8051 ADC 模块应用笔记

模数转换器（ADC）

Rev. 1.01

目录

1. 概述

1.1 目的

本文档介绍了中微 8051 芯片 ADC 模块的特性，如何实现软/硬件触发模数转换器进行模数转换，以及相关应用注意事项。

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
ADC	模数转换器
PWM	脉冲宽度调制
ADC-LDO	ADC 参考电压

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 ADC 硬件触发模式。

第 4 章 ADC 软件触发模式。

第 5 章 ADC 软件触发结合硬件触发。

第 6 章 注意事项。

2. 模块概述

模数转换器（ADC）可以将模拟输入信号转换为表示该信号的一个 12 位二进制数，ADC 结构框图如下图所示。

端口模拟输入信号和内部模拟信号经过多路选择器之后与模数转换器的输入相连。模数转换器采用逐次逼近法产生一个 12 位二进制结果，并将该结果保存在 ADC 结果寄存器（ADRESL 和 ADRESH）中，ADC 在转换完成之后可以产生一个中断。ADC 转换结果与 ADC 比较数据寄存器（ADCMPL 和 ADCMPH）的值进行比较，比较的结果存放在 ADCMPO 标志位中。

ADC 参考电压始终为内部产生，可选择 AVDD 提供，也可由内部 ADC-LDO 提供。

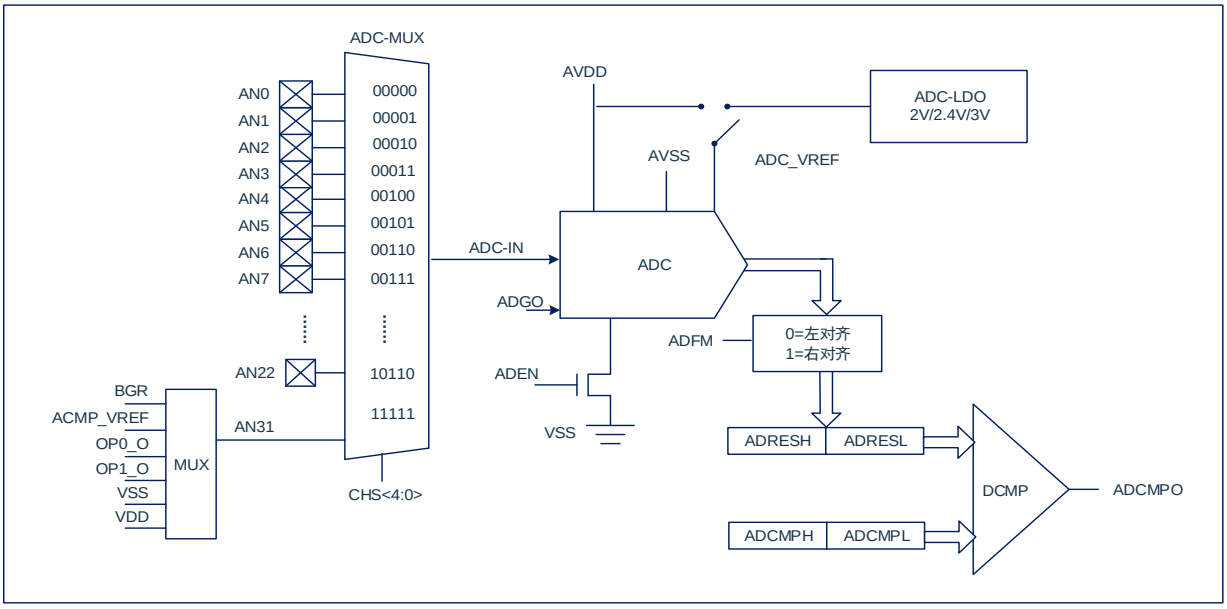

图 2-1：ADC 结构框图

注：部分 8051 芯片没有运算放大器和比较器模块，具体的 ADC 结构框图以手册为准。

2.1 模块特性

配置和使用 ADC 时，必须考虑如下因素：

- 端口配置
- 通道选择
- ADC 转换时钟源
- 中断控制
- 结果的存储格式

2.2 ADC 软件触发启动

使用软件触发 ADC 需要将 ADGO 置 1，转换完毕后 ADGO 位将自动清零。

2.3 ADC 硬件触发启动

除了软件触发 ADC 转换，该 ADC 模块还提供了硬件触发启动的方式。一种为外部端口边沿触发方式，另一种为 PWM 的边沿或周期触发方式。

2.4 寄存器说明

详情可参考芯片手册。

3. ADC 硬件触发模式

3.1 操作实例

实例目标：实现 PWM 零点触发 ADC 转换。

操作步骤：

1) 配置 ADC 运行模式。
2) 配置 ADC 转换通道。
3) 使能 LDO 位，选择 ADC 参考电压。
4) 使能 ADC 硬件触发位，同时选择 ADC 硬件触发源(PWM 零点触发)。
5) 开启 ADC 中断。
6) 开启 ADC。
7) 配置 P01 用于指示 ADC 中断。
8) 配置 PWM 运行模式。
9) 配置 PWM 周期和占空比。
10) 配置 PWM 输出极性。
11) 配置 PWM 输出 IO 口。
12) 开启 PWM。
13) 在 ADC 中断服务函数中翻转 P01 输出电平。

3.2 样例代码

```
int main(void)
{
        ADC_Config();                                           //ADC 配置
        PWM_Config();                                           //PWM 配置

        while(1)
        {
                ;
        }
}

void ADC_Config(void)
{
        /* (1)设置 ADC 的运行模式 */
        ADC_ConfigRunMode(ADC_KS4_1,ADC_CLK_KS4_1_DIV_6,ADC_RESULT_RIGHT);

