

### 概述

PWM 在电力电子系统中有着至关重要的作用，广泛地应用在电机控制、开关电源及 UPS 中。SPC1068 中带有 7 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。

---

注意： 本文档主要以 SPC1068 为例进行介绍。

---

# 目录

<b>1</b>	<b>PWM 单元</b> .....	<b>7</b>
1.1	概述 .....	7
1.2	PWM 单元特性 .....	7
<b>2</b>	<b>快速 PWM 波形配置</b> .....	<b>9</b>
2.1	设定单通道独立 PWM.....	9
2.2	设定互补 PWM 对 .....	12
2.3	PWM Trip Zone 功能介绍 .....	14
2.4	外部管脚触发停止 PWM.....	15
2.5	COMP 停止 (Trip) PWM 波形功能介绍 .....	16
<b>3</b>	<b>PWM 产生原理及范例</b> .....	<b>19</b>
3.1	例 1: 产生占空比周期性变化的 PWM 波.....	22
3.2	例 2: 产生两路互补、带死区 PWM 波.....	24
3.3	例 3: 产生两对同步 PWM 波 (互补、带死区) .....	27
3.4	例 4: 外部信号封锁 PWM 输出 (Trip Zone) .....	30
<b>4</b>	<b>PWM 在电机上的应用</b> .....	<b>33</b>
4.1	设置电机三相 PWM 代码.....	33
4.2	驱动电机 PWM 范例: PWM Trigger ADC 功能.....	34

## 图片列表

图 1-1: PWM 单元总览 .....	7
图 1-2: PWM 单元功能框图 .....	8
图 2-1: PWM 单独立通道示例 .....	10
图 2-2: PWM 互补对通道示例 .....	13
图 2-3: Comparator 提供相电流防护示意图 .....	16
图 3-1: PWM 单元的功能框图 .....	19
图 3-2: 先上后下计数, 对称 PWM 输出 .....	21
图 3-3: 先上后下计数, 非对称 PWM 输出 .....	21
图 3-4: PWM 波生成示意图 .....	22
图 3-5: 死区控制单元框图 .....	24
图 3-6: 生成带死区互补 PWM 示意图 .....	25
图 3-7: PWM 同步 (向下计数) .....	27
图 3-8: PWM 同步 (向上计数) .....	27
图 3-9: 多个 PWM 同步信号流 .....	28
图 3-10: 按周期封锁 PWM 信号逻辑 .....	30
图 3-11: 一次性封锁 PWM 信号逻辑 .....	30
图 3-12: Trip Zone 功能框图 .....	31
图 4-1: PWM 互补对通道示例 .....	35

## 表格列表

表 2-1: PWM 单元驱动函数 .....	9
表 2-2: PWM_SingleChannellnit()函数介绍.....	9
表 2-3: PWM_ComplementaryPairChannellnit()函数介绍.....	12
表 2-4: PWM_EnableOneShotTripFromComp()函数介绍 .....	16
表 3-1: 向上计数时各事件优先级 .....	20
表 3-2: 向下计数时各事件优先级 .....	20

SPIN TROL

## 版本历史

版本	日期	作者	状态	变更
C/0	2024-02-26	周佳莉	Released	首次发布。

SPIN TROL

## 术语或缩写

术语或缩写	描述
PWM	Pulse Width Modulation, 脉冲宽度调制

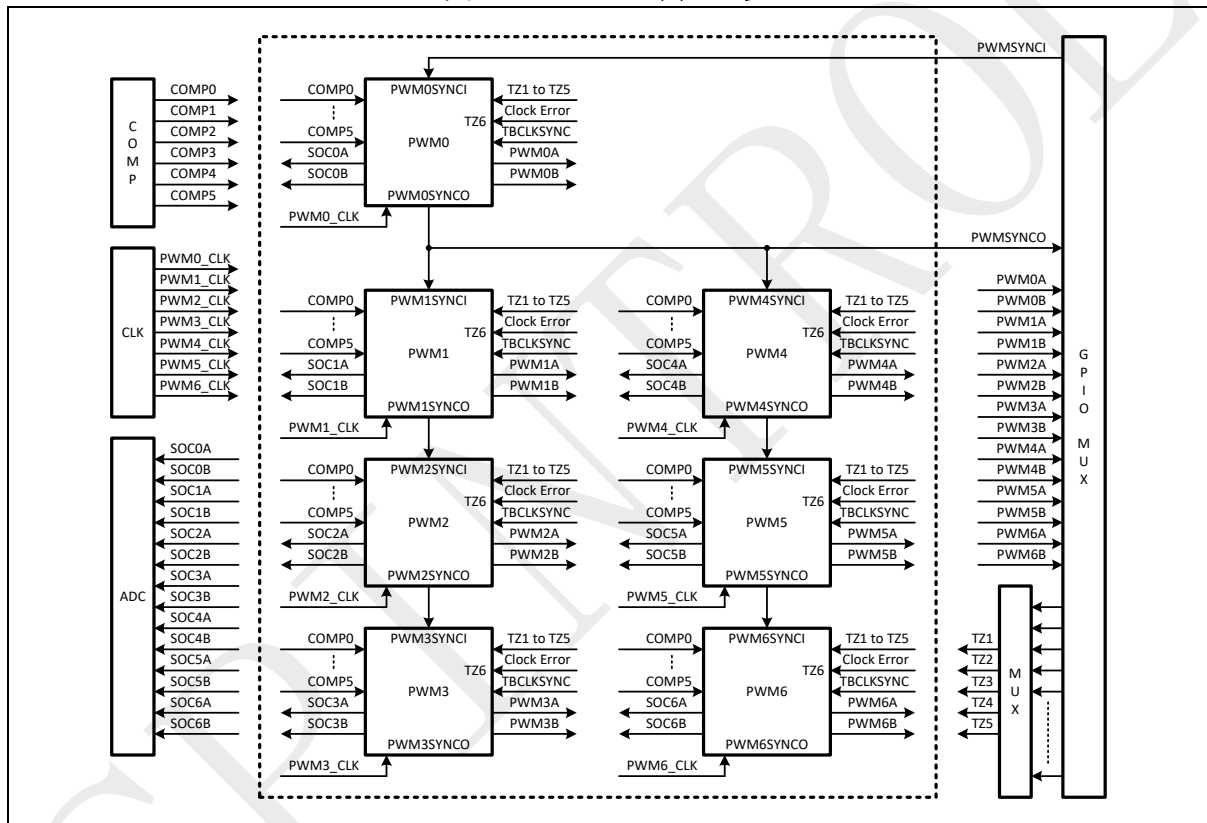
SPIN TROL

# 1 PWM 单元

## 1.1 概述

PWM 在电力电子系统中有着至关重要的作用，广泛地应用在电机控制、开关电源及 UPS 中。SPC1068 中带有 7 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。

图 1-1: PWM 单元总览

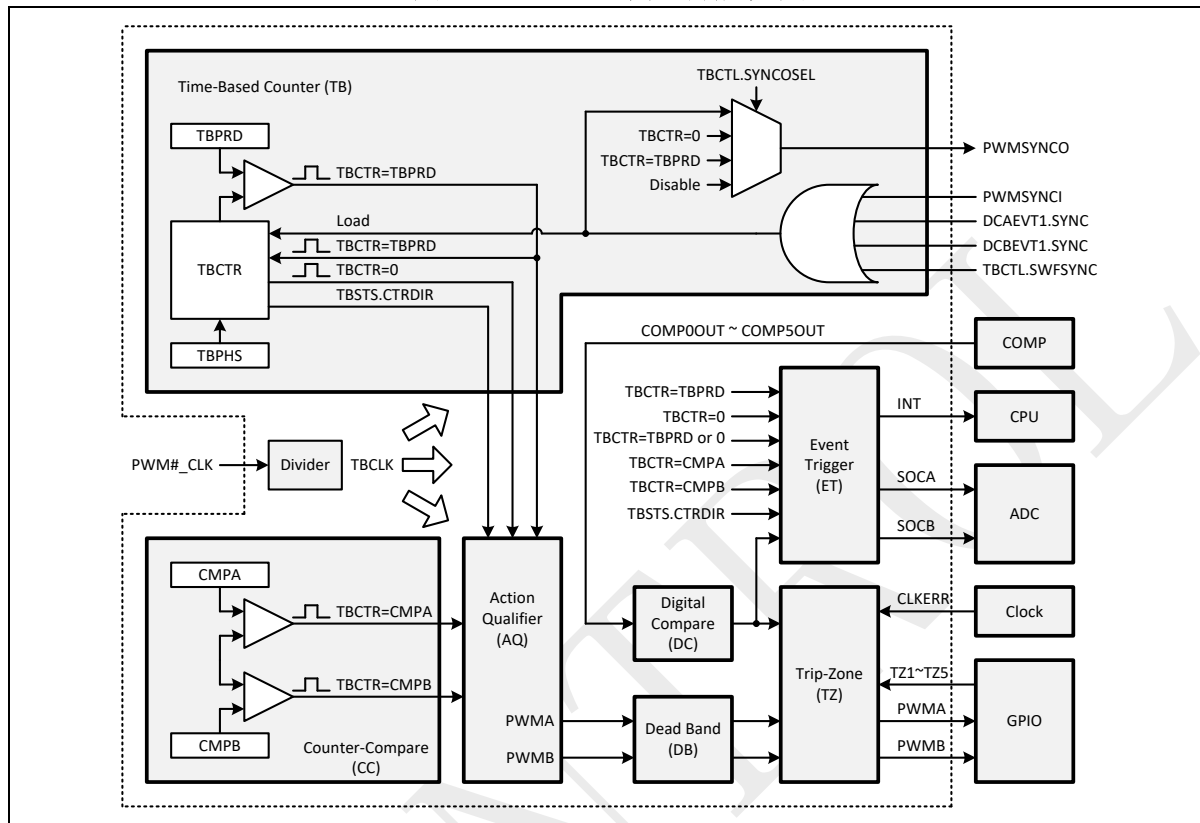


## 1.2 PWM 单元特性

- 七组 PWM 单元，每组提供两个互补 PWM 输出，亦可编程为双独立输出，提供共 14 组 PWM 输出管脚；
- PWM 的 Clock 频率最高可达 150MHz，可提供高精度 PWM 控制；
- 功能强大的 PWM 触发 ADC 功能，所有的 PWM 事件都可以触发 CPU 中断或者触发 ADC 转换（SOC）；
- 可用软件强制 PWM 输出为特定电平；
- 支持 Cycle-by-cycle 关断 PWM，支持 One-shot 连续关断 PWM；

