# STM32F4 ADC介绍

## 简介

模数转换器，将模拟信号转换为数字信号。转换原理主要为逐次逼近型、双积分型、电压频率转换型三种。而本ADC呢是逐次逼近型的模拟数字转换器。

STM32F4系列一般都有**3个ADC**，这些ADC可以独立使用，也可以使用**双重/三重模式**（提高采样率）。STM32F4的ADC是**12位逐次逼近型**的模拟数字转换器。

1. 多达19个复用通道，可以测量来自16个外部源、2个内部源和Vbat通道的信号。

2. 这些通道的A/D转换可以**单次、连续、扫描或间断模式**执行。

3. 结果可以左对齐（4-15位）或右对齐（0-11位）的方式存储在16位数据寄存器中。

4. ADC具有**模拟看门狗特性**，允许应用程序检测输入电压是否超出用户定义的阀值上限或下限。

## ADC框图

按照顺序逐步分析ADC的框图，以便编程。

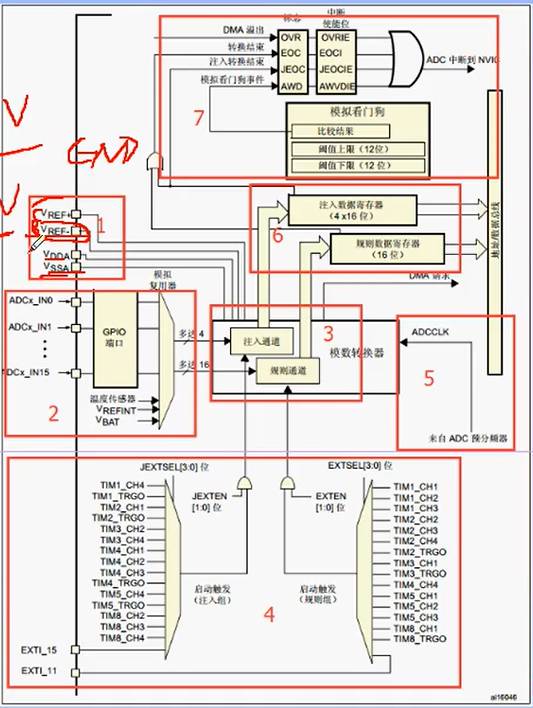


图 1 STM32F4 ADC内部框图

### 标号1：参考电压

VREF+和VDDA连接0~3.3V的参考电压，是ADC转换的参考电压。

VREF-和VSSA接GND。

### 标号2：输入通道

总共有3个ADC转换器以及16个外部输入通道，连接需要进行模/数转换的电压。另外还有3个内部通道，只有ADC1。

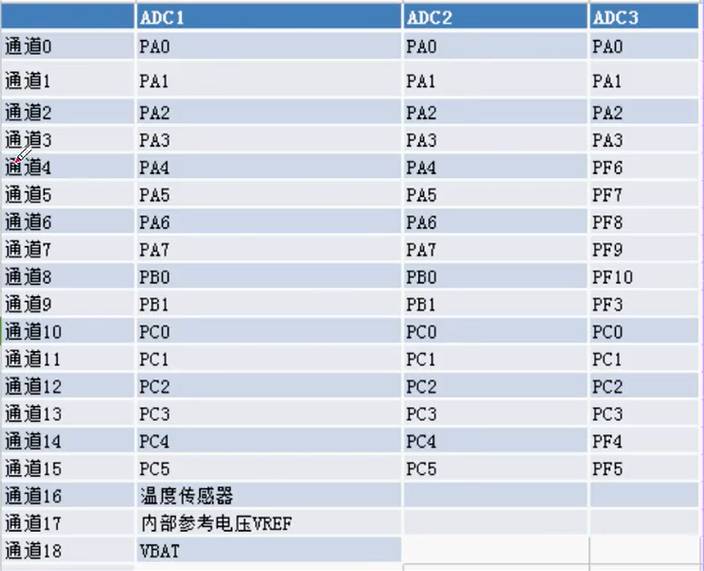


图 2 ADC通道对应管脚标号

### 标号3：通道转换顺序

外部的16个通道再转换的时候可分为2组通道：**规则通道组**和**注入通道组**，其中规则通道组最多有16路，注入通道组最多有4路。

**规则通道组：**从名字来理解，规则通道就是一种规矩的通道，类似于正常执行的程序，通常我们使用的都是这个通道。

**注入通道组：**从名字来理解，注入即为插入，是一种不安分的通道，类似于中断。当程序正常往下执行时，中断可以打断程序的执行。同样如果再规则通道的转换过程中，有注入通道插入，那么就要先转换完注入通道，等注入通道转换完成后再回到规则通道的转换流程。

每个组包含一个转换序列，该序列可按任意顺序在任意通道上完成。

### 标号4：触发

分为外部触发和内部触发。

### 标号5：分频器

对ADC内部84M的时钟进行分频，分频有2、4、6、8四种分频参数。当选择4时，则输入ADC的时钟为84/4=21M的时钟。

### 标号6：数据寄存器

ADC转换后的数据根据转换组的不同，规则组的数据放在ACD\_DR寄存器内，注入组的数据放在JDR×内。如果时使用双重或者三重模式，那规矩组的数据是存放在通用规矩寄存器ADC\_CDR内的。

因为STM32F4的ADC是12位转换精度，而数据寄存器是16位，所以ADC在存放数据的时候就有左对齐和右对齐区分。如果是左对齐，AD转换完成数据存放在ADC\_DR寄存器的[4:15]位内；如果是右对齐，则存放在ADC\_DR寄存器的[0:11]位内。具体选择何种存放方式，需通过ADC\_CR2的11位ALIGN设置。