        /* (2)设置 ADC 转换通道 */
        ADC_EnableChannel(ADC_CH_0);
        GPIO_SET_MUX_MODE(P00CFG, GPIO_P00_MUX_AN0);

        /* (3)设置 ADC LDO */
        ADC_EnableLDO();
        ADC_ConfigADCVref(ADC_VREF_3V);

        /* (4)设置 ADC 触发方式 */
        ADC_EnableHardwareTrig();
        ADC_ConfigHardwareTrig(ADC_TG_PWM0, ADC_TG_PWM_PE);   //PWM0 零点触发

        /* (5)设置 ADC 中断 */
        ADC_EnableInt();
        IRQ_SET_PRIORITY(IRQ_ADC,IRQ_PRIORITY_HIGH);
        IRQ_ALL_ENABLE();

        /* (6)开启 ADC */
        ADC_Start();

        /* (7)配置 P01 以用于观察 ADC*/
        GPIO_SET_MUX_MODE(P01CFG, GPIO_MUX_GPIO);
        GPIO_ENABLE_OUTPUT(P0TRIS, GPIO_PIN_1);
        P01 = 1;
}
```

```
void PWM_Config(void)
{
	/* (1)设置 EPWM 为同步模式*/
	EPWM_ConfigRunMode((EPWM_WFG_COMPLEMENTARY    |        /*设置为互补模式*/
	                    EPWM_COUNT_UP_DOWN     |
	                    EPWM_OCU_SYMMETRIC     |          /*设置为中心对称计数模式*/
	                    EPWM_OC_INDEPENDENT));           /*通道各自独立控制*/

	/* (2)设置 EPWM 通道周期以及占空比*/
	EPWM_ConfigChannelClk(EPWM0,EPWM_CLK_DIV_1);
	EPWM_ConfigChannelPeriod(EPWM0 , 4800);
	EPWM_ConfigChannelSymDuty(EPWM0, 1200);
	EPWM_EnableAutoLoadMode(EPWM_CH_0_MSK);

	/* (3)设置 EPWM 输出极性      */
	EPWM_DisableReverseOutput(EPWM_CH_0_MSK | EPWM_CH_1_MSK);

	/* (4)设置 EPWM io 口复用*/
	GPIO_SET_MUX_MODE(P13CFG, GPIO_MUX_PG0);
	GPIO_SET_MUX_MODE(P14CFG, GPIO_MUX_PG1);
	EPWM_EnableOutput(EPWM_CH_0_MSK | EPWM_CH_1_MSK);

	/* (5)开启 PWM 计数*/
	EPWM_Start(EPWM_CH_0_MSK);
}

void ADC_IRQHandler(void)  interrupt ADC_VECTOR
{
	if(ADC_GetIntFlag())
	{
		P01 ^= 1;
		ADC_ClearIntFlag();
	}
}
```

