CMS32M53xx/55xx 应用笔记

高速模数转换器（ADC1）

Rev. 1.00

目录

1. 概述 3
1.1 目的 3
1.2 定义、首字母缩略词和缩写词 3
1.3 内容提要 3
2. 模块概述 4
2.1 模块特性 4
2.2 ADC1 软件触发启动 4
2.3 ADC1 硬件触发启动 4
2.4 ADC1 校准 4
2.5 寄存器说明 4
3. ADC1 硬件触发模式 5
3.1 操作实例 5
3.2 样例代码 6
3.3 波形图 8
4. ADC1 软件触发模式 9
4.1 操作实例 9
4.2 样例代码 10
4.3 波形图 12
5. 注意事项 13
6. 更多信息 14
7. 版本修订说明 15

1. 概述

1.1 目的

本文档介绍了 CMS3253xx/55xx 高速模数转换器 ADC1 的特性以及如何实现软/硬件触发高速模数转换器进行模数转换。

1.2 定义、首字母缩略词和缩写词

表 1-1：定义、首字母缩略词和缩写词

缩写	说明
ADC1	高速模数转换器
PCLK	APB 时钟
EPWM	增强型 PWM
T_{ADCK}	ADC1 模块时钟

1.3 内容提要

本文档包含以下内容：

第 2 章 模块概述。

第 3 章 ADC1 硬件触发模式。

第 4 章 ADC1 软件触发模式。

第 5 章 注意事项。

2. 模块概述

2.1 模块特性

- 端口配置模拟输入电压范围：AVSS(VSS) ~ AVDD(VDD)
- 最大采样速率：1.2Msps
- 多达 24 路单端模拟输入通道
- 单次转换时间为：23×T_{ADCK}(采样时间设置为 10.5×T_{ADCK})
- 单次模式：对指定通道执行一次 A/D 转换
- 连续模式：对所有选定的通道都执行 A/D 转换
- 支持外部输入信号触发 ADC 转换
- 支持转换完毕产生中断
- 内置 AD 转换结果比较器
- 每个通道的转换结果都存储在对应的数据寄存器中
- 通道 30 可测试内部模拟电压信号(包括 OP0/1 输出、PGA0/1 输出、内部 1.2V 基准电压)

2.2 ADC1 软件触发启动

使用软件触发 ADC1 需要将寄存器 ADCCON2.ADCST 位中写入 1，将启动 ADC1 转换。转换完毕后，该位硬件自动清零。

在 ADC1 转换期间，任何软件和硬件触发启动信号将被忽略。

2.3 ADC1 硬件触发启动

除了软件触发 ADC1 转换，该 ADC1 模块还提供了硬件触发启动的方式。硬件触发源的种类有五种，分别是外部触发、内部触发、EPWM 输出通道触发、EPWM 计数比较器 0 触发和 EPWM 计数比较器 1 触发。

不同种类触发源可同时有效，且同一种类触发源可能含有不同的触发信号；详情可参考芯片手册。

2.4 ADC1 校准

ADC1 模块在测量模拟电压之前，建议对 ADC1 模块进行校准，以便得到更高的性能与分辨率。详情可参考芯片手册。

2.5 寄存器说明

详情可参考芯片手册。

3. ADC1 硬件触发模式

3.1 操作实例

实例目标：实现 EPWM0 零点硬件触发 ADC1 转换。

操作步骤：

1) 设置芯片 APB 时钟（PCLK）为 48Mhz。
2) 开启 ADC1 模块时钟使能位。
3) 设置 ADC1 模块为连续转换模式，时钟为 APB 时钟(PCLK)的 8 分频，采样保持时间 10.5×T_{ADCK}。
4) 设置 ADC1 转换通道，设置 ADC1 硬件触发模式为 EPWM0 零点触发。
5) 设置 ADC1 中断及中断优先级。
6) 开启 ADC1。
7) 开启 ADC1 校准，使用 ADC1 的校准值进行 ADC 转换。
8) 设置 P14 为输出模式，用于指示 ADC1 中断。
9) 设置 EPWM 模块为中心对称计数模式，互补输出模式。
10) 开启 EPWM 时钟使能位。
11) 设置 EPWM 模块 EPWM0 分频、周期、占空比、加载模式。
12) 设置 EPWM 中断及开启中断。
13) 设置 EPWM0 输出 IO 口，使能输出。
14) 开启 EPWM。
15) 设置 P13 为输出模式，用于指示 EPWM0 零点中断。
16) 在 ADC1 中断服务函数执行 P14 输出电平翻转。
17) 在 EPWM 中断服务函数执行 P13 输出电平翻转。

