XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

32 位微控制器

TIMER4 与 ADC 模块在电机 FOC 控制中单电阻采样联动操作说明

应用笔记

Rev1.0　2023 年 12 月

适用对象

产品系列	产品型号
F 系列	HC32F448 HC32F472 HC32F460 HC32F451 HC32F452 HC32F4A0

声　明

★ 小华半导体有限公司（以下简称：“XHSC”）保留随时更改、更正、增强、修改小华半导体产品和/或本文档的权利，恕不另行通知。用户可在下单前获取最新相关信息。XHSC 产品依据购销基本合同中载明的销售条款和条件进行销售。

★ 客户应针对您的应用选择合适的 XHSC 产品，并设计、验证和测试您的应用，以确保您的应用满足相应标准以及任何安全、安保或其它要求。客户应对此独自承担全部责任。

★ XHSC 在此确认未以明示或暗示方式授予任何知识产权许可。

★ XHSC 产品的转售，若其条款与此处规定不同，XHSC 对此类产品的任何保修承诺无效。

★ 任何带有“®”或“™”标识的图形或字样是 XHSC 的商标。所有其他在 XHSC 产品上显示的产品或服务名称均为其各自所有者的财产。

★ 本通知中的信息取代并替换先前版本中的信息。

目　录

1 概述

本篇应用笔记主要介绍关于小华半导体通用 MCU Cortex M4 系列控制定时器模块（TIMER4）与 ADC 模块在电机 FOC 控制中采用单电阻采样时的联动操作说明。