- 内部比较器 Comparator 可关断任意 PWM，只需简单配置。

图 1-2: PWM 单元功能框图





## 2 快速 PWM 波形配置

Spintrol 提供了简易的 PWM 设定函数，可快速协助用户设定 PWM 波形。主要的两个 API 函数如表 2-1 所示。

表 2-1: PWM 单元驱动函数

API 名称	描述
void PWM_SingleChannelInit(PWM_REGS* PWMx, PWM_ChannelEnum ePWM_CH, uint32_t u32PWMPFreqHz)	单个 PWM 信道初始化配置
void PWM_ComplementaryPairChannelInit(PWM_REGS* PWMx, uint32_t u32PWMPFreqHz, uint32_t u32DeadTimeNs)	互补 PWM 输出初始化配置
PWM_SetCMPA(PWMx,u16Val)	设定 PWMxA 的比较值
PWM_SetCMPB(PWMx,u16Val)	设定 PWMxB 的比较值

### 2.1 设定单通道独立 PWM

利用 PWM\_SingleChannelInit(), 用户可以配置独立单信道 PWM 波形，有关 PWM\_SingleChannelInit()的三个输入自变量配置如下表所介绍。

表 2-2: PWM\_SingleChannelInit()函数介绍

PWM_SingleChannelInit(PWMx, ePWM_CH, u32PWMPFreqHz)	
Parameter	Description
PWMx	PWM 单元: PWM0~PWM6
ePWM_CH	指定初始化的是 PWMxA 或是 PWMxB: PWM_CHA 与 PWM_CHB
u32PWMPFreqHz	PWM 载波频率(Hz)

一个 PWM 通道设定建议步骤如下:

1. 务必确认在 main 函数一开始已经调用 Sys\_Init 与 Clock\_Init
2. 调用 PWM\_SingleChannelInit (), 初始化基本波形
3. 启动 PWM clock, 注意 SPC1068 的 PWM counter 分为四种模式
  - a) Counter Stop
  - b) Count Up and Down
  - c) Count Up
  - d) Count Down
4. 设定 PWM 的 Duty, 在默认的配置中, CMPA 越高, 则 Duty 越小
5. 将管脚配置由 GPIO 改为 PWM 输出, 真正将 PWM 波形输出于管脚上。

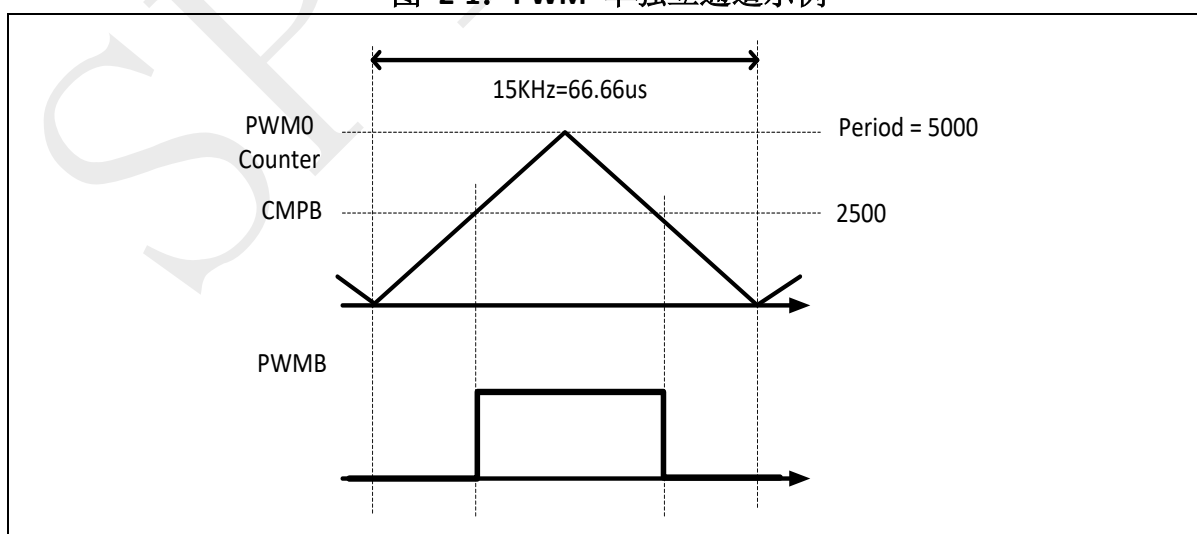
以下代码初始化了 PWM0B，可发送任意 DUTY 的 PWM 波形。

#### Example Code

```
void main()
{
    /**Step 1: System initial***/
    Sys_Init();
    /** Step 2: Clock initial***/
    CLOCK_InitWithRCO(CLOCK_HCLK_150MHZ);
    /**Step 3: PWM initial***/
    PWM_SingleChannelInit(PWM0,PWM_CHB,15000);
    /**Start PWM clock with counting up and down***/
    PWM_CounterRunControl(PWM0,COUNT_UP_DOWN_AND_RUN);
    /**Select GPIO23 as PWM0B***/
    GPIO_SetPinChannel(GPIO_23,GPIO23_PWM0B);
    /**Drive PWM0B output 50% Duty***/
    PWM_SetCMPB(PWM0,2500);/* Now period=15000000/15000/2 =
5000 */
}
```

依照以上 Example，波形如下所得：

图 2-1: PWM 单独立通道示例



PWM\_SingleChannellnit()配置重点如下:

1. 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:

$$\text{Period} = \text{PWM\_Clock\_Freq} / \text{PWM\_Freq}/2$$

以上述为例, PWM 的 Clock 与 HCLK 同频, 因此为 150MHz, 设定的 PWM 载波率为 15KHz, 因此周期为 66.67us。故此时 Period = 5000。

请调用函数 PWM\_CounterRunControl(PWM0,COUNT\_UP\_DOWN\_AND\_RUN);

此 API 让 PWM 的 counter 上下计数。

2. 预设当 counter 在往上数遇到 CMP 时, 将波形拉低, 往下数遇到 CMP 时, 将波形拉高。注意: 当以此种波形配置时, CMP 越高, 代表最终输出的 PWM 波形 Duty 越低, 反之亦然。
3. 务必先执行 Sys\_Init()与 CLOCK\_InitWithRCO()或是 CLOCK\_Init(), 否则 PWM clock 可能会有错误。
4. 本函数不配置 PWM 波形的 dead time, 用户若有控制需求, 请参阅函数 PWM\_ComplementaryPairChannellnit()。
5. 若使用者需要 PWM clock 跑得比 HCLK 慢, 可配置 PWM 内部除频器。

## 2.2 设定互补 PWM 对

利用 PWM\_ComplementaryPairChannelInit()可配置独立单信道 PWM 波形，三个输入自变量配置如下表所介绍。

表 2-3: PWM\_ComplementaryPairChannelInit()函数介绍

PWM_ComplementaryPairChannelInit (PWMx, u32PWMFreqHz, u32DeadTimeNs)	
Parameter	Description
PWMx	PWM 单元: PWM0~PWM6
u32PWMFreqHz	PWM 载波频率(Hz)
u32DeadTimeNs	PWM 上下臂死区时间，单位为纳秒(ns)