3.2 样例代码

```
int main(void)
{
	SYS_DisableIOCFGProtect();		/*关闭 IOCONFIG 写保护*/
	SYS_DisableGPIO0Protect();		/*关闭 GPIO0 的相关寄存器写保护*/
	SYS_DisableGPIO1Protect();		/*关闭 GPIO1 的相关寄存器写保护*/
	SYS_DisableGPIO2Protect();		/*关闭 GPIO2 的相关寄存器写保护*/
	SYS_DisableGPIO3Protect();		/*关闭 GPIO3 的相关寄存器写保护*/
	SYS_DisableGPIO4Protect();		/*关闭 GPIO4 的相关寄存器写保护*/
	SYS_ConfigHSI(SYS_CLK_HSI_48M);		/*设置内部高速时钟为 48Mhz*/
	SYS_EnableHSI();			/*开启高速时钟*/
	SYS_ConfigAHBClock(SYS_CLK_SEL_HSI,SYS_CLK_DIV_1);	/*设置 AHB 时钟为高速时钟的 1 分频*/
	SYS_ConfigAPBClock(AHB_CLK_DIV_1);			/*设置 APB 时钟为 AHB 时钟的 1 分频*/

	ADC1_EPWM_Trigger_Mode();		/*设置 ADC1 模块*/
	EPWM_Trigger_ADC1();			/*设置 EPWM 模块*/
	while(1)
	{
		;
	}
}

void ADC1_EPWM_Trigger_Mode(void)
{
	/*(1)设置 ADC1 时钟*/
	SYS_EnablePeripheralClk(SYS_CLK_ADC1_MSK);		/*使能 ADC1 模块时钟*/
	ADC1_ConfigRunMode(ADC1_CONVERT_CONTINUOUS,ADC1_CLK_DIV_8,ADC1_HOLD_10P5_CLK);
	//使能连续转换，TADCK=PCLK/8，采样保持时间为 10.5×TADCK

	/*(2)设置 ADC1 转换触发方式*/
	ADC1_EnableHardwareTrigger(ADC1_TG_EPWM0_ZERO);		/*EPWM0 的零点触发*/
	ADC1_SetTriggerDelayTime(0);				/*硬件触发延时*/

	/*(3)设置 ADC1 通道使能*/
	ADC1_EnableEPWMTriggerChannel(ADC1_CH_13_MSK);		/*EPWM 触发的通道选择 AN13 */
	ADC1_EnableScanChannel(ADC1_CH_13_MSK);
	SYS_SET_IOCFG(IOP21CFG,SYS_IOCFG_P21_AN13);		/*关闭 P21 的数字功能*/

	/*(4)设置 ADC1 中断*/
	ADC1_EnableChannelInt(ADC1_CH_13_MSK);			/*开 AN13 转换中断*/
	NVIC_EnableIRQ(ADC1_IRQn);

	/*(5)设置优先级*/
	NVIC_SetPriority(ADC1_IRQn,3);				/*优先级 0~3， 0 最高、3 最低*/

	/*(6)开启 ADC1*/
	ADC1_Start();

	/*(7)开启 ADC1 校准*/
	ADC1_StartAdjust();

	/*(8)使用 ADC1 校准值的 ADC 转换*/
	ADC1_EnableAdjust();

	/*(9)设置 P14 指示 ADC1 中断*/
	SYS_SET_IOCFG(IOP14CFG, SYS_IOCFG_P14_GPIO);
	GPIO_CONFIG_IO_MODE(GPIO1,GPIO_PIN_4,GPIO_MODE_OUTPUT);
	GPIO1->DO_f.P4= 1;
}
```

```
void EPWM_Trigger_ADC1(void)
{
    /*(1)设置 EPWM 运行模式*/
    EPWM_ConfigRunMode(   EPWM_COUNT_UP_DOWN  |          /*中心对称计数模式*/
                          EPWM_OCU_SYMMETRIC   |         /*对称模式*/
                          EPWM_WFG_COMPLEMENTARYK   |    /*互补模式*/