3.3 波形图

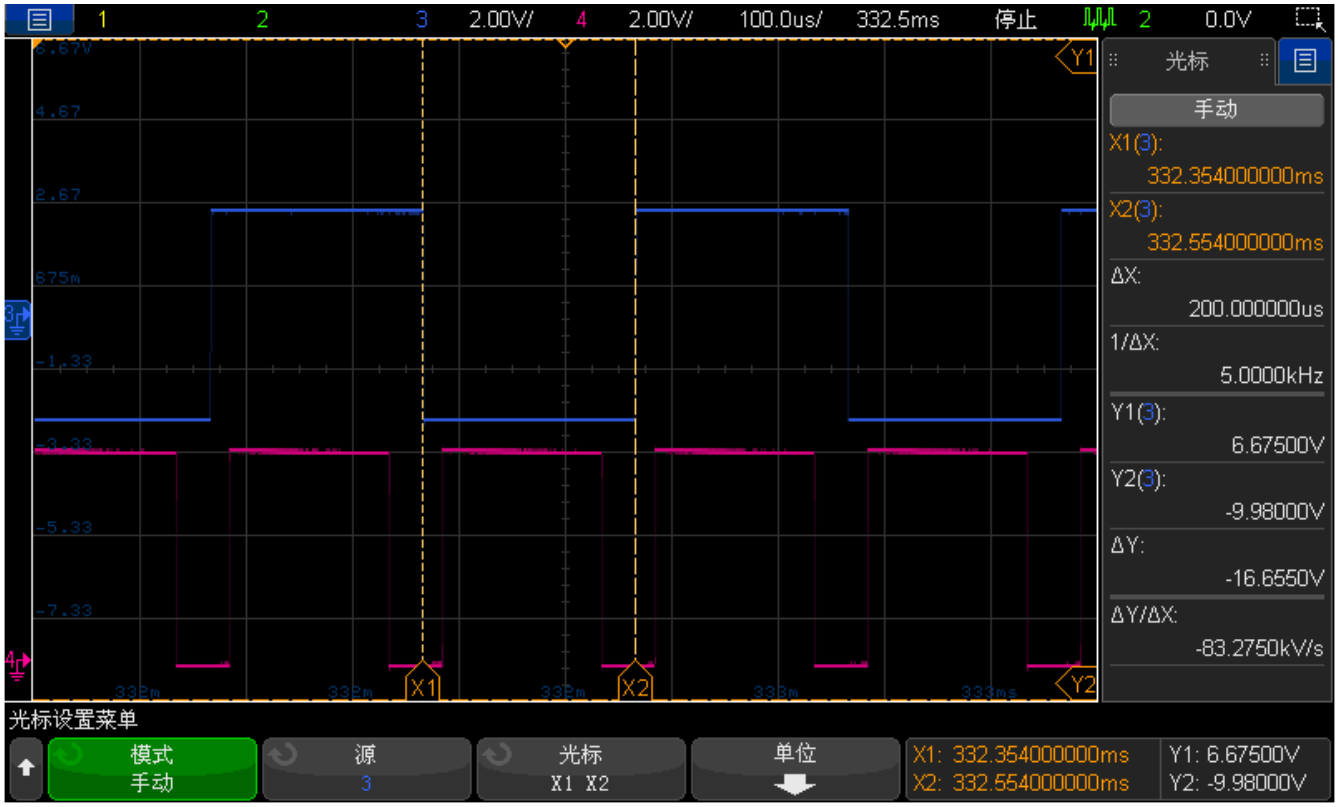

图 3-1：硬件触发波形图

如上图 3-1 所示，蓝线代表 P01（ADC 硬件中断信号），红线代表 PWM0 输出波形。

4. ADC 软件触发模式

4.1 操作实例

实例目标：实现软件触发 ADC 转换。

操作步骤：

1) 配置 ADC 运行模式。
2) 配置 ADC 转换通道。
3) 使能 LDO 位，选择 ADC 参考电压。
4) 开启 ADC 中断。
5) 开启 ADC。
6) 配置 P01 用于指示 ADC 中断。
7) 配置 P02 用于指示软件触发信号。
8) 在主程序中循环软件触发 ADC 转换，同时在触发前后置位 P02。
9) 在 ADC 中断服务函数中翻转 P01 输出电平。

