芯海科技
CHIPSEA

CS32F0XX ADC 外设模块

应用笔记

REV 1.0

芯海科技（深圳）股份有限公司
地　址：深圳市南山区蛇口南海大道1079号花园城数码大厦A座9楼
电　话：+(86 755)86169257　　传　真：+(86 755)86169057
网　站：www.chipsea.com　　邮　编：518067
微信号：芯海科技

版本历史

历史版本	修改内容	版本日期
REV 1.0	初版生成	2019-11-15

目录

1 使用概要

1.1 前言

本应用笔记旨在展示使用 CS32F0xx 微控器，提高 A/D 转化精度的应用。帮助 ADC 模块用户了解 CS32 微控器提供的一些高级应用并加快开发周期。所介绍的每种模式都提供一个应用示例，以方便用户快速移植。

本应用笔记分为三部分:

- 独立 ADC 模式
- 降低 ADC 误差固件的方法
- 在 VDD 跳变的情况下使用 ADC 模式

下表列出了本应用笔记覆盖的 CS32 微控器类型及 PACK 版本号。

表 1.1 适用类型

类型	编号
微控制器	CS32F03x(CS32F030, CS32F031)
PACK 包	Chipsea.CS32F0xx_DFP.1.0.0

2 独立模式

A/D 转换器(ADC)是将输入模拟信号转换成数字信号的装置。一般由 4 个部分组成，模拟部分(如参考电压、比较器、可控积分器等)、采样保持部分、数字或数据产生部分、数据输出部分。ADC 可以配置为单通道、多通道、连续、非连续等采样模式，传输获取采样值分为查询、中断、DMA 传输获取等方式。下文给出了本笔记覆盖的应用方式。

2.1 单通道、单次转换模式

2.1.1 原理演示

此种模式是十分通用的 ADC 模式，在此模式下，ADC 模块执行单个通道 x 的单次采样，并在转换结束之后停止。转换示例见图 2.1 所示。

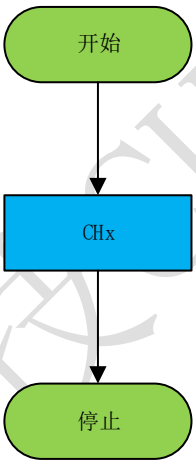

图 2.1 单通道单次转换

2.1.2 应用场景

此模式用于电压测量，在启动系统前可以测量一下电池电压：若电量正常，可以启动系统；若电量过低，则不要启动系统。

- 硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接
- 软件代码：见附录 1 单通道单次转换

2.2 多通道（扫描）、单次转换模式

2.2.1 原理演示

此模式用于在独立模式下对于一些通道进行依次转换的场景。可以通过配置 ADC 的寄存器，使 ADC 以不同的采样时间和采样顺序对任意序列的通道进行配置（最多 12 个通道）。例如：可以执行图 2.2 所示序列，用户可以设定 ADC 顺序以不同的采样时间转换 6 个通道。通过该种模式，ADC 可以多个通道的单次转化（见图 2.3），用户不必在转化过程中停止 ADC，即可以不同的采样时间配置下一个通道。此模式可以避免 MCU 的负担。

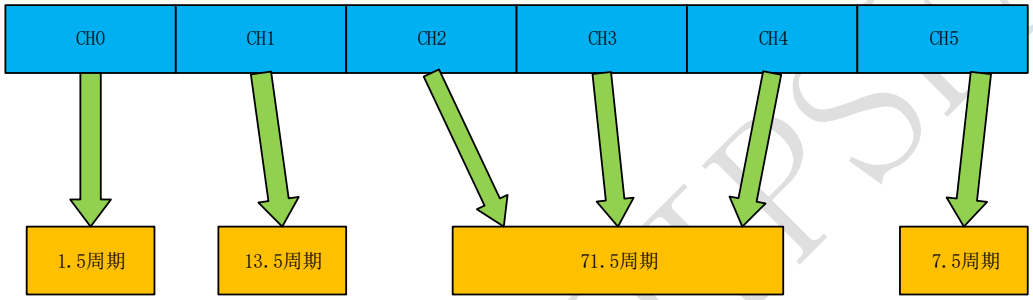

图 2.2 ADC 正序扫描以不同的采样时间转换 6 个通道

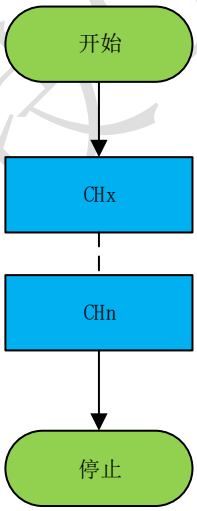

图 2.3 多通道单次转换

2.2.2 应用场景

此模式用于不需要频繁采样 ADC 的场景，如每隔一段时间同时检测电量值、温度值、外部传感器的值。

还可以用于单次测量多个信号（温度、电流、电压、压力）以确定是否可以启动系统，从而确保设备与人身安全。

在联动系统中，需要确定一些参数（例如机器人系统中，需要知道机器人顶端的坐标）时，可以使用这种模式。在这种系统中，在上电之前必须确定每个关节的位置，才能确定机器人顶端坐标。

也可以用于物体不同应变方向和值的场景，以用来对应变仪的信号进行转换。

- **硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接**
- **软件代码：见附录 2**

2.3 单通道、连续转换模式

2.3.1 原理演示

单通道连续转换模式可以在常规通道中对单个通道连续不断的进行转换。此种模式下允许 ADC 模块在后台工作。因此 ADC 在没有 MCU 的干预的情况下连续转换通道。另外，还可以在循环模式下使用 DMA，从而降低 MCU 的负载。

本应用笔记中提供了 ADC+DMA 模式实现对单通道的连续转换，转换的原理如图 2.4 所示。

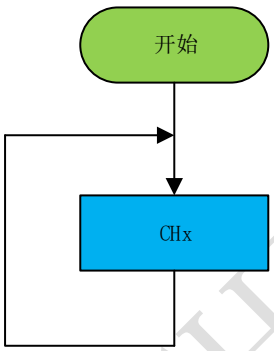

图 2.4 单通道连续转换

2.3.2 应用场景

此模式用于 ADC 不同监视电池电压、MCU 温度和调控孵化箱温度等场景。在用于调控孵化箱温度时，系统将不断读取温度值并与用户设定的温度值相比较。当温度低时，打开加热电阻的电源；当温度高时，关闭加热电阻的电源，从而保证孵化箱保持在用户设定的恒温值。