                          EPWM_OC_INDEPENDENT);          /*独立输出模式*/

    /*(2)设置 EPWM 时钟周期与死区*/
    SYS_EnablePeripheralClk(SYS_CLK_EPWM_MSK);           /*开启 EPWM 时钟*/
    EPWM_ConfigChannelClk( EPWM0, EPWM_CLK_DIV_1);       /*EPWM0 时钟为 PCLK 的 1 分频*/
    EPWM_ConfigChannelPeriod(EPWM0,   4800);             /*EPWM 计数值 4800，EPWM 周期 200us*/
    EPWM_ConfigChannelSymDuty(EPWM0, 2400);
    EPWM_DisableDeadZone(EPWM_CH_0_MSK | EPWM_CH_1_MSK);   /*关闭死区*/

    /*(3)设置 EPWM 反向输出*/
    EPWM_DisableReverseOutput( EPWM_CH_0_MSK | EPWM_CH_1_MSK);      /*关闭反相输出*/

    /*(4)设置 EPWM 加载方式*/
    EPWM_EnableAutoLoadMode(EPWM_CH_0_MSK);              /*EPWM 模式需自动加载模式*/

    /*(5)设置中断*/
    EPWM_EnableZeroInt( EPWM_CH_0_MSK);                  /*开启零点中断*/
    NVIC_EnableIRQ(EPWM_IRQn);

    /*(6)设置优先级*/
    NVIC_SetPriority(EPWM_IRQn,2);                       /*优先级 0~3，  0 最高、3 最低*/
    /*(7)设置 IO 口输出*/

    SYS_SET_IOCFG(IOP01CFG, SYS_IOCFG_P01_EPWM0);        /*EPWM0 输出口*/
    SYS_SET_IOCFG(IOP04CFG, SYS_IOCFG_P04_EPWM1);
    EPWM_EnableOutput(EPWM_CH_0_MSK | EPWM_CH_1_MSK);    /*使能 EPWM0、EPWM1 输出*/

    /*(8)开启 EPWM*/
    EPWM_Start(EPWM_CH_0_MSK);

   /*(9)设置 P13  指示 EPWM0 零点中断*/
    SYS_SET_IOCFG(IOP13CFG, SYS_IOCFG_P13_GPIO);
    GPIO_CONFIG_IO_MODE(GPIO1,GPIO_PIN_3,GPIO_MODE_OUTPUT);
    GPIO1->DO_f.P3 = 1;
}

void ADC1_IRQHandler(void)
{
    if(ADC1_GetChannelIntFlag(ADC1_CH_13))               /*判断 ADC1 通道中断状态*/
    {
        GPIO1->DO_f.P4 = 1;                              /*P14 置 1*/
        ADC1_ClearChannelIntFlag(ADC1_CH_13);            /*清除 ADC1 通道中断标志*/
        GPIO1->DO_f.P4 = 0;                              /*P14 清 0*/
    }
}

void EPWM_IRQHandler(void)
{
    if(EPWM_GetZeroIntFlag(EPWM0))                       /*判断 EPWM 中断标志*/
    {
        GPIO1->DO_f.P3 = 1;                              /*P13 置 1*/
        EPWM_ClearZeroIntFlag(EPWM0);                    /*清除 EPWM0 中断标志*/
        GPIO1->DO_f.P3 = 0;                              /*P13 清 0*/
    }
}
```

