嵌入式系统原理及实验

顾　震

信息科学与工程学院自动化系

华东理工大学

Email: guzhen@ecust.edu.cn

课程大纲

1. 嵌入式系统导论
2. Cortex-M3微处理器
3. STM32最小系统及开发环境
4. 嵌入式C语言
5. 通用输入输出GPIO模块
6. 中断
7. 定时器原理与应用
8. USART通信原理及实现
9. DMA控制器
10. SPI与I2C通信原理及实现
11. 模数转换原理及实现
12. 人工智能辅助的嵌入式项目开发
13. 嵌入式应用前沿

11. 模数转换原理及实现

本章知识与能力要求

- 理解和掌握ADC的转换过程、主要技术参数；
- 理解STM32的ADC内部结构；
- 掌握基于HAL库的ADC开发。

11. 模数转换原理及实现

11.1 ADC基础理论知识

11.2 STM32的ADC模块

11.3 ADC模块的标准外设库接口函数及应用

11.3 ADC模块的HAL库接口函数及应用

11.1 ADC基础理论知识

11.1.1 AD转换过程

11.1.2 AD转换的主要技术参数

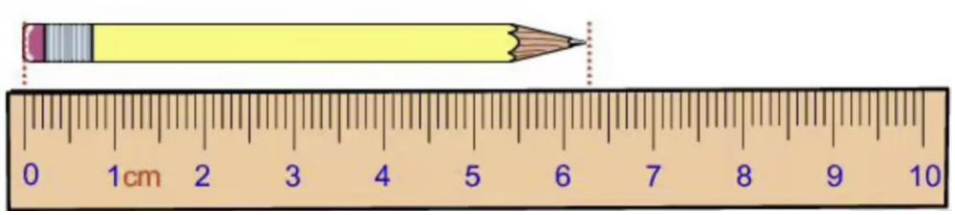
11.1.3 AD转换分类及原理

11.1.4 AD转换输入模式

11.1.1 A/D转换过程

- ADC（Analog to Digital Converter）即模数转换器，用来将**模拟信号**转换为**数字信号**。

尺子测量长度

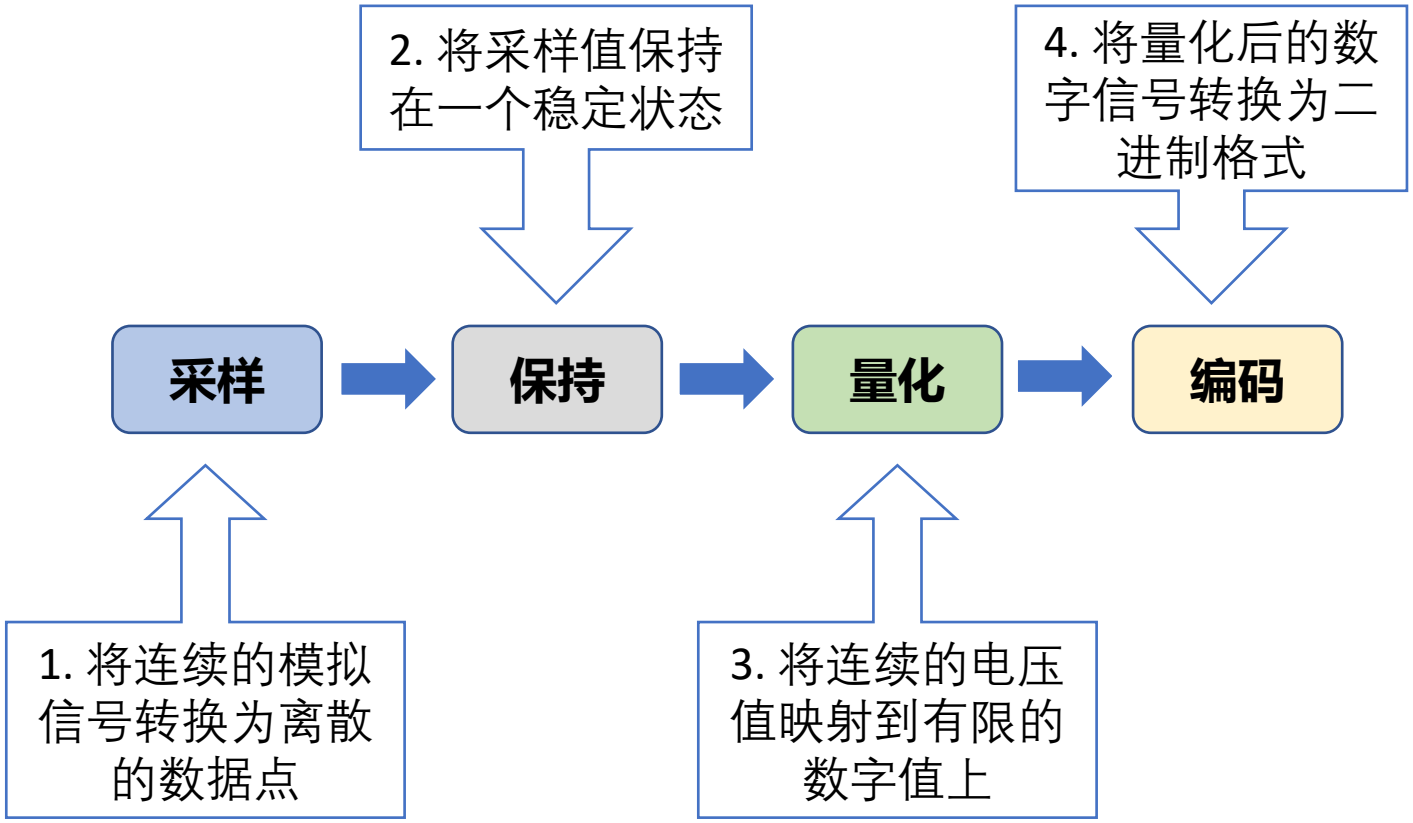


- 零刻度线
- 量程
- 分度值

ADC模数转换器相当于用来测量电量（通常为电压）的“尺子”。

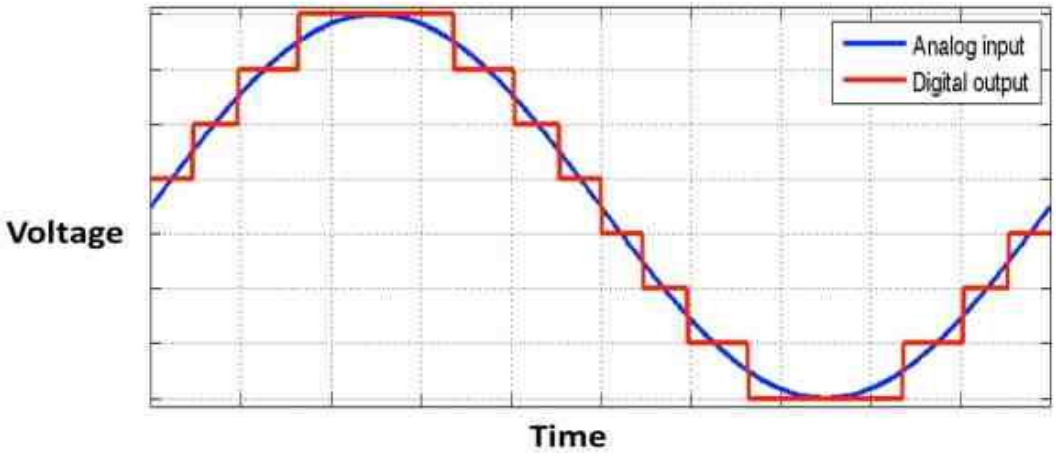
11.1.1 A/D转换过程

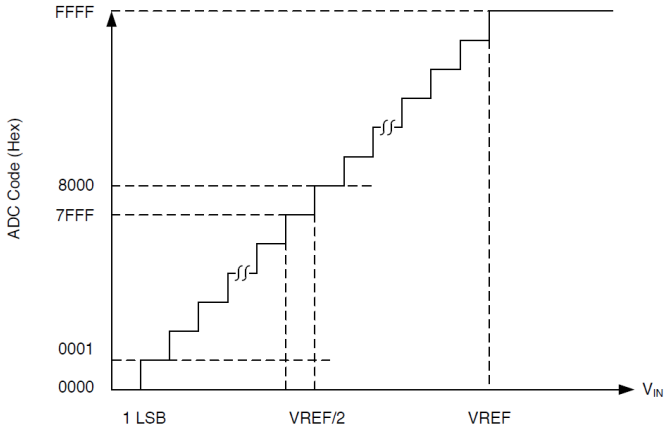
- **模数转换（ADC）是将模拟信号转换为数字信号的过程，主要包括采样、保持、量化和编码四个步骤。**



