32 位微控制器

HC32M120 系列的模数转换器 ADC

适用对象

系列	产品型号
HC32M120	HC32M120J6TB HC32M120F6TB

目　录

1　摘要 4
2　ADC 简介 4
2.1　功能简介 4
2.2　主要特性 5
2.3　引脚配置 6
3　ADC 应用 7
3.1　模拟输入引脚与通道的对应关系 7
3.2　模拟输入的采样时间和转换时间 8
3.3　模式和功能 8
3.4　触发源选择 8
3.5　序列 A 单次扫描模式 9
3.5.1　说明 9
3.5.2　应用 9
3.6　序列 A 连续扫描模式 10
3.6.1　说明 10
3.6.2　应用 10
3.7　双序列扫描模式 11
3.7.1　说明 11
3.7.2　应用 13
3.8　模拟看门狗 14
3.8.1　说明 14
3.8.2　应用 14
3.9　扩展通道 15
3.9.1　说明 15
3.9.2　应用 15
3.10　运算放大器 OPA 16
3.10.1　说明 16
3.10.2　应用 16
4　例程讲解 17
4.1　基本流程 17
4.2　应用程序源码说明 18

4.2.1 应用程序基本结构 18
4.2.2 重要源码讲解 19
4.2.3 程序执行现象 23
5 总结 24
6 版本信息 & 联系方式 25

1 摘要

本应用笔记主要介绍 HC32M120 系列 MCU 的模数转换器（以下简称 ADC）的特点及使用方法，包括基本用法、外部触发源的用法、扩展通道的用法和模拟看门狗的用法。

2 ADC 简介

2.1 功能简介

HC32M120 系列 MCU 内部集成一个 ADC 模块（系统框图如图 2-1），可配置 12 位、10 位和 8 位分辨率，支持最多 12 个外部模拟输入通道。这些模拟输入通道可以任意组合成一个序列（序列 A 或序列 B），一个序列可以进行单次扫描（扫描包括两个动作：采样和转换），或连续扫描。ADC 模块还搭载模拟看门狗（以下简称 AWD）功能，可对指定通道的转换结果进行监视，检测是否超出设定的阀值。

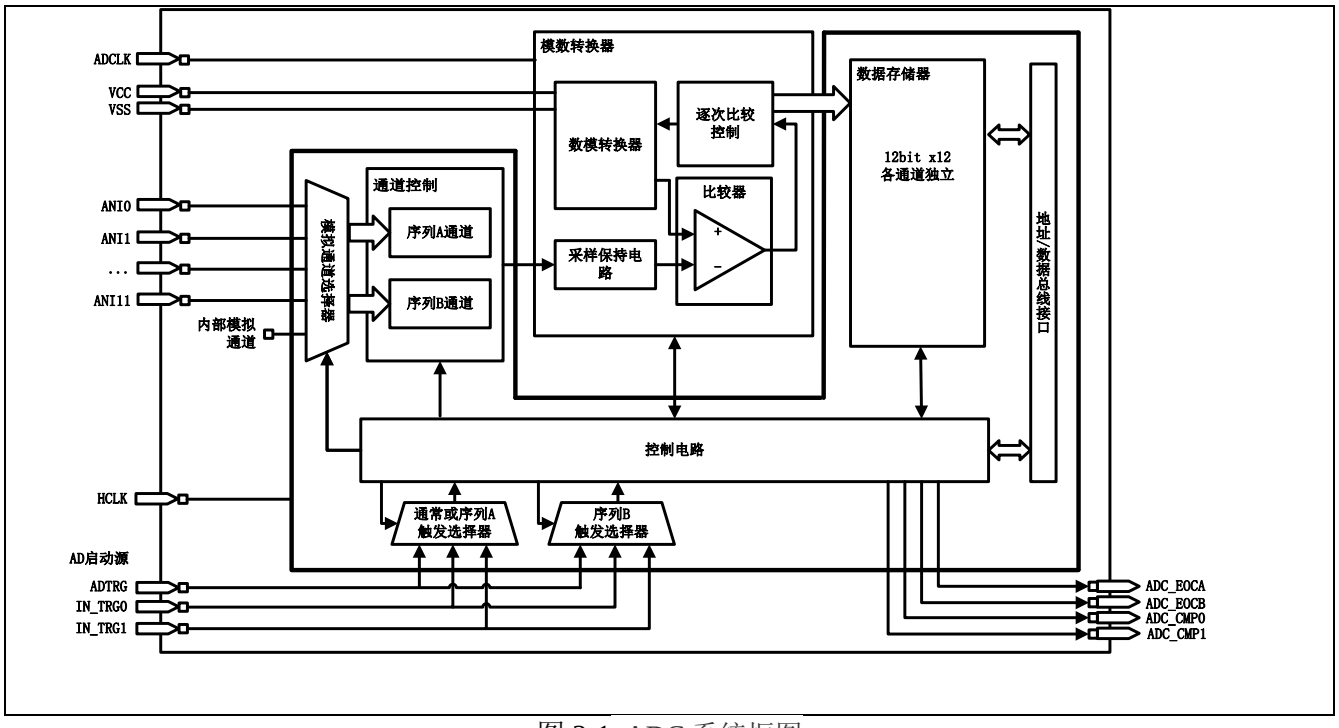

图 2-1 ADC 系统框图

2.2 主要特性

HC32M120 系列 MCU 的 ADC 具有如下主要特性：

1) 高性能

- 可配置 12 位、10 位和 8 位分辨率
- A/D 模拟电路时钟 ADCLK（用于 ADC 采样、转换等动作）的频率可以选择为 HCLK 的 1，2，4，8，16 或 32 分频
- 支持 1MSPS 采样率
- 采样时间可编程
- 各通道独立数据寄存器
- 数据寄存器可配置数据对齐方式
- 模拟看门狗，监视转换结果
- 不使用时可将 ADC 模块设定为停止状态