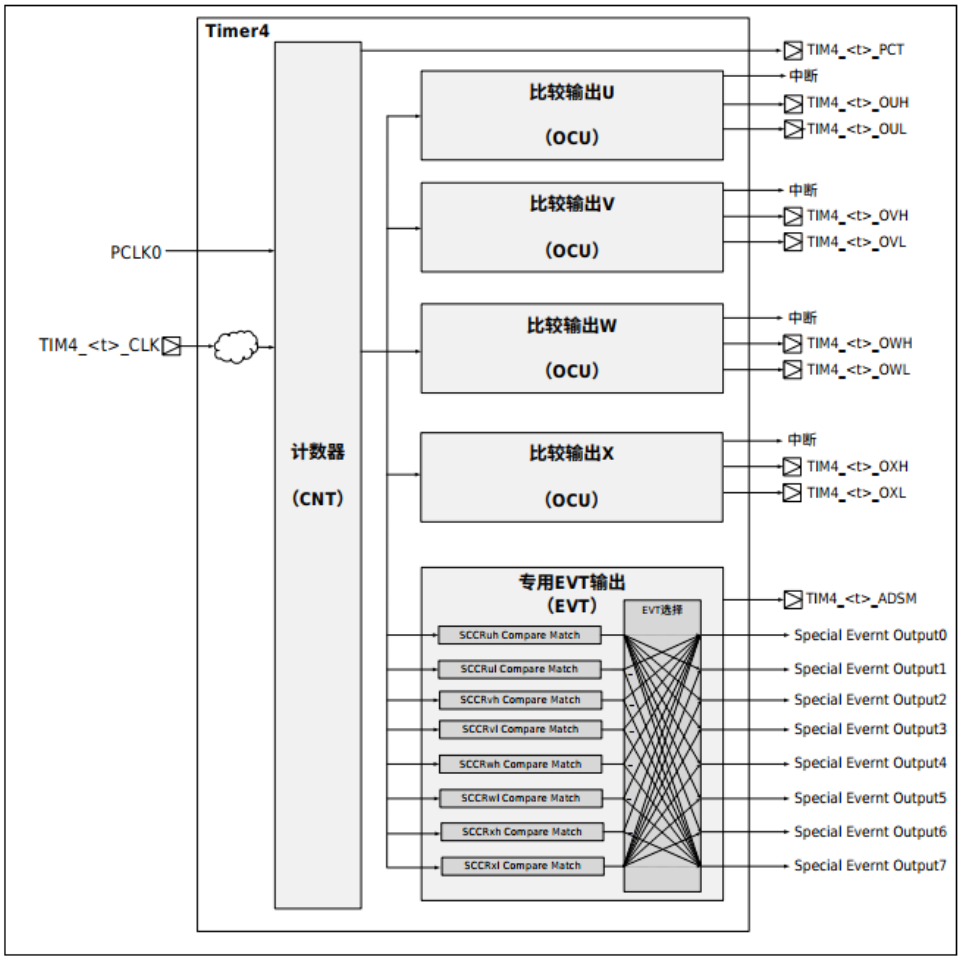
2 单电阻采样的实现方式

电机变频控制中，单电阻采样会遇到一些挑战，空间矢量脉宽调制器（SVPWM）在空间矢量的扇区边界和低调制区域的时候，会存在占空比两长一短和两短一长以及三个几乎一样长的时刻。这种情况下，有效矢量持续的时间少于电流采样时间，如果不进行移相，则会出现采样错误。当采取移相的方式实现时，FOC 变频控制 PWM 占空比的寄存器值发生改变，但是如果采用此寄存器进行 ADC 采样，此刻硬件开关的噪声会给采样带来较大的误差，影响控制精度及噪声。所以在单电阻采样时，需要考虑采样延时。此时通用 MCU 芯片需要有可以存放相位移动后的占空比值，并且能够产生触发条件触发 ADC 同步采样。

小华通用 MCU Cortex M4 系列具有 AOS 硬件外设自动互联功能，可以联动实现 FOC 控制的定时器外设同步延时触发 ADC 采样功能。本定时器触发有两种方式可以实现。一种采用专用比较器的值（SCCR）立即生效比较触发 ADC，另一种采用生成 PWM 波的 OCCR 寄存器加上 SCCR 寄存器做为延时时间的比较触发 ADC 模式。两种方式的具体实现在下文中进行详细描述。

2.1 定时器 TIMER4 专用比较器比较触发 ADC 实现单电阻采样模式

小华半导体通用控制中的通用控制定时器（TIMER4）模块是一个用于三相电机控制的定时器模块，提供各种不同应用的三相电机控制方案，该定时器支持三角波和锯齿波两种波形模式，可生成各种 PWM 波形；支持缓存功能；支持专用比较器匹配，支持 AOS 触发功能，支持 EMB 控制，Cortex M4 系列产品中搭载 3 个单元的 TIMER4。下图为 TIMER4 的基本框图。从图中可以看出 timer 具有硬件插入死区的 4 路互补 PWM，用来输出 FOC 控制中的 PWM 输出。另外还有专用 EVT 输出，此 EVT 输出可以产生 8 路比较匹配，并且产生特殊事件输出，此输出可以关联到 AOS 功能产生移相后的 ADC 采样触发。



2.1.1 硬件设定实现互补 PWM 输出

在死区定时器模式（POCR.PWMMD=01）下，通用比较基准寄存器（OCCR*l）的值发生比较匹配产生的内部输出信号（in_op*l）和 PWM 死区控制寄存器（PDAR/ PDBR）的设定值通过时序偏移，以硬件方式实现互补 PWM 输出。在该模式下，TIM4_<t>_O*H 端口输出的极性与 in_op*l 相同，TIM4_<t>_O*L 端口输出的极性与 in_op*l 相反。

2.1.2 专用比较匹配中断及事件

TIMER4 的 6/8 个专用比较基准寄存器（SCCRm）对应产生 6/8 个专用事件输出信号可以用于选择触发别的模块，如启动 ADC 等。

时钟计数过程中，专用比较基准值（SCCRm）发生计数比较匹配事件，产生相应的有效请求信号，该请求信号可以被配置到任意的事件 EVT 输出信号上（由 SCSR.EVTOS 位设定）用于触发其它模块。此时若设定 SCIR.ITEN 使能中断，则对应的专用比较匹配中断请求（TMR4_<t>_SCM）也会被触发。

该事件请求信号的输出可以选择比较启动模式或延时启动模式。在比较启动模式时（SCSR.EVTMS=0），产生 SCCR 的计数比较匹配事件后，专用事件输出信号直接输出；在延时启动模式时

（SCSR.EVTMS=1），产生 OCCR*h 或 OCCR*l（由 SCSR.EVTDS 位选择）的计数比较匹配事件后，经过 SCCR 设定的基准周期时间后，专用事件输出信号输出。