利用 PWM\_ComplementaryPairChannelInit ()可配置独立单信道 PWM 波形，以下代码初始化了 PWM0A 与 PWM0B。

### Example Code

```
void main()
{
    /** Step 1: System initial***/
    Sys_Init();

    /** Step 2: Clock initial***/
    CLOCK_InitWithRCO(CLOCK_HCLK_150MHZ);

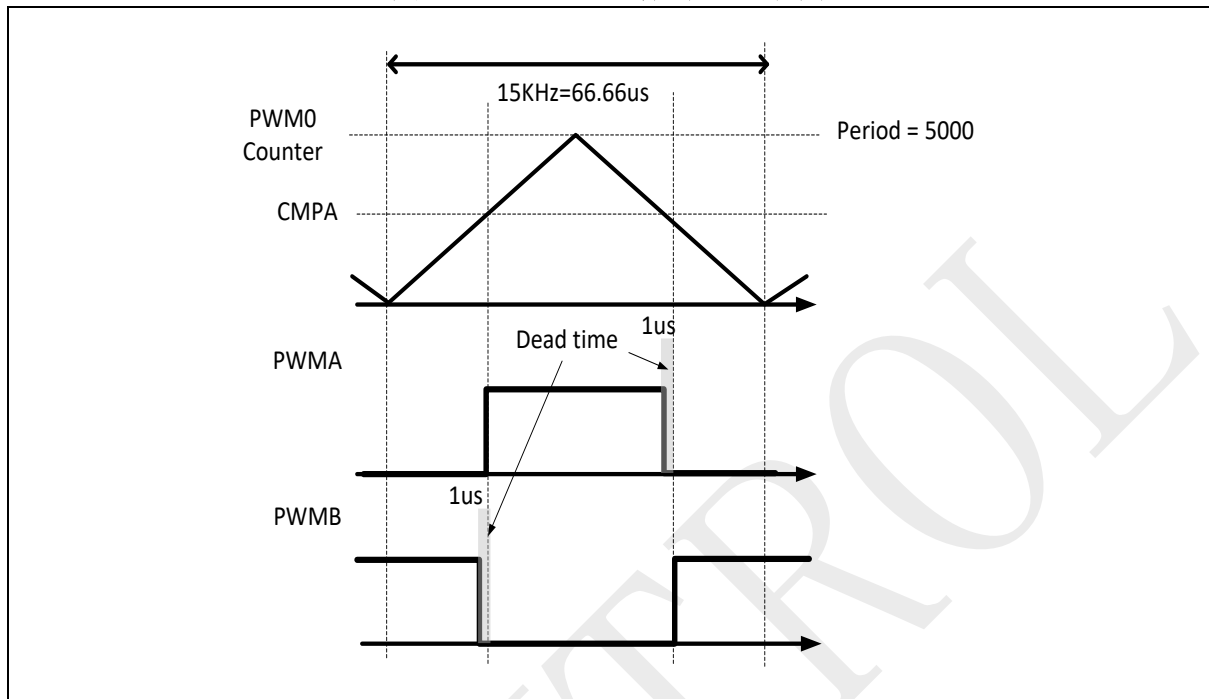
    /** Step 3: PWM initial***/
    PWM_ComplementaryPairChannelInit(PWM0 , 15000, 1000*/);

    /**Start PWM clock with counting up and down***/
    PWM_CounterRunControl(PWM0,COUNT_UP_DOWN_AND_RUN);

    /**Select GPIO23 as PWM0B***/
    GPIO_SetPinChannel(GPIO_23,GPIO23_PWM0B);
    /* Drive PWM0B output 50% Duty */
    PWM_SetCMPA(PWM0,2500); /* Now period = 15000000/15000/2 =
5000 */
}
```

以上配置如下图

图 2-2: PWM 互补对通道示例



PWM\_ComplementaryPairChannelInit()配置重点如下:

1. 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:

$$\text{Period} = \text{PWM\_Clock\_Freq} / \text{PWM\_Freq} / 2$$

以上述为例, PWM 的 Clock 与 HCLK 同频, 因此为 150MHz, 设定的 PWM 载波频率为 15KHz, 因此周期为 66.67us。故此时 Period = 5000。

请调用函数 PWM\_CounterRunControl(PWM0,COUNT\_UP\_DOWN\_AND\_RUN);

此 API 让 PWM 的 counter 上下计数。

2. 预设当 counter 在往上数遇到 CMPA 时, 将波形拉低, 往下数遇到 CMPA 时, 将波形拉高。注意: 当以此种波形配置时, CMPA 越高, 代表最终输出的 PWM 波形 Duty 越低, 反之亦然。
3. CMPB 的配置在本配置中不影响波形输出。
4. 务必先执行 Sys\_Init()与 CLOCK\_InitWithRCO()或是 CLOCK\_Init(), 否则 PWM clock 可能会有错误。
5. 本函数配置 dead time, 时间为 1000ns = 1us。
6. 若使用者需要 PWM clock 跑得比 HCLK 慢, 可配置 PWM 内部除频器。

## 2.3 PWM Trip Zone 功能介绍

PWM trip 功能泛指 PWM 因为特殊状况，可在第一时间将 PWM 关闭，而等待适当的时间重新启动 PWM 波形。SPC1068 支持 Cycle by cycle 与 one-shot(continuous)的 PWM 停止功能。Cycle by cycle 启动时，PWM 波形停止后，会在下一个 PWM 周期重新启动，适合用于恒流斩波，如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。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。

## 2.4 外部管脚触发停止 PWM

外部管脚触发功能为 Active Low，亦即当外部管脚由高电位转为低电位后，可触发 PWM 波形停止功能。SPC1068 提供的 SDK 只需输入 GPIO 的编号与欲停止的 PWM 模块编号，即可设定完成，注意执行以下代码之后，该 Pin 将自动被初始化为 GPIO 功能。

### Example Code

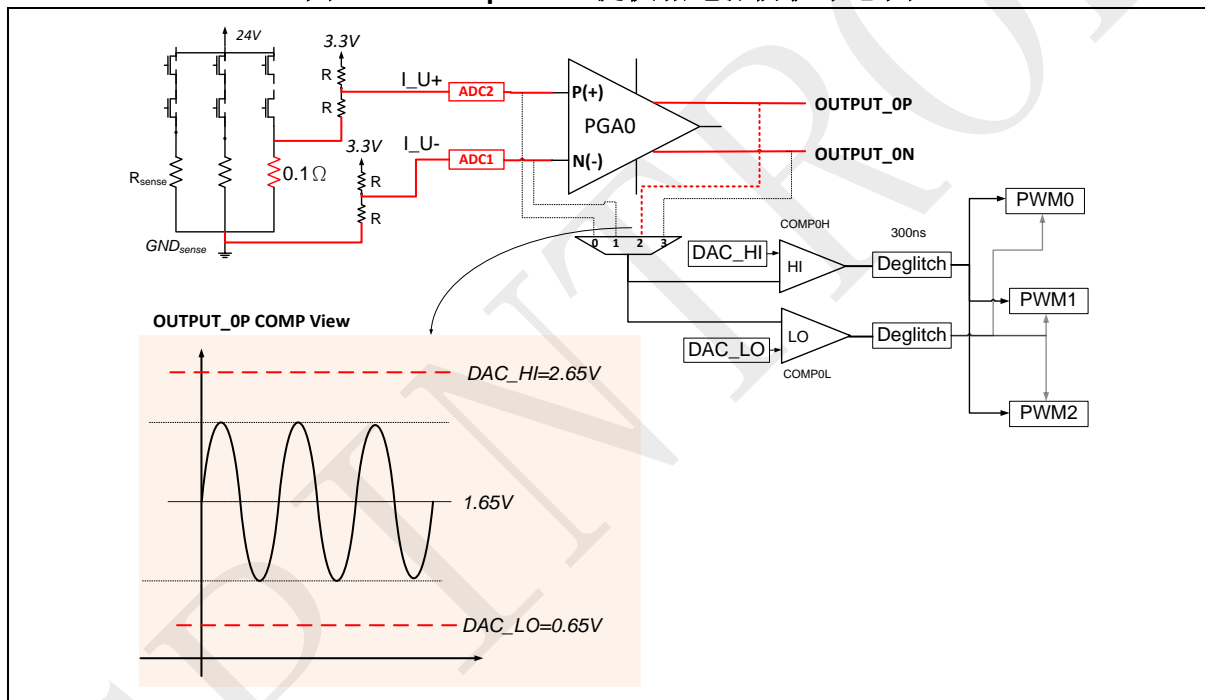
```
/* Trip zone from TZn */  
PWM_SetOneshotTripFromExtPin(PWM0, GPIO_9);  
PWM_SetOneshotTripFromExtPin(PWM1, GPIO_9);  
PWM_SetOneshotTripFromExtPin(PWM2, GPIO_9);
```

## 2.5 COMP 停止 (Trip) PWM 波形功能介绍

SPC1068 的 COMP 专为相电流防护做特殊设计。当 PGA 作为三相电流采集的时候，SPC1068 的 COMP 提供了每一相独立的电流防护，较一般的总电流防护更加安全，以下代码以 U 相电流为例，当 PGA0 输出超过 3V 时，COMP0H 触动 PWM 输出停止，当 PGA0 的电流小于 1V，由 COMPL 触发 PWM 停止。Spintrol 的 COMP 触发 PWM 停止可达成任一 COMP 同时停止所有 PWM 输出波形功能，详见 PWM trip zone 章节。

举例来说，COMP0H 与 COMP0L，只要任一种状况发生，均可以同时三相 PWM 停止，同理其他四个 COMP 亦具有同样功能，三相电流一共有六个 COMP，只要其中一个发生触发，均可同时将 PWM 停止，对于相电流防护来说是很重要的一个功能。

图 2-3: Comparator 提供相电流防护示意图



相比于大部分 IC 需要复杂的设定，Spintrol 亦提供了简单的 API 设定 COMP 触发 PWM trip 功能。当 eCOMP 代表的比较器发生输出由 L 转 H 的瞬间，关闭 PWMx 的 CHA 与 CHB 输出。可重复呼叫设定多种组合，请见范例码。

表 2-4: PWM\_EnableOneShotTripFromComp()函数介绍

PWM_EnableOneShotTripFromComp (PWMx, eCOMP)	
Parameter	Description
PWMx	PWM 单元: PWM0~PWM6
eCOMP	Comparator 单元: COMP_0_LO COMP_1_LO COMP_2_LO



	COMP_0_HI COMP_1_HI COMP_2_HI
--	-------------------------------------

以下为范例码:

