CMS32M53xx/55xx 应用笔记

低速模数转换器（ADC0）

Rev. 1.00

目录

1. 概述

1.1 目的

本文档介绍了 CMS3253xx/55xx 低速模数转换器 ADC0 的特性以及如何实现软/硬件触发低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
ADC0	低速模数转换器
PCLK	APB 时钟
Timer0	定时器 0
T_{ADCK}	ADC0 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 ADC0 硬件触发模式。

第 4 章 ADC0 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：100Ksps
- 多达 24 路单端模拟输入通道
- 单次转换时间为：18.5×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 ADC0 软件触发启动

使用软件触发 ADC0 需要将寄存器 ADCCON2.ADCST 位中写入 1，将启动 ADC0 转换。转换完毕后，该位硬件自动清零。

在 ADC0 转换期间，任何软件和硬件触发启动信号将被忽略。

2.3 ADC0 硬件触发启动

除了软件触发 ADC0 转换，该 ADC0 模块还提供了硬件触发启动的方式。硬件触发包括外部触发和内部触发。

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 寄存器说明

详情可参考芯片手册。

3. ADC0 硬件触发模式

3.1 操作实例

实例目标：实现 Timer0 中断信号硬件触发 ADC0 转换。

操作步骤：

1) 设置芯片 APB 时钟（PCLK）为 48Mhz。
2) 开启 ADC0 模块时钟使能位。
3) 设置 ADC0 模块为连续转换模式，时钟为 APB 时钟(PCLK)的 8 分频。
4) 设置 ADC0 转换通道。
5) 设置 ADC0 硬件触发模式。
6) 设置 ADC0 中断以及中断优先级。
7) 开启 ADC0。
8) 设置 P14 为输出模式，用于指示 ADC0 中断。
9) 开启 Timer0 模块时钟使能位。
10) 设置 Timer0 模块时钟为 APB 时钟（PCLK）的 1 分频。
11) 设置 Timer0 模块为周期运行模式，设置 Timer0 模块运行周期。
12) 设置 Timer0 模块周期点中断，设置中断优先级。
13) 开启 Timer0。
14) 设置 P13 为输出模式，用于指示 Timer0 中断。
15) 在 ADC0 中断服务函数执行 P14 输出电平翻转。
16) 在 Timer0 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*/

    ADC0_TMR_Trigger_Mode();            /*设置 ADC0 定时器触发模式*/
    TMR0_Period_Count_Config();         /*设置定时器 0 定时模式*/

    while(1)
    {
        ;
    }

}

void ADC0_TMR_Trigger_Mode(void)
{
    /*(1)设置 ADC0 时钟*/
    SYS_EnablePeripheralClk(SYS_CLK_ADC0_MSK);                          /*使能 ADC0 模块时钟*/
    ADC0_ConfigRunMode(ADC0_CONVERT_CONTINUOUS,ADC0_CLK_DIV_8);     //使能连续转换，TADCK=PCLK/8

    /*(2)设置 ADC0 通道使能*/
    ADC0_EnableChannel(ADC0_CH_13_MSK);                             /*选择 AN13*/
    SYS_SET_IOCFG(IOP21CFG,SYS_IOCFG_P21_AN13);                     /*关闭 P21 的数字功能*/

    /*(3)设置 ADC0 转换触发方式*/
    ADC0_EnableHardwareTrigger(ADC0_TG_INTNEL_TMR0);                /*定时器 0 触发*/

    /*(4)设置 ADC0 中断*/
    ADC0_EnableChannelInt(ADC0_CH_13_MSK);                              /*开 AN13 转换中断*/
    NVIC_EnableIRQ(ADC0_IRQn);

    /*(5)设置优先级*/
    NVIC_SetPriority(ADC0_IRQn,3);                                  /*优先级 0~3， 0 最高、3 最低*/

    /*(6)开启 ADC0*/
    ADC0_Start();

    /*(7)设置 P14 指示 ADC0 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 ADC0_IRQHandler(void)
{
    if(ADC0_GetChannelIntFlag(ADC0_CH_13))                  /*判断 ADC0 中断触发通道*/
    {
        GPIO1->DO_f.P4 =1;                                      /*P14 置 1*/
        ADC0_ClearChannelIntFlag(ADC0_CH_13);                   /*清除 ADC0 中断通道标志位*/
        GPIO1->DO_f.P4 =0;                                      /*P14 清 0*/
    }
}
```