注意：在延迟计数运行中，如果再次发生 *OCCR* 与计数器匹配事件，延迟计数器重新加载计数值并重新进行递减计数。因此，如果 *OCCR* 匹配事件时间间隔小于设定的延迟时间 *SCCR* 时，专用事件输出的请求信号可能一直不会产生。

本文档采用 SCCR 比较启动模式产生 EVT 做为触发 ADC 采样触发源为样例进行说明。

2.2 ADC 模块单电阻采样应用

12 位 ADC 是一种采用逐次逼近方式的模拟数字转换器，模拟输入通道可以任意组合成一个序列，一个序列可以进行单次扫描转换或连续扫描转换。支持对任意指定通道进行连续多次转换并对转换结果进行平均。ADC 模块还搭载模拟看门狗功能，对任意指定通道的转换结果进行监视，检测其是否超出用户设定的范围。

2.2.1 ADC 通道与触发源选择

ADC 模块有多个通道，可配置为两个序列：序列 A、序列 B 进行转换。序列 A 和 B 配有独立的通道选择寄存器 ADC_CHSELRA，ADC_CHSELRB。ADC 模块支持通道映射，即模块中的虚拟通道与实际的物理通道间的映射。虚拟通道是指 ADC 模块中假定的通道，如寄存器 ADC_CHSELRA 设置为 0x1 表示序列 A 选择转换 CH0，这个 CH0 就是虚拟通道，而寄存器 ADC_DR0，是虚拟通道 CH0 的转换结果寄存器。物理通道是指实际存在的模拟通道，即外部引脚的模拟输入 ADCx_INy。虚拟通道与物理通道的映射可以通过寄存器 ADC_CHMUXR 进行配置，即同一个物理通道可以匹配为多个虚拟通道进行采样。具体参考寄存器说明，详情请参考 MCU 参考手册。

序列 A（顺序采样），序列 B（插队采样）独立选择触发源。可选择的触发源包括外部端口 ADTRGx，内部事件 IN_TRGx0，IN_TRGx1。其中，端口 ADTRGx 下降沿输入有效。IN_TRGx0，IN_TRGx1 由寄存器 ADC_TRGSEL0，1 设置，可以选择芯片内部丰富的事件源，其中就包括 TIMER4 的 EVT 事件源及 AOS 其他触发事件源。序列 B 的优先级高于序列 A。

而在实际应用中，电机控制实时性要求较高，电流采样要求实时反馈，所以建议选择序列 B 做为单电阻采样的配置。

3 M4 系列单电阻采样应用配置说明

小华 M4 系列通过控制定时器（TIMER4）模块在电机 FOC 单电阻采样控制，根据单电阻移相的方式不同，可以配置的方式也不同，下述操作以 HC32F448 芯片 PWM 波中心对称，上升沿触发 ADC 为例进行配置说明。其他实现方式也可以参考。

3.1 操作概述

小华半导体通用控制定时器模块在电机 FOC 单电阻采样控制中的操作流程为：

1) 产生 3 组互补带死区，带缓存的 PWM 信号写入比较配置寄存器，用于驱动三相电机。

2) 使能专用比较器 SCCR 功能，选择立即生效，在特定时刻产生 ADC 触发信号，触发 ADC 对电流信号的采样。

3) 使能 ADC 转换完成触发 DMA 功能，配置 DMA 源地址，目的地址进行 ADC 输出搬运或者触发中断使能在中断中读取数据。

4) 启动 ADC，启动 TIMER4 计数，开始输出及采样。

3.2 操作流程

3.2.1 互补带死区的 PWM 信号生成

1) 配置外设时钟，计数器计数值，计数模式，IO 状态
在 FOC 控制中，大部分应用场景中的载波选择三角波模式，生成具有目标周期频率的三角载波，产生计数上溢中断或计数下溢中断（以计数上溢中断为例）。

2) 配置比较输出电平，更新模式，周期链接功能以及缓存使能
设定比较输出所对应的端口输出选择，比较输出信号来源，使能周期缓存，周期链接等功能，以及比较输出模式。

3) PWM 输出流程
设定 PWM 的输出模式（互补带死区模式）、PWM 输出极性是否翻转、GPIO 初始输出电平，死区的宽度等 PWM 输出信息。

4) EVT 特殊事件触发 ADC 采样配置
配置专用比较器比较通道、周期链接、缓存功能信息以及触发模式等信息。

3.2.2 ADC 触发信号生成

TIMER4 模块运行时，因为采用单电阻采样，在设定的计数方向上，需要在【0-周期值】的任意数值上匹配 ADC 采样两次，并且发出的两次占空比值的间距不能小于电流采样的时间。所以需要采用两个专用比较器做为触发 ADC 事件的触发匹配数据，即需要两个 SCCRx/y 比较寄存器存储比较值产生比较匹