11.1.1 A/D转换过程

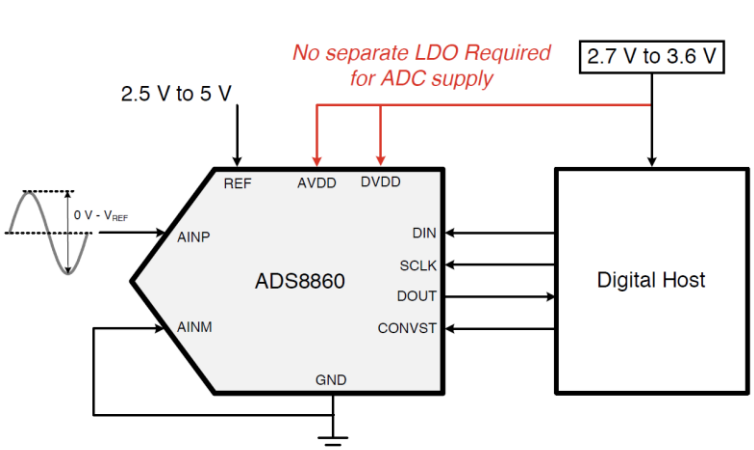
- 通过连续周期性A/D转换，可获得动态变化信号的波形。

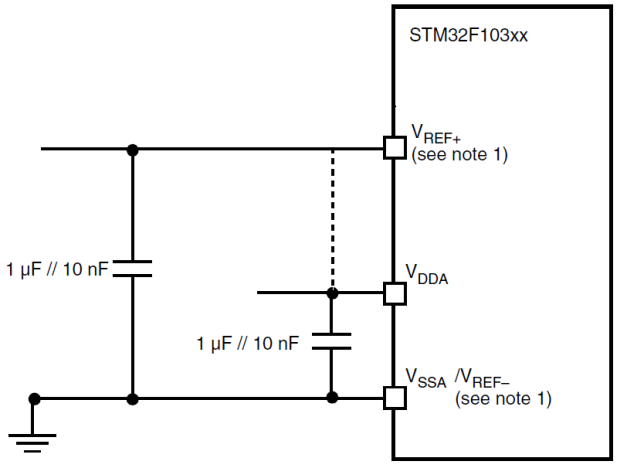




11.1.1 A/D转换过程

- **参考电压：进行模拟到数字转换时，ADC所能测量的最大模拟输入电压。通常可由外部电压源为ADC芯片提供，以使芯片可灵活的用于不同量程。**





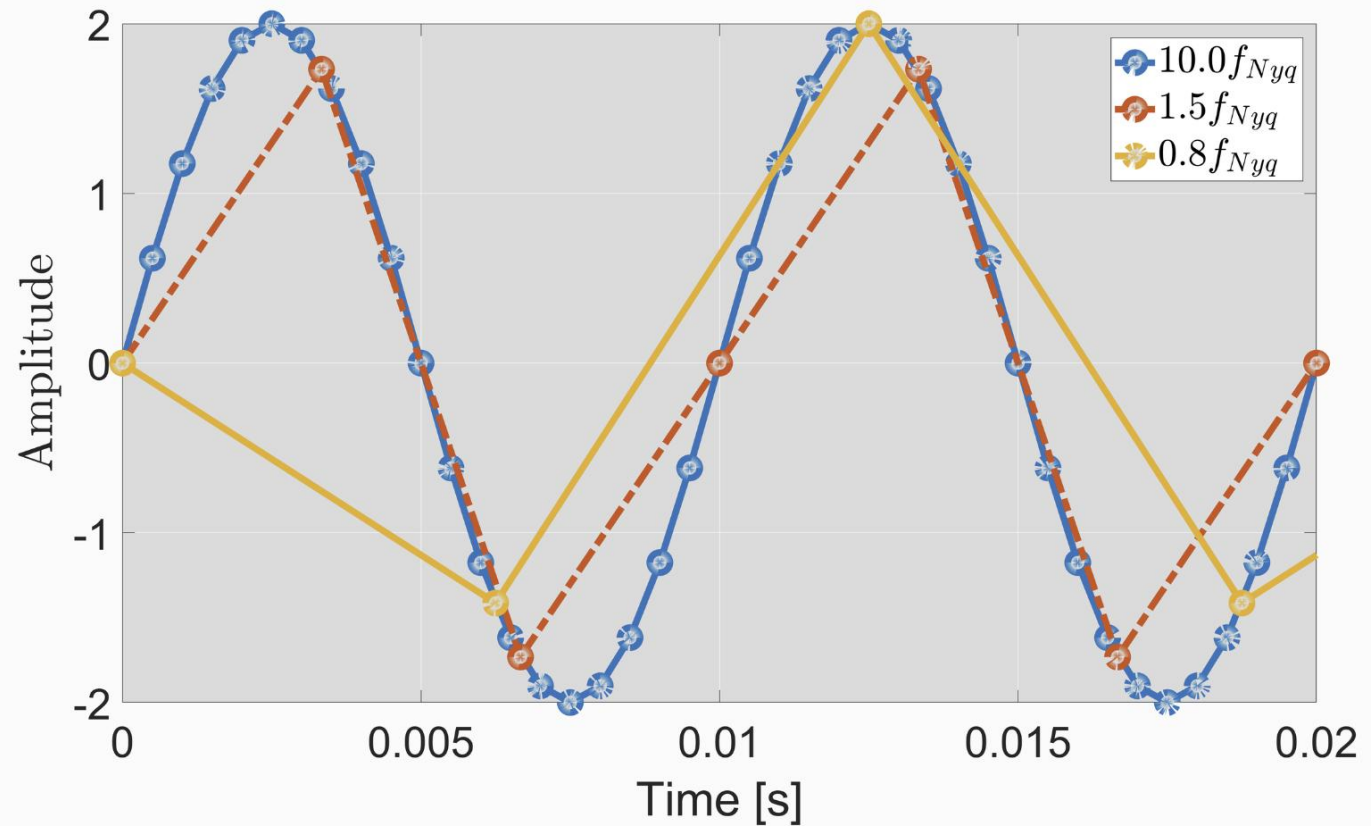
决定输入电压范围：ADC的输入电压根据输入模式在0到参考电压（Vref）之间，或±Vref之间.