#### Example Code

```
void MotorPWM_InitTripZoneAction(void)
{
    /* Note: Please initial PGA pin before comparator initial */
    COMP_Init(COMP_0_HI,
              FROM_PGAP_IN, /* From PGA0P output */
              2650          /* DAC HI compare voltage mV */,
              300          /* Output deglitch time(ns) */);

    COMP_Init(COMP_0_LO,
              FROM_PGAP_IN, /* From PGA0P output */
              650          /* DAC LO compare voltage mV */,
              300          /* Output deglitch time(ns) */);

    /* Step 3: PWM U,V,W will be triped when comparator0 output
    high.

    This code is prepared for total current protection */
    PWM_EnableOneShotTripFromComp(PWM_U,COMP_0_HI);

    PWM_EnableOneShotTripFromComp(PWM_U,COMP_0_LO);

    PWM_EnableOneShotTripFromComp(PWM_V,COMP_0_HI);

    PWM_EnableOneShotTripFromComp(PWM_V,COMP_0_LO);

    PWM_EnableOneShotTripFromComp(PWM_W,COMP_0_HI);

    PWM_EnableOneShotTripFromComp(PWM_W,COMP_0_LO);

    /* Step 5: Set PWM reaction after one shot or cycle by cycle
    event */
    PWM_SetCHAOutputWhenTrip(PWM_U,TRIP_AT_LOW);
    PWM_SetCHBOutputWhenTrip(PWM_U,TRIP_AT_LOW);

    PWM_SetCHAOutputWhenTrip(PWM_V,TRIP_AT_LOW);
    PWM_SetCHBOutputWhenTrip(PWM_V,TRIP_AT_LOW);
```

```
PWM_SetCHAOutputWhenTrip(PWM_W,TRIP_AT_LOW);  
PWM_SetCHBOutputWhenTrip(PWM_W,TRIP_AT_LOW);  
  
/* Step 6: Enable one shot interrupt in this application */  
PWM_U->TZIE.bit.OST=TZIE_BIT_OST_ENABLE;  
PWM_V->TZIE.bit.OST=TZIE_BIT_OST_ENABLE;  
PWM_W->TZIE.bit.OST=TZIE_BIT_OST_ENABLE;  
}
```

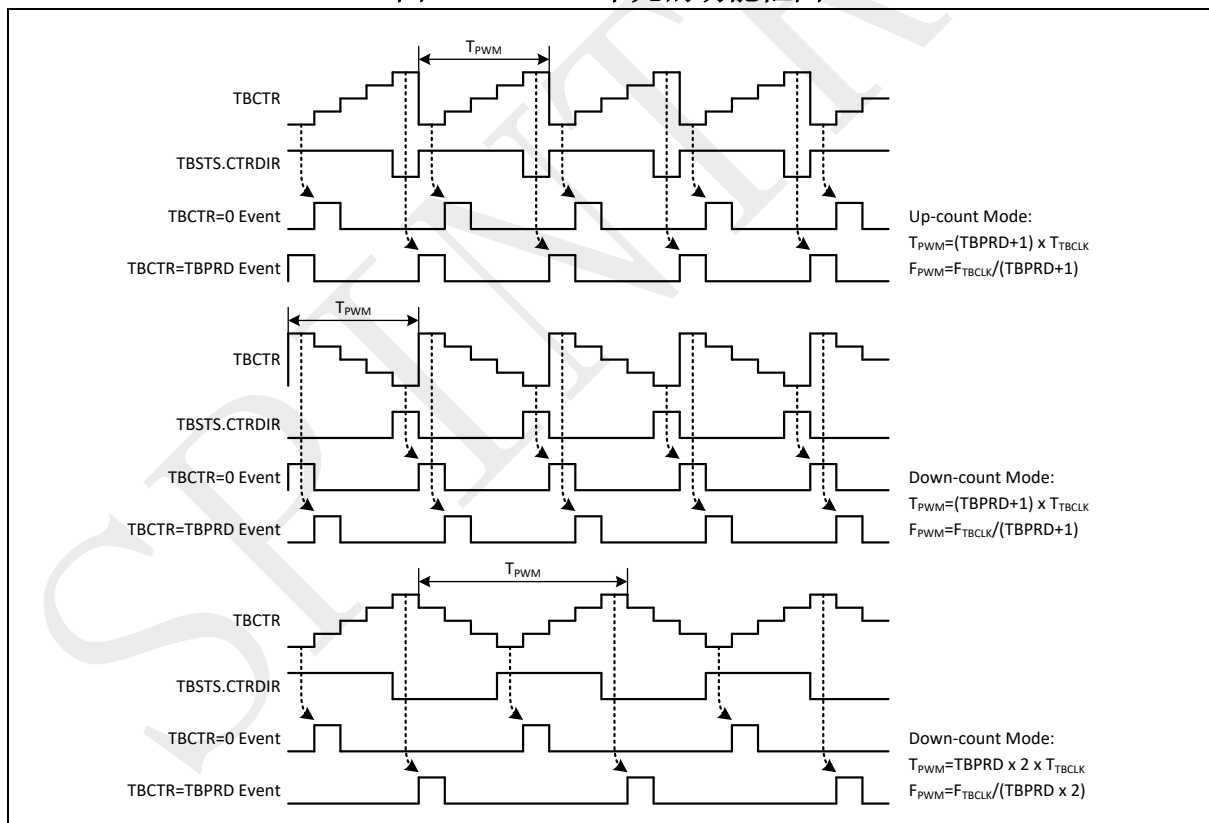
### 3 PWM 产生原理及范例

当 PWM 单元的时钟工作以后，PWM 单元内部的计数器 TBCTR 就会按照以下三种计数方式进行计数：

- 向上计数
- 向下计数
- 先向上再向下

在计数过程中，TBCTR 的最大值是周期寄存器 TBPRD 中保存的值，而最小值为 0。同时在计数器为 0 或者等于 TBPRD 时，均可产生事件，触发中断。三种计数模式如下图，其中 TBSTS.CTRDIR 表征计数器工作的方向

图 3-1: PWM 单元的功能框图



同时，在 TBCTR 计数过程中，它的值会与 0，TBPRD，CMPA 以及 CMPB 寄存器的值相比较，比较的结果可以控制 PWM 的输出结果。这些行为可以通过配置 Action-Qualifier (AQ) 寄存器来完成。总结起来，共有以下几种事件可以控制 PWM 的输出：

- TBCTR=TBPRD
- TBCTR=0

- TBCTR=0
- TBCTR=CMPA
- TBCTR=CMPB
- 软件强制控制
- 其中每件事件都可以被配置为：
  - PWM 输出置高（Set）
  - PWM 输出置低（Reset）
  - 翻转 PWM 输出（Toggle）
  - 无影响

这些事件又根据计数器的计数方向，且根据他们的优先级，可以总结为下表。

**表 3-1: 向上计数时各事件优先级**

优先级	事件
1(最高)	软件强制控制
2	TBCTR=TBPRD (PRD)
3	TBCTR=CMPB on up-count (CBU)
4	TBCTR=CMPA on up-count (CAU)
5(最低)	TBCTR=0 (ZRO)

**表 3-2: 向下计数时各事件优先级**

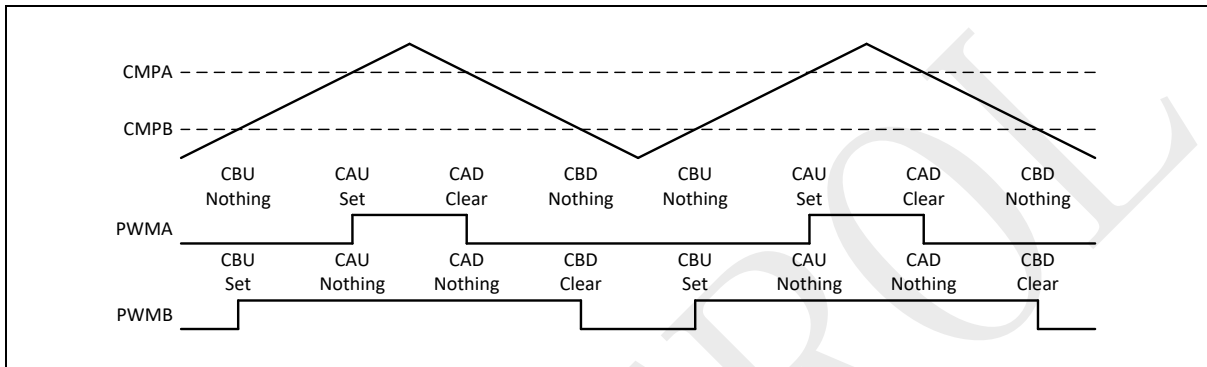
优先级	事件
1(最高)	软件强制控制
2	TBCTR=0 (ZRO)
3	TBCTR=CMPB on down-count (CBD)
4	TBCTR=CMPA on down-count (CAD)
5(最低)	TBCTR=TBPRD (PRD)

下图展示了 PWM 输出在各个事件产生点受到的影响。

在图 3-2 的例子中，当计数器向上计数时，若  $CMPA=TBCRT$ ，则 PWMA 置高；当计数器向下计数时，若  $CMPA=TBCTR$ ，则 PWMA 置低；

对于 PWMB 的输出，当计数器向上计数时，若  $CMPB=TBCRT$ ，则 PWMB 置高；当计数器向下计数时，若  $CMPB=TBCTR$ ，则 PWMB 置低。

图 3-2: 先上后下计数，对称 PWM 输出

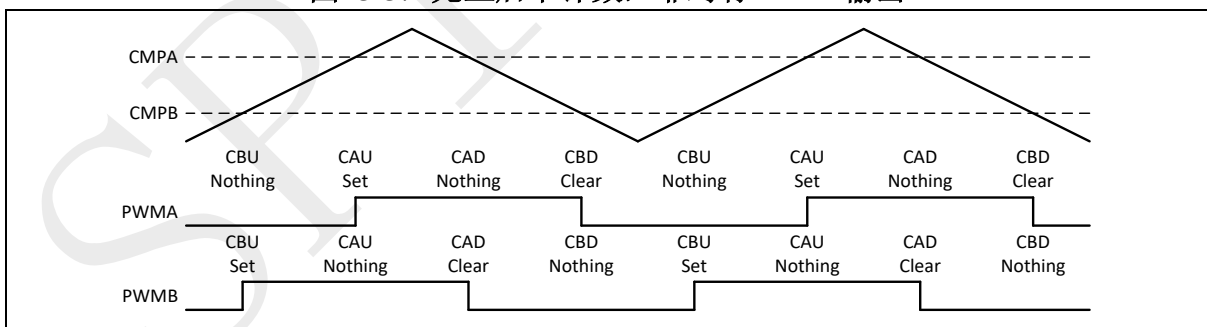


在图 3-3 的例子中，PWMA 和 PWMB 的波形受到更复杂的控制，PWMA 和 PWMB 同时受到 CMPA 和 CMPB 配置的影响，这样可以生成非对称的 PWM 波。