- 硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接
- 软件代码：见附录 3

2.4 多通道（扫描）、连续转换模式

2.4.1 原理演示

多通道（扫描）模式可用于 ADC 处于独立模式下对一组通道按照设定顺序进行依次转化。可以通过配置 ADC 的寄存器，使 ADC 以不同的采样时间和采样顺序对任意序列的通道进行配置（最多 12 个通道）。此模式的工作原理与多通道单次转换模式相似，不同点在一组序列的完成转换之后不会停止转换，而是继续从用户设定的第一个通道依次无限转换下去。

本应用笔记中提供了 ADC+DMA 模式实现对多通道的连续转换，转换的原理如图 2.5 所示。

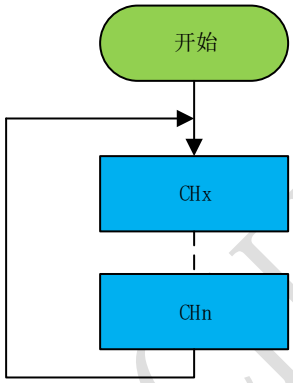

图 2.5 多通道连续转换

2.4.2 应用场景

此模式用于系统持续监视多个电压与温度模块。例如在电子烟示例种，系统持续读取内部参考电压、外部电池电压、内部 MCU 温度以及外部烤烟温度等。当电压或温度达到用户设定的最大值时，将切断加热模块或供电模块。

- 硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接
- 软件代码：见附录 4

3 提高 A/D 转化精度的技巧

3.1 软件滤波

软件滤波是一个非常有用的技巧，即取多次模拟输入值求和之后取均值的方法。此方法既有助于消除模拟输入上的噪声，又有利于减小错误转换的影响。

3.1.1 采样滤波算法

当使用该种方法，对转化值求和取平均时的除法可以通过右移位操作的方法，会使取平均计算更有效率。这样节省了 MCU 的运行时间，所以采样个数 N 应为 2 的倍数。在 Cortex-Mx 内核中，右移操作仅花费 1 个 CPU 周期。平均技巧的图形见下图 3.1 所示。

应用场景：用于测量一个模拟输入引脚上的电压。

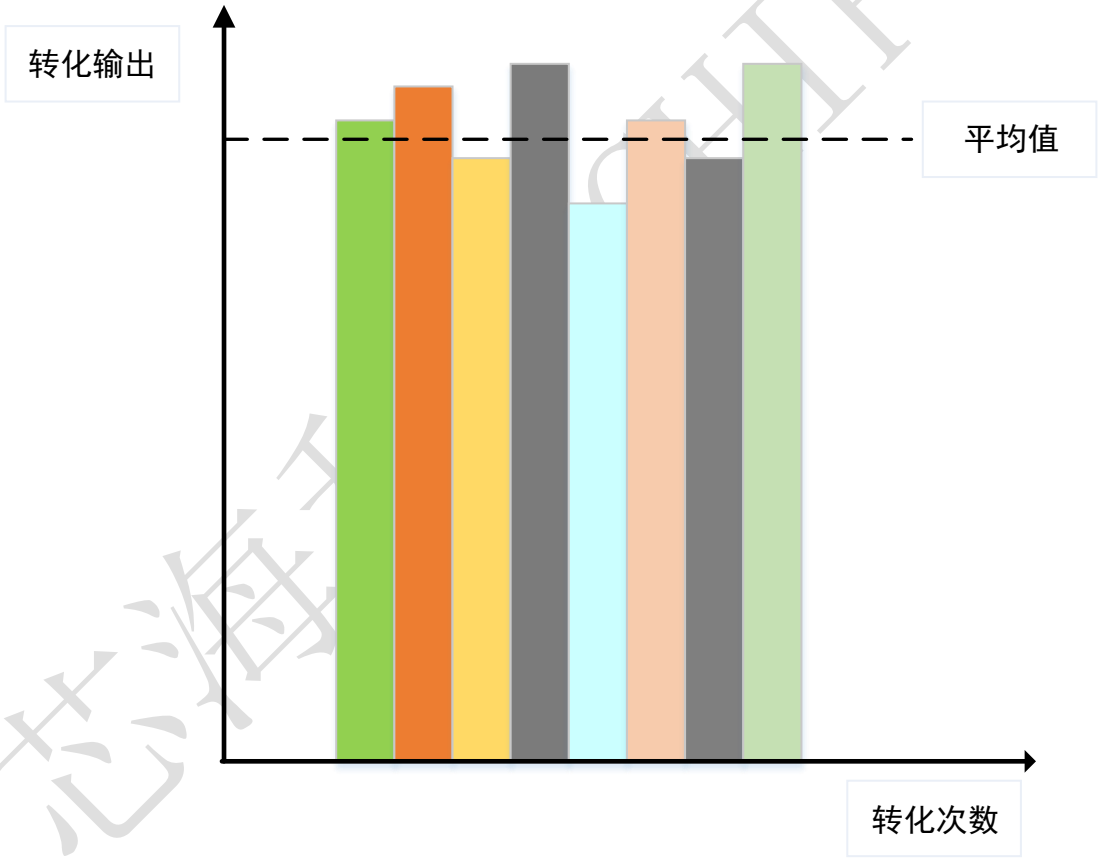

图 3.1N 次采样滤波算法图

ADC 模块的采样时间是可编程的，采样时间可以根据输入电压的阻抗、转换精度以及转换时间综合考虑。采样时间以 ADC 时钟周期为单位，采样时间直接关系到 ADC 模块的转换速率，其 ADC 转换时间的公式计算：