配（SCCRxy Compare Match），而 TIMER4 中的 EVT 事件可以关联到 Special Event Output（x)中的任意一个，可以多个 Match 匹配同一个 Event Output。详情可以查阅参考手册中 TIMER4 的基本框图。当采用专用事件比较匹配立即生效时，TIMER4 当前的计数值和设定的专用比较值相等时，产生 ADC 触发信号，进行 FOC 控制中电流信号的采样。

1) 配置外设 ADC、AOS 时钟，使能 ADC 序列 B 触发模式；

2) 配置 ADC 通道，采样时间以及选择 AOS 模块触发源；

3) 使能 DMA 或者中断接收采样值。

3.2.3 启动 TIMER4 计数

启动 TIMER4 计数，开始进行输出。

```
void TMR4_Start(CM_TMR4_TypeDef *TMR4x)
{
    DDL_ASSERT(IS_TMR4_UNIT(TMR4x));
    CLR_REG16_BIT(TMR4x->CCSR, TMR4_CCSR_STOP);
}
```

3.2.4 注意事项

1) 缓存功能建议开启，确保 FOC 控制中时序，防止产生未预期的问题。

2) 可以根据采样电流重构方式的不同，更改上升沿及下降沿触发方式。

3) TIMER4 的 8 路专用比较器触发可以匹配到同一个 EVT 事件进行输出，不为一一对应关系。

4) 需要根据采样时钟及外部负载设定采样时间。

5) 关于 ADC 采样精度提升可以参考 AN《AN_如何提高 ADC 采样精度_Rev1.0》。

6) 采样完成后可以由中断读取 ADC 数据也可以用 DMA 传送，注意采样数据保存的时序。

4 样例代码

以下是介绍本 AN 基于 TIMER4 及 ADC 模块在电机 FOC 控制中的相关外设配置。实例以 HC32F448 芯片为例。

4.1 M4 系列中 TIMER4 输出死区 PWM 及触发单电阻采样样例配置

```
void TIM4_1_Conf(uint16_t u16_CPSR, uint16_t u16_DeadTimeCnt, uint8_t u8PeakIsrMaskCnt)
{
    stc_tmr4_init_t             stcTmr4Init;
    stc_tmr4_oc_init_t          stcTmr4OcInit;
    stc_tmr4_pwm_init_t         stcTmr4PwmInit;
    un_tmr4_oc_ocmrl_t          unTmr4OcOcmrl;
    un_tmr4_oc_ocmrh_t          unTmr4OcOcmrh;
    stc_tmr4_evt_init_t         stcTmr4EventInit;

    /*配置外设时钟，计数器计数值，计数模式，IO 状态*/

    /* Enable TMR4 peripheral clock */
    FCG_Fcg2PeriphClockCmd(FCG2_PERIPH_TMR4_1, ENABLE);
    /* Initialize PWM I/O */
    GPIO_SetFunc(GPIO_PORT_B, GPIO_PIN_13 | GPIO_PIN_14 |GPIO_PIN_15, GPIO_FUNC_2);// UL、VL、WL

    GPIO_SetFunc(GPIO_PORT_A, GPIO_PIN_08 | GPIO_PIN_09 |GPIO_PIN_10, GPIO_FUNC_2);// UH、VH、WH

    /************************* Configure TMR4 counter *************************/
    (void)TMR4_StructInit(&stcTmr4Init);
    stcTmr4Init.u16ClockSrc = TMR4_CLK_SRC_INTERNCLK;
    stcTmr4Init.u16CountMode = TMR4_MD_TRIANGLE;
    stcTmr4Init.u16ClockDiv = TMR4_CLK_DIV1;
    stcTmr4Init.u16PeriodValue = u16_CPSR;
    (void)TMR4_Init(CM_TMR4_1, &stcTmr4Init);
    TMR4_SetCountValue(CM_TMR4_1, 0);
    TMR4_SetCountIntMaskTime(CM_TMR4_1,TMR4_INT_CNT_PEAK,u8PeakIsrMaskCnt);
    /*Enable    the TMR4 counter period buffer function.*/
    TMR4_PeriodBufCmd(CM_TMR4_1, ENABLE);
    /* TMR4 OC low channel: initialize */
    (void)TMR4_OC_StructInit(&stcTmr4OcInit);
    stcTmr4OcInit.u16CompareValue = 0xffff;
    stcTmr4OcInit.u16BufLinkTransObject = TMR4_OC_BUF_CMP_VALUE|TMR4_OC_BUF_CMP_MD;
    stcTmr4OcInit.u16OcInvalidPolarity = TMR4_OC_INVD_LOW;
    stcTmr4OcInit.u16CompareModeBufCond = TMR4_OC_BUF_COND_VALLEY;
    stcTmr4OcInit.u16CompareValueBufCond = TMR4_OC_BUF_COND_VALLEY;
    (void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_UL, &stcTmr4OcInit);
    (void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_VL, &stcTmr4OcInit);
    (void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_WL, &stcTmr4OcInit);
```

```
/* TMR4 OC low channel: initialize */