当计数器向上计数时，若  $CMPA=TBCRT$ ，则 PWMA 置高；当计数器向下计数时，若  $CMPB=TBCTR$ ，则 PWMA 置低；

对于 PWMB 的输出，当计数器向上计数时，若  $CMPB=TBCRT$ ，则 PWMB 置高；当计数器向下计数时，若  $CMPA=TBCTR$ ，则 PWMB 置低。

图 3-3: 先上后下计数，非对称 PWM 输出

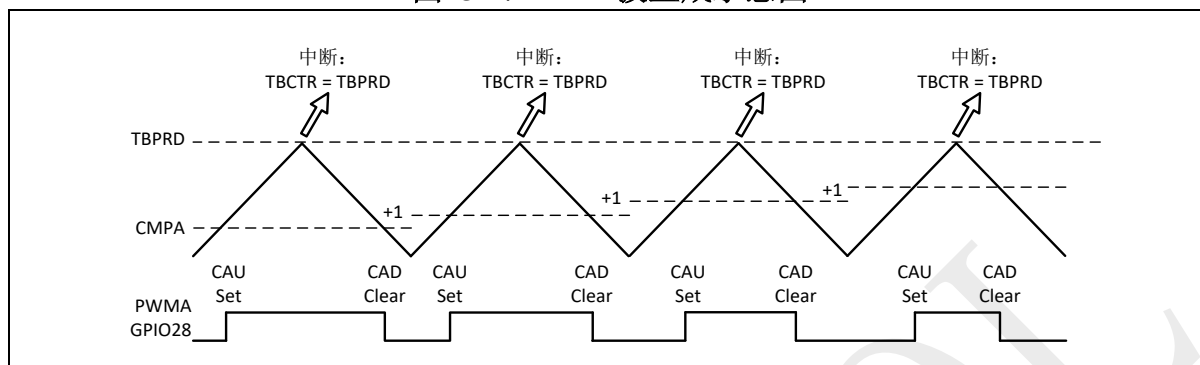


在 SP1068 的库函数中，已经包含了一些供 PWM 使用的库函数，抽象化的描述了 PWM 的行为，可供用户更加方便的配置 PWM 单元。

- 注意：
- PWM CLK 不能设置得比 PCLK 慢，PCLK 必须是最慢的时钟；
  - PWM 的 software force 和 Continuous force 触发的点位不是一致的，continuous force 在输出管脚，而 one time 在死区之前触发。

### 3.1 例 1：产生占空比周期性变化的 PWM 波

图 3-4：PWM 波生成示意图



具体代码如下：

#### Example Code

```
#define PWM_Frequency      10000      /* 10kHz PWM */

{
    uint16_t u16PrdVal = 0;

    /* Enable PWM1 clock */
    CLOCK_EnableModule(PWM1_MODULE);

    /* Set GPIO pin as PWM output pin */
    GPIO_SetPinChannel(GPIO_28, GPIO28_PWM1A);

    /* Set frequency of PWM */
    u16PrdVal = CLOCK_GetPWModuleClk(PWM1) / PWM_Frequency / 2;
    PWM_SetPeriodValue(PWM1, u16PrdVal);
    PWM_SetCMPA(PWM1, u16PrdVal / 2);      /* PWM Duty = 50% */

    /* Following 3 configurations are by default */
    PWM1->TBCTL.bit.PRDL = TBCTL_BIT_PRDL_TBPRD_FROM_SHADOW;
    PWM1->CMPCTL.bit.SHDWAMODE =
    CMPCTL_BIT_SHDWAMODE_SHADOW_MODE;
    PWM1->CMPCTL.bit.LOADAMODE =
    CMPCTL_BIT_LOADAMODE_LOAD_ON_ZERO;

    /* Configure output action */
    PWM1->AQCTLA.bit.CAU = AQCTLA_BIT_CAU_SET_HIGH;
    PWM1->AQCTLA.bit.CAD = AQCTLA_BIT_CAD_SET_LOW;

    /* Start PWM */
}
```

```
PWM_CounterRunControl (PWM1, COUNT_UP_DOWN_AND_RUN);

/* Enable PWM1 TBCTR = TBPRD event */
PWM_SetTimeEvtTiming (PWM1, EQU_PERIOD);
PWM_SetTimeEvtPeriod (PWM1, ON_1ST_EVENT);
PWM_EnableTimeEvtINT (PWM1);

/* Enable PWM1 Interrupt in CPU side */
NVIC_EnableIRQ (PWM1_IRQn);

while (1)
{
}

void PWM1_IRQHandler (void)
{
    uint16_t u16CMPAVal = 0;

    u16CMPAVal = PWM1->CMPA.all;

    /* Change PWM duty */
    if ((u16CMPAVal + 1) >= PWM_GetPeriodValue (PWM1))
    {
        PWM_SetCMPA (PWM1, 0);
    }
    else
    {
        PWM_SetCMPA (PWM1, u16CMPAVal + 1);
    }

    /* Clear interrupt flag */
    PWM_ClearTimeEvtInt (PWM1);
}
```

### 3.2 例 2：产生两路互补、带死区 PWM 波

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。

SPC1068 的 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，一般是通过一个 PWM 波，增加死区时间后，取反得到带死区的互补的 PWM 波。

下面的两幅图，展示了从一路 PWM 波，如何：

- 一分为二
- 分别增加上升沿、下降沿死区
- 不翻转、翻转

最终成为一对带死区互补 PWM 波的过程。

图 3-5：死区控制单元框图

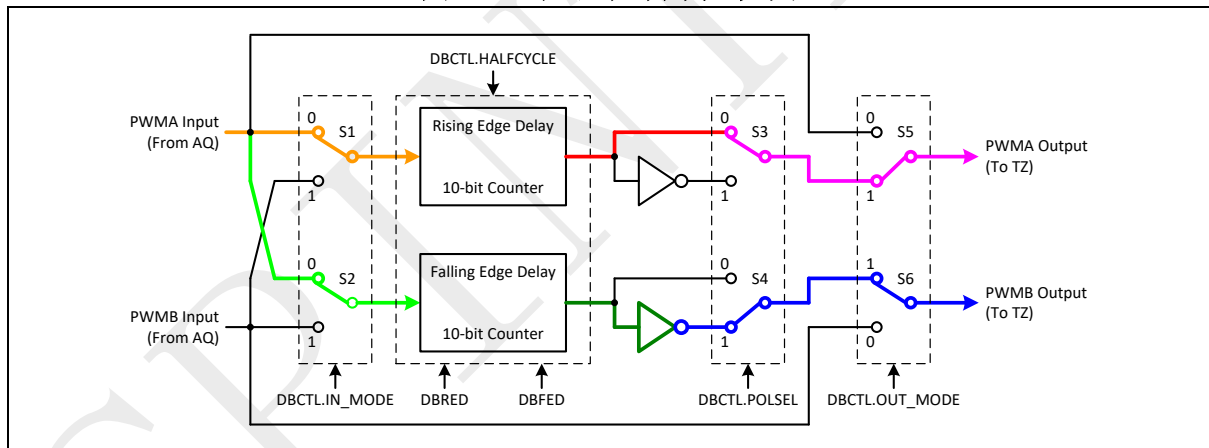
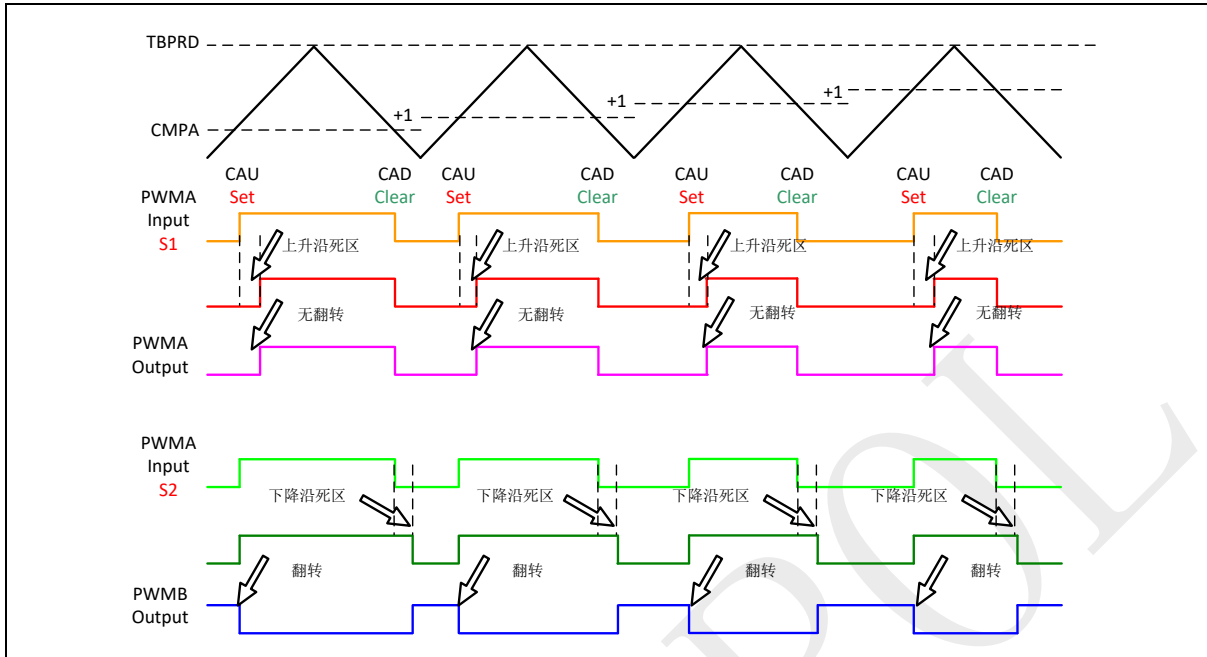




