BAT32G137（库函数版本）

Rev 1.0

修订历史

版本	日期	修订人	修订内容
Rev1.1	22.9.22	缪勤文	

目 录

1. 前言

BAT32G137 系列的不同芯片 A/D 转换器的模拟输入通道数不一样，相同产品不同 PIN 脚的输入通道数也不一样； 具体采集通道数需要查看芯片的用户手册以及 datasheet。

2. AD 单元功能

AD 转换器精度：12 位分辨率

AD 转换时间：以 BAT32G137 为例，高速模式下：13.5 个采样时钟+逐次比较时钟 31.5 个 ADC CLK；低电流模式下：13.5+40.5 = 54 ADC CLK；

AD 工作模式：

- 选择模式
- 扫描模式

AD 转换模式：

- 单次转换模式
- 连续转换模式

AD 触发方式：

- 软件触发（通过软件操作开始转换）
- 硬件触发无等待（通过硬件触发检测来开始转换）
- 硬件触发等待（在切断 A/D 电源的转换待机状态下，检测硬件触发来开始转换）

应用例程给出了选择模式连续转换、选择模式单次转换、扫描模式单次转换、扫描模式连续转换；触发无等待以及触发等待的例程；

3. 中微 BAT32G137 ADC 应用库简介

中微 BAT32G137 ADC 应用库是一个便于移植的标准库代码风格，用户只需要对软件接口相关参数进行简单配置、以及封装接口函数调用即可实现所需功能，节约时间，提高开发效率。

使用方式：

需将应用层 adc_demo.c adc_demo.h 驱动层 adc.c adc.h、gpio.c gpio.h、isr.c 、isr.h 加入到工程中去；若搭配使用 DMA，则需要将相应 DMA 驱动文件以及 dma_demo 程序加入。

3.1.应用例程使用

观察芯片引脚图，在使用 ADC 采样时候，需要根据芯片引脚功能，以 BAT32G137 64PIN 为例：ANI0 代表 adc 采集通道 channel0，ANI1 代表 adc 采集通道 channel1，ANI2 为 adc 采集通道 channel2 使用引脚为 P22……以此类推；因此在代码配置以及硬件设计时候需要查看芯片引脚图确定采样通道和对应的 ADC 采样通道。

3.1.1. 选择模式

```
1.  /*************************************************************
```

```
2.  Function Name: Adc_Init
3.  * @brief  This function initial the AD converter
4.  * @param  none
5.  * @return none
6.  ************************************************************/
7.  void Adc_Init()
8.  {
9.   GPIO_InitTypeDef      GPIO_InitStructure ={0};
10.  ADC_InitTypeDef       ADC_InitStructure ={0};
11.
12.  GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_2;  //设置adc 采样通道
13.  GPIO_InitStructure.GPIO_Ctrl = GPIO_Control_ANA; //模拟输入
14.  GPIO_Init(GPIO_PORT2, &GPIO_InitStructure);//初始化
15.
16.  ADC_InitStructure.ADC_Mode      = ADC_Mode_Select; //选择模式
17.  ADC_InitStructure.ADC_ConvSpeed = ADC_ConvSpeed_High; //高速变换模式
18.  ADC_InitStructure.ADC_Prescaler = ADC_Prescaler_Div32;//预分频
19.  ADC_InitStructure.ADC_RefVoltage = ADC_Ref_Vdd; //使用VDD 参考电压
20.  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrig_Software;
     //软件触发
21.  ADC_InitStructure.ADC_ContinuousConvMode = ADC_Conv_Continuous;//开
     启连续转换
22.  ADC_InitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_13p5C
     ycles; //默认采样间隔为13.5 个clk
23.  ADC_InitStructure.ADC_UpLimit   = ADC_UpLimit_Setting; //adc 采样高
     8 位 采样比较最大值，默认值
24.  ADC_InitStructure.ADC_LowLimit  = ADC_LowLimit_Setting; //adc 采样高
     8 位 采样比较最低值，默认值
25.  ADC_Init(&ADC_InitStructure);
26.
27.  ISR_Register(ADC_IRQn,adc_interrupt); //中断服务路径注册
28.  ADC_Start(ADC_Channel_2);
29. }
```