ADC 转换时间=采样时间+逐次逼近时间

总转换时间=（采样数 N*ADC 转换时间）+计算时间

因此，根据模拟信号的变化范围及计算可用的时间，对总转换时间和采样个数之间进行折中取值。工作流程图如下图 3.2 所示。

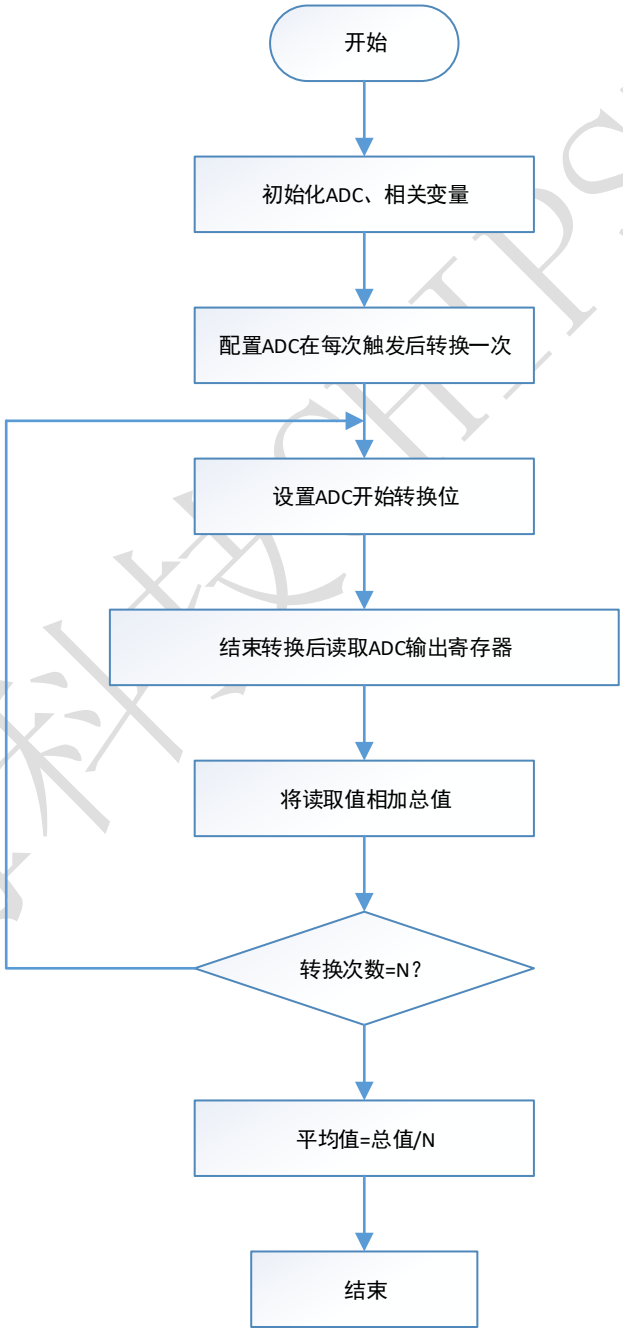

图 3.2 N 次采样算法流程图

注：采样算法函数实现原型为 uint16_t adc_get_ave(uint8_t n)。

硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接

软件代码：详见附录 5

3.1.2 移动滤波算法

此方法采样 N 个 ADC 码值，将数值按从大到小或从小到大排序，然后删除两端的 X 个采样的码值。此处删除了影响平均的两端值，更有效的减少了误差。此处 N 与 X 同样建议为 2 的倍数，通过位运算提高效率。N-X 个采样平均算法工作流程图如下图 3.3 所示。

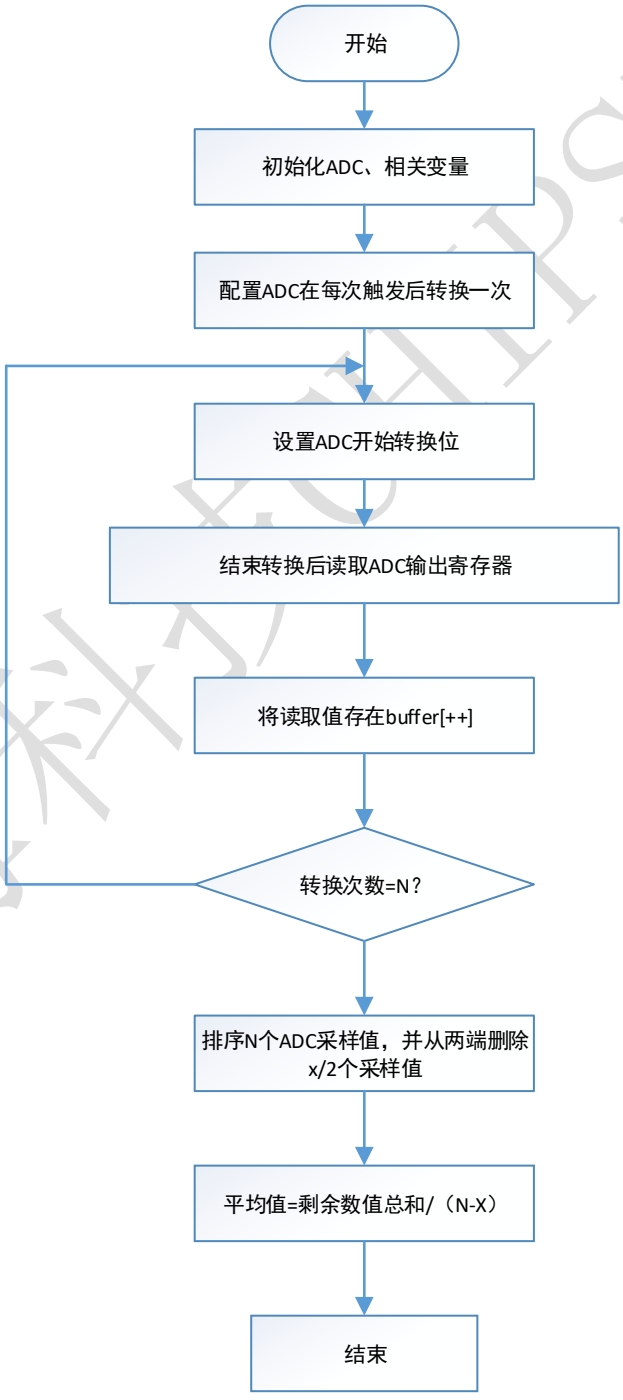

图 3.3 N-X 个采样算法流程图

注：n-x 采样值算法函数实现原型为 uint16_t adc_get_ave(uint8_t n,uint8_t x)，排序函数实现原型为 void sort_adc(uint16_t tab[],uint8_t number)。

硬件连接：将 PC 串口的 RX 和 TX 引脚分别与芯片对应的 PA9 与 PA10 引脚相连接

软件代码：详见附录 6

3.2 其他技巧

ADC 转换结果为输入采样电压与参考电压的比值。所以参考电压若受噪声影响，ADC 转换的码值也会不准确。因此提高转换精度，可以通过下面方法降低噪声。

- **硬件上：**

  MCU 在执行指令时在内部电源上产生一些噪声，若要过滤此噪声在硬件设计上使用 MCU 封装上的 Vdda 和 Vssa 模拟供电引脚，还可以连接一个电容滤波器至电源引脚以过滤高频噪声。

- **软件上：**

  1. **为了避免将噪声引入模拟输入引脚中，应避免在同一个 I/O 端口上切换数字输出。**
  2. **为了避免在电源中产生电源纹波，在转换的过程中避免切换高灌电流 I/O 口。**
  3. **为了避免供电电压中产生噪声，在 ADC 转换过程中避免启动外设的数据传输。**