编程要注意这些寄存器的设置，如果不是特别清楚最好参考一下手册上的资料。

# STM32F4 ADC配置步骤

## 步骤

### 使能端口时钟和ADC时钟，设置引脚模式为模拟输入

1. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);
2. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE);
3. GPIO\_InitStructure, GPIO\_Mode = GPIO\_Mode\_AN;//模拟输入模式

### 设置ADC通用控制寄存器CCR，包括ADC模式、ADC输入时钟分频等

1. Void ADC\_CommonInit(ADC\_CommonInitTypeDef\* ADC\_CommonInitStruct);
2. **typedef** **struct**
3. {
4. unit32\_t ADC\_Mode; //ADC模式选择
5. unit32\_t ADC\_Prescaler; //ADC分频系数
6. unit32\_t ADC\_DMAAccessMode;//ADC DMA模式配置
7. unit32\_t ADC\_TwoSamplingDelay; //ADC采样延迟
8. }ADC\_CommonInitTypedef;

3. 初始化ADC，包括ADC分辨率、转换模式、数据对齐方式等

1. **void** ADC\_Init(ADC\_TypeDef\* ADC×, ADC\_InitTypeDef\* ADC\_InitStruct);
2. **typedef** **struct**
3. {
4. unit32\_t ADC\_Resolution; //ADC分辨率选择
5. FunctionalState ADC\_ScanConvMode;//ADC扫描模式选择
6. FunctionalState ADC\_ContinuousConvMode;//ADC连续转换模式选择
7. unit32\_t ADC\_ExternalTriConvEdge;//外部触发极性
8. unit32\_t ADC\_ExternalTrigConv;//ADC外部触发选择
9. unit32\_t ADC\_DataAlign;//ADC数据对齐方式
10. unit8\_t ADC\_NbrOfConversion;//ADC规则序列长度
11. }ADC\_InitTypeDef;

### 开启ADC

1. **void** ADC\_Cmd(ADC\_TypeDef\* ADC×, FunctionalState NewState);
2. ADC\_Cmd(ADC1\_ENABLE);//开启AD转换器