- 配置 GPIO，由于我们使用 ADC channel2 因此配置 P22 引脚，模拟输入
- 配置 ADC 工作模式为选择模式
- 配置 ADC 进行高速转换
- 配置 ADC CLK 为 sysclk/ ADC_Prescaler
- 选择 ADC 参考电压：：Vdd / 外部参考电压/内部基准电压 1.45V
- 连续转换模式：开启/关闭，关闭就代表单次转换
- ADC 采样间隔：13.5clk，一般不能改变

- ADC 转换结果上限值与下限值设定：在设置设定范围内产生中断，即转换结果的高 8 位于其比较在设定范围内产生中断

例程位于 adcBurnTets 中，例程使用 DMA 重复方式，将 ADC 采样值 ADC->ADCR 到 buf 中之后，再次打开 DMA 使能，保证对 P22 一直采样的值通过 DMA 传送出来。

3.1.2. 扫描模式

扫描模式，一次最多可以扫描 4 个通道，并且这 4 个通道号必须是连续的；比如设置起始的扫描通道为 channel2，设置连续扫描通道号为 4,则扫描通道为 ch2~ch5，采集相对应的引脚为 P22~P25；配置扫描模式代码如下：

```
1.  void Adc_Scan_Init()
2.  {
3.   GPIO_InitTypeDef      GPIO_InitStructure ={0};
4.   ADC_InitTypeDef       ADC_InitStructure;
5.
6.   GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_2;
7.   GPIO_InitStructure.GPIO_Ctrl = GPIO_Control_ANA; //模拟输入
8.   GPIO_Init(GPIO_PORT2, &GPIO_InitStructure);//初始化
9.
10.  ADC_InitStructure.ADC_Mode = ADC_Mode_Scan; //扫描模式
11.  ADC_InitStructure.ADC_ScanConf.ADC_ScanNum = ADC_ScanNum_2;  //2 通道扫描
12.  ADC_InitStructure.ADC_ScanConf.ADC_ChannelSign = ADC_Sign_Enable; //转换结果高 4 位显示通道号
13.  ADC_InitStructure.ADC_ConvSpeed = ADC_ConvSpeed_High; //高速变换模式
14.  ADC_InitStructure.ADC_Prescaler = ADC_Prescaler_Div32;//预分频
15.  ADC_InitStructure.ADC_RefVoltage = ADC_Ref_Vdd; //使用内部参考电压
16.  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrig_Software;
     //软件触发
17.  ADC_InitStructure.ADC_ContinuousConvMode = ADC_Conv_Continuous ;//开启单次转换 ADC_Conv_Oneshot ADC_Conv_Continuous
18.  ADC_InitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_13p5Cycles; //默认采样间隔为 13.5 个 clk
19.  ADC_InitStructure.ADC_UpLimit = ADC_UpLimit_Setting; //adc 采样高 8 位 采样比较最大值，默认值
20.  ADC_InitStructure.ADC_LowLimit = ADC_LowLimit_Setting; //adc 采样高 8 位 采样比较最低值，默认值
21.  ADC_Init(&ADC_InitStructure);
```

```
22.
23.  ISR_Register(ADC_IRQn,adc_interrupt); //中断服务路径注册
24. }
25.
```

- 配置 GPIO，由于我们使用 ADC channel2 因此配置 P22 引脚，模拟输入
- 配置 ADC 工作模式为扫描模式，以及扫描结果是否标识通道号
- 配置 ADC 进行高速转换
- 配置 ADC CLK 为 sysclk/ ADC_Prescaler
- 选择 ADC 参考电压：：Vdd / 外部参考电压/内部基准电压 1.45V
- 连续转换模式：开启/关闭，关闭就代表单次转换
- ADC 采样间隔：13.5clk，一般不能改变
- ADC 转换结果上限值与下限值设定：在设置设定范围内产生中断，即转换结果的高 8 位于其比较在设定范围内产生中断

