripInt(PWM_W);
    }
}

void PWM5TZ_IRQHandler(void)
{
    uint32_t u32TZStatus = PWM5->TZSTS.all;

    if (u32TZStatus & TZSTS_ALL_DCAEVT0_OCCUR)
    {
        printf("Digital compare A event 0 one-shot trip event occurred\n");

        PWM5->TZSTSCLR.bit.DCAEVT0 = 1;
    }

    PWM_ClearTripGlobalInt(PWM5);
}
```

2.5.3 电焊机示例

在电焊机领域，通常要求一个 PWM 能产生一对带死区的互补波形，之后会将两路波形通过板级电路整合成一个波，如图 2-14 所示。在合成波的高电平期间，会给目标电容 $C_{voltage}$ 充电，使用 COMP 对目标电容 $C_{voltage}$ 的电压进行比较，当充电电压处于图中蓝色线时，COMP 将触发 PWM 封波信号，对应的 PWM 波将被拉低，电容 $C_{voltage}$ 开始放电，电压降低，在随后的合成波高电平时重新充电。

图 2-14：电焊机波形需求

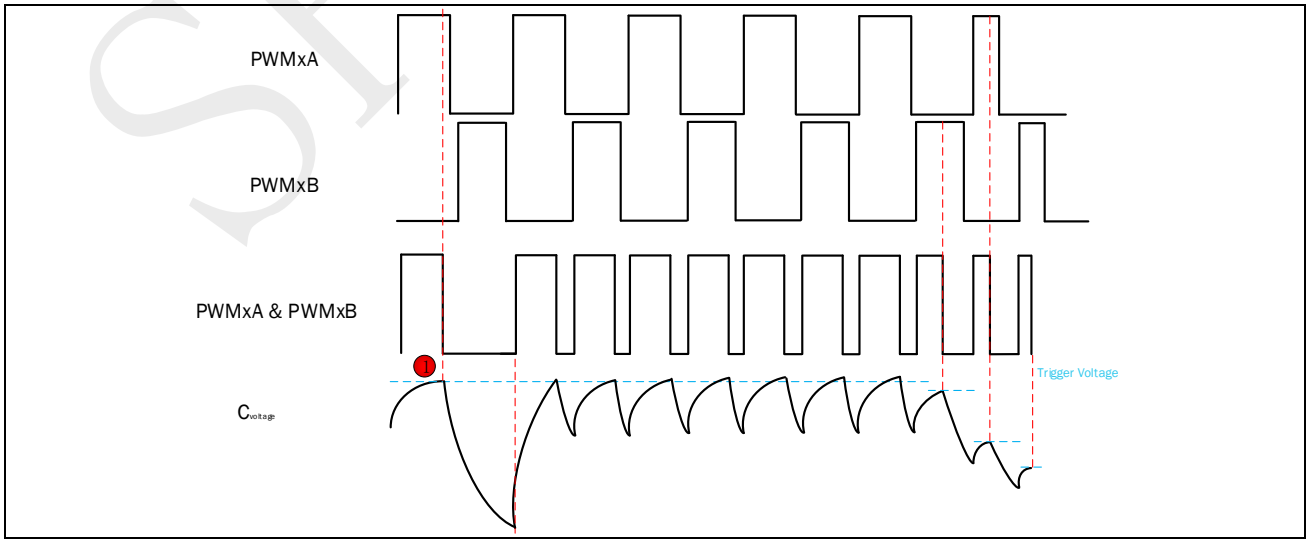


在本实例中，有两个条件需要满足：

- 一对带死区的互补波形；
- 采用 COMP 触发 PWM 封波；

由于这两个条件需要同时满足，使用一个 PWM 输出两路带死区的互补波形不可行，其原因在于：如图 2-15 所示，若合成波形来自于同一个 PWM，在 1 时刻产生封波事件，这将会导致 PWM 的两路波形都被封锁，从合成波的角度来看，这个封锁信号同时对两个脉冲产生了影响，这在应用中不可接受。

图 2-15：使用一个 PWM 产生带死区互补波形



所以在这种情况下，只能采用两个 PWM，分别产生单路波形，且这连个 PWM 的单路波形，相互之间是带死区且互补的波形，这样才能满足实际的需求。

代码实现如下所示：

Example Code