### 读取ADC转换值

设置规则序列通道以及采样周期的库函数是：

1. **void** ADC\_RegularChannelConfig(ADC\_TypeDef\* ADC×, unit8\_t ADC\_Channel, unit8\_t Rank, unit8\_t ADC\_SampleTime);
3. ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_5, 1, ADC\_SampleTime\_480Cycles);

设置好规则序列通道及采样周期，接下来就要开启转换，由于我们采用的是软件触发，库函数

1. **void** ADC\_SoftwareStartConv(ADC\_TypeDef\* ADC×);

开启转换之后，就可以获取ADC转换结果数据，调用的库函数是：

1. unit16\_t ADC\_GetConversionValue(ADC\_TypeDef\* ADC×);

例如我们要判断 ADC1 的转换是否结束，方法是：

1. **while**(!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC));//等待转换结束

# 编写ADC控制程序

本章所要实现的功能是：通过 ADC1 通道5采样外部电压值，将采样的AD值和转换后的电压值通过串口打印出来，同时 D1 指示灯闪烁，提示系统正常运行。程序框架如下：

1. 初始化 ADC1\_IN5 相关参数，开启 ADC1

2. 编写获取 ADC1\_IN5 的AD转换值函数

3. 编写主函数

## 初始化 ADC1\_IN5 相关参数，开启 ADC1

1. **void** ADC×\_Init()
2. {
3. GPIO\_InitTypeDef GPIO\_InitStructure;//定义结构体变量
4. ADC\_CommonInitTypeDef ADC\_CommonInitSructure;//定义ADC配置结构体变量
5. ADC\_InitTypeDef ADC\_InitStructure;
7. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);
8. RCC\_AHB2PeriphClockCmd(RCC\_AHB2Periph\_ADC1, ENABLE);
10. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AN;//模拟输入模式
11. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;//管脚设置PA5
12. GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;//浮空
13. GPIO\_Init(GPIOA, &GPIO\_InitStructure;//初始化结构体
15. ADC\_CommonInitStructure.ADC\_Mode = ADC\_Mode\_Independent;
16. ADC\_CommonInitStructure.ADC\_TwoSamplingDelay=ADC\_TwoSamplingDelay\_5Cycles;
17. ADC\_CommonInitStructure.ADC\_DMAAccessMode=ADC\_DMAAccessMode\_Disabled;
18. ADC\_CommonInitStructure.ADC\_Prescaler=ADC\_Prescaler\_Div4;
19. ADC\_CommonInit(&ADC\_CommonInitStructure);
21. ADC\_InitStructure.ADC\_Resolution =  ADC\_Resolution\_12b;//12位模式
22. ADC\_InitStructure.ADC\_ScanConvMode = DISABLE;//非扫描模式
23. ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE;//关闭连续转换
24. ADC\_InitStructure.ADC\_ExternalTrigConvEdge = ADC\_ExternalTrigConvEdge\_None;//禁止触发检测
25. ADC\_InitStructure.ADC\_DataAlign =   ADC\_DataAlign\_Right;//右对齐
26. ADC\_InitStructure.ADC\_NbrOfConversion = 1;//1个转换在规则序列中，也就是只转换规则序列1
27. ADC\_Init(ADC1, &ADC\_InitStructure);//ADC初始化
29. }

## 编写获取 ADC1\_IN5 的AD转换值函数

1. u16 Get\_ADC\_Value(u8 ch, u8 times)
2. {
3. u32 temp\_val = 0;
4. u8 t;
6. ADC\_RegularChannelConfig(ADC1, ch, 1, 480Cycles);
7. ADC\_SoftwareStartConv(ADC1);
9. fot(t = 0; t < times; t++)
10. {
11. **while**(!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC));//等待转换结束
13. temp\_val += ADC\_GetConversionValue(ADC1);
14. }
16. **return** temp\_val/times;
17. }