```
void TMR0_Period_Count_Config(void)
{
    /*(1)设置 Timer0  的时钟*/
    SYS_EnablePeripheralClk(SYS_CLK_TIMER01_MSK);              /*打开 Timer0/1 的时钟*/
    TMR_ConfigClk(TMR0,TMR_CLK_SEL_APB,TMR_CLK_DIV_1);         /*Timer0 时钟为 PCLK/1*/

    /*(2)设置 Timer0  运行模式*/
    TMR_ConfigRunMode(TMR0,TMR_COUNT_PERIOD_MODE, TMR_BIT_32_MODE);    /*周期计数模式*/
    TMR_DisableOneShotMode(TMR0);                                      /*关闭单次模式*/

    /*(3)设置 Timer0  运行周期*/
    TMR_SetPeriod(TMR0,4800);                  /* 100us*/

    /*(4)设置 Timer0  中断*/
    TMR_EnableOverflowInt(TMR0);
    NVIC_EnableIRQ(TMR0_IRQn);

    /*(5)设置 Timer0  中断优先级*/
    NVIC_SetPriority(TMR0_IRQn,2);

    /*(6)开启 Timer0*/
    TMR_Start(TMR0);

    /*(7)设置 P13 指示定时器 0 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 TMR0_IRQHandler(void)
{
    if(TMR_GetOverflowIntFlag(TMR0))                    /*判断 Timer0 中断标志*/
    {
        GPIO1->DO_f.P3 = 1;                                 /*P13 置 1*/
        TMR_ClearOverflowIntFlag(TMR0);                   /*清除 Timer0 中断标志位*/
        GPIO1->DO_f.P3 = 0;                                 /*P13 清 0*/
    }
}
```

3.3 波形图

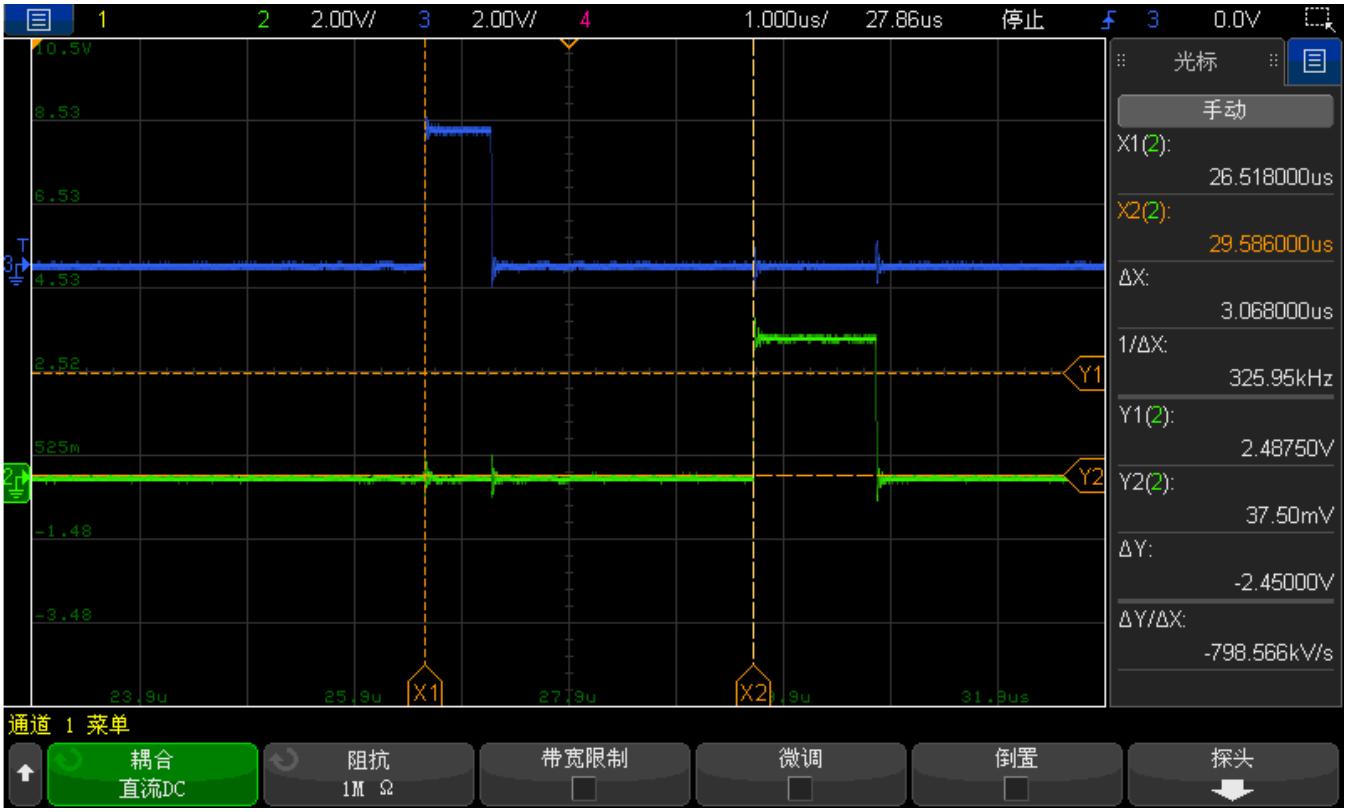

图 3-1：硬件触发波形图

如上图 3-1 所示，蓝线代表 P13（Timer0 溢出中断信号），绿线代表 P14（ADC0 硬件中断信号）。

4. ADC0 软件触发模式

4.1 操作实例

实例目标：实现软件触发 ADC0 中断。

操作步骤：

1. 设置芯片 APB 时钟（PCLK）为 48Mhz。
2. 开启 ADC0 模块时钟使能位。
3. 设置 ADC0 模块为连续转换模式，时钟为 APB 时钟(PCLK)的 8 分频。
4. 设置 ADC0 转换通道。
5. 设置 ADC0 中断及中断优先级。
6. 开启 ADC0。
7. 设置 P14 为输出模式，用于指示 ADC0 中断。
8. 设置 P13 为输出模式，用于指示软件触发信号。
9. 在主函数 while(1)中循环软件触发 ADC0。

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*/

    ADC0_Software_Trigger_Mode();                           /*设置 ADC0 定时器触发模式*/
    /*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 = 1;