3.3 波形图

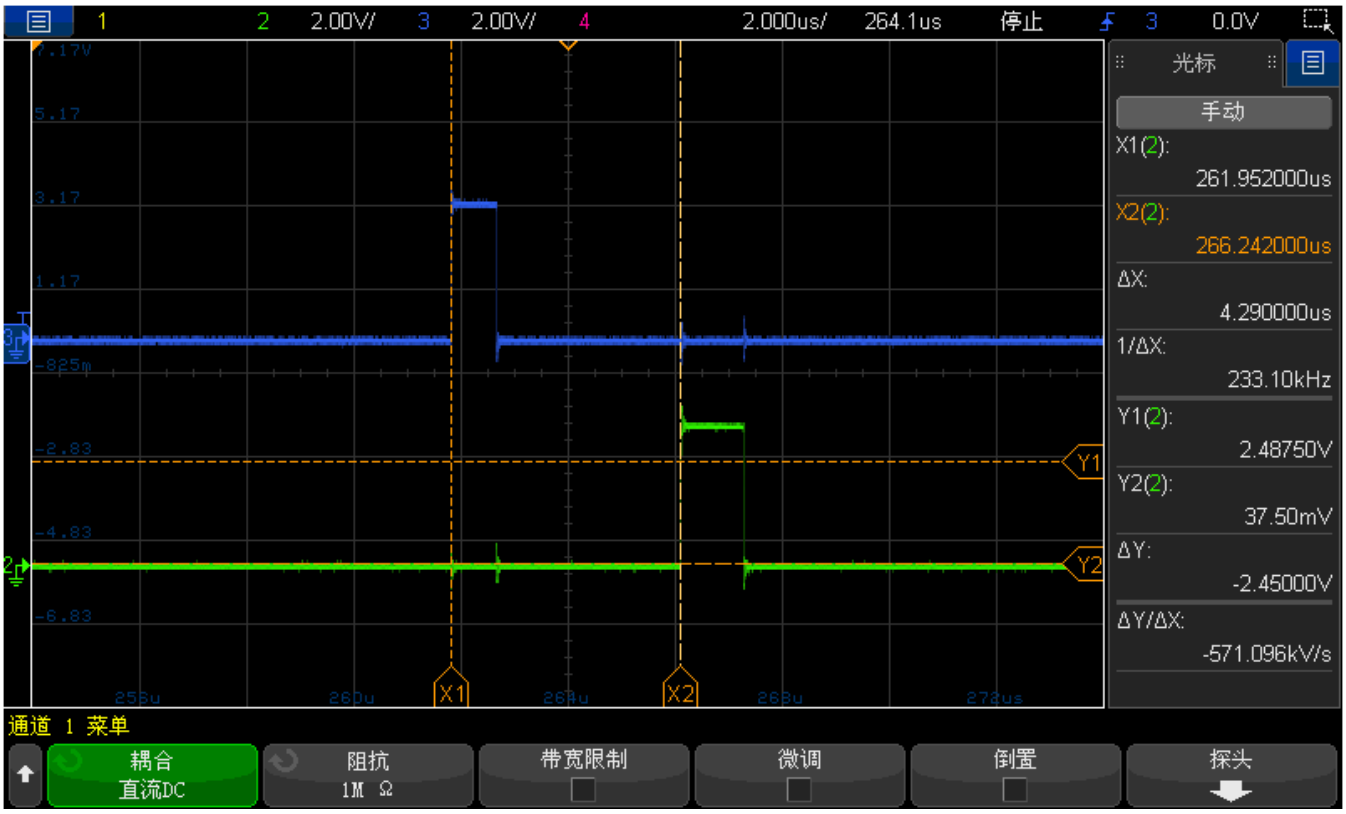

图 3-1：硬件触发波形图

如上图 3-1 所示，蓝线代表 P13（EPWM0 零点中断信号），绿线代表 P14（ADC1 硬件中断信号）。

4. ADC1 软件触发模式

4.1 操作实例

实例目标：实现软件触发 ADC1 中断。

操作步骤：

1) 设置芯片 APB 时钟（PCLK）为 48Mhz。
2) 使能 ADC1 模块时钟。
3) 设置 ADC1 模块为连续转换模式，时钟为 APB 时钟(PCLK)的 8 分频，采样保持时间 $10.5\times T_{ADCK}$。
4) 设置 ADC1 软件转换通道。
5) 设置 ADC1 中断以及中断优先级。
6) 开启 ADC1。
7) 开启 ADC1 校准，使用 ADC1 的校准值进行 ADC 转换。
8) 设置 P14 为输出模式，用于指示 ADC1 中断。
9) 设置 P13 为输出模式，用于指示软件触发信号。
10) 在主函数 while(1)中循环软件触发 ADC1。