4 内部参考电压跳变下计算输入电压

这个 12 位 ADC 是一个逐次逼近型模数转换器。CS32F03x 支持最多 13 个通道，包括 10 个外部通道和 3 个内部通道。不同通道的转换模式包括单次、扫描模式。在扫描模式下，将自动对选定的模拟输入通道组进行转换。

ADC 支持 DMA 访问。模拟看门狗特性允许应用程序监控一个、多个或者所有通道的转换电压。当转换电压超出软件编程的范围时，将会产生一个中断。

4.1 内部参考电压

内部电压参考（Vrefint）为 ADC 提供一个稳定的电压输出。Vrefint 被内部接到 ADIN_17 输入通道。每颗芯片的 Vrefint 精确电压在量产测试时被测量并存储在系统存储区，它是只读的。

表 4. 1 内部参考电压校准值

校准值名称	描述	存储地址
VREFINT_CAL	数值在 30℃（±5℃），V_{DDA}=3.3V（±10mV）获得	0x1FFF F7BA - 0x1FFF F7BB

4.2 使用内部参考电压计算实际 V_{DDA}

在 MCU 的 V_{DDA} 电源电压发生跳变的情况下，可以通过内部出厂嵌入的电压基准 Vrefint，和 ADC 在 V_{DDA}=3.3V 时获取的标定数据，用于计算实际跳变中的 V_{DDA} 电压值。

计算为设备供电的实际 V_{DDA} 电压公式：

$$V_{DDA}=3.3V* VREFINT\_CAL / VREFINT\_DATA$$

其中：

- VREFINT_CAL 是内部校准值，存放在 0x1FFF F7BA - 0x1FFF F7BB 地址中
- VREFINT_DATA 是通过实际 ADC 转换的码值

4.3 将 ADC 码值转化为绝对电压

在某些情况下，有必要将测量外部电压的比值转化为与 V_{DDA} 无关的电压。可以通过下面公式得到转化的绝对电压值：

$$Vchannelx=Vdda*ADC\_DATAx/FUALL\_SCALE$$

对于 V_{DDA} 未知的应用程序，必须使用 4.2 中的公式将 V_{DDA} 替换掉，替换后的结果如下：

$$Vchannelx=3.3V* VREFINT\_CAL * ADC\_DATAx/(VREFINT\_DATA* FUALL\_SCALE)$$

其中：

VREFINT_CAL **是内部校准值，存放在 0x1FFF F7BA - 0x1FFF F7BB 地址中**

ADC_DATAx 是通道 x 上测量 ADC 码值（右对齐）

VREFINT_DATA 是 ADC 转换的实际 VREFINT 输出值，通道 17ADC 码值

FUALL_SCALE 是 ADC 输出的最大值。若选择的是 12 采样，该值为 4095；若选择 8 位采样，该值为 255

注：ADC 转换的数据输出必须为右对齐的方式，若不是该方式在计算之前必须转化为兼容格式。

附录 1 单通道单次转换

```
/**
 * @file   ADC/ADC_BasicExample/main.c
 * @brief  Main program body
 * @author  ChipSea MCU Group
 * @version V1.0.0
 * @date   2018.11.01
 * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
 * @note
 * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
 *
 *
 */
#include  "main.h"
/** @addtogroup CS32F0xx_StdPeriph_Examples
 * @{
 */

/** @addtogroup ADC_sigleExample
 * @{
 */
void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf, uint8_t len);

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{

        uint32_t sample_value;
```

```
    uint32_t real_value;

    usart_com_init();
    adc_config();

    while (1)
    {
        while(adc_flag_status_get(ADC1, ADC_FLAG_EOCH) == RESET);
        sample_value = adc_conversion_value_get(ADC1);
        real_ value = (sample_value *3300)/0xFFF;
        printf("ADC_CH0 = %d mV",real_value);

    }

}

/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
    usart_config_t usart_configStruct;
    gpio_config_t  gpio_configStruct;

   // Clock Config
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);

   // GPIO MF Config
    gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
    gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

    gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;//PA9,PA10 USART
```

```
    gpio_configStruct.gpio_mode = GPIO_MODE_MF;
    gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
    gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
    gpio_init(GPIOA, &gpio_configStruct);

    // USART Config
    usart_def_init(USART1);
    usart_configStruct.usart_rate = 115200;
    usart_configStruct.data_width = USART_DATA_WIDTH_8;
    usart_configStruct.stop_bits = USART_STOP_BIT_1;
    usart_configStruct.usart_parity = USART_PARITY_NO;
    usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
    usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
    usart_init(USART1, &usart_configStruct);
    usart_enable_ctrl(USART1,ENABLE);
}

/**
 * @fn  void adc_config(void)
 * @brief  ADC Configuration
 * @param  None
 * @return None
 */
void adc_config(void)
{
    adc_config_t  adc_configStruct;
    gpio_config_t  gpio_configStruct;

    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE);

    //Configure ADC CH0 GPIO as analog input.
    gpio_configStruct.gpio_pin = GPIO_PIN_0 ;
    gpio_configStruct.gpio_mode = GPIO_MODE_AN;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
```

```
    gpio_init(GPIOA, &gpio_configStruct);

    adc_def_init(ADC1);
    adc_config_struct_init(&adc_configStruct);

    // Configure the ADC1 in continuous mode
    adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
    adc_configStruct.conversion_mode = DISABLE;      //A single conversion
    adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
    adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
    adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
    adc_init(ADC1, &adc_configStruct);

    // Set the ADC1 CH0 with 239.5 Cycles
    adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 ,
ADC_SAMPLE_TIMES_239_5);
    adc_calibration_value_get(ADC1); // ADC Calibration.
    adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.

    while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
    adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
    uint8_t i = 0;

    for(i = 0; i < len; i++)
    {
```

```
            usart_data_send(USART1,pbuf[i]);
            while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
        }
}
/**
 * @brief  Retargets the C library printf function to the USART.
 * @param  None
 * @return None
 */
int fputc(int ch, FILE *f)
{
  usart_data_send(USART1, (uint8_t) ch);
  while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
  return ch;
}
```