影响数字输出值：参考电压决定了ADC输出的数字值与输入模拟信号之间的映射关系。

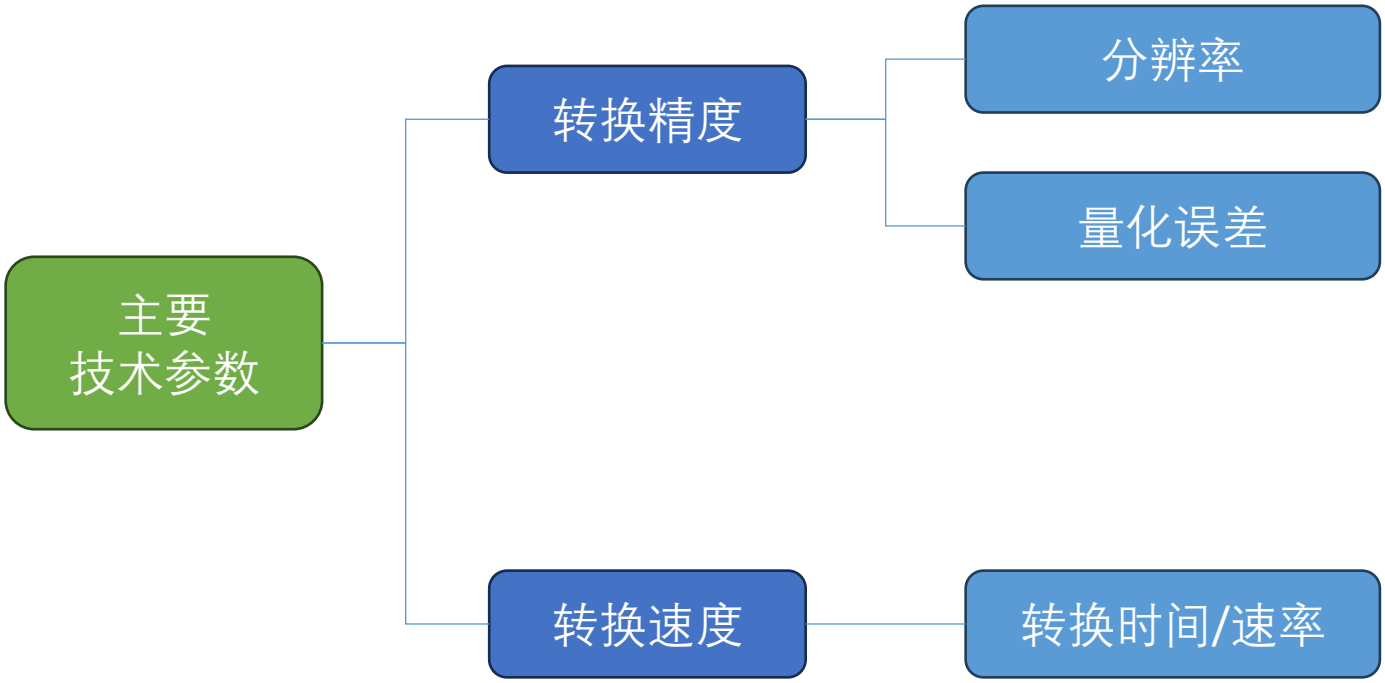
影响转换精度：参考电压的稳定性对ADC的转换精度至关重要。

11.1.1 A/D转换过程

- **Nyquist–Shannon(奈奎斯特-香农)采样定理**：为了不失真地恢复模拟信号，采样频率应该不小于模拟信号频谱中**最高频率的2倍**。



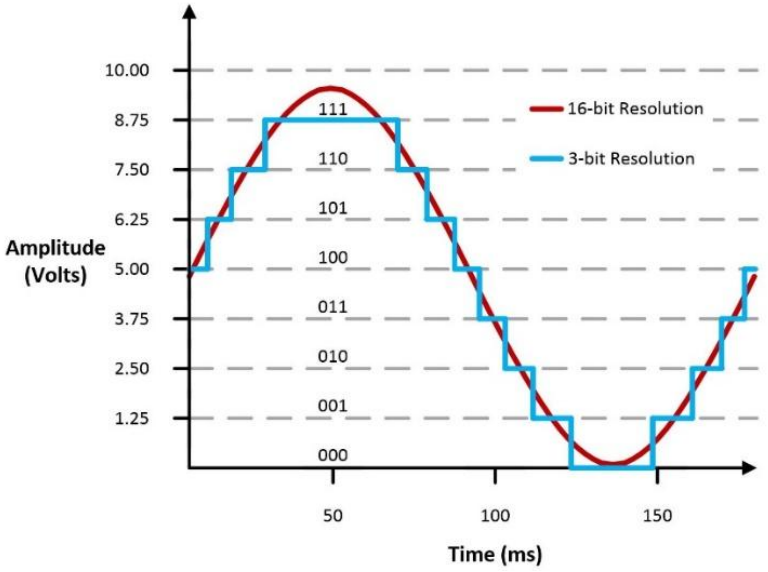
11.1.2 A/D转换的主要技术参数