(void)TMR4_OC_StructInit(&stcTmr4OcInit);
stcTmr4OcInit.u16CompareValue = 0xffff;

stcTmr4OcInit.u16BufLinkTransObject = TMR4_OC_BUF_CMP_VALUE|TMR4_OC_BUF_CMP_MD;

stcTmr4OcInit.u16OcInvalidPolarity = TMR4_OC_INVD_LOW;
stcTmr4OcInit.u16CompareModeBufCond = TMR4_OC_BUF_COND_VALLEY;
stcTmr4OcInit.u16CompareValueBufCond = TMR4_OC_BUF_COND_VALLEY;
(void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_UH, &stcTmr4OcInit);
(void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_VH, &stcTmr4OcInit);
(void)TMR4_OC_Init(CM_TMR4_1, TMR4_OC_CH_WH, &stcTmr4OcInit);

/*配置比较输出电平，事件更新模式以及缓存使能*/

unTmr4OcOcmrh.OCMRx = 0x0000;
TMR4_OC_SetHighChCompareMode(CM_TMR4_1,TMR4_OC_CH_UH,unTmr4OcOcmrh);
TMR4_OC_SetHighChCompareMode(CM_TMR4_1,TMR4_OC_CH_VH,unTmr4OcOcmrh);
TMR4_OC_SetHighChCompareMode(CM_TMR4_1,TMR4_OC_CH_WH,unTmr4OcOcmrh);
/* TMR4 OC low channel: compare mode OCMR[31:0] */
unTmr4OcOcmrl.OCMRx_f.OCFDCL = TMR4_OC_HOLD;      /* bit[0]     0 */
unTmr4OcOcmrl.OCMRx_f.OCFPKL = TMR4_OC_HOLD;      /* bit[0]     0 */
unTmr4OcOcmrl.OCMRx_f.OCFUCL = TMR4_OC_HOLD;      /* bit[0]     0 */
unTmr4OcOcmrl.OCMRx_f.OCFZRL = TMR4_OC_HOLD;      /* bit[0]     0 */
unTmr4OcOcmrl.OCMRx_f.OPDCL    = TMR4_OC_LOW;     /* bit[5:4]     10 */
unTmr4OcOcmrl.OCMRx_f.OPPKL    = TMR4_OC_LOW;     /* bit[7:6]     10 */
unTmr4OcOcmrl.OCMRx_f.OPUCL    = TMR4_OC_HOLD;    /* bit[9:8]     00 */
unTmr4OcOcmrl.OCMRx_f.OPZRL    = TMR4_OC_LOW;     /* bit[11:10] 10 */
unTmr4OcOcmrl.OCMRx_f.OPNPKL   = TMR4_OC_HIGH;    /* bit[13:12] 01 */
unTmr4OcOcmrl.OCMRx_f.OPNZRL   = TMR4_OC_LOW;     /* bit[15:14] 10 */
unTmr4OcOcmrl.OCMRx_f.EOPNDCL = TMR4_OC_HOLD;     /* bit[17:16] 00 */
unTmr4OcOcmrl.OCMRx_f.EOPNUCL = TMR4_OC_HIGH;     /* bit[19:18] 01 */
unTmr4OcOcmrl.OCMRx_f.EOPDCL   = TMR4_OC_LOW;     /* bit[21:20] 10 */
unTmr4OcOcmrl.OCMRx_f.EOPPKL   = TMR4_OC_LOW;     /* bit[23:22] 10 */
unTmr4OcOcmrl.OCMRx_f.EOPUCL   = TMR4_OC_HIGH;    /* bit[25:24] 01 */
unTmr4OcOcmrl.OCMRx_f.EOPZRL   = TMR4_OC_HIGH;    /* bit[27:26] 01 */
unTmr4OcOcmrl.OCMRx_f.EOPNPKL = TMR4_OC_HIGH;     /* bit[29:28] 01 */
unTmr4OcOcmrl.OCMRx_f.EOPNZRL = TMR4_OC_HIGH;     /* bit[31:30] 01 */
TMR4_OC_SetLowChCompareMode(CM_TMR4_1, TMR4_OC_CH_UL, unTmr4OcOcmrl);
TMR4_OC_SetLowChCompareMode(CM_TMR4_1, TMR4_OC_CH_VL, unTmr4OcOcmrl);
TMR4_OC_SetLowChCompareMode(CM_TMR4_1, TMR4_OC_CH_WL, unTmr4OcOcmrl);
/* TMR4 OC low channel: enable */
(void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_UL, ENABLE);
(void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_VL, ENABLE);
(void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_WL, ENABLE);
/* TMR4 OC low channel: enable */
(void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_UH, DISABLE);
```

```
    (void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_VH, DISABLE);
    (void)TMR4_OC_Cmd(CM_TMR4_1, TMR4_OC_CH_WH, DISABLE);