获取扫描结果函数 ADC_Converse_Scan

```
1. /*****************************************************
2. Function Name: ADC_Converse_Scan
3. * @brief  This function starts the AD converter and returns the conv
   ersion result in the buf
4. *         It is suitable for software trigger mode with polling mode
   .
5. * @param  ch - specify the ad channel
6. * @param  scannum - specify scan channels
7. * @param  times - set the times of ad conversion
8. * @param  buf - the address where to write the conversion result
9. * @return average value
10.*****************************************************/
11.uint16_t ADC_Converse_Scan(ADC_Channel_t ch, uint8_t scannum, uint32
   _t times, uint16_t *buf)
```

扫描通道函数，形参为扫描通道数、次数、以及存放扫描结果的 buf；具体使用例程可以参考 adcScan

3.1.3. 硬件触发

硬件触发源有如下几种：

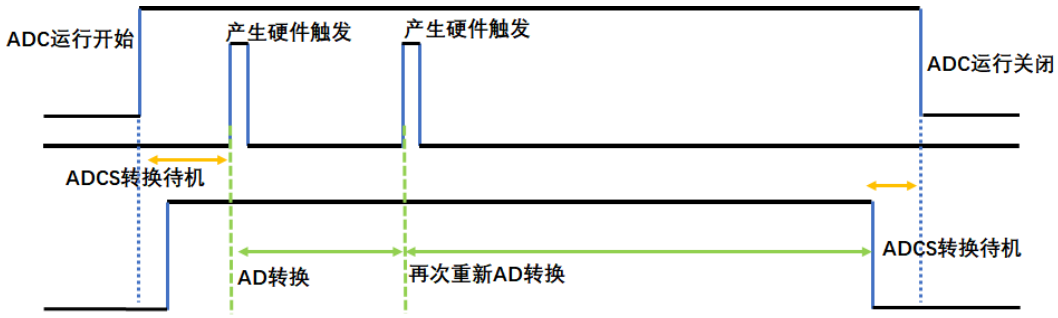
INTTMO1： 定时器 TIMER40 的通道 1 产生的中断

ELC ：时间联动器的事件信号

INTRTC：实时时钟的中断信号

INTIT： 15 位间隔定时器产生的计数中断

硬件触发无等待：在 ADCS 为 1 时候，如果检测到硬件触发信号，才会进行 AD 转换，AD 转换结束就会将结果保存在 ADCR ADCRH，并且产生中断信号；时序图如下：