2) 模拟输入通道

- 最多 12 个外部模拟输入通道
- 1 个内部模拟检测通道，用于内部基准电压采样或温度传感器采样

3) 转换开始条件

- 软件启动开始转换（只支持序列 A）
- 外设事件触发开始转换（支持序列 A 和序列 B）
- 外部引脚触发开始转换（支持序列 A 和序列 B）

4) 转换模式

- 序列 A 单次扫描
- 序列 A 连续扫描
- 双序列扫描

5) 中断与事件信号输出

- 序列 A 扫描结束中断和事件 ADC_EOCA

- 序列 A 扫描结束中断和事件 ADC_EOCB
- 模拟看门狗 0 比较中断和事件 ADC_CMP0
- 模拟看门狗 1 比较中断和事件 ADC_CMP1

2.3 引脚配置

HC32M120 系列 MCU 的 ADC 有 12 个采样通道，最多支持 12 个外部模拟输入引脚。通道 0~10 用于外部模拟输入引脚的输入检测，通道 11（也叫扩展通道）可配置为外部模拟输入引脚或内部模拟输入源（包括内部基准电压和内部温度传感器）的输入检测。

3 ADC 应用

3.1 模拟输入引脚与通道的对应关系

HC32M120 系列 MCU 的 ADC 模块模拟输入引脚等配置如表 3-1。

ADC通道	外部模拟输入引脚	
CH0	ANI0(P20)	
CH1	ANI1(P21)	
CH2	ANI2(P22)	
CH3	ANI3(P23)	
CH4	ANI4(P24)	
CH5	ANI5(P25)	
CH6	ANI6(P26)	
CH7	ANI7(P27)	
CH8	ANI8(P147)	
CH9	ANI9(P00)	
CH10	ANI10(P01)	
CH11	ANI11(P120)	
	内部模拟输入	内部基准电压
		内部温度传感器

表 3-1 ADC 引脚与通道

ADC 的模拟输入通道可配置为两个序列，序列 A 和序列 B，只要将序列对应的通道选择寄存器的相应位置 1，即选择了对应的模拟输入引脚。如将序列 A 的通道选择寄存器 ADC_CHSELRA0 的 bit0 和 bit2 置 1，即序列 A 选择了两个通道，对应模拟输入引脚 ANI0 和 ANI2；序列 B 的通道选择寄存器 ADC_CHSELRB0 的 bit4、bit5 和 bit8 置 1，即序列 B 选择了三个通道，对应模拟输入引脚 ANI4、ANI5 和 ANI8。特别注意，序列 A 和序列 B 不可选择相同的通道。

3.2 模拟输入的采样时间和转换时间

关于 ADC 时间的详细说明，请参考用户手册“模拟输入的采样时间和转换时间”一节中，寄存器 ADC_SSTR 和 ADC 电气特性部分，请严格按照手册要求设置采样时间。在满足应用需求的情况下，请尽量将采样时间设置得大一些，尤其在对多个通道采样时，如果通道的采样时间设置偏小，可能会导致相邻通道（如序列 A 配置了 CH0、CH5、CH7，那么 CH0 和 CH5、CH5 和 CH7 都是相邻通道）之间透过采样电容发生耦合，而使转换结果不准确。

3.3 模式和功能

ADC 的通道可配置为序列 A 或序列 B，可分别设置触发源，共有四种扫描方式：

- 序列 A 单次扫描；
- 序列 A 连续扫描；
- 序列 A 单次扫描，序列 B 单次扫描；
- 序列 A 连续扫描，序列 B 单次扫描。

3.4 触发源选择

序列 A 和序列 B 可设置独立的触发源。可选择的触发源包括外部引脚 ADTRG（ADTRG_A、ADTRG_B 和 ADTRG_C），内部事件 IN_TRG0 和 IN_TRG1。其中 ADTRG 下降沿输入有效，IN_TRG0 和 IN_TRG1 由寄存器 ADC_ITRGSELR0 和 ADC_ITRGSELR1 设置，可选择芯片内部丰富的事件源。此外，写寄存器 ADC_STR 可生成软件触发信号，软件触发信号只能在 ADC 处于待机状态时使用，并且只适用于序列 A。

寄存器 ADC_TRGSR 用于设置序列 A 和序列 B 的触发源。序列 A 和序列 B 不可设置相同的触发源，即，如果序列 A 选择了触发源 ADTRG，那么序列 B 就不可再选择 ADTRG；如果序列 A 选择了 IN_TRG0，那么序列 B 就不可再选择 IN_TRG0（或包含 IN_TRG0 的设置）；如果序列 A 选择了 IN_TRG0 和 IN_TRG1 同时作为触发源，那么序列 B 就只能选择 ADTRG 作为触发源，反之亦然。

3.5 序列 A 单次扫描模式

3.5.1 说明

在此模式下，ADC 执行单个或多个通道的单次扫描，并在转换完成后停止，示意如图 3-1。

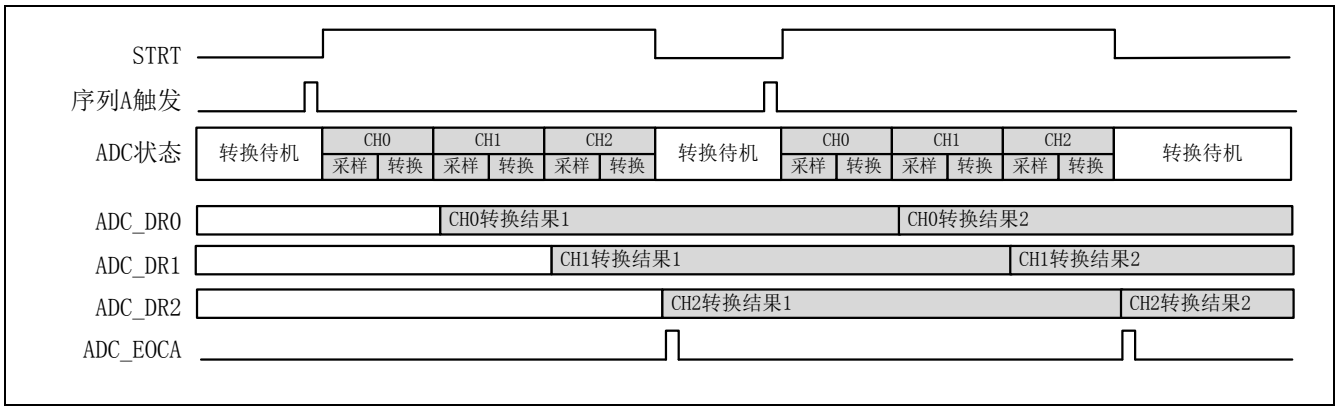

图 3-1 ADC 单次扫描模式

3.5.2 应用

该模式可以设置单个或多个通道。设置多个通道时，可实现对多个通道进行依次扫描，这些通道可设置不同的采样时间，用户不必在扫描过程中停止 ADC，即可以不同的采样时间重新扫描下一个通道，可避免额外的 CPU 负载以及繁重的软件开发。