4.2 样例代码

```
int main(void)
{
	ADC_Config();                                    //ADC 配置
	GPIO_SET_MUX_MODE(P02CFG, GPIO_MUX_GPIO);        //配置 P02 用于观察软件触发信号
	P02 = 0;
	GPIO_ENABLE_OUTPUT(P0TRIS, GPIO_PIN_2);
	while(1)
	{
		if(!ADC_IS_BUSY)                             //判断是否正在转换
		{
			_nop_();
			P02 = 1;
			ADC_GO();                                //开启 ADC 转换
			P02 = 0;
			while(ADC_IS_BUSY);                      //等到 ADC 转换结束
		}
	}
}

void ADC_Config(void)
{
	/* (1)设置 ADC 的运行模式 */
	ADC_ConfigRunMode(ADC_KS4_1,ADC_CLK_KS4_1_DIV_6,ADC_RESULT_RIGHT);

	/* (2)设置 ADC 转换通道 */
	ADC_EnableChannel(ADC_CH_0);
	GPIO_SET_MUX_MODE(P00CFG, GPIO_P00_MUX_AN0);

	/* (3)设置 ADC LDO */
	ADC_EnableLDO();
	ADC_ConfigADCVref(ADC_VREF_3V);

	/* (4)设置 ADC 中断 */
	ADC_EnableInt();
	IRQ_SET_PRIORITY(IRQ_ADC,IRQ_PRIORITY_HIGH);
	IRQ_ALL_ENABLE();

	/* (5)开启 ADC */
	ADC_Start();

	/* (6)配置 P01 以用于观察 ADC*/
	GPIO_SET_MUX_MODE(P01CFG, GPIO_MUX_GPIO);
	GPIO_ENABLE_OUTPUT(P0TRIS, GPIO_PIN_1);
	P01 = 1;
}

void ADC_IRQHandler(void)  interrupt ADC_VECTOR
{
	if(ADC_GetIntFlag())
	{
		P01 ^= 1;
		ADC_ClearIntFlag();        //清除 ADC 中断标志
	}
}
```