图 3-6: 生成带死区互补 PWM 示意图



实际配置过程包括:

- 单路 PWM 波配置和中断配置与例 1 相同, 这里不再赘述
- 死区时间配置

其中, 死区的配置代码如下:

#### Example Code

```
#define PWM_Frequency      10000      /* 10kHz PWM */
#define PWM_Deadband_NS    2000      /* unit is ns */

{
    uint16_t u16PrdVal = 0;
    uint16_t u16DeadbandDelay = 0;

    /* Configure Deadband */
    PWM1->DBCTL.all |= DBCTL_ALL_INMODE_0
                      | DBCTL_ALL_POLSEL_MODE_AHL      /* Active
High complementary */
                      | DBCTL_ALL_OUTMODE_3            /* PWMxA and
PWMxB are all from dead band generator */
                      | DBCTL_ALL_HALFCYCLE_DISABLE;

    u16DeadbandDelay = (PWM_Deadband_NS *
(CLOCK_GetPwmModuleClk(PWM1)/100000)) / 10000 ;
    /* Deadband falling edge delay */
}
```

```
PWM_DeadBandFallingDelay (PWM1, u16DeadbandDelay);  
/* Deadband rising edge delay */  
PWM_DeadBandRisingDelay (PWM1, u16DeadbandDelay);  
}
```

其中关键参数有三个：

1. PWM\_DBCTL\_ALL\_IN\_MODE\_0

这个参数把 PWMA 分别输入到死区控制单元的两个通道，完成了信号一分为二的动作，实际是控制开关 S1（选择 0）和 S2（选择 0）

2. PWM\_DBCTL\_ALL\_POLSEL\_MODE\_AHL

这个参数是控制死区控制单元中两路信号是否翻转，实际是控制开关 S3（选择 0）和 S4（选择 1）

3. PWM\_DBCTL\_ALL\_OUT\_MODE\_3

这个参数是控制最终输出信号是否经过死区控制单元，实际是控制 S5（选择 1）和 S6（选择 1）

### 3.3 例 3：产生两对同步 PWM 波（互补、带死区）

#### 1. PWM 同步原理

SPC1068 的 PWM 单元可以通过外部信号进行同步，当同步信号到来时，PWM 单元会把 TBPHS 中的数值载入到计数器（TBCTR）中，而计数器的计数方向则取决于 TBCTL.PHSDIR 的配置。当配置为同步后计数方向向下时（默认状态），其行为如图 3-7 所示；图 3-8 是向上计数时的行为。

图 3-7：PWM 同步（向下计数）

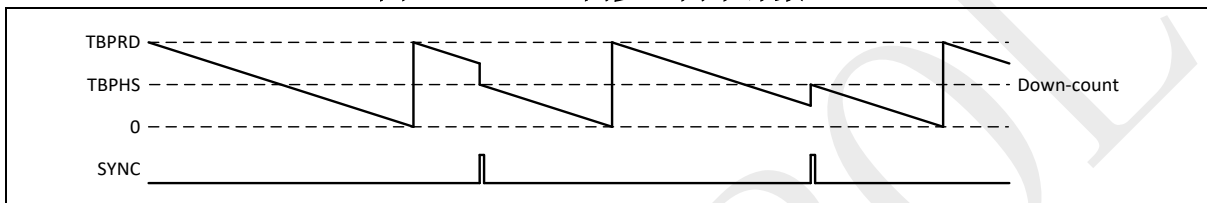
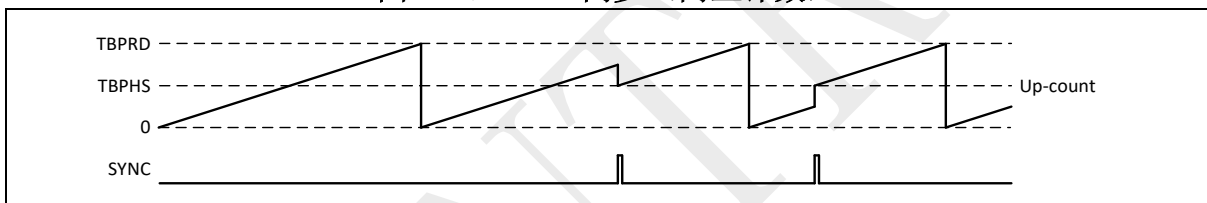


图 3-8：PWM 同步（向上计数）

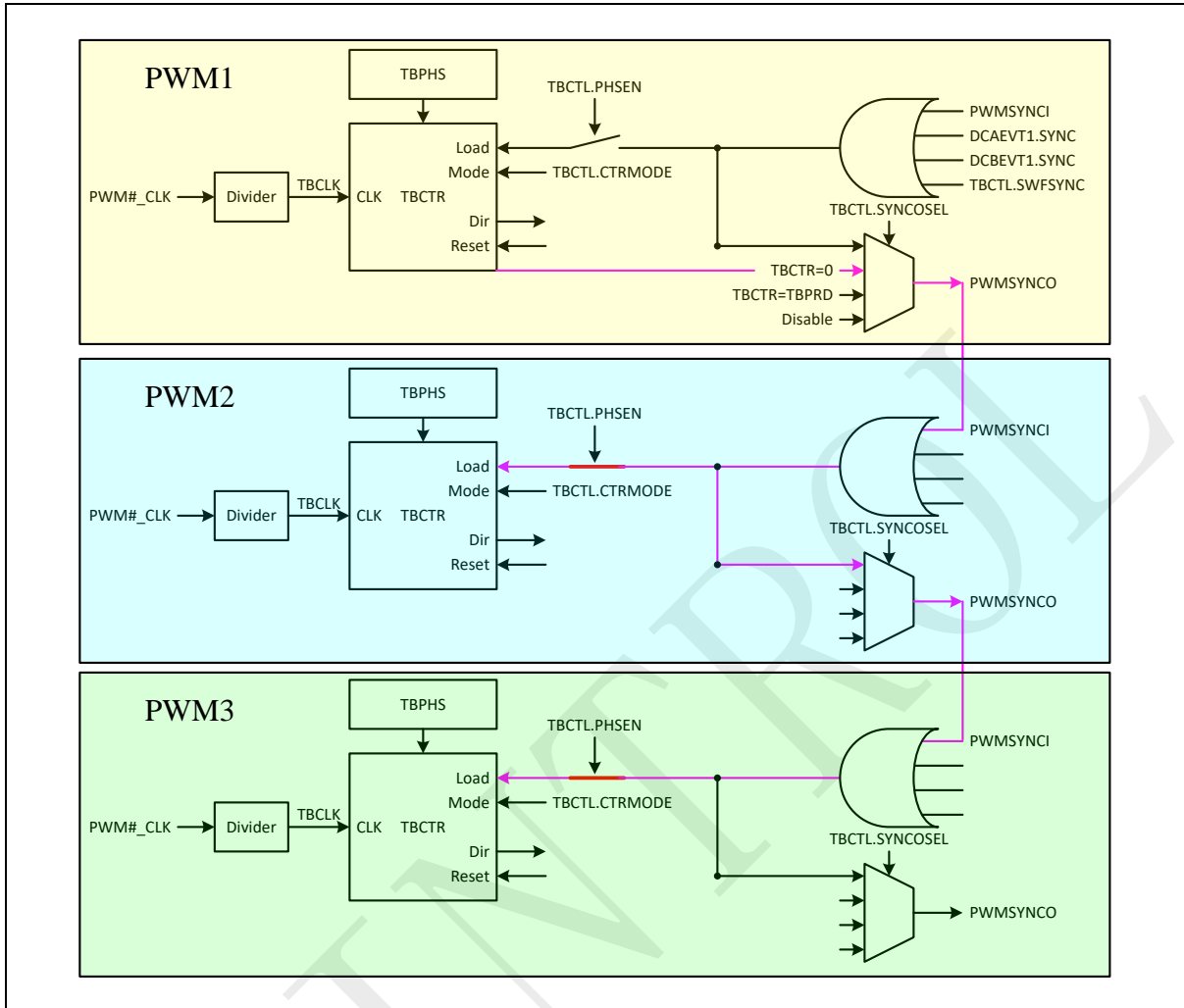


同步信号可以在以下选项中选择：

- 软件触发
- PWM 同步信号
- 比较器信号

如果选择为 PWM 同步信号，则可以实现多个 PWM 单元间的同步，这对于多相 PWM 控制（比如三相电机控制）非常有帮助。例如，在图 3-9 中，PWM1 在 TBCTR=0 时候发出同步信号，该信号传送到 PWM2 中，而 PWM2 又把该同步信号转发给 PWM3；同时在 PWM2 和 PWM3 中都打开了 PWM 同步的使能。

图 3-9: 多个 PWM 同步信号流



## 2. 配置过程

这个例子是基于之前的 PWM 例 2，所以关于 PWM 生成部分这里不多解释，主要介绍 PWM 的同步配置。

### a) 两组互补 PWM 配置

#### Example Code

```
/* Set GPIO pin as PWM2 output pin */
GPIO_SetPinChannel(GPIO_30, GPIO30_PWM2A);
GPIO_SetPinChannel(GPIO_31, GPIO31_PWM2B);
```

### b) 配置 PWM2 同步 PWM1

#### Example Code

```
/* Enable PWM1 send out SYNC signal when TBCNT == 0 */
PWM1->TBCTL.bit.SYNCOSEL = TBCTL_BIT_SYNCOSEL_TBCNT_EQU_ZERO;
/* PWM2 phase delay */
PWM2->TBPHS.all = 0;
```

```
/* PWM2 count up after SYNC */
PWM2->TBCTL.bit.PHSDIR                                =
TBCTL_BIT_PHSDIR_COUNT_UP_AFTER_SYNC;
/* Enable PWM2 SYNC function */
PWM2->TBCTL.bit.PHSEN = ENABLE;
```