该模式是最简单的 ADC 模式，应用方式灵活。如在系统启动前，可以用这种模式检测系统的一些状态信息，如电压、压力、温度等，以确定系统是否可以正常启动；在系统运行中，可用这种模式，按需检测系统状态，以获取系统实时状态。

应用例程 adc_01_sa_base 给出了该模式的具体用法。

3.6 序列 A 连续扫描模式

3.6.1 说明

连续扫描模式可对单个通道或多个通道进行连续不断的扫描，如图 3-2。连续扫描模式允许 ADC 在后台工作。因此，ADC 可在没有任何 CPU 干预的情况下对通道进行连续（循环）扫描通道，直到通过软件停止。

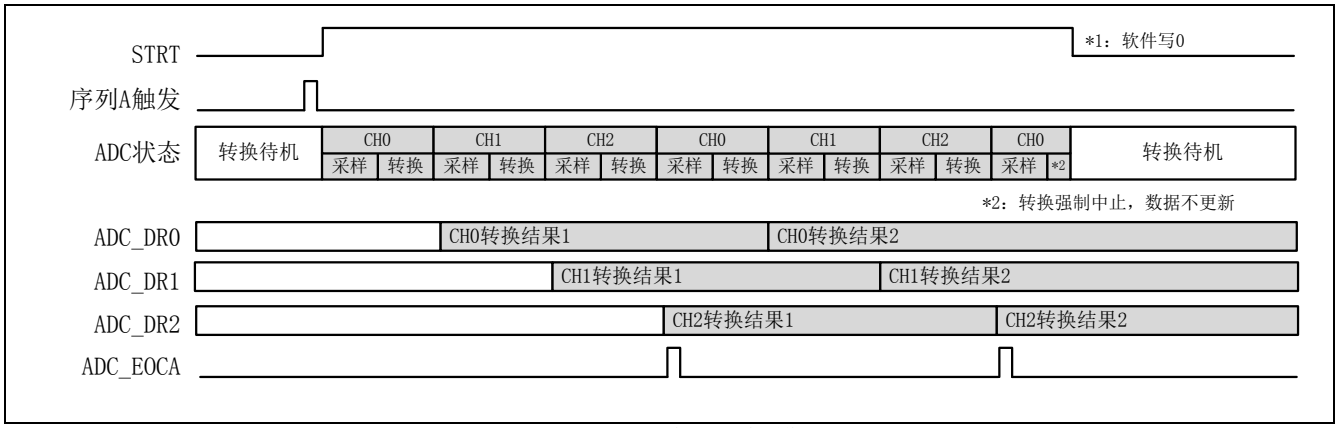

图 3-2 ADC 连续扫描模式

3.6.2 应用

此模式设置单个通道时，可用于监视电池电压、测量和调节烤箱温度等应用。在用于调节烤箱温度时，系统将读取温度并与用户设置的温度进行比较。当烤箱温度达到所需温度时，关闭加热电阻器的电源。

设置多个通道时，与多通道单次扫描模式类似，只是在完成序列的最后一个通道后不会停止扫描，而是从第一个通道重新开始扫描并无限循环下去。多通道连续扫描模式，可用于监视多电池充电器中的多个电压和温度。系统在充电过程中读取每节电池的电压和温度。当电压或温度达到最大值时，将切断相应电池与充电器的连接。

3.7 双序列扫描模式

3.7.1 说明

此处将“序列 A 单次扫描、序列 B 单次扫描”和“序列 A 连续扫描、序列 B 单次扫描”两种模式整合为双序列扫描模式进行介绍。双序列扫描模式，只是在前两种模式中增加了序列 B 的扫描。双序列扫描模式下，序列 B 必须由外部引脚或内部事件触发转换，软件启动对序列 B 无效，序列 A 可由软件启动扫描，也可由外部引脚或内部事件触发扫描。序列 B 的优先级高于序列 A，其与序列 A 的竞争关系如表 3-2。

A/D转换	触发信号发生	处理方式	
		ADC_CR1.RSCHSEL=0	ADC_CR1.RSCHSEL=1
序列A转换过程中	序列A触发	触发信号无效	
	序列B触发	1）序列A的转换被中断，开始序列B转换 2）序列B的转换完成后，序列A从被中断的通道开始继续转换	1）序列A的转换被中断，开始序列B转换 2）序列B的转换完成后，序列A从第一个通道开始重新转换
序列B转换过程中	序列A触发	序列B全部通道转换完成后，开始序列A转换	
	序列B触发	触发信号无效	
AD空闲中，序列A，B同时触发		序列B先启动，全部通道转换完成后，开始序列A转换	