    /*配置 PWM 输出*/


    /* TMR4 PWM: initialize */
    (void)TMR4_PWM_StructInit(&stcTmr4PwmInit);
    stcTmr4PwmInit.u16Polarity = TMR4_PWM_OXH_HOLD_OXL_HOLD;
    stcTmr4PwmInit.u16ClockDiv = TMR4_PWM_CLK_DIV1;

    stcTmr4PwmInit.u16Mode = TMR4_PWM_MD_DEAD_TMR;                    //死区定时器模式

    (void)TMR4_PWM_Init(CM_TMR4_1, TMR4_PWM_CH_U, &stcTmr4PwmInit);
    (void)TMR4_PWM_Init(CM_TMR4_1, TMR4_PWM_CH_V, &stcTmr4PwmInit);
    (void)TMR4_PWM_Init(CM_TMR4_1, TMR4_PWM_CH_W, &stcTmr4PwmInit);
    /* TMR4 PWM: set dead time count */
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1, TMR4_PWM_CH_U, TMR4_PWM_PDAR_IDX, u16_DeadTimeCnt);
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1, TMR4_PWM_CH_U, TMR4_PWM_PDBR_IDX, u16_DeadTimeCnt);
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1, TMR4_PWM_CH_V, TMR4_PWM_PDAR_IDX, u16_DeadTimeCnt);
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1, TMR4_PWM_CH_V, TMR4_PWM_PDBR_IDX, u16_DeadTimeCnt);
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1,        TMR4_PWM_CH_W,        TMR4_PWM_PDAR_IDX,
u16_DeadTimeCnt);
    TMR4_PWM_SetDeadTimeValue(CM_TMR4_1,        TMR4_PWM_CH_W,        TMR4_PWM_PDBR_IDX,
u16_DeadTimeCnt);
    /* TMR4 PWM: when EMB occured,the status of PWM state */
    /* MOE: 0 ; AOE 0 */
    TMR4_PWM_SetOEEffectTime(CM_TMR4_1, TMR4_PWM_OE_EFFECT_COUNT_VALLEY);
    TMR4_PWM_EmbHWMainOutputCmd(CM_TMR4_1,DISABLE);
    TMR4_PWM_MainOutputCmd(CM_TMR4_1,DISABLE);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OUH, TMR4_PWM_ABNORMAL_PIN_HIZ);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OUL, TMR4_PWM_ABNORMAL_PIN_HIZ);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OVH, TMR4_PWM_ABNORMAL_PIN_HIZ);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OVL, TMR4_PWM_ABNORMAL_PIN_HIZ);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OWH, TMR4_PWM_ABNORMAL_PIN_HIZ);
    TMR4_PWM_SetAbnormalPinStatus(CM_TMR4_1, TMR4_PWM_PIN_OWL, TMR4_PWM_ABNORMAL_PIN_HIZ);
    /* TMR4 PWM: set port output normal */
    TMR4_PWM_SetPortOutputMode(CM_TMR4_1, TMR4_PWM_PIN_OUH, TMR4_PWM_PIN_OUTPUT_NORMAL);
    TMR4_PWM_SetPortOutputMode(CM_TMR4_1, TMR4_PWM_PIN_OUL, TMR4_PWM_PIN_OUTPUT_NORMAL);
    TMR4_PWM_SetPortOutputMode(CM_TMR4_1, TMR4_PWM_PIN_OVH, TMR4_PWM_PIN_OUTPUT_NORMAL);
    TMR4_PWM_SetPortOutputMode(CM_TMR4_1, TMR4_PWM_PIN_OVL, TMR4_PWM_PIN_OUTPUT_NORMAL);
   TMR4_PWM_SetPortOutputMode(CM_TMR4_1,TMR4_PWM_PIN_OWH,TMR4_PWM_PIN_OUTPUT_NORMAL);
   TMR4_PWM_SetPortOutputMode(CM_TMR4_1,TMR4_PWM_PIN_OWL,TMR4_PWM_PIN_OUTPUT_NORMAL);
    /*Set TMR4 OC OCCR/OCMR buffer transfer condition.*/
   TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_UH,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
```

```
        TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_UL,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
        TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_VH,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
        TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_VL,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
        TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_WH,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
        TMR4_OC_SetCompareBufCond(CM_TMR4_1,TMR4_OC_CH_WL,TMR4_OC_BUF_CMP_VALUE,TMR4_OC_BUF_C
OND_VALLEY);
        /* EVT 特殊事件触发 ADC 采样配置*/
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UH */
        (void)TMR4_EVT_StructInit(&stcTmr4EventInit);
        stcTmr4EventInit.u16CompareValue = 0x10;
        stcTmr4EventInit. u16MatchCond = TMR4_EVT_MATCH_CNT_UP|TMR4_EVT_MATCH_CNT_PEAK;
        stcTmr4EventInit.u16OutputEvent = TMR4_EVT_OUTPUT_EVT1;
        stcTmr4EventInit.u16Mode = TMR4_EVT_MD_CMP;
        (void)TMR4_EVT_Init(CM_TMR4_1, TMR4_EVT_CH_UH, &stcTmr4EventInit);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UH SCSR */
        TMR4_EVT_BufIntervalReponseCmd(CM_TMR4_1,TMR4_EVT_CH_UH, ENABLE);
        TMR4_EVT_SetOutputEvent(CM_TMR4_1, TMR4_EVT_CH_UH, TMR4_EVT_OUTPUT_EVT1);
        TMR4_EVT_SetCompareBufCond(CM_TMR4_1, TMR4_EVT_CH_UH, TMR4_EVT_BUF_COND_VALLEY);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UH SCMR */
        TMR4_EVT_SetMaskTime(CM_TMR4_1,TMR4_EVT_CH_UH,TMR4_EVT_MASK0);
        TMR4_EVT_EventIntervalReponseCmd(CM_TMR4_1,
        TMR4_EVT_CH_UH,TMR4_EVT_MASK_VALLEY|TMR4_EVT_MASK_PEAK,ENABLE);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UH SCER */
        TMR4_EVT_SetOutputEventSignal(CM_TMR4_1,TMR4_EVT_OUTPUT_EVT0_SIGNAL);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UL */
        (void)TMR4_EVT_StructInit(&stcTmr4EventInit);
        stcTmr4EventInit.u16CompareValue = 0x1000;
        stcTmr4EventInit.u16MatchCond = TMR4_EVT_MATCH_CNT_UP|TMR4_EVT_MATCH_CNT_PEAK;
        stcTmr4EventInit.u16OutputEvent = TMR4_EVT_OUTPUT_EVT1;
        stcTmr4EventInit.u16Mode = TMR4_EVT_MD_CMP;
        (void)TMR4_EVT_Init(CM_TMR4_1, TMR4_EVT_CH_UL, &stcTmr4EventInit);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UL SCSR */
        TMR4_EVT_BufIntervalReponseCmd(CM_TMR4_1,TMR4_EVT_CH_UL, ENABLE);
        TMR4_EVT_SetOutputEvent(CM_TMR4_1, TMR4_EVT_CH_UL, TMR4_EVT_OUTPUT_EVT1);
        TMR4_EVT_SetCompareBufCond(CM_TMR4_1,TMR4_EVT_CH_UL, TMR4_EVT_BUF_COND_VALLEY);
        /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UL SCMR */
        TMR4_EVT_SetMaskTime(CM_TMR4_1,TMR4_EVT_CH_UL,TMR4_EVT_MASK0);
    TMR4_EVT_EventIntervalReponseCmd(CM_TMR4_1,TMR4_EVT_CH_UL,TMR4_EVT_MASK_VALLEY|TMR4_EVT_MAS
    K_PEAK,ENABLE);
```

```
    /* Initialize TIMER4 SEVT.TMR4_EVT_CH_UH SCER */
    TMR4_EVT_SetOutputEventSignal(CM_TMR4_1,TMR4_EVT_OUTPUT_EVT1_SIGNAL);
    /* Enable interrupt */
    TMR4_IntCmd(CM_TMR4_1, TMR4_INT_CNT_PEAK, ENABLE);//上溢中断
}
```