图二 硬件触发无等待

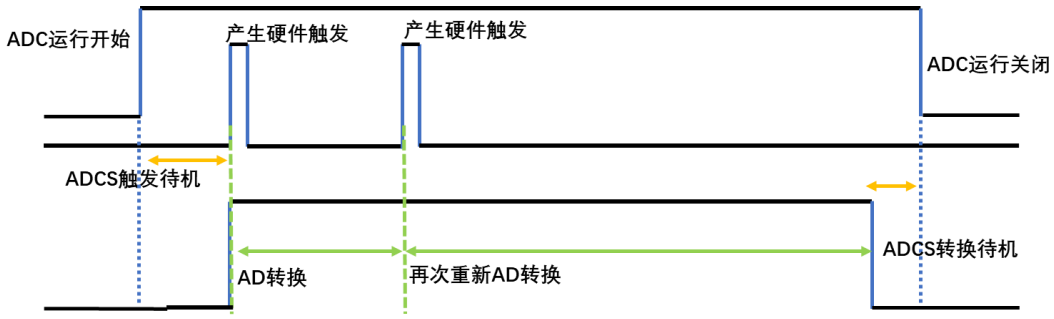
注意：即使 AD 在转换过程中，检测到硬件触发，则也要立即停止转换，重新开始转换。

```
1.  void Adc_HardNoWait_Init(void)
2.  {
3.   GPIO_InitTypeDef       GPIO_InitStructure ={0};
4.   ADC_InitTypeDef        ADC_InitStructure ={0};
5.
6.   GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_2;  //设置 adc 采样通道
7.   GPIO_InitStructure.GPIO_Ctrl = GPIO_Control_ANA; //模拟输入
8.   GPIO_Init(GPIO_PORT2, &GPIO_InitStructure);//初始化
9.
10.  ADC_InitStructure.ADC_Mode = ADC_Mode_Select; //选择模式
11.  ADC_InitStructure.ADC_ConvSpeed = ADC_ConvSpeed_High; //高速变换模式
12.  ADC_InitStructure.ADC_Prescaler = ADC_Prescaler_Div32;//预分频
13.  ADC_InitStructure.ADC_RefVoltage = ADC_Ref_Vdd; //使用内部参考电压
14.  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrig_Hardware_
    NoWait; //硬件触发
15.  ADC_InitStructure.ADC_HardwareTrigSour = ADC_HardwareTrig_INTIT;
16.  ADC_InitStructure.ADC_ContinuousConvMode = ADC_Conv_Continuous;//单
    次转换
17.  ADC_InitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_13p5C
    ycles; //默认采样间隔为13.5 个clk
18.  ADC_InitStructure.ADC_UpLimit = ADC_UpLimit_Setting; //adc 采样高 8
    位 采样比较最大值，默认值
19.  ADC_InitStructure.ADC_LowLimit = ADC_LowLimit_Setting; //adc 采样高
    8 位 采样比较最低值，默认值
20.  ADC_Init(&ADC_InitStructure);
```

```
21.
22.  ISR_Register(ADC_IRQn,adc_interrupt); //中断服务路径注册
23. }
```

- 配置 GPIO，由于我们使用 ADC channel2 因此配置 P22 引脚，模拟输入
- 配置 ADC 工作模式为选择模式
- 配置 ADC 进行高速转换
- 配置 ADC CLK 为 sysclk/ ADC_Prescaler
- 选择 ADC 参考电压：：Vdd / 外部参考电压/内部基准电压 1.45V
- 连续转换模式：开启/关闭，关闭就代表单次转换
- ADC 采样间隔：13.5clk，一般不能改变
- ADC 转换结果上限值与下限值设定：在设置设定范围内产生中断，即转换结果的高 8 位于其比较在设定范围内产生中断
- 选择硬件触发，触发源选择 INTIT 15 位间隔定时器进行触发

硬件触发等待：不同于硬件触发无等待时，在 ADCS 下进行的硬件触发；无等待，在检测到硬件触发时，立即将 ADCS 置 1；转换结束之后产生中断信号，将结果保存在 ADCR ADCRH 中；时序图如下：



在初始化时候，在选择触发时候，将硬件无等待触发，改为硬件触发等待即可

3.2. 例程说明

硬件触发无等待；例程演示，利用间隔定时器的中断触发 ADC 进行 AD 转换，通过触发 DMA 将 AD 连续转换结果搬移到 buf 中；

```
// Systick setting
//------------------------------------------------------------------------
    SystemCoreClockUpdate();
    msCnt = SystemCoreClock / 1000;
    SysTick_Config(msCnt);

    Uart0_Init(19200);

#if ADC_ENABLE
    /*the adc hardwareTriggerNoWait test(use interval timer trigger ADC convertor)*/
    printf("Hello, UART\n");
    Interval_Init(IT_FIL, 10000);
    IT_Start();
    INTC_DisableIRQ(IT_IRQn);       /* disable INTIT interrupt */
    Adc_HardNoWait_Init();
//  ADC_Set_HardTrigger(0, ADC_HardwareTrig_INTIT);
    ADC_Start(ADC_Channel_2);
    Dma_Adc_Config(DMA_VECTOR_ADC,DMA_Mode_Normal,(uint16_t *)&ADC->ADCR,get_value,8);

//  ADC->ADTES = 0x01;  // zero code test
//  ADC->ADTES = 0x03;  // half code test
//  ADC->ADTES = 0x05;  // full code test

//  __WFI();    // Sleep mode

    while(g_AdcIntTaken < 1);
    for(i=0; i<8; i++)
    {
        printf("ADC Get Value = 0x%04X\n", get_value[i]);
    }
    IT_Stop();
    ADC_Stop();
#endif
}
```