表 3-2 序列 A 和序列 B 的竞争关系

配置 ADC_CR1.RSCHSEL 为 0 时，当序列 A 被中断后，恢复时，从被中断通道继续扫描，如图 3-3。

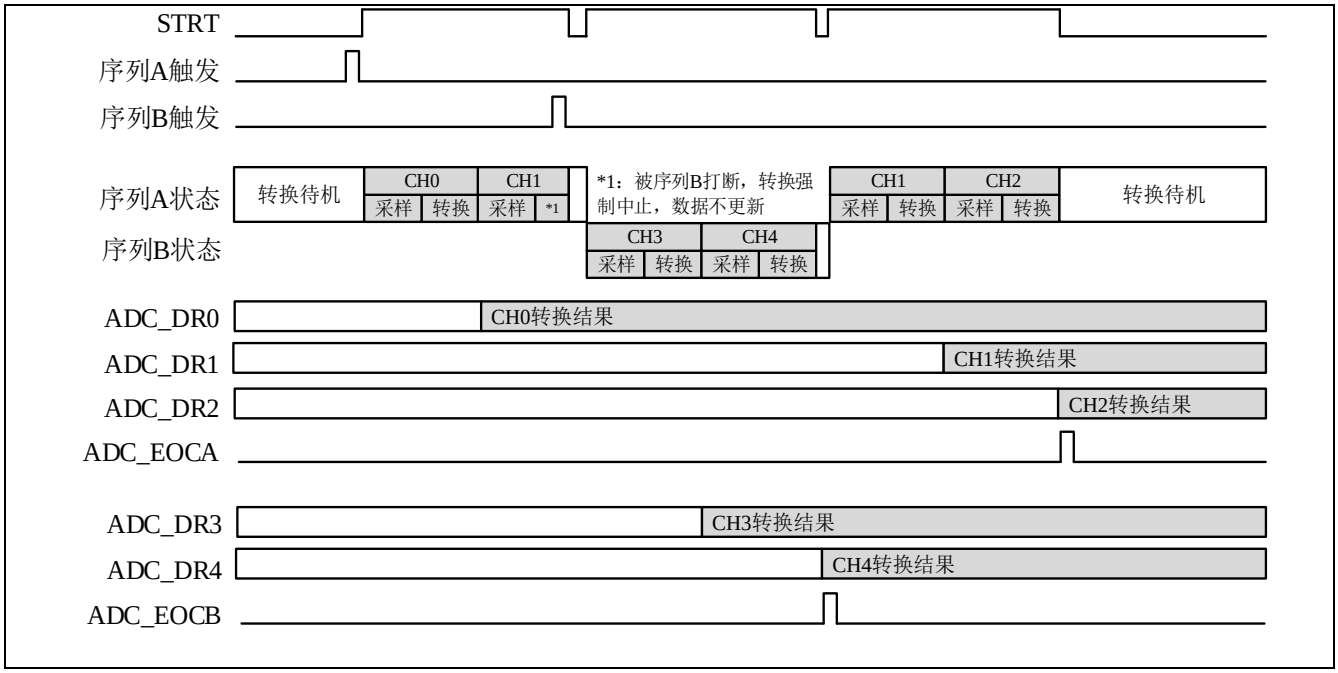

图 3-3 序列 A 从被中断通道恢复扫描

配置 ADC_CR1.RSCHSEL 为 1 时，当序列 A 被中断后，恢复时，从序列第一个通道重新开始扫描，如图 3-4。

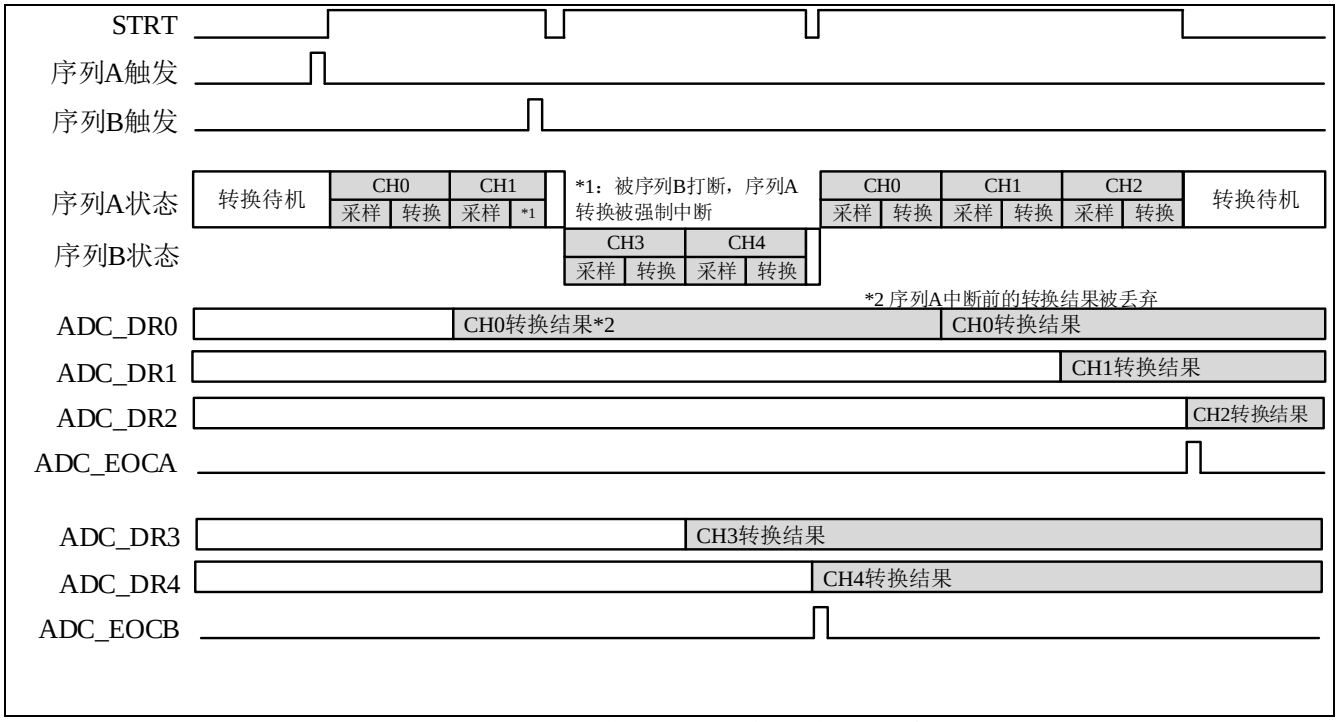

图 3-4 序列 A 从本序列第一个通道恢复扫描

注意：

- 序列 A 和序列 B 不能选择相同的通道，不能选择相同的触发源。

3.7.2 应用

可在前两种模式的应用中，加入需要实时响应（更高优先级）扫描的通道，将其配置为序列 B。例程 adc_02_sa_sb_event_trigger 实现了双序列扫描的基本用法。