4.3 波形图

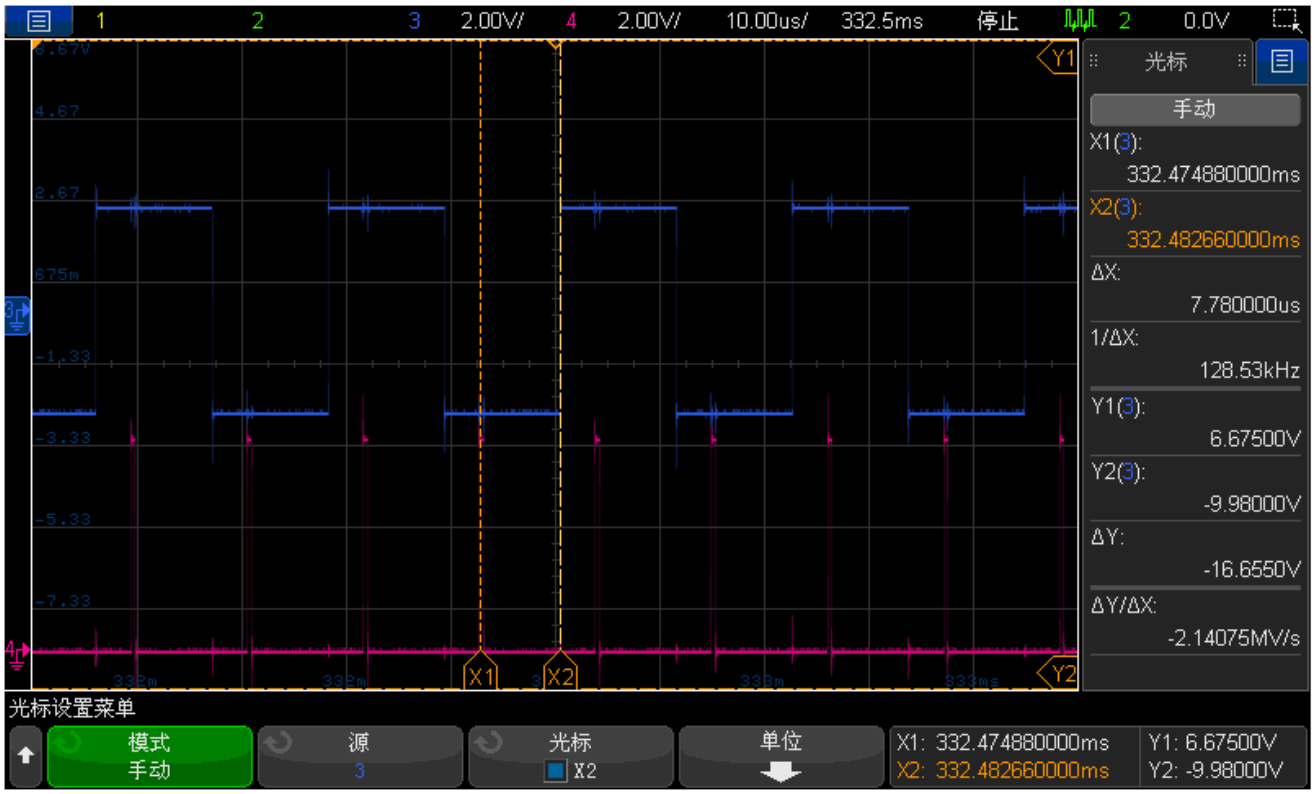

图 4-1：软件触发波形图

如上图 4-1 所示，蓝线代表 P01（ADC 软件中断信号），红线代表 P02（ADC 软件触发信号）。

5. ADC 软件触发结合硬件触发

5.1 操作实例

实例目标：实现 PWM 零点触发 ADC 转换，在主循环中软件触发 ADC 转换。

操作步骤：

1) 配置 ADC 运行模式。
2) 配置 ADC 转换通道。
3) 使能 LDO 位，选择 ADC 参考电压。
4) 使能 ADC 硬件触发位，同时配置 ADC 硬件触发方式（PWM 零点触发）。
5) 开启 ADC 中断。
6) 开启 ADC。
7) 配置 P01 用于指示 ADC 中断。
8) 配置 PWM 运行模式。
9) 配置 PWM 周期和占空比。
10) 配置 PWM 输出极性。
11) 配置 PWM 输出 IO 口。
12) 开启 PWM。
13) 配置 P02 用于指示软件触发信号。
14) 在主程序中循环软件触发 ADC 转换，同时在触发前后置位 P02。
15) 在 ADC 中断服务函数中翻转 P01 输出电平。