    while(1)
    {
        for(i = 240; i>0; i--);
        if( !ADC0_IS_BUSY() )                               /*判断 ADC 是否正在转换*/
        {
            GPIO1->DO_f.P3 = 1;                             /*P13 置 1*/
            ADC0_Go();                                      /*软件启动 ADC0*/
            GPIO1->DO_f.P3 = 0;                             /*P13 清 0*/
            while( ADC0_IS_BUSY() );                        /*等待 ADC 转换结束*/
        }
    }

}
```

```
void ADC0_Software_Trigger_Mode(void)
{
    /*(1)设置 ADC0 时钟*/
    SYS_EnablePeripheralClk(SYS_CLK_ADC0_MSK);                          /*使能 ADC0 模块时钟*/
    ADC0_ConfigRunMode(ADC0_CONVERT_CONTINUOUS,ADC0_CLK_DIV_8);//使能连续转换,TADCK=PCLK/8

    /*(2)设置 ADC0 通道使能*/
    ADC0_EnableChannel(ADC0_CH_13_MSK);                         /*选择 AN13*/
    SYS_SET_IOCFG(IOP21CFG,SYS_IOCFG_P21_AN13);                 /*关闭 P21 的数字功能*/

    /*(3)设置 ADC0 中断*/
    ADC0_EnableChannelInt(ADC0_CH_13_MSK);                          /*开 AN13 转换中断*/
    NVIC_EnableIRQ(ADC0_IRQn);

    /*(4)设置优先级*/
    NVIC_SetPriority(ADC0_IRQn,2);                                  /*优先级 0~3， 0 最高、3 最低*/

    /*(5)开启 ADC0*/
    ADC0_Start();

    /*(6)设置 P14 为输出模式，指示 ADC0 中断*/
    SYS_SET_IOCFG(IOP14CFG,SYS_IOCFG_P14_GPIO);                     /*设置 P14 为 GPIO 模式*/
    GPIO_CONFIG_IO_MODE(GPIO1,GPIO_PIN_4,GPIO_MODE_OUTPUT);
    GPIO1->DO_f.P4 = 1;
}

void ADC0_IRQHandler(void)
{
    if(ADC0_GetChannelIntFlag(ADC0_CH_13))                      /*判断 ADC0 中断触发通道*/
    {
        GPIO1->DO_f.P4 = 1;                                         /*P14 置 1*/
        ADC0_ClearChannelIntFlag(ADC0_CH_13);                   /*清除 ADC0 中断通道标志位*/
        GPIO1->DO_f.P4 = 0;                                         /*P14 置 1*/
    }
}
```