3.8 模拟看门狗

3.8.1 说明

HC32M120 系列 MCU 的模拟看门狗，包含比较窗口 0 和比较窗口 1，可配置为上下限比较或区间比较，如图 3-5。

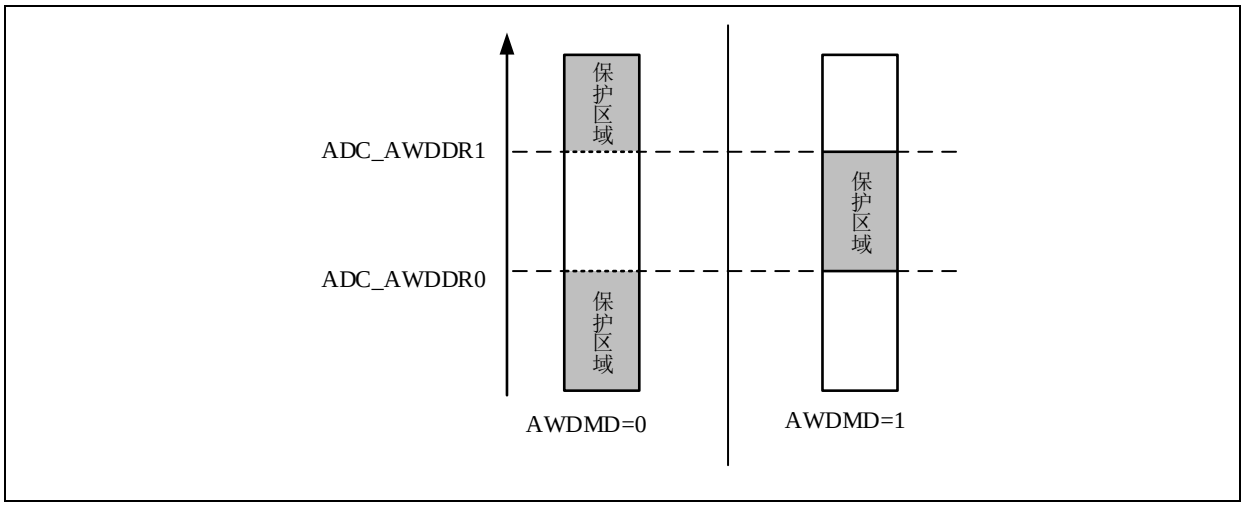

图 3-5 模拟看门狗比较模式

用户可预先设置比较条件和相应的上下限或区间。在通道转换结束后，模拟看门狗对转换结果进行比较，如果满足比较条件，则产生比较中断和事件 ADC_CMP0 或 ADC_CMP1。

两个比较窗口可组合使用。当窗口组合功能有效时，在窗口 1 选择的通道转换结束后，比较中断 ADC_CMP1 输出的不再是窗口 1 单独的比较结果，而是按照设置，对窗口 0 和窗口 1 的比较结果进行逻辑或、逻辑与或者逻辑异或后产生的组合结果。使用窗口组合功能的软件流程与窗口单独使用时相似，只是需要在设置好窗口 0 和窗口 1 之后，追加设置 ADC_AWDCR.AWDCM[1:0]寄存器选择组合方式。如果使能了窗口组合模式，请保证窗口 1 所选通道的转换在窗口 0 所选通道转换结束之后。

3.8.2 应用

在一些控制系统中，需要严格监测电压、压力、温度等信号的范围，使用模拟看门狗能够快速地检测到这些信号的异常状况，并做出相应的应对措施，以确保设备安全。例程 adc_05_awd_base 给出了模拟看门狗的配置和基本应用方法；但通常，模拟看门狗的中断用法更为高效，应用也更为普遍，例程 adc_06_awd_interrupt 给出了模拟看门狗的中断配置和用法。

3.9 扩展通道

3.9.1 说明

扩展通道（通道 11），其输入源可配置，可配置为外部模拟输入引脚 ANI11 或内部模拟输入，其中内部模拟输入又包括内部基准电压和内部温度传感器。如图 3-6。

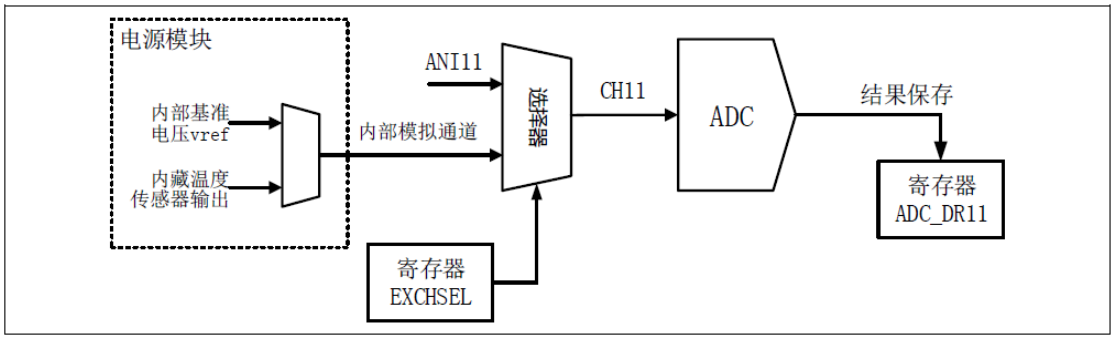

图 3-6 内部模拟通道选择

3.9.2 应用

1. 在一些系统中，可能由于某些原因导致 ADC 的参考电压不稳定，从而无法知道模拟输入的实际电压值，这时，可用 ADC 的内部检测通道，来检测内部基准电压（在系统工作电压正常时恒为 1.575V），来反推 ADC 当前的参考电压，从而得知模拟输入当前的实际电压值。具体实现如下：

   1) 在某已知参考电压 VREF1 下测得内部基准电压的 ADC 值为 VAL1；

   2) 随着系统的运行，参考电压可能随着供电源（如电池）的电压降低而下降，此时测得内部基准电压的 ADC 值为 VAL2，设此时参考电压为 VREF2；

   3) 由于内部基准电压是恒定的，所以有：

      VAL1 × VREF1 = VAL2 × VREF2;