11.1.2 A/D转换的主要技术参数

- 分辨率为A/D转换器对输入模拟量微小变化的分辨能力，通常用二进制数的有效位表示。

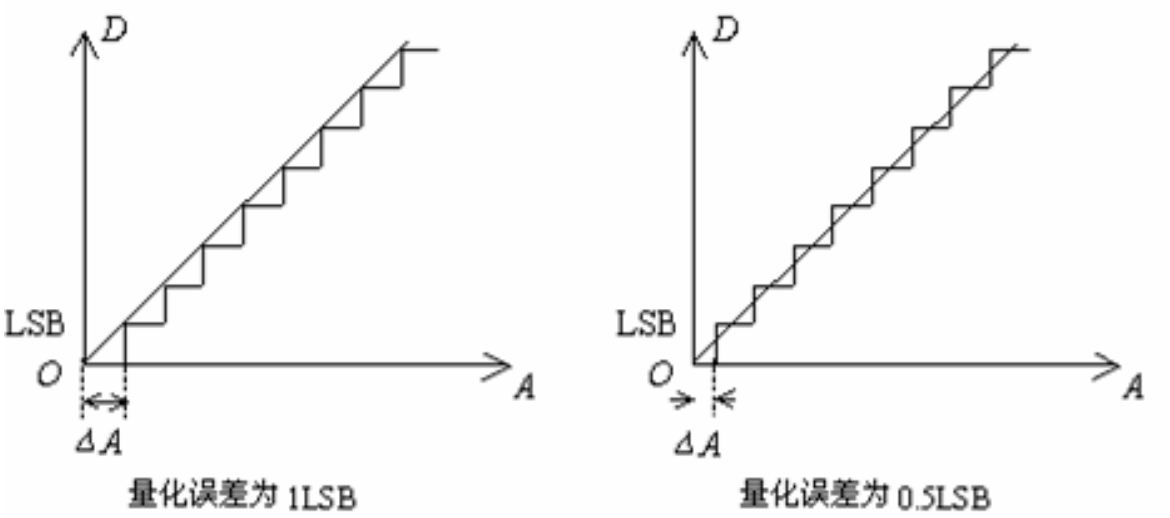
位数	级数	分辨率（参考电压10V）
8	$2^8 = 256$	39.1 mV
10	$2^{10} = 1024$	9.77 mV
12	$2^{12} = 4096$	2.44 mV
14	$2^{14} = 16384$	0.61 mV
16	$2^{16} = 65536$	0.15 mV