5.2 样例代码

```
int main(void)
{
	uint16_t i;
	ADC_Config();                                   //ADC 配置
	PWM_Config();                                   //PWM 配置
	GPIO_SET_MUX_MODE(P02CFG, GPIO_MUX_GPIO);       //配置 P02 用于观察软件触发信号
	P02 = 0;
	GPIO_ENABLE_OUTPUT(P0TRIS, GPIO_PIN_2);
	while(1)
	{
		for(i = 200; i >= 1; i--);
		if(!ADC_IS_BUSY)                            //判断 ADC 是否正在转换
		{
			_nop_();
			P02 = 1;
			ADC_DisableHardwareTrig();          //关闭硬件触发
			ADC_GO();                           //软件触发 ADC 转换
			P02 = 0;
			while(ADC_IS_BUSY);                 //等待 ADC 转换完毕
			ADC_EnableHardwareTrig();           //开启硬件触发
		}
	}
}

void ADC_Config(void)
{
	/* (1)设置 ADC 的运行模式 */
	ADC_ConfigRunMode(ADC_KS4_1,ADC_CLK_KS4_1_DIV_6,ADC_RESULT_RIGHT);

	/* (2)设置 ADC 转换通道 */
	ADC_EnableChannel(ADC_CH_0);
	GPIO_SET_MUX_MODE(P00CFG, GPIO_P00_MUX_AN0);

	/* (3)设置 ADC LDO */
	ADC_EnableLDO();
	ADC_ConfigADCVref(ADC_VREF_3V);

	/* (4)设置 ADC 触发方式 */
	ADC_EnableHardwareTrig();
	ADC_ConfigHardwareTrig(ADC_TG_PWM0, ADC_TG_PWM_PE);   //PWM0 零点触发

	/* (5)设置 ADC 中断 */
	ADC_EnableInt();
	IRQ_SET_PRIORITY(IRQ_ADC,IRQ_PRIORITY_HIGH);
	IRQ_ALL_ENABLE();

	/* (6)开启 ADC */
	ADC_Start();

	/* (7)配置 P01 以用于观察 ADC*/
	GPIO_SET_MUX_MODE(P01CFG, GPIO_MUX_GPIO);
	GPIO_ENABLE_OUTPUT(P0TRIS, GPIO_PIN_1);
	P01 = 1;
}
```

```
void PWM_Config(void)
{
	/* (1)设置 EPWM 为同步模式*/
	EPWM_ConfigRunMode((EPWM_WFG_COMPLEMENTARY    |        /*设置为互补模式*/
	                    EPWM_COUNT_UP_DOWN     |
	                    EPWM_OCU_SYMMETRIC     |    /*设置为中心对称计数模式*/
	                    EPWM_OC_INDEPENDENT));             /*通道各自独立控制*/

	/* (2)设置 EPWM 通道周期以及占空比*/
	EPWM_ConfigChannelClk(EPWM0,EPWM_CLK_DIV_1);
	EPWM_ConfigChannelPeriod(EPWM0 , 4800);
	EPWM_ConfigChannelSymDuty(EPWM0, 1200);
	EPWM_EnableAutoLoadMode(EPWM_CH_0_MSK);

	/* (3)设置 EPWM 输出极性       */
	EPWM_DisableReverseOutput(EPWM_CH_0_MSK | EPWM_CH_1_MSK);

	/* (4)设置 EPWM io 口复用*/
	GPIO_SET_MUX_MODE(P13CFG, GPIO_MUX_PG0);
	GPIO_SET_MUX_MODE(P14CFG, GPIO_MUX_PG1);
	EPWM_EnableOutput(EPWM_CH_0_MSK | EPWM_CH_1_MSK);

	/* (5)开启 PWM 计数*/
	EPWM_Start(EPWM_CH_0_MSK);
}

void ADC_IRQHandler(void)  interrupt ADC_VECTOR
{
	if(ADC_GetIntFlag())
	{
		P01 ^= 1;
		ADC_ClearIntFlag();
	}
}
```