4.2 样例代码

```
int main(void)
{
    uint16_t i;
    SYS_DisableIOCFGProtect();              /*关闭 IOCONFIG 写保护*/
    SYS_DisableGPIO0Protect();              /*关闭 GPIO0 的相关寄存器写保护*/
    SYS_DisableGPIO1Protect();              /*关闭 GPIO1 的相关寄存器写保护*/
    SYS_DisableGPIO2Protect();              /*关闭 GPIO2 的相关寄存器写保护*/
    SYS_DisableGPIO3Protect();              /*关闭 GPIO3 的相关寄存器写保护*/
    SYS_DisableGPIO4Protect();              /*关闭 GPIO4 的相关寄存器写保护*/
    SYS_ConfigHSI(SYS_CLK_HSI_48M);         /*设置内部高速时钟为 48Mhz*/
    SYS_EnableHSI();                        /*开启高速时钟*/
    SYS_ConfigAHBClock(SYS_CLK_SEL_HSI,SYS_CLK_DIV_1);      /*设置 AHB 时钟为高速时钟的 1 分频*/
    SYS_ConfigAPBClock(AHB_CLK_DIV_1);                      /*设置 APB 时钟为 AHB 时钟的 1 分频*/

    ADC1_Software_Trigger_Mode();           /*设置 ADC1 软件触发模式*/
    /*设置 P13 指示软件触发信号*/
     SYS_SET_IOCFG(IOP13CFG, SYS_IOCFG_P13_GPIO);
    GPIO_CONFIG_IO_MODE(GPIO1,GPIO_PIN_3,GPIO_MODE_OUTPUT);
    GPIO1->DO_f.P3 = 0;
    while(1)
    {
        for(i = 240; i > 0; i--);
        if( !ADC1_IS_BUSY() )                   /*判断 ADC 是否正在转换*/
        {
            GPIO1->DO_f.P3 = 1;                 /*P13 置 1*/
            ADC1_Go();                          /*软件启动 ADC1*/
            GPIO1->DO_f.P3 = 0;                 /*P13 清 0*/
            while(ADC1_IS_BUSY());              /*等待转换结束*/
        }
    }
}

void ADC1_Software_Trigger_Mode(void)
{
    /*(1)设置 ADC1 时钟*/
    SYS_EnablePeripheralClk(SYS_CLK_ADC1_MSK);              /*使能 ADC1 模块时钟*/
    ADC1_ConfigRunMode(ADC1_CONVERT_CONTINUOUS,ADC1_CLK_DIV_8,ADC1_HOLD_10P5_CLK);
    //使能连续转换，TADCK=PCLK/8，采样保持时间为 10.5×TADCK

    /*(2)设置 ADC1 通道使能*/
    ADC1_EnableScanChannel(ADC1_CH_13_MSK);
    SYS_SET_IOCFG(IOP21CFG,SYS_IOCFG_P21_AN13);     /*设置 P21 为输入模式，ADC1 采集 P21*/

    /*(3)设置 ADC1 中断*/
    ADC1_EnableChannelInt(ADC1_CH_13_MSK);          /*开 AN13 转换中断*/
    NVIC_EnableIRQ(ADC1_IRQn);
    /*(4)设置优先级*/
    NVIC_SetPriority(ADC1_IRQn,3);                  /*优先级 0~3， 0 最高、3 最低*/

    /*(5)开启 ADC1*/
    ADC1_Start();

    /*(6)开启 ADC1 校准*/
    ADC1_StartAdjust();

    /*(7)使用 ADC1 校准值的 ADC 转换*/
    ADC1_EnableAdjust();

    /*(8)设置 P14 指示 ADC1 中断*/
    SYS_SET_IOCFG(IOP14CFG, SYS_IOCFG_P14_GPIO);
    GPIO_CONFIG_IO_MODE(GPIO1,GPIO_PIN_4,GPIO_MODE_OUTPUT);     /*指示 ADC1 中断*/
    GPIO1->DO_f.P4 = 1;
}
```

```
void ADC1_IRQHandler(void)
{
    if(ADC1_GetChannelIntFlag(ADC1_CH_13))                    /*判断 ADC1 通道中断状态*/
    {
        GPIO1->DO_f.P4 = 1;                                    /*P14 置 1*/
        ADC1_ClearChannelIntFlag(ADC1_CH_13);                  /*清除 ADC1 通道中断标志*/
        GPIO1->DO_f.P4 = 0;                                    /*P14 清 0*/
    }
}
```