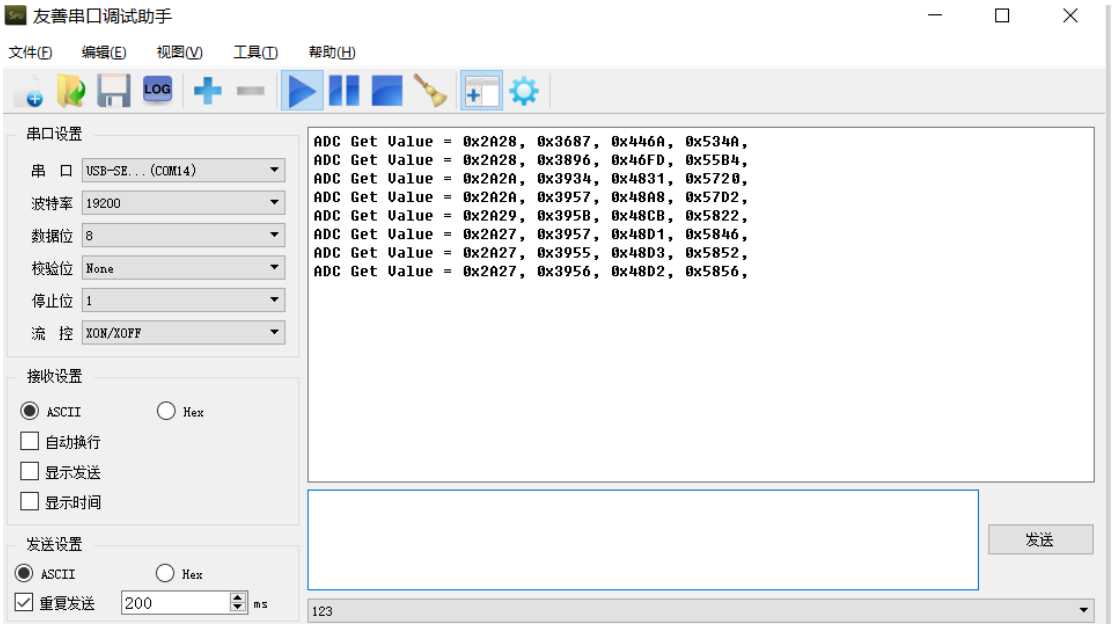
硬件触发等待：通过间隔定时器触发单次采样，每间隔定时时间触发单次采样，采样 8 次；

扫描：

```
    uint32_t msCnt;     // count value of lms
    uint32_t width =0,tempAdc;
    int i=0,j=0;
//------------------------------------------------------------------------
// Systick setting
//------------------------------------------------------------------------
    SystemCoreClockUpdate();
    msCnt = SystemCoreClock / 1000;
    SysTick_Config(msCnt);
    delay_init(SystemCoreClock);

    Uart0_Init(19200);

#if ADC_ENABLE
    /*the adc scans 2 adc channel and output adc result */
    Adc_Scan_Init();
    tempAdc = ADC_Converse_Scan(ADC_Channel_2, ADC_ScanNum_4, 8, get_value);
    for(i=0; i<8; i++)
    {
        printf("ADC Get Value = ");
        for(j=0; j<4; j++)
        {
            //printf("0x%04X, ", get_value[i][j]);
            printf("0x%04X, ", get_value[i*ADC_ScanNum_4+j]);
        }
        printf("\n");
    }
#endif

}
```

对 P22~P254 同奥进行扫描，扫描 8 次，然后将数据打印出来；

4. 示例演示



图一 扫描模式

对于扫描模式，由于配置了高 4 位指示通道号，因此在打印结果时候，0x2xxx 显示了 ADC channel2，采样值为 0x2A7