### 3.4 例 4：外部信号封锁 PWM 输出（Trip Zone）

#### 1. 封锁 PWM 原理

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。

SPC1068 同样提供了这项功能，可以让用户把 PWM 配置成为当外部紧急信号到来时候，马上停止或者在下一个周期停止。

SPC1068 的 Trip Zone 可以实现以下功能：

- 按周期（Cycle-by-Cycle, CBC）封锁 PWM 输出
- 一次性封锁（One Shot, OST）PWM 输出

图 3-10：按周期封锁 PWM 信号逻辑

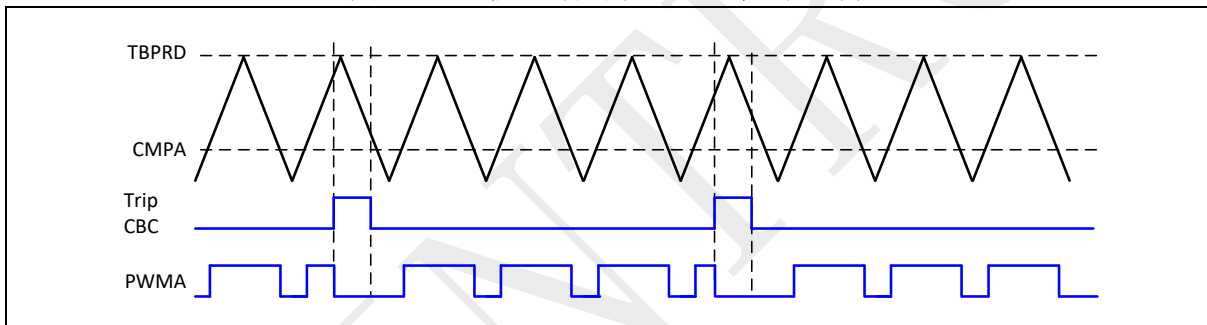


图 3-11：一次性封锁 PWM 信号逻辑

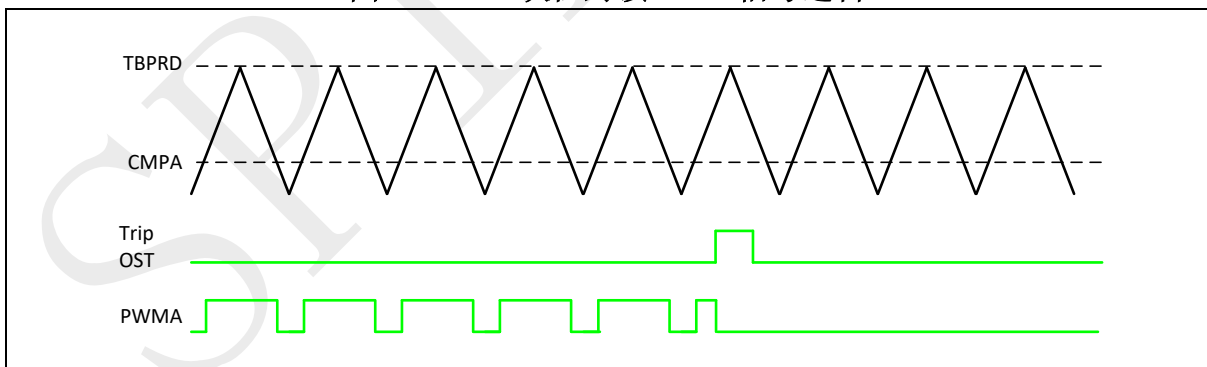
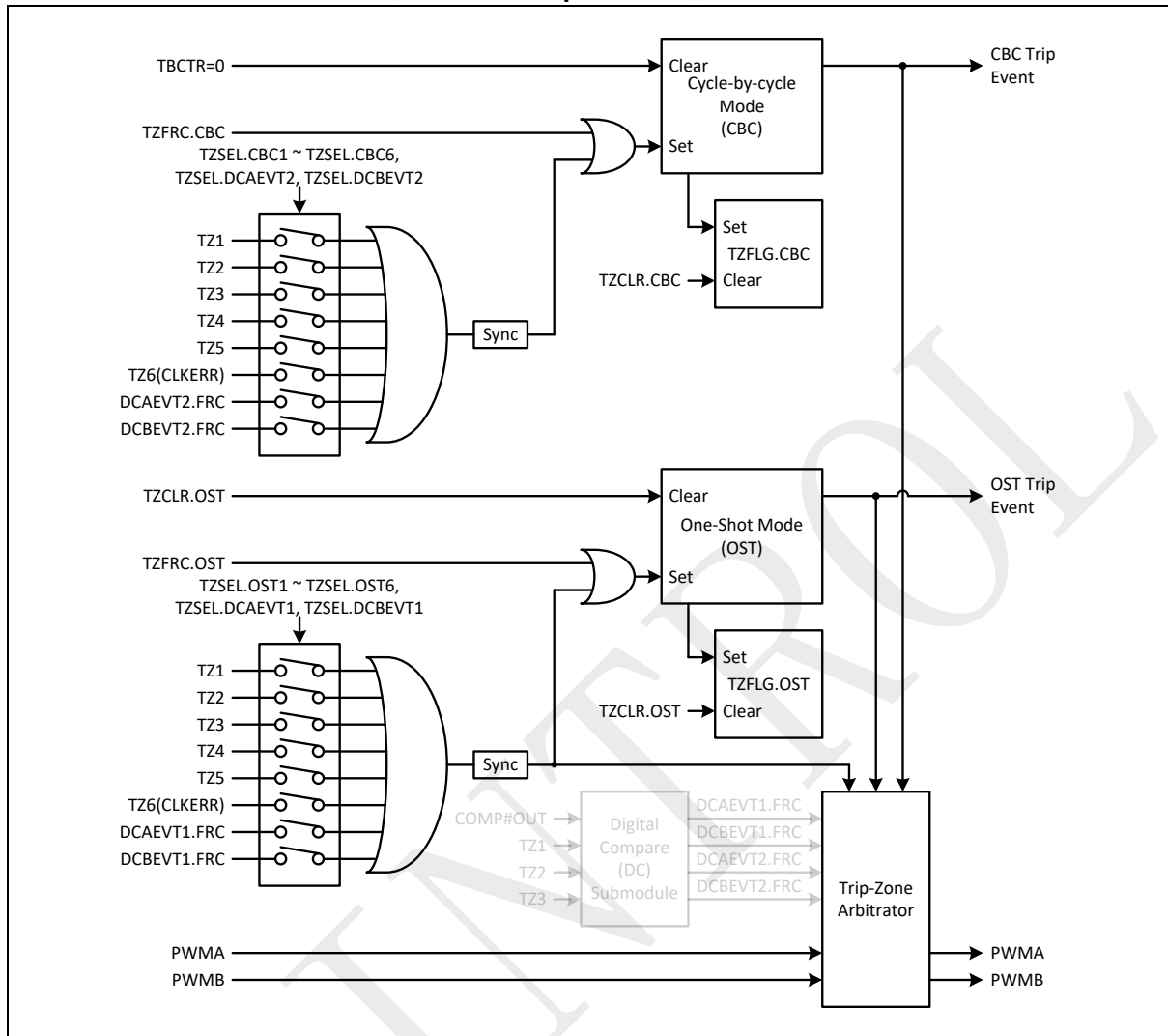


图 3-12 是 Trip Zone 的功能框图，从图中可以看出，可以引起 PWM 封锁的信号有 8 个不同的信号，可以针对每个信号分别使能。使能的信号，任意一个都能触发封锁功能。而且，封锁信号还可以通过软件触发。

Trip Zone 的按周期和一次性可以同时配置，如果两个信号同时产生，则一次性封锁的信号将覆盖按周期的信号，即一次性封锁优先级高于按周期封锁。

图 3-12: Trip Zone 功能框图



## 2. 配置过程

配置过程分为 PWM 配置过程和 Trip Zone 配置过程。

- PWM 配置过程，配置两对同步的互补 PWM，这部分采用例 3 中的配置，这里不再描述
- Trip Zone 配置

### Example Code

```

/* Trip Configuration */
GPIO_SetPinChannel(GPIO_5, GPIO5_GPIO5);
GPIO_SetPinDir(GPIO_5, GPIO_INPUT);
/* Select GPIO5 as TZ1 input */
GLOBAL->GPIOMISC.bit.TZ1SEL = GPIOMISC_BIT_TZ1SEL_GPIO5;

/* Enable TZ1 for PWM1 */

```

```
PWM1->TZSEL.all |= TZSEL_ALL_CBC1_ENABLE;
/* Config Trip Zone action on PWM1A */
PWM1->TZCTL.bit.TZA = TZCTL_BIT_TZA_SET_LOW;
/* Config Trip Zone action on PWM1B */
PWM1->TZCTL.bit.TZB = TZCTL_BIT_TZB_SET_HIGH;

/* Enable TZ1 for PWM2 */
PWM2->TZSEL.all |= TZSEL_ALL_CBC1_ENABLE;
/* Config Trip Zone action on PWM2A */
PWM2->TZCTL.bit.TZA = TZCTL_BIT_TZA_SET_LOW;
/* Config Trip Zone action on PWM2B */
PWM2->TZCTL.bit.TZB = TZCTL_BIT_TZB_SET_HIGH;
```

由上面的步骤可以看出，Trip Zone 的配置可以分为以下几个关键步骤：

- 选择 TZx 的输入引脚；
- 针对每个 PWMx，使能 TZx CBC 或 TZx OST 的输入；
- 针对每个 PWMx 的两路输出 PWMA 和 PWMB，配置 Trip Zone 信号来时，每个输入对应的动作。