5.3 波形图

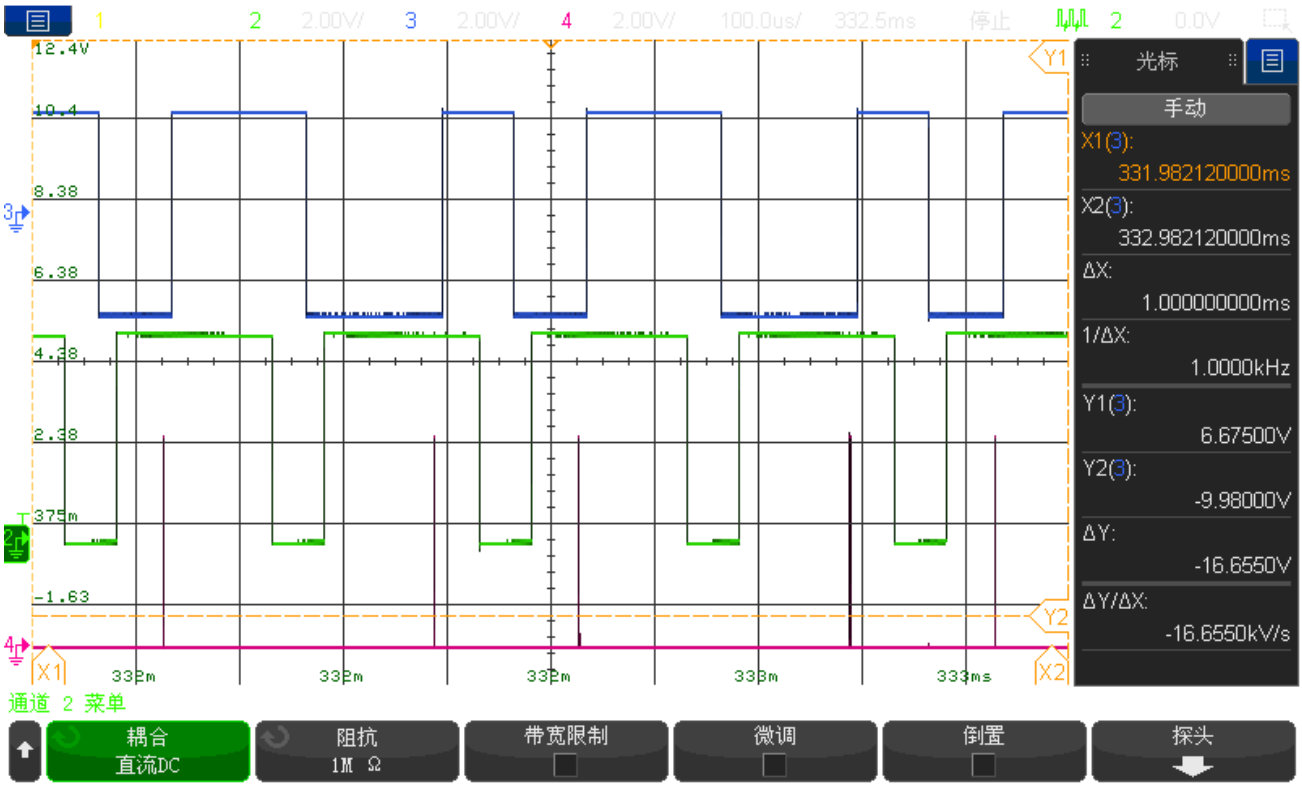

图 5-1：软件触发结合硬件触发波形图

如上图 5-1 所示，蓝线代表 P01（ADC 中断信号），绿线代表 PWM0 输出波形，红线代表 P02（ADC 软件触发信号）。

6. 注意事项

1) ADC 在转换过程中忽略该时间段内所有硬件和软件触发信号。
2) 使用 ADC 硬件触发时，相邻两触发信号时间间隔应大于 ADC 转换时间，避免在 ADC 转换期间产生触发信号。
3) 使用 ADC 软件触发时，应遵循以下流程图：

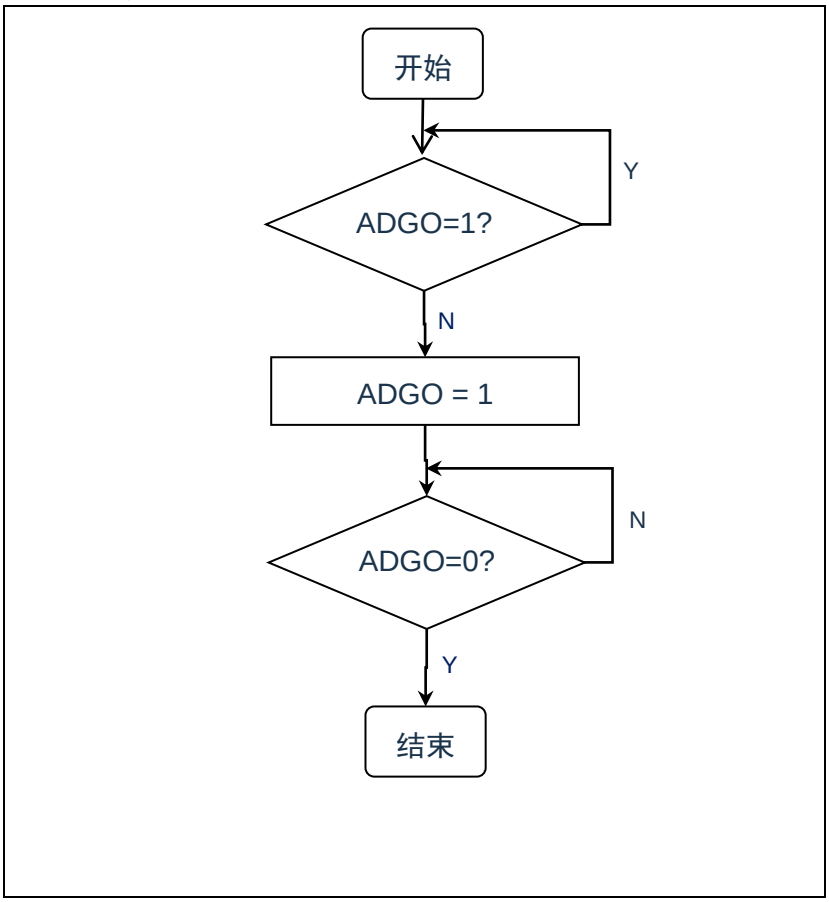

图 6-1：ADC 软件触发流程图

代码如下：

```
if(!ADC_IS_BUSY)                    //判断 ADC 是否正在转换
{
        _nop_();
        ADC_GO();                   //软件触发 ADC 转换
        while(ADC_IS_BUSY);         //等待 ADC 转换完毕
}
```