      VREF2 = (VAL1 × VREF1) / VAL2;

   由此已知当前参考电压为 VREF2，就不难得到模拟输入的实际电压了。

2. 利用内藏温度传感器获取温度。将内部通道设置对内藏温度传感器进行 ADC 采样，可获取芯片内部温度，其计算公式为：

   t = 25 - (((sen_vol - 1.03) * 1000) / 3.5)

   其中，t 是温度值，单位摄氏度；sen_vol 是温度传感的电压值，单位伏特。

例程 adc_04_internal_channel 展示了扩展通道的各种配置和简单用法。

3.10 运算放大器 OPA

3.10.1说明

HC32M120 系列 MCU 搭载有一个运算放大器 OPA，通过设置寄存器 ADC_OPACR 使能，其工作示意图如图 3-7。

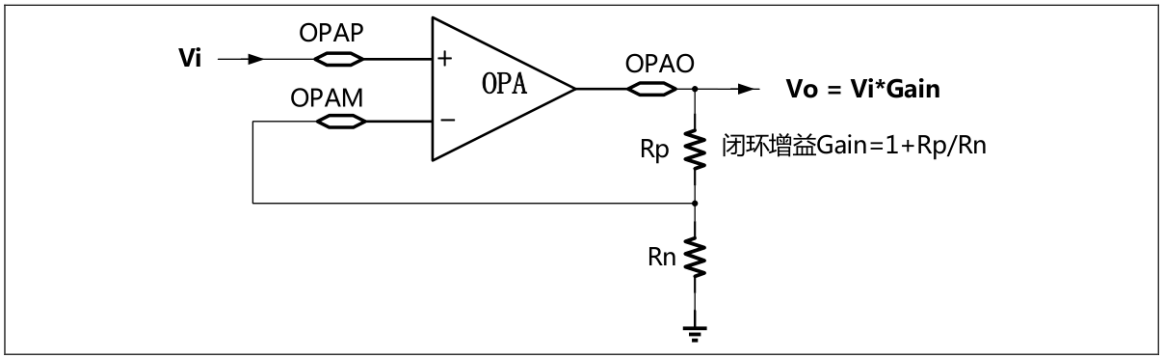

图 3-7 OPA 工作示意图

OPA 可独立运行，对输入 OPAP 进行放大并输出至 OPAO；也可结合 ADC 使用，通过 ADC 采样获取 OPAO 的电压；也可结合电压比较器 CMP 使用，对 OPAO 进行电压比较。OPA 的控制寄存器属于 ADC 模块，在应用 OPA 之前，需先使能 ADC 外设时钟，然后使能 OPA 功能，并配置 OPAM 引脚的功能。

由图 3-7 可知，OPA 的放大系数（Gain=1+Rp/Rn）取决于 Rp 和 Rn，用户需根据实际应用选用合适的 Rp 和 Rn。OPAP 的输入电压范围以及 OPAO 的负载能力请参考电气特性部分。

3.10.2应用

例程 adc_08_opa 展示了 OPA 独立使用和结合 ADC 使用的基本配置。

4 例程讲解

4.1 基本流程

ADC 应用程序流程可简单用图 4-1 表示。

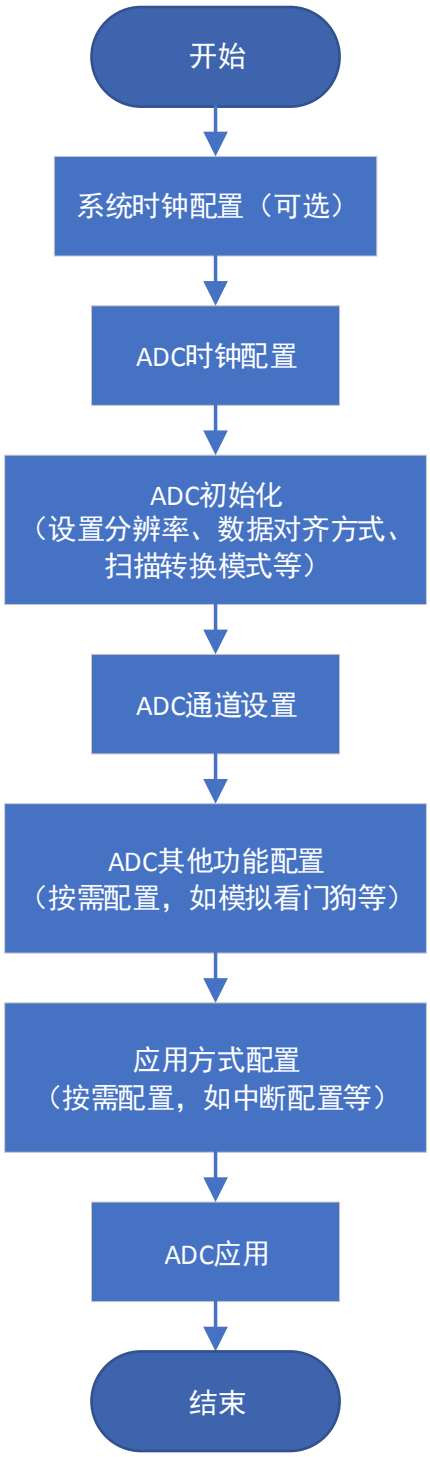

图 4-1 ADC 应用流程图

4.2 应用程序源码说明

用户可以根据 4.1 节的流程图编写自己的 ADC 应用程序，也可以通过华大半导体的网站下载 HC32M120 系列 MCU 的设备驱动库（Device Driver Library，DDL），参考其中的 ADC 例程。这里以其中的例程 adc_06_awd_interrupt，对 ADC 应用程序的源码做一个简单的说明。

4.2.1 应用程序基本结构

ADC 应用程序例程基本结构如程序清单 4-1。

```
int32_t main(void)
{
    /* Config a new system clock. */
    SystemClockConfig();

    /* Configures ADC. */
    AdcConfig();

    /* Configures and starts TIMER2.
     TIMER2 generates the event EVT_TMR2_GCMP every 100 milliseconds to trigger the ADC. */
    Timer2Config();
    /****************** Configuration end, application start **************/

    while (1u)
    {
        // Your application code.
    }
}
```