```
#define             PWM_BlankWIN_Size      100                          /* 100
TBCNT CLK */
#define             PWM_Blank_Offset       50                           /* 50
TBCNT CLK */
#define             COMP_FilterWIN         300                          /*
300ns */
/* This macro is used to get the PWM period with a specified frequency */
#define             PWMPeriod(u32PWMFreqHz)
((CLOCK_GetModuleClock(PWM_MODULE))/u32PWMFreqHz)/2;
#define             PWM_FREQ                        20000                /* 20kHz
PWM */

uint32_t          u32PWMPeriod;
uint32_t          u32Duty = 75;

void PWMX_Set_Digiter_Compare(PWM_REGS *PWMx)
{
    /* Set DCAH from COMP0H */
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0H);

    PWM_SetRawDCAEVT1(PWMx, DCH_HIGH_DCL_X);

    /* Set the filtered event as the final DCAEVT1, and trigger TZ sb-model when
DCH is high */
    PWM_SetDCAEVT1(PWMx, DCEVT_FILTERED);

    /*
     * 1.Configure DC event filter
     *
     * 2.Input is DCAEVT1, apply blank window with original polarity
     *
     * 3.Set the blank window position relative to TBCNT=0
     */
    PWM_SetDCFilter(PWMx, DCF_FROM_RAW_DCAEVT1, DCF_ALIGN_ON_ZERO);

    /* Set blank window size as 100 TBCLK and offset as 50 TBCLK */
    PWM_SetDCFilterBlankWindow(PWMx, PWM_BlankWIN_Size, PWM_Blank_Offset);

    /* Enable PWM DC filter blank function */
    PWM_EnableDCFilterBlank(PWMx);

    PWM_SetCBCTripEvent(PWMx, TRIP_EVENT_DCAEVT, TRIP_OUTPUT_LATCH);
}


int main()
{
    FLASH_WALLOW();

#if defined(SPC1158)
    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_100MHZ);
#else
```

```
    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
#endif

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* Select output of PWM0 respectively */
    GPIO_SetPinChannel(GPIO_18, GPIO18_PWM0A);

    /* Select output of PWM1 respectively */
    GPIO_SetPinChannel(GPIO_20, GPIO20_PWM1A);

    /* Init PWM on both channel A */
    PWM_SingleChannelInit(PWM0, PWM_CHA, PWM_FREQ);
    PWM_SingleChannelInit(PWM1, PWM_CHA, PWM_FREQ);

    /* Set PWMxA output duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM0, ((u32PWMPeriod * u32Duty) / 100));
    PWM_SetCMPA(PWM1, ((u32PWMPeriod * (100 - u32Duty)) / 100));

    PWM0->AQCTLA.all  =  AQCTLA_ALL_ZRO_SET_HIGH    /* set high when CNT=Zero */
                       | AQCTLA_ALL_CAU_SET_LOW;     /* set low when TBCNT=CMPA and
TBCNT is counting up */

    PWM1->AQCTLA.all  =  AQCTLA_ALL_PRD_SET_HIGH    /* set high when TBCNT=TBPRD
*/
                       | AQCTLA_ALL_CAD_SET_LOW;  /* set low when TBCNT=CMPA and
TBCNT is counting down */

    /* Use the SYNCO to Sync phase */
    PWM_SetSyncOutEvent(PWM0, TBCTL_BIT_SYNCOSEL_TBCNT_EQU_ZERO);
    PWM_SetSyncOutEvent(PWM1, TBCTL_BIT_SYNCOSEL_DISABLE);

    /* Enable PWM synchronization */
    PWM_EnableSync(PWM1);

    /* Set PWMx counting up after SYNC */
    PWM_SetCounterDirAfterSync(PWM1, COUNT_UP);

    /* Set the re-load value when SYNC signal happened */
    PWM_SetSyncReloadValue(PWM1, 0);