附录 2 多通道单次转换

```
/**
 * @file   ADC/ADC_BasicExample/main.c
 * @brief  Main program body
 * @author  ChipSea MCU Group
 * @version V1.0.0
 * @date   2018.11.01
 * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
 * @note
 * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
 *
 *
 */
#include  "main.h"

/** @addtogroup CS32F0xx_StdPeriph_Examples
 * @{
 */

/** @addtogroup ADC_multExample
 * @{
 */

#define ADC1_OUTDAT_REG_ADDRESS     0x40012440
#define TAB_SIZE                (sizeof(adc_conv)/2)
volatile uint16_t adc_conv[6];
void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf,uint8_t len);
void adc_dma_config(void);

/**
 * @brief  Main program.
 * @param  None
```

```
 * @retval None
 */
int main(void)
{
 uint8_t i;

 adc_dma_config();
 usart_com_init();
 adc_config();

        while (1)
        {
                while((dma_flag_status_get(DMA1_FLAG_CMP1)) == RESET );
                dma_flag_clear(DMA1_FLAG_CMP1); //Clear DMA CMP1 flag.
                for(i=0;i<TAB_SIZE;i++)
                {
                   printf("ADC_CH %d = %d \r\n ",i,adc_conv[i]);
                }
        }
}

/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
        usart_config_t usart_configStruct;
        gpio_config_t  gpio_configStruct;

       // Clock Config
        rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
        rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);
```

```
    // GPIO MF Config
    gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
    gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

    gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;//PA9,PA10 USART
    gpio_configStruct.gpio_mode = GPIO_MODE_MF;
    gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
    gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
    gpio_init(GPIOA, &gpio_configStruct);

    // USART Config
    usart_def_init(USART1);
    usart_configStruct.usart_rate = 115200;
    usart_configStruct.data_width = USART_DATA_WIDTH_8;
    usart_configStruct.stop_bits = USART_STOP_BIT_1;
    usart_configStruct.usart_parity = USART_PARITY_NO;
    usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
    usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
    usart_init(USART1, &usart_configStruct);
    usart_enable_ctrl(USART1,ENABLE);
}

/**
 * @fn  void dma_config(void)
 * @brief  DMA channel1 configuration
 * @param  None
 * @return None
 */
void adc_dma_config(void)
{
    dma_config_t  dma_configStruct;

    //Enable DMA1 clock
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_DMA1, ENABLE);
```

```
    // DMA1 Channel1 Config
    dma_def_init(DMA1_CHANNEL1);
    dma_configStruct.peri_base_addr = (uint32_t)ADC1_OUTDAT_REG_ADDRESS;
    dma_configStruct.mem_base_addr = (uint32_t)adc_conv;
    dma_configStruct.transfer_direct = DMA_TRANS_DIR_FROM_PERI;
    dma_configStruct.buf_size = TAB_SIZE;
    dma_configStruct.peri_inc_flag = DMA_PERI_INC_DISABLE;
    dma_configStruct.mem_inc_flag = DMA_MEM_INC_ENABLE;
    dma_configStruct.peri_data_width = DMA_PERI_DATA_WIDTH_HALFWORD;
    dma_configStruct.mem_data_width = DMA_MEM_DATA_WIDTH_HALFWORD;
    dma_configStruct.operate_mode = DMA_OPERATE_MODE_NORMAL; //DMA
single
    dma_configStruct.priority_level = DMA_CHANNEL_PRIORITY_HIGH;
    dma_configStruct.m2m_flag = DMA_M2M_MODE_DISABLE;
    dma_init(DMA1_CHANNEL1, &dma_configStruct);

    dma_enable_ctrl(DMA1_CHANNEL1, ENABLE); //DMA1 Channel1 enable
}


/**
 * @fn  void adc_config(void)
 * @brief  ADC Configuration
 * @param  None
 * @return None
 */
void adc_config(void)
{
    adc_config_t  adc_configStruct;
    gpio_config_t  gpio_configStruct;

  //GPIOA  clock enable.
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
    //ADC1  clock enable.
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE);
```

```
        //Configure ADC CH0 1 2 3 4 5 GPIO as analog input.
        gpio_configStruct.gpio_pin =
        GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5;
        gpio_configStruct.gpio_mode = GPIO_MODE_AN;
        gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
        gpio_init(GPIOA, &gpio_configStruct);

        adc_def_init(ADC1);
        adc_config_struct_init(&adc_configStruct);

        // Configure the ADC1 in continuous mode
        adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
        adc_configStruct.conversion_mode = DISABLE;      //A single conversion
        adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
        adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
        adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
        adc_init(ADC1, &adc_configStruct);

        // Set the ADC1 CH0 with 1.5 Cycles
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 , ADC_SAMPLE_TIMES_1_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_1 ,
ADC_SAMPLE_TIMES_13_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_2 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_3 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_4 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_5 , ADC_SAMPLE_TIMES_7_5);

        adc_calibration_value_get(ADC1); // ADC Calibration.
        adc_dma_mode_set(ADC1, ADC_DMA_MODE_SINGLE); // ADC DMA single mode .

        // Enable ADC_DMA
        adc_dma_enable_ctrl(ADC1, ENABLE);
        adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.
```

```
        while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
        adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
        uint8_t i = 0;

        for(i = 0; i < len; i++)
        {
                usart_data_send(USART1,pbuf[i]);
                while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
        }
}

/**
 * @brief  Retargets the C library printf function to the USART.
 * @param  None
 * @return None
 */
int fputc(int ch, FILE *f)
{
  usart_data_send(USART1, (uint8_t) ch);
  while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
  return ch;
}
```