程序清单 4-1 ADC 应用程序例程基本结构

说明：

1. SystemClockConfig()：配置新的系统时钟，可选。当前应用程序（adc_06_awd_interrupt）使用的默认系统时钟 HRC8MHz；
2. AdcConfig()：ADC 的所有配置，包括初始化、通道设置等；
3. Timer2Config()：TIMER2 配置，如果用到其他外设产生的事件来触发 ADC，则需对外设进行相应配置，本例用 TIMER2 产生的事件 EVT_TMR2_GCMP 来触发 ADC 的序列 A。

4.2.2 重要源码讲解

下面对详细介绍该应用程序中 ADC 的配置。当前应用程序的 ADC 配置包括时钟（ADCLK）配置、初始化配置、通道配置、触发源配置和中断配置，如程序清单 4-2。

```
static void AdcConfig(void)
{
   AdcClockConfig();
   AdcInitConfig();
   AdcChannelConfig();
   AdcTriggerConfig();
   AdcIrqConfig();
}
```

程序清单 4-2 ADC 配置

说明：

1. AdcClockConfig()：配置 HCLK 的 2 分频（4MHz）作为 ADCLK，见程序清单 4-3。

```
static void AdcClockConfig(void)
{
   CLK_SetADClkDiv(CLK_HCLK_DIV2);
}
```

程序清单 4-3 配置 HCLK 的 2 分频作为 ADCLK

2. AdcInitConfig()：默认情况下，通过函数 ADC_StructInit()将 ADC 的初始化结构体作如下初始化设置：

   - 序列 A 单次扫描
   - 10 位分辨率
   - 禁止自动清除数据寄存器
   - ADC 数据右对齐
   - 采样时间为 20 个 ADCLK 周期
   - 序列 A 被序列 B 打断后，恢复扫描时，从被打断的通道开始继续扫描

   所有的初始化设置，在满足手册相关规定的前提下，都可以根据实际应用重新设置。该例程只用到了序列 A，且由 TIMER2 每 100 毫秒触发一次，所以保持默认扫描方式不变——序列 A 单次扫描；分辨率修改为 12 位；采样时间设置为 30 个 ADCLK 周期。详见程序清单 4-4。

```
static void AdcInitConfig(void)
{
    stc_adc_init_t stcInit;

    /* Set a default value. */
    ADC_StructInit(&stcInit);

    /* User configurations. */
    stcInit.u16Resolution = ADC_RESOLUTION;
    stcInit.u8SampTime    = ADC_SAMPLE_TIME;

    /* 1. Enable ADC peripheral. */
    CLK_FcgPeriphClockCmd(CLK_FCG_ADC, Enable);

    /* 2. Initializes ADC. */
    ADC_Init(&stcInit);
}
```

程序清单 4-4 ADC 初始化

3. AdcChannelConfig()：本例程中，AWD 的两个比较窗口各设置了一个通道，均配置到了序列 A。ADC 通道的基本配置，包括模拟输入引脚模式设置（设置为模拟输入模式）和 ADC 通道添加，本例程还包括模拟看门狗的配置。配置详见程序清单 4-5。

```
static void AdcChannelConfig(void)
{
    uint8_t u8AwdNbr;
    stc_awd_cfg_t stcAwdCfg;

    /* 1. Set the ADC pin to analog mode. */
    AdcSetChannelPinAnalogMode(ADC_SA_CHANNEL);

    /* 2. Add ADC sequence A channels. */
    ADC_AddAdcChannel(ADC_SEQ_A, ADC_SA_CHANNEL);

    /* 3. Configrues AWD. */
    u8AwdNbr = ADC_AWD_0;
    stcAwdCfg.u8AdcPin = AWD0_PIN;
    stcAwdCfg.u16AwdMd = ADW0_MD;
    stcAwdCfg.u16Dr0   = AWD0_DR0_LOWER;
    stcAwdCfg.u16Dr1   = AWD0_DR1_UPPER;
    ADC_AwdConfig(u8AwdNbr, &stcAwdCfg);

    u8AwdNbr = ADC_AWD_1;
    stcAwdCfg.u8AdcPin = AWD1_PIN;
    stcAwdCfg.u16AwdMd = ADW1_MD;
    stcAwdCfg.u16Dr0   = AWD1_DR0_LOWER;
    stcAwdCfg.u16Dr1   = AWD1_DR1_UPPER;
    ADC_AwdConfig(u8AwdNbr, &stcAwdCfg);

#ifdef AWD_COMBINATION_MODE
    /* If combination mode is
valid(ADC_AWD_COMB_OR/ADC_AWD_COMB_AND/ADC_AWD_COMB_XOR) and
       the channles selected by the AWD0 and AWD1 are deferent, make sure that the channel
       of AWD1 is converted after the channel conversion of AWD0 ends. */
```

```
    ADC_AwdConfigCombMode(AWD_COMBINATION_MODE);
#endif

    ADC_AwdCmd(ADC_AWD_0, Enable);
    ADC_AwdCmd(ADC_AWD_1, Enable);
}
```

程序清单 4-5 ADC 通道配置

4. AdcTriggerConfig()：ADC 触发源设置。本例设置 TIMER2 产生的事件 ADC_TRGSRC_IN_EVT0 作为序列 A 的触发源，如程序清单 4-6 所示。

```
static void AdcTriggerConfig(void)
{
    uint8_t u8Seq;
    stc_adc_trg_cfg_t stcTrgCfg;

    /* Set a default value for stcTrgCfg.  */
    ADC_TriggerSrcStructInit(&stcTrgCfg);

    /* Configrues sequence A's trigger source. */
    u8Seq = ADC_SEQ_A;
    stcTrgCfg.u16TrgSrc = ADC_TRGSRC_IN_EVT0;
    stcTrgCfg.u32Event0 = EVT_TMR2_GCMP;

    /* 1. Enable AOS. */
    CLK_FcgPeriphClockCmd(CLK_FCG_AOS, Enable);
    /* 2. Configrues the trigger source of sequence A. */
    ADC_ConfigTriggerSrc(u8Seq, &stcTrgCfg);
    /* 3. Enable the trigger source. */
    ADC_TriggerSrcCmd(u8Seq, Enable);
}
```