    /* Start counting */
    PWM_RunCounter(PWM0);
    PWM_RunCounter(PWM1);
```

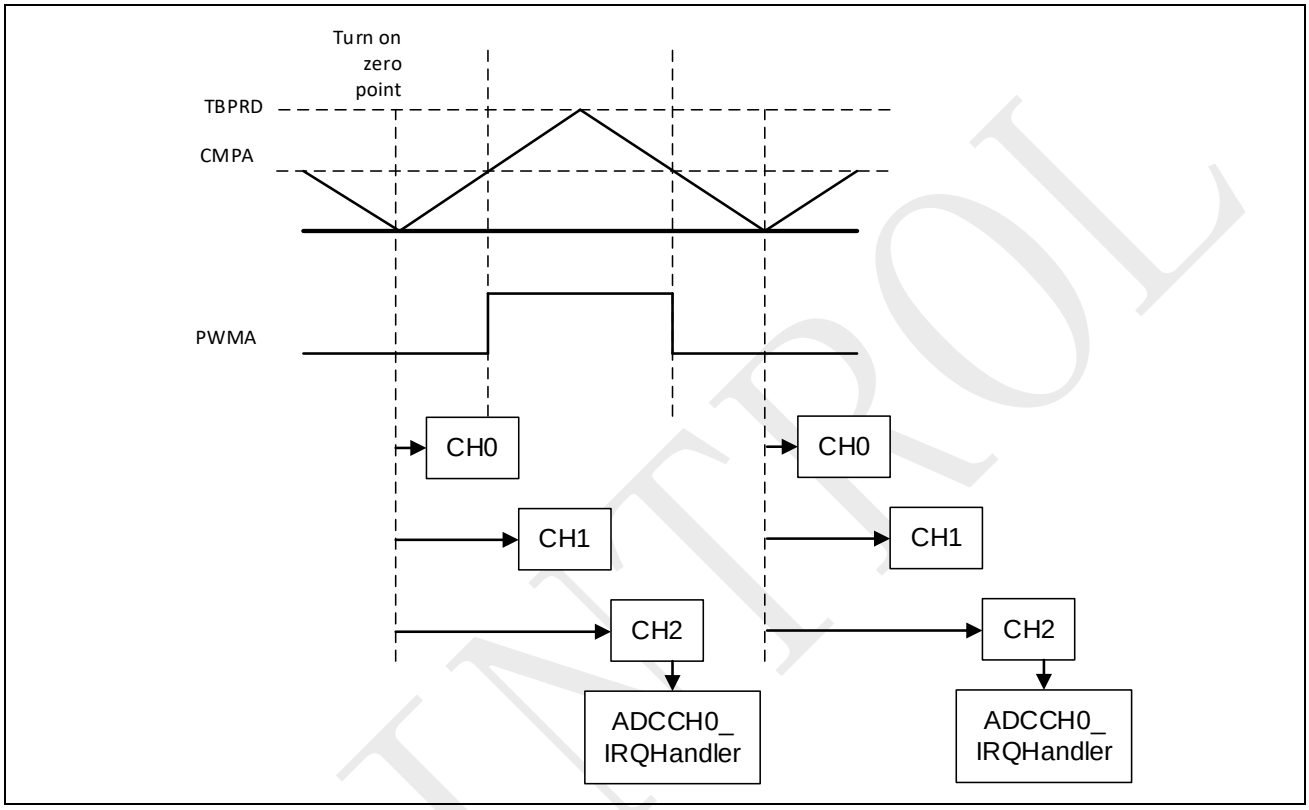
```
    /*
     * As the title described, we suppose GPIO8(ADC8) can be used to sample the
voltage
     * of current sampling resistor in the BLDC circuit. so, we need to set the
GPIOs as ADCs.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    COMP_Init(COMP_0_HI, COMP0_FROM_ADC8, 3000, COMP_FilterWIN);

    /* Set the DC sub-module */
    PWMX_Set_Digiter_Compare(PWM0);
    PWMX_Set_Digiter_Compare(PWM1);

    /* Set output to tri-state upon TZ trip event */
    PWM_SetCHAOutputWhenTrip(PWM0, TZU_TRIP_AS_LOW |
                             TZD_TRIP_DO_NOTHING |
                             DCEVT0U_TRIP_DO_NOTHING |
                             DCEVT0D_TRIP_DO_NOTHING |
                             DCEVT1U_TRIP_DO_NOTHING |
                             DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHAOutputWhenTrip(PWM1, TZU_TRIP_DO_NOTHING |
                             TZD_TRIP_AS_LOW |
                             DCEVT0U_TRIP_DO_NOTHING |
                             DCEVT0D_TRIP_DO_NOTHING |
                             DCEVT1U_TRIP_DO_NOTHING |
                             DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableDCAEVT1TripInt(PWM0);
    PWM_EnableDCAEVT1TripInt(PWM1);

    NVIC_EnableIRQ(PWM0TZ_IRQn);
    NVIC_EnableIRQ(PWM1TZ_IRQn);
    while (1)
    {

    }
}

void PWM0TZ_IRQHandler(void)
{
    uint32_t u32TZStatus = PWM0->TZSTS.all;

    if (u32TZStatus & TZSTS_ALL_DCAEVT1_OCCUR)
    {
        printf("PWM0 DCAEVT1_OCCUR occurred\n");

        PWM0->TZSTSCLR.bit.DCAEVT1 = 1;
    }

    PWM_ClearTripGlobalInt(PWM0);
}

void PWM1TZ_IRQHandler(void)
{
    uint32_t u32TZStatus = PWM1->TZSTS.all;

    if (u32TZStatus & TZSTS_ALL_DCAEVT1_OCCUR)
    {
        printf("PWM1 DCAEVT1_OCCUR occurred\n");
```