4.3 波形图

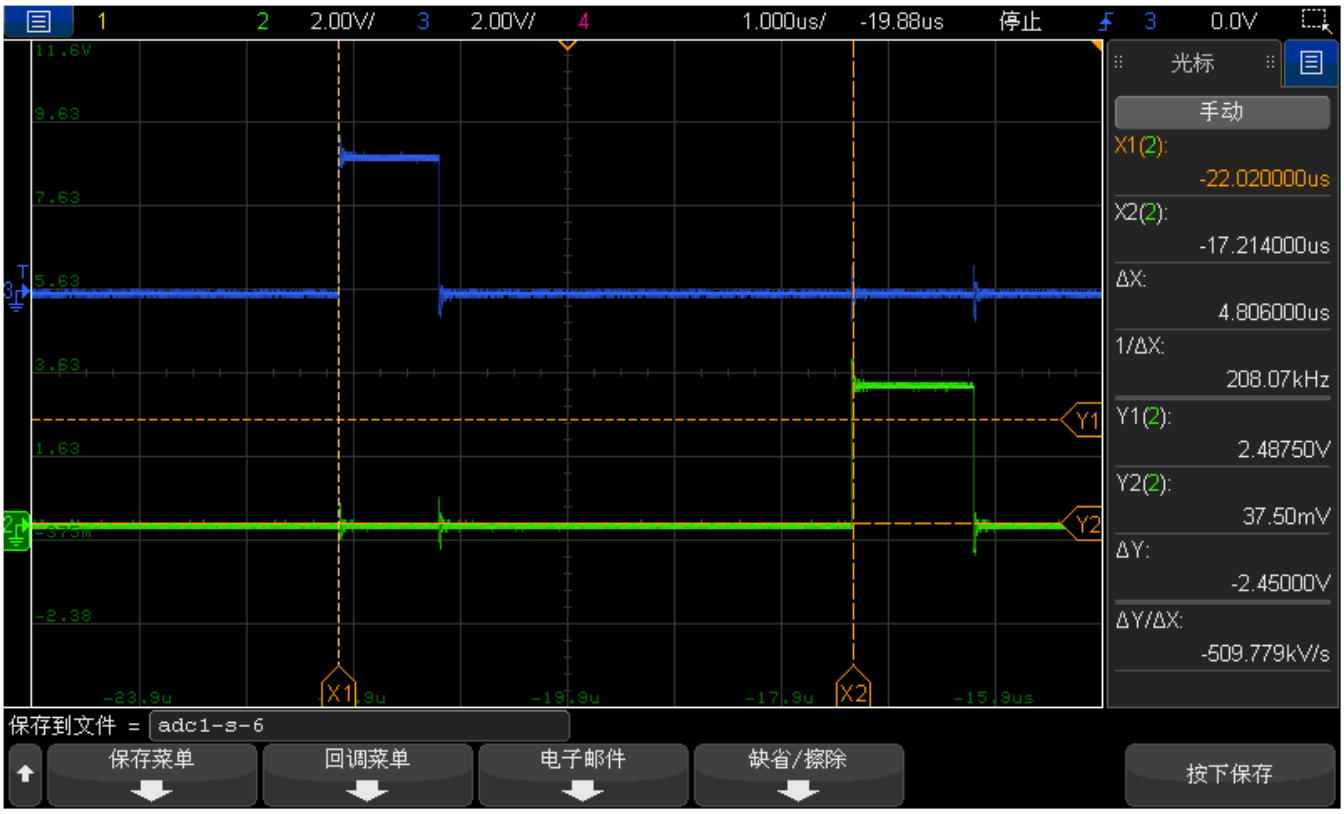

图 4-1：软件触发波形图

如上图 4-1 所示，蓝线代表 P13（软件触发信号）,绿线代表 P14（ADC0 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) 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