## 4 PWM 在电机上的应用

### 4.1 设置电机三相 PWM 代码

下例为初始化三相电机所需的 PWM，建议的初始化流程如下：

1. 务必确认在 main 函数一开始已经调用 Sys\_Init 与 Clock\_Init
2. 调用 PWM\_ComplementaryPairChannelInit(), 初始化基本互补波形
3. 启动 PWM clock, 注意 SPC1068 的 PWM counter 分为四种模式
  - Counter Stop
  - Count Up and Down
  - Count Up
  - Count Down
4. 设定 PWM 的 Duty, 在默认的配置中, CMPA 越高, 则 Duty 越小
5. 将管脚配置由 GPIO 改为 PWM 输出。

#### Example Code

```
#define PWM_U          PWM1
#define PWM_V          PWM2
#define PWM_W          PWM3

/* PWM IO define */
#define GPIO_PWM_U_H   GPIO_28
#define GPIO_PWM_V_H   GPIO_30
#define GPIO_PWM_W_H   GPIO_32

#define GPIO_PWM_U_H_SEL   GPIO28_PWM1A
#define GPIO_PWM_V_H_SEL   GPIO30_PWM2A
#define GPIO_PWM_W_H_SEL   GPIO32_PWM3A

#define GPIO_PWM_U_L   GPIO_29
#define GPIO_PWM_V_L   GPIO_31
#define GPIO_PWM_W_L   GPIO_33

#define GPIO_PWM_U_L_SEL   GPIO29_PWM1B
#define GPIO_PWM_V_L_SEL   GPIO31_PWM2B
#define GPIO_PWM_W_L_SEL   GPIO33_PWM3B

void MotorPWM_ExampleInit()
{
    Sys_Init();
    CLOCK_InitWithRCO(CLOCK_HCLK_150MHZ);
    /* Step 1: Initial Basic Complementary PWM-*/
    PWM_ComplementaryPairChannelInit(PWM_U , 15000, 1000);
    PWM_ComplementaryPairChannelInit(PWM_V , 15000, 1000);
    PWM_ComplementaryPairChannelInit(PWM_W , 15000, 1000);
    /* Step 2: PWM Start Run Up and Down*/
    PWM_CounterRunControl(PWM_U, COUNT_UP_DOWN_AND_RUN);
}
```

```

    PWM_CounterRunControl(PWM_V,COUNT_UP_DOWN_AND_RUN);
    PWM_CounterRunControl(PWM_W,COUNT_UP_DOWN_AND_RUN);
/* Step 3: PWM Set CMPA*/
    PWM_SetCMPA(PWM_U,2500);
    PWM_SetCMPA(PWM_V,2500);
    PWM_SetCMPA(PWM_W,2500);
/*Step 3: PWM Start Run Up and Down*/
    GPIO_SetPinChannel(GPIO_PWM_U_H,GPIO_PWM_U_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_V_H,GPIO_PWM_V_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_W_H,GPIO_PWM_W_H_SEL);

    GPIO_SetPinChannel(GPIO_PWM_U_L,GPIO_PWM_U_L_SEL);
    GPIO_SetPinChannel(GPIO_PWM_V_L,GPIO_PWM_V_L_SEL);
    GPIO_SetPinChannel(GPIO_PWM_W_L,GPIO_PWM_W_L_SEL);
}

```

以上的 PWM 配置为简单的波形输出，实际上为了驱动电机，产生 ADC 触发，PWM 尚有寄存器需要配置，请参见下一个章节。

## 4.2 驱动电机 PWM 范例：PWM Trigger ADC 功能

以下以一个驱动 U 相开关的 PWM 触发三相电流采样为例。

### Example Code

```

void MotorPWM_ExampleTrigADC()
{
    Sys_Init();
    CLOCK_InitWithRCO(CLOCK_HCLK_150MHZ);
/*Step 1: Initial Basic Complementary PWM*/
    PWM_ComplementaryPairChannelInit(PWM1 , 15000, 1000);
    PWM_ComplementaryPairChannelInit(PWM2 , 15000, 1000);
    PWM_ComplementaryPairChannelInit(PWM3 , 15000, 1000);
/*Step 2: PWM Trig Select*/
    PWM1      ->ETSEL.all          |=          ETSEL_ALL_SOCAEN_ENABLE
|ETSEL_ALL_SOCASEL_TBCTR_EQU_ZERO ;
    PWM1 ->ETPS.bit.SOCAPRD = ETPS_BIT_SOCAPRD_GEN_SOCA_ON_1ST_EVENT;
/*Step 3: ADC initial*/
    ADC_EasyInit1(ADC_SOC_0,GPIO_3,ADCTRIG_PWM1A);
    ADC_EasyInit1(ADC_SOC_1,GPIO_5,ADCTRIG_PWM1A);
    ADC_EasyInit1(ADC_SOC_2,GPIO_7,ADCTRIG_PWM1A);
/*Step 4. Interrupt service routine configuration*/
    NVIC_SetPriority(ADC2_IRQn,1);
    NVIC_EnableIRQ(ADC2_IRQn);
}

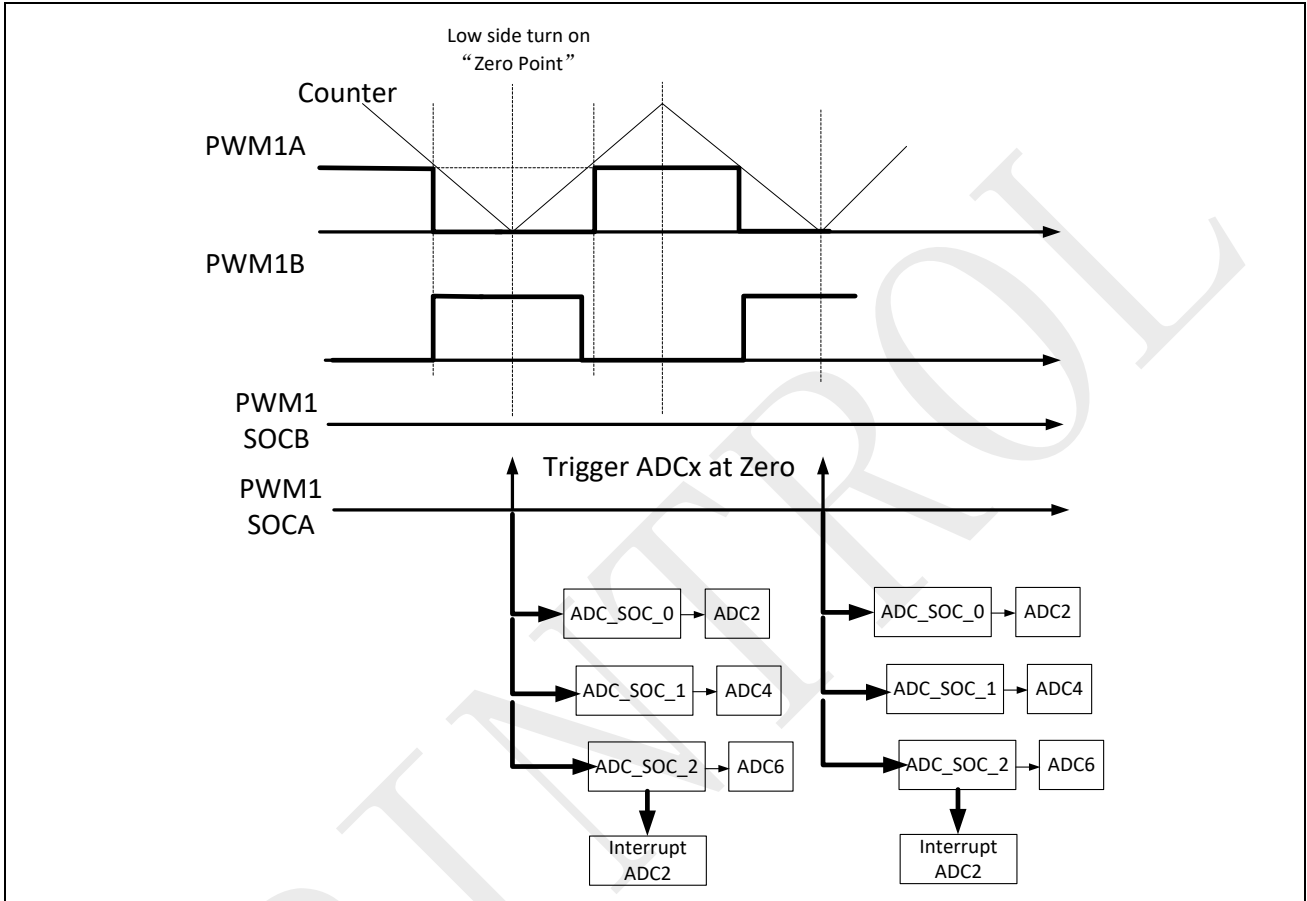
```

以上先使用基本的初始化互补对，PWM1~PWM3 依次控制 UVW 三相，设定 PWM1 之 Counter 过零时触发 ADC，一次同时触发 SOC0~SOC2 分别负责转换三相电流，当 SOC2 转换完之后，进入中断服务程序(SOC2 对应的中断服务程序为 ADC2\_IRQ)，进行电流采样。

其中设定 PWM1 由 SOCA 送出触发讯号，ADC 相对的设定为被 PWM1A 所触发。

以上的配置如下图所绘。

图 4-1: PWM 互补对通道示例



当 SOC0~SOC2 同时被触发时，根据优先层级，SOC0 会优先被转换，最后是 SOC2，上图设定为每一个 PWM 周期均送出 SOC 触发讯号。