```
        PWM1->TZSTSCLR.bit.DCAEVT1 = 1;
    }

    PWM_ClearTripGlobalInt(PWM1);
}
```

2.6 PWM 触发 ADC 采样

在 PWM 触发三相电流采样的场景里，设定 PWM0 之 TBCNT 过零时触发 ADC，一次同时触发 CH0～CH2 分别负责转换三相电流，当 CH2 转换完之后，进入中断服务程序(ADCCH0_IRQHandler)，获取采样值，如图 2-16 所示。

图 2-16：PWM 触发三相电流采样



代码实现如下所示：

Example Code

```
int main()
{
    FLASH_WALLOW();

    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
```

```
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* Initial Basic Complementary PWM */
    PWM_ComplementaryPairChannelInit(PWM0, PWM_FREQ, PWM_DB_NS);

    /* Set PWM5B output 50% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM0, (u32PWMPeriod * 3) / 4);

    /* Start counting */
    PWM_RunCounter(PWM0);

    /* PWM trigger timing selecting */
    PWM_SetSOCATiming(PWM0, EQU_ZERO);

    /* PWM trigger period selecting */
    PWM_SetSOCAPeriod(PWM0, ON_15TH_EVENT);

    /* Enable PWM SOCA trigger */
    PWM_EnableSOCA(PWM0);

    /*
     * ADC SOC0 Init
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHA_P_ANA_IN8, ADC_SHA_N_GND, SHA_AND_SHB_SHC,
ADCTRIG_PWM0SOCA);

    /* Set SOC0 sample 4 times (2^2) at every convert */
    ADC_SetAverageCnt(ADC_SOC_0, ADCSOCCTL0_BIT_AVGCNT_AVG_4);

    /*
     * ADC SOC1 Init, the configuration of SH, trigger source,
     * averaging times, sampling count, conversion count will be
     * ignored and keep the same as SOC0.
     */
    GPIO_SetPinChannel(GPIO_4, GPIO4_ADC4);
    ADC_Init(ADC_SOC_1, ADC_SHB_P_ANA_IN4, ADC_SHB_N_GND, SHB,
ADCTRIG_PWM0SOCA);

    GPIO_SetPinChannel(GPIO_14, GPIO14_ADC14);
    ADC_Init(ADC_SOC_2, ADC_SHC_P_ANA_IN14, ADC_SHC_N_GND, SHC,
ADCTRIG_PWM0SOCA);

    /* INT service routine configuration */
    NVIC_SetPriority(ADC0_IRQn, 1);
    NVIC_EnableIRQ(ADC0_IRQn);

    PWMCFG->SOCAOCTL.bit.SOCA0EN = SOCAOCTL_BIT_SOCA0EN_ENABLE;
    PWMCFG->SOCAOCTL.bit.POL = SOCAOCTL_BIT_POL_ACTIVE_HIGH;
    PWMCFG->SOCAOCTL.bit.DURATION = SOCAOCTL_ALL_DURATION_32_PWM_CLK;
    GPIO_SetPinChannel(GPIO_8, GPIO8_PWMSOC);

    while (1)
    {

    }
}
```

```
void ADC0_IRQHandler(void)
{
    /* The result of SHA/B/C is stored in RESULT0/1/2 respectively */
    /* Result gotten from 'ADC_GetResult()' can be a negative value or a
positive value*/
    i32result0 = ADC_GetResult(ADC_SOC_0);
    i32result1 = ADC_GetResult(ADC_SOC_1);
    i32result2 = ADC_GetResult(ADC_SOC_2);
    printf("ADC SOC0 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32result0), i32result0);

    printf("ADC SOC1 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32result1), i32result1);

    printf("ADC SOC2 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32result2), i32result2);

    /* Clear SOC0 INT */
    ADC_ClearInt0();

    Delay_Ms(3000);
}
```

2.6.1 观测 PWM 触发 ADC 采样信号

可以将 PWM 触发 ADC 采样信号通过具有 PWMSOC 功能的引脚送出，输出给示波器观测。

Example Code

```
    PWMCFG->SOCAOCTL.bit.SOCA0EN = SOCAOCTL_BIT_SOCA0EN_ENABLE;
    PWMCFG->SOCAOCTL.bit.POL = SOCAOCTL_BIT_POL_ACTIVE_HIGH;
    PWMCFG->SOCAOCTL.bit.DURATION = SOCAOCTL_ALL_DURATION_32_PWM_CLK;
    GPIO_SetPinChannel(GPIO_8, GPIO8_PWMSOC);
```