4.2 M4 系列 ADC 单电阻采样

```
void ADC_Conf(void)
{
    stc_adc_init_t stcAdcInit;
    stc_gpio_init_t stcGpioInit;
    stc_aos_intsfttrg_bit_t AOS_Config;
    /* 1. Enable ADC peripheral clock. */
    FCG_Fcg3PeriphClockCmd(FCG3_PERIPH_ADC1, ENABLE);
    FCG_Fcg0PeriphClockCmd(FCG0_PERIPH_AOS, ENABLE);
    /* 2. Modify the default value depends on the application. */
    (void)ADC_StructInit(&stcAdcInit);
    stcAdcInit.u16DataAlign    = ADC_DATAALIGN_RIGHT;
    stcAdcInit.u16Resolution = ADC_RESOLUTION_12BIT;
    stcAdcInit.u16ScanMode = ADC_MD_SEQA_BUF_SEQB_SINGLESHOT;//序列 A/B 单次扫描模式
    /* 3. Initializes ADC. */
    (void)ADC_Init(CM_ADC1, &stcAdcInit);
    /* 4. ADC channel configuration. */
    /* 4.1 Set the ADC pin to analog input mode. */
    (void)GPIO_StructInit(&stcGpioInit);//
    stcGpioInit.u16PinAttr = PIN_ATTR_ANALOG;
    (void)GPIO_Init(GPIO_PORT_A, GPIO_PIN_02, &stcGpioInit);
    /* 4.2 Enable ADC channels and set sample time. */
    ADC_ChRemap(CM_ADC1,ADC_CH0,ADC1_PIN_PA2);
    ADC_ChCmd(CM_ADC1, ADC_SEQ_A, ADC_CH0, ENABLE);//-PA2--U
    ADC_ChRemap(CM_ADC1,ADC_CH1,ADC1_PIN_PA2);
    ADC_ChCmd(CM_ADC1, ADC_SEQ_A, ADC_CH1, ENABLE);//-PA2--U
    //采样时间最小不能小于 5 ,否则 AD 采样异常,进不了 DMA ,设置范围(5-255)
    ADC_SetSampleTime(CM_ADC1, ADC_CH0, 20);
    ADC_SetSampleTime(CM_ADC1, ADC_CH1, 20);
    AOS_CommonTriggerCmd(AOS_ADC1_0,AOS_COMM_TRIG1,ENABLE);
    AOS_SetTriggerEventSrc(AOS_ADC1_0,EVT_SRC_TMR4_1_SCMP_UL);
    ADC_TriggerConfig(CM_ADC1, ADC_SEQ_A, ADC_HARDTRIG_EVT0);
    ADC_TriggerCmd(CM_ADC1, ADC_SEQ_A, ENABLE);
}
```

版本修订记录

版本号	修订日期	修订内容
Rev1.0	2023/12/21	初版发布。