附录 3 单通道连续转换

```
/**
 * @file   ADC/ADC_BasicExample/main.c
 * @brief  Main program body
 * @author  ChipSea MCU Group
 * @version V1.0.0
 * @date   2018.11.01
 * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
 * @note
 * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
 *
 *
 */
#include  "main.h"

#define ADC1_OUTDAT_REG_ADDRESS     0x40012440
#define TAB_SIZE                    (sizeof(adc_conv)/2)
volatile uint16_t adc_conv[1];
void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf,uint8_t len);
void adc_dma_config(void);

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
 uint8_t i;

   adc_dma_config();
   usart_com_init();
```

```
    adc_config();

        while (1)
        {
            //wait DMA1 CMP1 flag.
            while((dma_flag_status_get(DMA1_FLAG_CMP1)) == RESET );
            dma_flag_clear(DMA1_FLAG_CMP1); //Clear DMA CMP1 flag.
            for(i=0;i<TAB_SIZE;i++)
            {
            printf("ADC_CH %d = %d \r\n ",i,adc_conv[i]);
            }
        }
}

/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
        usart_config_t usart_configStruct;
        gpio_config_t  gpio_configStruct;

        // Clock Config
        rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
        rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);

       // GPIO MF Config
        gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
        gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

        gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;//PA9,PA10 USART
        gpio_configStruct.gpio_mode = GPIO_MODE_MF;
        gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
```

```
    gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
    gpio_init(GPIOA, &gpio_configStruct);

    // USART Config
    usart_def_init(USART1);
    usart_configStruct.usart_rate = 115200;
    usart_configStruct.data_width = USART_DATA_WIDTH_8;
    usart_configStruct.stop_bits = USART_STOP_BIT_1;
    usart_configStruct.usart_parity = USART_PARITY_NO;
    usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
    usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
    usart_init(USART1, &usart_configStruct);
    usart_enable_ctrl(USART1,ENABLE);
}

/**
 * @fn  void dma_config(void)
 * @brief  DMA channel1 configuration
 * @param  None
 * @return None
 */
void adc_dma_config(void)
{
    dma_config_t  dma_configStruct;

    //Enable DMA1 clock
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_DMA1, ENABLE);

    // DMA1 Channel1 Config
    dma_def_init(DMA1_CHANNEL1);
    dma_configStruct.peri_base_addr = (uint32_t)ADC1_OUTDAT_REG_ADDRESS;
    dma_configStruct.mem_base_addr = (uint32_t)adc_conv;
    dma_configStruct.transfer_direct = DMA_TRANS_DIR_FROM_PERI;
    dma_configStruct.buf_size = TAB_SIZE;
    dma_configStruct.peri_inc_flag = DMA_PERI_INC_DISABLE;
```

```
    dma_configStruct.mem_inc_flag = DMA_MEM_INC_ENABLE;
    dma_configStruct.peri_data_width = DMA_PERI_DATA_WIDTH_HALFWORD;
    dma_configStruct.mem_data_width = DMA_MEM_DATA_WIDTH_HALFWORD;
   //DMA circular mode
    dma_configStruct.operate_mode = DMA_OPERATE_MODE_CIRCULAR;
    dma_configStruct.priority_level = DMA_CHANNEL_PRIORITY_HIGH;
    dma_configStruct.m2m_flag = DMA_M2M_MODE_DISABLE;
    dma_init(DMA1_CHANNEL1, &dma_configStruct);

    dma_enable_ctrl(DMA1_CHANNEL1, ENABLE);  //DMA1 Channel1 enable
}


/**
 * @fn  void adc_config(void)
 * @brief  ADC Configuration
 * @param  None
 * @return None
 */
void adc_config(void)
{
    adc_config_t  adc_configStruct;
    gpio_config_t gpio_configStruct;

   //GPIOA  clock enable.
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
  //ADC1  clock enable.
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE);

    //Configure ADC CH0  GPIO as analog input.
    gpio_configStruct.gpio_pin = GPIO_PIN_0;
    gpio_configStruct.gpio_mode = GPIO_MODE_AN;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
    gpio_init(GPIOA, &gpio_configStruct);

    adc_def_init(ADC1);
```

```
    adc_config_struct_init(&adc_configStruct);

    // Configure the ADC1 in continuous mode
    adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
    adc_configStruct.conversion_mode = ENABLE;  //A circular conversion
    adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
    adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
    adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
    adc_init(ADC1, &adc_configStruct);

    // Set the ADC1 CH0 with 55.5 Cycles
    adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 ,
ADC_SAMPLE_TIMES_55_5);

    adc_calibration_value_get(ADC1);          // ADC Calibration.
    adc_dma_mode_set(ADC1, ADC_DMA_MODE_CIRCULAR); // ADC DMA circle
mode .

    // Enable ADC_DMA
    adc_dma_enable_ctrl(ADC1, ENABLE);
    adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.

    while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
    adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
    uint8_t i = 0;
```

```
    for(i = 0; i < len; i++)
    {
        usart_data_send(USART1,pbuf[i]);
        while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
    }
}

/**
 * @brief  Retargets the C library printf function to the USART.
 * @param  None
 * @return None
 */
int fputc(int ch, FILE *f)
{
  usart_data_send(USART1, (uint8_t) ch);
  while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
  return ch;
}
```