程序清单 4-6 ADC 触发源设置

5. AdcIrqConfig()：ADC 中断配置，主要展示了 ADC 中断的独立中断配置和共享中断的配置方式，见程序清单 4-7。

```
static void AdcIrqConfig(void)
{
    stc_irq_regi_config_t stcIrqRegiConf;

    /* Configures AWD0 interrupt.
       The following 2 configurations of interrupt are both valid. */
#ifdef SHARE_INTERRUPT
    /* Share interrupt. */
    stcIrqRegiConf.enIntSrc    = INT_ADC_CMP0;
    stcIrqRegiConf.enIRQn      = Int030_IRQn;
    INTC_ShareIrqCmd(stcIrqRegiConf.enIntSrc, Enable);
    NVIC_ClearPendingIRQ(stcIrqRegiConf.enIRQn);
    NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_02);
    NVIC_EnableIRQ(stcIrqRegiConf.enIRQn);
#else
    /* Independent interrupt. */
```

```
    stcIrqRegiConf.enIntSrc    = INT_ADC_CMP0;
    stcIrqRegiConf.enIRQn      = Int020_IRQn;
    stcIrqRegiConf.pfnCallback = &AdcCmp0_IrqHandler;
    INTC_IrqRegistration(&stcIrqRegiConf);
    NVIC_ClearPendingIRQ(stcIrqRegiConf.enIRQn);
    NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_02);
    NVIC_EnableIRQ(stcIrqRegiConf.enIRQn);
#endif // #ifdef SHARE_INTERRUPT

    /* Configures AWD1 interrupt. */
    stcIrqRegiConf.enIntSrc    = INT_ADC_CMP1;
    stcIrqRegiConf.enIRQn      = Int022_IRQn;
    stcIrqRegiConf.pfnCallback = &AdcCmp1_IrqHandler;
    INTC_IrqRegistration(&stcIrqRegiConf);
    NVIC_ClearPendingIRQ(stcIrqRegiConf.enIRQn);
    NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_03);
    NVIC_EnableIRQ(stcIrqRegiConf.enIRQn);

    /* Enable the interrupts. */
    ADC_AwdIntCmd(ADC_AWD_0, Enable);
    ADC_AwdIntCmd(ADC_AWD_1, Enable);
}
```

程序清单 4-7 ADC 中断配置

6. 中断回调函数 AdcCmp0_IrqHandler()和 AdcCmp1_IrqHandler()，见程序清单 4-8。

```
void AdcCmp0_IrqHandler(void)
{
    if (ADC_AwdGetFlag(ADC_FLAG_AWD0) == Set)
    {
        ADC_AwdClrFlag(ADC_FLAG_AWD0);
        m_u32AdcIrqFlag |= ADC_AWD0_IRQ_BIT;
    }
}

void AdcCmp1_IrqHandler(void)
{
#ifdef AWD_COMBINATION_MODE
    if (ADC_AwdGetFlag(ADC_FLAG_AWD_COMB) == Set)
    {
        ADC_AwdClrFlag(ADC_FLAG_AWD_COMB);
        m_u32AdcIrqFlag |= ADC_AWD1_IRQ_BIT;
    }
#else
    if (ADC_AwdGetFlag(ADC_FLAG_AWD1) == Set)
    {
        ADC_AwdClrFlag(ADC_FLAG_AWD1);
        m_u32AdcIrqFlag |= ADC_AWD1_IRQ_BIT;
    }
#endif // #ifdef AWD_COMBINATION_MODE
}
```

程序清单 4-8 ADC 中断回调函数

4.2.3 程序执行现象

将工程 adc_06_awd_interrupt 编译，下载至目标板，在中断回调函数处设置断点，然后全速运行，拉低 ADTRGA（P10）以触发序列 A 动作，继而触发序列 B 动作，看到图 4-2 和图 4-3 所示的运行情况。

```
396 /**
397  * @brief  ADC AWD0 IRQ callback.
398  * @param  None
399  * @retval None
400  */
401 void AdcCmp0_IrqHandler(void)
402 {
403     if (ADC_AwdGetFlag(ADC_FLAG_AWD0) == Set)
404     {
405         ADC_AwdClrFlag(ADC_FLAG_AWD0);
406         m_u32AdcIrqFlag |= ADC_AWD0_IRQ_BIT;
407     }
408 }
```

图 4-2 模拟看门狗窗口 0 中断

```
410 /**
411  * @brief  ADC AWD1 IRQ callback.
412  * @param  None
413  * @retval None
414  */
415 void AdcCmp1_IrqHandler(void)
416 {
417 #ifdef AWD_COMBINATION_MODE
418     if (ADC_AwdGetFlag(ADC_FLAG_AWD_COMB) == Set)
419     {
420         ADC_AwdClrFlag(ADC_FLAG_AWD_COMB);
421         m_u32AdcIrqFlag |= ADC_AWD1_IRQ_BIT;
422     }
423 #else
424     if (ADC_AwdGetFlag(ADC_FLAG_AWD1) == Set)
425     {
426         ADC_AwdClrFlag(ADC_FLAG_AWD1);
427         m_u32AdcIrqFlag |= ADC_AWD1_IRQ_BIT;
428     }
429 #endif // #ifdef AWD_COMBINATION_MODE
430 }
```

图 4-3 模拟看门狗窗口 1 中断

5 总结

本应用笔记简要介绍了 HC32M120 系列 ADC 模块的各种功能以及可能的应用场景，并给出了 ADC 模块应用的基本流程，在实际开发中，用户可根据具体应用场景按需配置和应用 ADC 模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2019/10/31	Rev1.0	初版发布

如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email：mcu@hdsc.com.cn

网址：www.hdsc.com.cn

通信地址：上海市浦东新区中科路 1867 号 A 座 10 层

邮编：201203