4) 当程序中同时使用软件和硬件触发 ADC 时，应在软件触发 ADC 转换前关闭硬件触发。

```
if(!ADC_IS_BUSY)                            //判断 ADC 是否正在转换
{
        _nop_();
        ADC_DisableHardwareTrig();          //关闭硬件触发
        ADC_GO();                           //软件触发 ADC 转换
        while(ADC_IS_BUSY);                 //等待 ADC 转换完毕
        ADC_EnableHardwareTrig();           //开启硬件触发
}
```

上述应用注意事项 1)、2)、3)、4)适用于以下产品型号：

1. CMS8S3660/3680
2. CMS8S5880/5887/5888/5889
3. CMS8S6980/6990
4. CMS8M35XX
5. CMS8S78XX
6. CMS80F261X
7. CMS80F253X
8. CMS80F251X/751X
9. CMS8S6997/6998/6999
10. CMS8S589X, CMS8S369X, CMS80F1616

5) ADC 分频比操作注意事项：检测到上一次 ADC 转换完成后，如果需要对 ADC 分频比进行再次操作，建议至少等待 4 个 ADC 转换时钟 Tadck。说明如下：

- ✓ 如图 6-2 e 步骤⑥ 所示：检测到上一次 ADC 转换完成后，如果需要对 ADC 分频比进行再次操作，建议至少等待 4 个 ADC 转换时钟 Tadck；
- ✓ 如图 6-2 a、b、c、d 所示：再次使用 ADC 时，只要不操作 ADC 分频比，不需要延时。

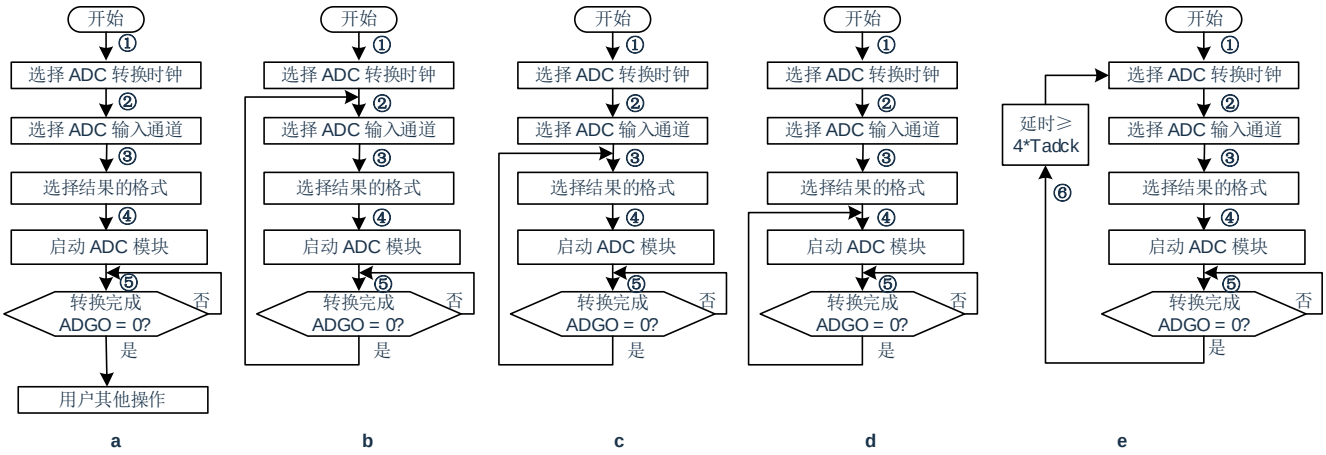

图 6-2：ADC 分频比操作注意示意图

上述应用注意事项 5)适用于所有 8051 产品型号。

7. 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。

8. 版本修订说明

版本号	时间	修改内容
V1.00	2021 年 11 月	初始版本
V1.01	2022 年 8 月	增加 6 注意事项 5)