附录 4 多通道连续转换

```
/**
 * @file   ADC/ADC_BasicExample/main.c
 * @brief  Main program body
 * @author  ChipSea MCU Group
 * @version V1.0.0
 * @date   2018.11.01
 * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
 * @note
 * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
 *
 *
 */
#include  "main.h"

/** @addtogroup CS32F0xx_StdPeriph_Examples
 * @{
 */

/** @addtogroup ADC_multconvExample
 * @{
 */
#define ADC1_OUTDAT_REG_ADDRESS    0x40012440
#define TAB_SIZE                   (sizeof(adc_conv)/2)
volatile uint16_t adc_conv[6];
void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf,uint8_t len);
void adc_dma_config(void);

/**
 * @brief  Main program.
 * @param  None
 * @retval None
```

```
 */
int main(void)
{
 uint8_t i;

        adc_dma_config();
         usart_com_init();
         adc_config();

         while (1)
         {
                //wait DMA1 CMP1 flag.
                 while((dma_flag_status_get(DMA1_FLAG_CMP1)) == RESET );
                 dma_flag_clear(DMA1_FLAG_CMP1); //Clear DMA CMP1 flag.
                  for(i=0;i<TAB_SIZE;i++)
                  {
                  printf("ADC_CH %d = %d \r\n ",i,adc_conv[i]);
                  }
         }

}

/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
         usart_config_t usart_configStruct;
         gpio_config_t  gpio_configStruct;

        // Clock Config
         rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
         rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);
```

```
    // GPIO MF Config
    gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
    gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

    gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;//PA9,PA10 USART
    gpio_configStruct.gpio_mode = GPIO_MODE_MF;
    gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
    gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
    gpio_init(GPIOA, &gpio_configStruct);

    // USART Config
    usart_def_init(USART1);
    usart_configStruct.usart_rate = 115200;
    usart_configStruct.data_width = USART_DATA_WIDTH_8;
    usart_configStruct.stop_bits = USART_STOP_BIT_1;
    usart_configStruct.usart_parity = USART_PARITY_NO;
    usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
    usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
    usart_init(USART1, &usart_configStruct);
    usart_enable_ctrl(USART1,ENABLE);
}


/**
 * @fn  void dma_config(void)
 * @brief  DMA channel1 configuration
 * @param  None
 * @return None
 */
void adc_dma_config(void)
{
    dma_config_t  dma_configStruct;

    //Enable DMA1 clock
```

```
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_DMA1, ENABLE);

    // DMA1 Channel1 Config
    dma_def_init(DMA1_CHANNEL1);
    dma_configStruct.peri_base_addr = (uint32_t)ADC1_OUTDAT_REG_ADDRESS;
    dma_configStruct.mem_base_addr = (uint32_t)adc_conv;
    dma_configStruct.transfer_direct = DMA_TRANS_DIR_FROM_PERI;
    dma_configStruct.buf_size = TAB_SIZE;
    dma_configStruct.peri_inc_flag = DMA_PERI_INC_DISABLE;
    dma_configStruct.mem_inc_flag = DMA_MEM_INC_ENABLE;
    dma_configStruct.peri_data_width = DMA_PERI_DATA_WIDTH_HALFWORD;
    dma_configStruct.mem_data_width = DMA_MEM_DATA_WIDTH_HALFWORD;
    dma_configStruct.operate_mode = DMA_OPERATE_MODE_CIRCULAR;  //DMA
circul
    dma_configStruct.priority_level = DMA_CHANNEL_PRIORITY_HIGH;
    dma_configStruct.m2m_flag = DMA_M2M_MODE_DISABLE;
    dma_init(DMA1_CHANNEL1, &dma_configStruct);

    dma_enable_ctrl(DMA1_CHANNEL1, ENABLE);  //DMA1 Channel1 enable
}

/**
 * @fn  void adc_config(void)
 * @brief  ADC Configuration
 * @param  None
 * @return None
 */
void adc_config(void)
{
    adc_config_t  adc_configStruct;
    gpio_config_t  gpio_configStruct;

   //GPIOA  clock enable.
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
   //ADC1  clock enable.
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE);
```

```
        //Configure ADC CH0 1 2 3 4 5 GPIO as analog input.
        gpio_configStruct.gpio_pin =
        GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5;
        gpio_configStruct.gpio_mode = GPIO_MODE_AN;
        gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
        gpio_init(GPIOA, &gpio_configStruct);

        adc_def_init(ADC1);
        adc_config_struct_init(&adc_configStruct);

        // Configure the ADC1 in continuous mode
        adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
        adc_configStruct.conversion_mode = ENABLE;      //A circul conversion
        adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
        adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
        adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
        adc_init(ADC1, &adc_configStruct);

        // Set the ADC1 CH0 with 1.5 Cycles
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 , ADC_SAMPLE_TIMES_1_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_1 ,
ADC_SAMPLE_TIMES_13_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_2 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_3 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_4 ,
    ADC_SAMPLE_TIMES_71_5);
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_5 , ADC_SAMPLE_TIMES_7_5);

        adc_calibration_value_get(ADC1); // ADC Calibration.
        adc_dma_mode_set(ADC1, ADC_DMA_MODE_CIRCULAR); // ADC DMA circul
mode .

        // Enable ADC_DMA
        adc_dma_enable_ctrl(ADC1, ENABLE);
```

```
    adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.

    while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
    adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
    uint8_t i = 0;

    for(i = 0; i < len; i++)
    {
        usart_data_send(USART1,pbuf[i]);
        while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
    }
}

/**
 * @brief  Retargets the C library printf function to the USART.
 * @param  None
 * @return None
 */
int fputc(int ch, FILE *f)
{
  usart_data_send(USART1, (uint8_t) ch);
  while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
  return ch;
}
```

附录 5 软件滤波

```
/**
  * @file    ADC/ADC_BasicExample/main.c
  * @brief   Main program body
  * @author  ChipSea MCU Group
  * @version V1.0.0
  * @date    2018.11.01
  * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
  * @note
  * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
  *
  *
  */
#include   "main.h"

/** @addtogroup CS32F0xx_StdPeriph_Examples
  * @{
  */

/** @addtogroup ADC_multExample
  * @{
  */

void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf,uint8_t len);
uint16_t adc_get_ave(uint8_t n);

/**
  * @brief  Main program.
  * @param  None
  * @retval None
  */
int main(void)
{
      uint32_t real_Value;

      usart_com_init();
      adc_config();

      real_Value = adc_get_ave(8);
      printf("ADC_CH0 = %d ",real_Value);
      while (1)
      {

      }

}

/**
 * @fn uint16_t adc_get_ave(uint8_t n)
 * @brief  Get n of ADC average data
 * @param  The number of sampling
 * @returnaverage data
 */

uint16_t adc_get_ave(uint8_t n)
```

```
{
  uint32_t all_adc=0x00;
      uint16_t adc_data[8]={0,0,0,0,0,0,0,0};
      uint8_t i=0;

  for(i=0;i<n;i++)
      {
        adc_conversion_start(ADC1);
        while(adc_flag_status_get(ADC1, ADC_FLAG_EOCH) == RESET);
        adc_data[i]=adc_conversion_value_get(ADC1);
      }

      for(i=0;i<n;i++)
      {
        all_adc+=adc_data[i];
      }

  all_adc/=n;

      return all_adc;
}



/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
      usart_config_t usart_configStruct;
      gpio_config_t  gpio_configStruct;

      // Clock Config
      rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
      rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);

      // GPIO MF Config
      gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
      gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

      gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;
      gpio_configStruct.gpio_mode = GPIO_MODE_MF;
      gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
      gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
      gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
      gpio_init(GPIOA, &gpio_configStruct);

      // USART Config
      usart_def_init(USART1);
      usart_configStruct.usart_rate = 115200;
      usart_configStruct.data_width = USART_DATA_WIDTH_8;
      usart_configStruct.stop_bits = USART_STOP_BIT_1;
      usart_configStruct.usart_parity = USART_PARITY_NO;
      usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
      usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
      usart_init(USART1, &usart_configStruct);
```

```
        usart_enable_ctrl(USART1,ENABLE);
}

/**
  * @fn  void adc_config(void)
  * @brief  ADC Configuration
  * @param  None
  * @return None
  */
void adc_config(void)
{
        adc_config_t   adc_configStruct;
        gpio_config_t  gpio_configStruct;

        rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE); //GPIOA
clock enable.
        rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE); //ADC1
clock enable.

        //Configure ADC CH0 GPIO as analog input.
        gpio_configStruct.gpio_pin = GPIO_PIN_0 ;
        gpio_configStruct.gpio_mode = GPIO_MODE_AN;
        gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
        gpio_init(GPIOA, &gpio_configStruct);

        adc_def_init(ADC1);
        adc_config_struct_init(&adc_configStruct);

        // Configure the ADC1 in continuous mode
        adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
        adc_configStruct.conversion_mode = DISABLE;         //A single conversion
        adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
        adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
        adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
        adc_init(ADC1, &adc_configStruct);

        // Set the ADC1 CH0 with 239.5 Cycles
        adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 , ADC_SAMPLE_TIMES_239_5);
        adc_calibration_value_get(ADC1); // ADC Calibration.
        adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.

        while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
        adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
        uint8_t i = 0;

        for(i = 0; i < len; i++)
        {
                usart_data_send(USART1,pbuf[i]);
                while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
```

```
        }
}

/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  None
  * @return None
  */
int fputc(int ch, FILE *f)
{
    usart_data_send(USART1, (uint8_t) ch);
    while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
    return ch;
}
```