4.3 波形图

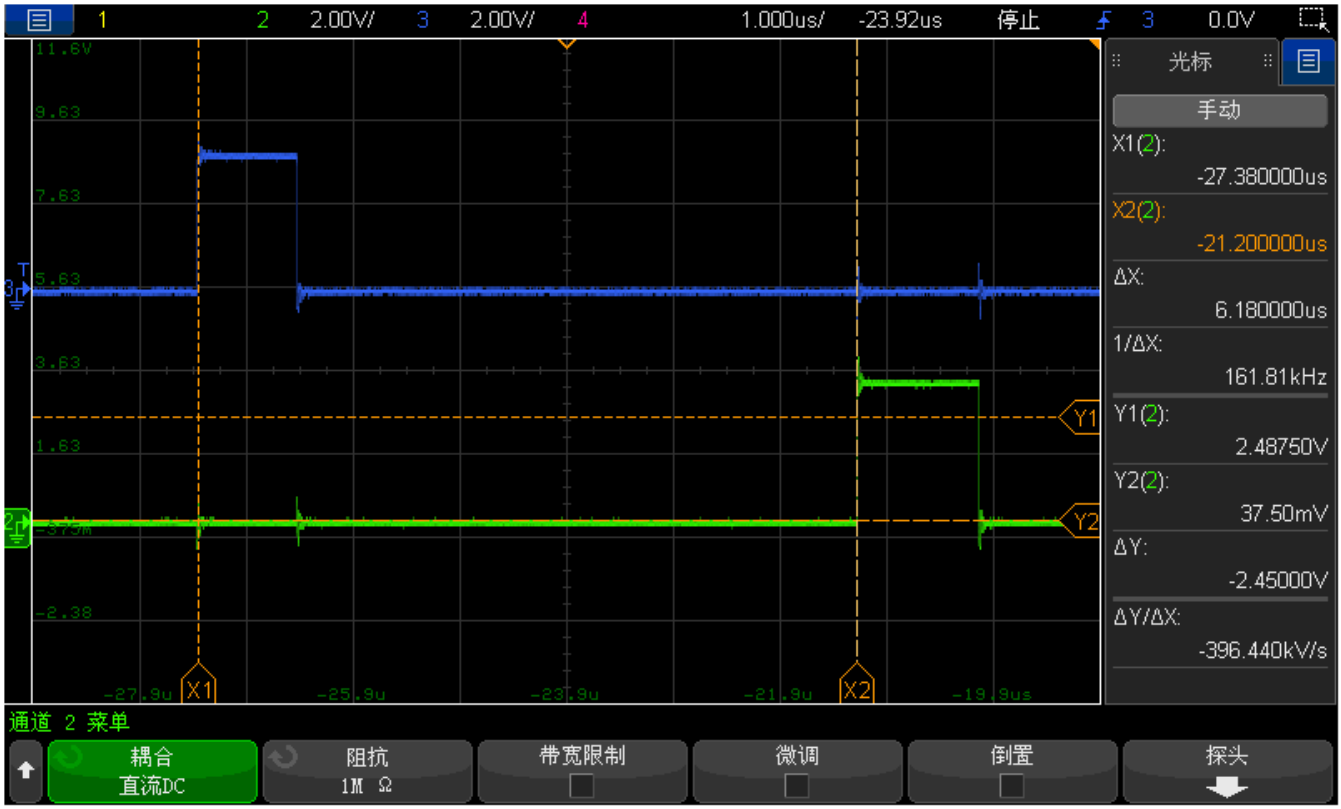

图 4-1：软件触发波形图

如上图 4-1 所示，蓝线代表 P13（软件触发信号），绿线代表 P14（ADC1 软件中断信号）。

5. 注意事项

1) CMS32M53xx/55xx 系列 ADC 模块部分寄存器为被保护的寄存器，可对寄存器进行加锁或解锁操作。详情可参考芯片手册。
2) ADC 转换未结束期间，其他的任何触发信号将被忽略。
3) ADC 采样选择内部模拟量 Bandgap（1.2V）时，ADC 时钟分频应选择 128 分频。
4) 上电后 ADC 模块建议校准一次（注：ADC 模块时钟四分频时只允许校准一次）。
5) ADC 校准期间避免触发 ADC 转换（硬件触发和软件触发）；例如使用 EPWM 硬件触发 ADC1，则需先配置 ADC1 模块，再配置 EPWM 模块。
6) ADC 时钟相关寄存器仅允许在 ADC 停止状态（ADCEN=0）下更改；例如在 ADC 启动后，需要对相关寄存器修改，则应先停止 ADC（ADCEN=0），再更改相关寄存器，最后重新启动 ADC（ADCEN=1）。

6. 更多信息

更多信息，请登录中微半导体网站查看 http://www.mcu.com.cn。

7. 版本修订说明

版本号	时间	修改内容
V1.00	2021 年 12 月	初始版本