ADC原理及应用

9.1 ADC概述

将模拟量转换为数字量的过程称为模式（A/D）转换，完成这一转换的器件称为模数转换器（ADC）。

9.1.1 STM32的ADC功能及结构

STM32F103RB系列产品内嵌2个12位的模拟/数字转换器（ADC）。ADC的分辨率位12位，供电为2.4 ~ 3.6V，输入范围为0 ~ 3.6V。

具体功能：

- 规则转换和注入转换均有外部触发选型。
- 在规则通道转换期间，可以产生DMA请求。
- 自校准，在每次ADC开始转换前进行一次自校准。
- 通道采样间隔时间可编程。
- 带内嵌数据一致性的数据对齐。
- 可设置成单次、连续、扫描、间断模式。
- 双ADC模式，带2个ADC设备ADC1和ADC2，有8种转换方式。

ADC硬件结构主要由如下4个部分组成：

1. 模拟信号通道。
2. A/D转换器。
3. 模拟看门狗部分。
4. 中断电路。

9.1.2 STM32的ADC工作模式

单次转换模式

ADC只执行一次转换。

连续转换模式

当前面ADC转换一结束马上就启动另一次转换。

扫描模式

此模式用来扫描一组模拟通道。

间断模式

1. 规则组
   设置一个段序列的n（n<=8）此转换。被转换的通道序列将会依次转换n个通道，并且不会自动从头开始。
   eg：n=3，被转换的通道 = 0, 1, 2, 3, 6, 7, 9, 10。
   则转换顺序为：0, 1, 2 ; 3, 6, 7; 9, 10 ; 0, 1, 2 ...
2. 注入组

模拟量和数字量之间的转换

$$\frac{A - V_{REF-}}{V_{REF+} - V_{REF-}} = \frac{D - 0}{(2^R - 1) - 0}$$

其中A是模拟电压，D是数字电压，R是分辨率。V_{REF-}和V_{REF+}是输入范围的下界和上界。

9.1.3 STM32的ADC库函数

ADC_Init()

void ADC_Init(ADC_TypeDef *ADCx, ADC_InitTypeDef *ADC_InitStruct)

其中ADC_InitTypeDef结构体的定义如下：

```
typedef struct
{
    u32 ADC_Mode;           // ADC工作模式（独立、双ADC）
    FunctionalState ADC_ScanConvMode;   // ADC转换工作在扫描还是单次模式
    FunctionalState ADC_ContinuousConvMode;     // 工作在连续模式还是单次模式
    u32 ADC_ExternalTrigConv;   // 使用外部触发规则来启动规则通道的模数转换
    u32 ADC_DataAlign;      // ADC数据向左边对齐还是向右边对齐
    u8 ADC_NbrOfChannel;    // 按顺序进行转换的ADC通道数目
} ADC_InitTypeDef;
```

ADC_StartCalibration()

void ADC_StartCalibration(ADC_TypeDef* ADCx)

开始指定ADC的校准状态。

ADC_SoftwareStartConvCmd()

void ADC_SoftwareStartConvCmd(ADC_TypeDef *ADCx, FunctionalState NewState)

使能或者失能指定的ADC的软件转换启动功能。

ADC_RegularChannelConfig()

void ADC_RegularChannelConfig(ADC_TypeDef *ADCx, u8 ADC_Channel, u8 Rank, u8 ADC_SampleTime)

设置指定ADC的规则组通道。

ADC_GetConversionValue()

void ADC_GetConversionValue(ADC_TypeDef *ADCx)

返回最近一次ADCx规则组的转换结果。

9.1.4 ADC实例说明

```
/*
    利用滑动变阻器产生的电压，通过STM32芯片的PC0引脚采集。
    并通过串口把这个电压值传送到电脑。
*/

// 初始化时钟
void rcc_init()
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|RCC_APB2Periph_ADC1|
    RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO, ENABLE);
}

// 初始化GPIO口
void gpio_init()
{
    // 初始化采集的PC0引脚
    GPIO_InitTypeDef gpio_init;
    gpio_init.GPIO_Pin = GPIO_Pin_0;
    gpio_init.GPIO_Mode = GPIO_Mode_AIN;
    gpio_init.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &gpio_init);

    // 初始化串口发送引脚
    gpio_init.GPIO_Pin = GPIO_Pin_9;
    gpio_init.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &gpio_init);
    gpio_init.GPIO_Pin = GPIO_Pin_10;
    gpio_init.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &gpio_init);
}

// 初始化串口
void usart_init()
{
    USART_InitTypeDef usart_init = {
        .USART_BaudRate = 9600,
        .USART_WordLength = USART_WordLength_8b,
        .USART_StopBits = USART_StopBits_1,
        .USART_Parity = USART_Parity_No,
        .USART_HardwareFlowControl = USART_HardwareFlowControl_None,
        .USART_Mode = USART_Mode_TX|USART_Mode_RX,
    }
    USART_Init(USART1, &usart_init);
    USART_Cmd(USART, ENABLE);
}

// 发送数据模块
void usart_send_data(unsigned char data)
{
    USART_SendData(data);

    while(!USART_GetFlagStatus(USART1, USART_FLAG_TC));
}

// 初始化ADC模块
```

```
// 初始化ADC模块
void adc_init()
{
    ADC_InitTypeDef adc_init;
    ADC_DeInit(ADC1);
    adc_init.ADC_Mode = ADC_Mode_Independent;
    adc_init.ADC_ScanConvMode = DISABLE;
    adc_init.ADC_ContinuousConvMode = ENABLE;
    adc_init.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    adc_init.ADC_DataAlign = ADC_DataAlign_Right
    adc_init.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &adc_init);
    RCC_ADCCLKConfig(RCC_PCLK2_Div2);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_239Cycles5);
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibrationsStatus(ADC1);
    while(ADC_GetCalibrationsStatus(ADC1));
    ADC_StartCalibrationStatus(ADC1);
    while(ADC_GetCalibrationsStatus(ADC1));
}

int main()
{
    rcc_init();
    gpio_init();
    usart_init();
    adc_init();
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
    while(1)
    {
        if(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC))
        {
            unsigned char voltage = 3.3f/4096 * ADC_GetConverionValue(ADC1);
            usart_send_data(voltage);
            delay_moment();
        }
    }
    return 0;
}
```

ADC配置总结

1. 初始化输入的引脚GPIO口。
2. 初始化ADC模块：
   初始化ADC_InitTypeDef类型的结构变量，并设定字段：
   ADC_Mode、ADC_ScanConvMode、ADC_ContinuousConvMode、ADC_ExternalTrigConv、ADC_DataAlign、ADC_NbrOfChannel。
   调用函数初始化ADC_Init(ADCx, &adc_init)。
3. 初始化ADC时钟：
   RCC_ADCCLKConfig(RCC_PCLK2_Div2);

4. 指定ADC规则组通道：
   ADC_RegularChannelConfig(ADCx, ADC_Channel_y, z, ADC_SampleTime_uCyclesi);
5. 开启ADC：

```
ADC_Cmd(ADCx, ENABLE);
ADC_ResetCalibrationStatus(ADCx);
while(ADC_GetCalibrationStatus(ADCx));
ADC_StartCalibrationStatus(ADCx);
while(ADC_GetCalibrationStatus(ADCx));
ADC_SoftwareStartConvCmd(ADCx, ENABLE);
```

6. 获取ADC转换信息：

```
if(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC))
    value = ADC_GetConversionValue(ADCx)
```