附录 6 移动滤波

```
/**
  * @file    ADC/ADC_BasicExample/main.c
  * @brief   Main program body
  * @author  ChipSea MCU Group
  * @version V1.0.0
  * @date    2018.11.01
  * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
  * @note
  * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
  *
  *
  */
#include   "main.h"

/** @addtogroup CS32F0xx_StdPeriph_Examples
  * @{
  */

/** @addtogroup ADC_multExample
  * @{
  */

void adc_config(void);
void usart_com_init(void);
void usart_send(uint8_t * pbuf,uint8_t len);
void sort_adc(uint16_t tab[],uint8_t number);
uint16_t adc_get_ave(uint8_t n,uint8_t x);

/**
  * @brief  Main program.
  * @param  None
  * @retval None
  */
int main(void)
{
      uint32_t real_value;

      usart_com_init();
      adc_config();

      real_value = adc_get_ave(8,2);
      printf("ADC_CH0 = %d ",real_value);
      while (1)
      {

      }

}

/**
 * @fn uint16_t adc_get_ave(uint8_t n,uint8_t x)
 * @brief  Get n-x of ADC average data
 * @param n: The number of sampling
 * @param x: The number of filter,it is even number
 * @returnaverage data
 */
```

```
uint16_t adc_get_ave(uint8_t n,uint8_t x)
{
    uint32_t all_adc=0x00;
      uint16_t adc_data[8]={0,0,0,0,0,0,0,0};
      uint8_t i=0;

    for(i=0;i<n;i++)
    {
      adc_conversion_start(ADC1);
      while(adc_flag_status_get(ADC1, ADC_FLAG_EOCH) == RESET); // Check EOC
flag
      adc_data[i]=adc_conversion_value_get(ADC1);
    }
    sort_adc(adc_data,n);
      for(i=(x/2);i<n-(x/2);i++)
      {
        all_adc+=adc_data[i];
      }

    all_adc/=(n-x);

      return all_adc;
}




/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
void usart_com_init(void)
{
      usart_config_t usart_configStruct;
      gpio_config_t  gpio_configStruct;

     // Clock Config
      rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
      rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);

      // GPIO MF Config
      gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
      gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

      gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10;        //PA9,PA10
USART
      gpio_configStruct.gpio_mode = GPIO_MODE_MF;
      gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
      gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
      gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
      gpio_init(GPIOA, &gpio_configStruct);

      // USART Config
      usart_def_init(USART1);
      usart_configStruct.usart_rate = 115200;
      usart_configStruct.data_width = USART_DATA_WIDTH_8;
      usart_configStruct.stop_bits = USART_STOP_BIT_1;
```

```
	usart_configStruct.usart_parity = USART_PARITY_NO;
	usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
	usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
	usart_init(USART1, &usart_configStruct);
	usart_enable_ctrl(USART1,ENABLE);
}

/**
  * @fn  void adc_config(void)
  * @brief  ADC Configuration
  * @param  None
  * @return None
  */
void adc_config(void)
{
	adc_config_t   adc_configStruct;
	gpio_config_t  gpio_configStruct;

	rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE); //GPIOA
clock enable.
	rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_ADC, ENABLE); //ADC1
clock enable.

	//Configure ADC CH0 GPIO as analog input.
	gpio_configStruct.gpio_pin = GPIO_PIN_0 ;
	gpio_configStruct.gpio_mode = GPIO_MODE_AN;
	gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL ;
	gpio_init(GPIOA, &gpio_configStruct);

	adc_def_init(ADC1);
	adc_config_struct_init(&adc_configStruct);

	// Configure the ADC1 in continuous mode
	adc_configStruct.adc_resolution = ADC_CONV_RES_12BITS;
	adc_configStruct.conversion_mode = DISABLE;        //A single conversion
	adc_configStruct.trigger_mode = ADC_TRIG_MODE_SEL_NONE;
	adc_configStruct.data_align = ADC_DATA_ALIGN_RIGHT;
	adc_configStruct.scan_direction = ADC_CONV_SEQ_DIR_UPWARD;
	adc_init(ADC1, &adc_configStruct);

	// Set the ADC1 CH0 with 239.5 Cycles
	adc_channel_config(ADC1, ADC_CONV_CHANNEL_0 , ADC_SAMPLE_TIMES_239_5);
	adc_calibration_value_get(ADC1); // ADC Calibration.
	adc_enable_ctrl(ADC1, ENABLE);  // Enable the ADC.

	while(!adc_flag_status_get(ADC1, ADC_FLAG_EOI)); // Wait the EOI flag.
	adc_conversion_start(ADC1);  //ADC1 regular Software Start Conv.
}
/**
 * @fn void sort_tab(uint16_t tab[],uint8_t number)
 * @brief  Sort two channel values
 * @param  tab[]:the data of adc convern
 * @param  number:the number of channel
 * @return none
 */

void sort_adc(uint16_t tab[],uint8_t number)
{
    uint8_t i=0,exchange=1;;
    uint16_t tmp=0;
```

```
    while(exchange==1)
    {
        for(i=0;i<number-1;i++)
        {
            if(tab[i]>tab[i+1])
            {
                tmp=tab[i];
                tab[i]=tab[i+1];
                tab[i]=tmp;
                exchange=1;

            }
        }
       exchange=0;
    }
}

/**
 * @fn void usart_send(uint8_t * pbuf,uint8_t len)
 * @brief  Transmits  data through the Usart peripheral.
 * @param  pbuf: The pointer to buffer being transmited data.
 * @param  len: The data length.
 * @return None
 */
void usart_send(uint8_t * pbuf,uint8_t len)
{
      uint8_t i = 0;

      for(i = 0; i < len; i++)
      {
            usart_data_send(USART1,pbuf[i]);
            while(usart_flag_status_get(USART1,USART_FLAG_TXE) == RESET);
      }
}

/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  None
  * @return None
  */
int fputc(int ch, FILE *f)
{
    usart_data_send(USART1, (uint8_t) ch);
    while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
    return ch;
}
```