11.1.2 A/D转换的主要技术参数

- 量化误差为有限分辨率而引起的误差，反映A/D转换器实际输出数字量和理想输出之间的差异。

通常为1个或半个最小数字量的模拟变化量，表示为1LSB、 1/2LSB （Least Significant Bit）。



两种转换特性

11.1.2 A/D转换的主要技术参数

- **转换时间是指从转换控制信号到ADC输出端得到稳定的数字量所需要的时间，转换时间的倒数即为转换速率。**

> 转换时间 = 采样保持时间 + 量化编码时间

通常说的**采样频率**为实际A/D转换中采用的数据采集频率，应小于或等于ADC芯片的最高转换速率。

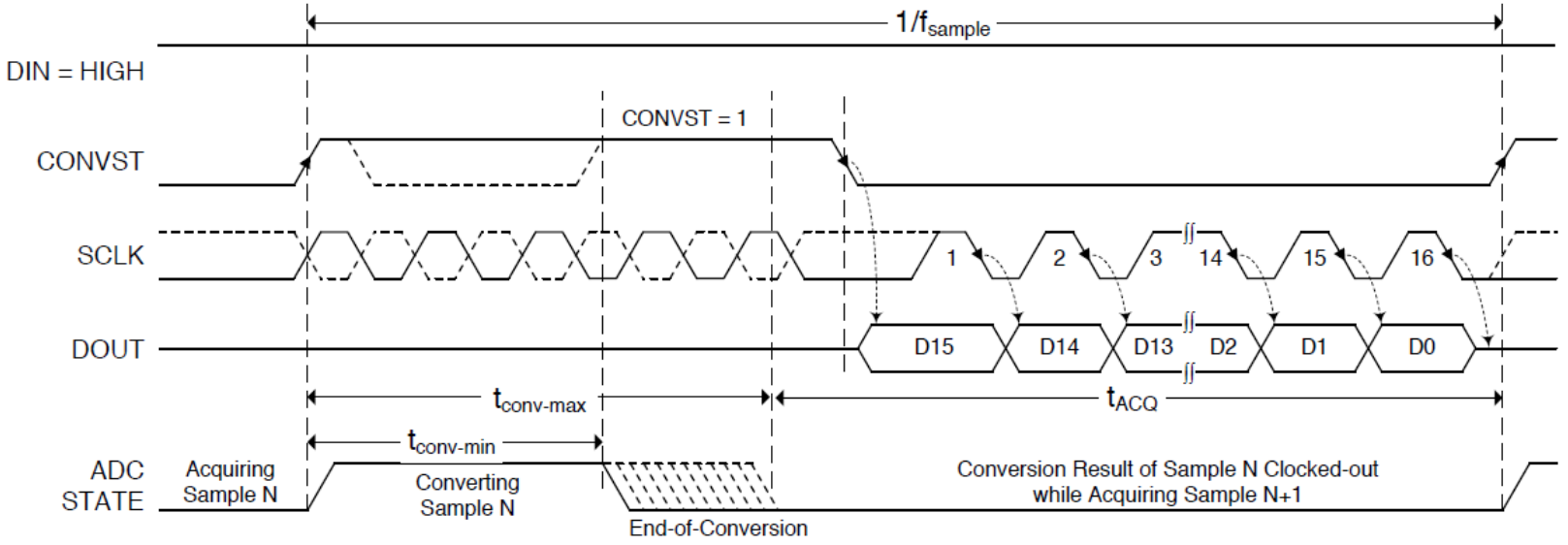


Figure 49. Interface Timing Diagram: 3-Wire $\overline{CS}$ Mode Without a Busy Indicator (DIN = 1)

11.1.3 A/D转换分类及原理

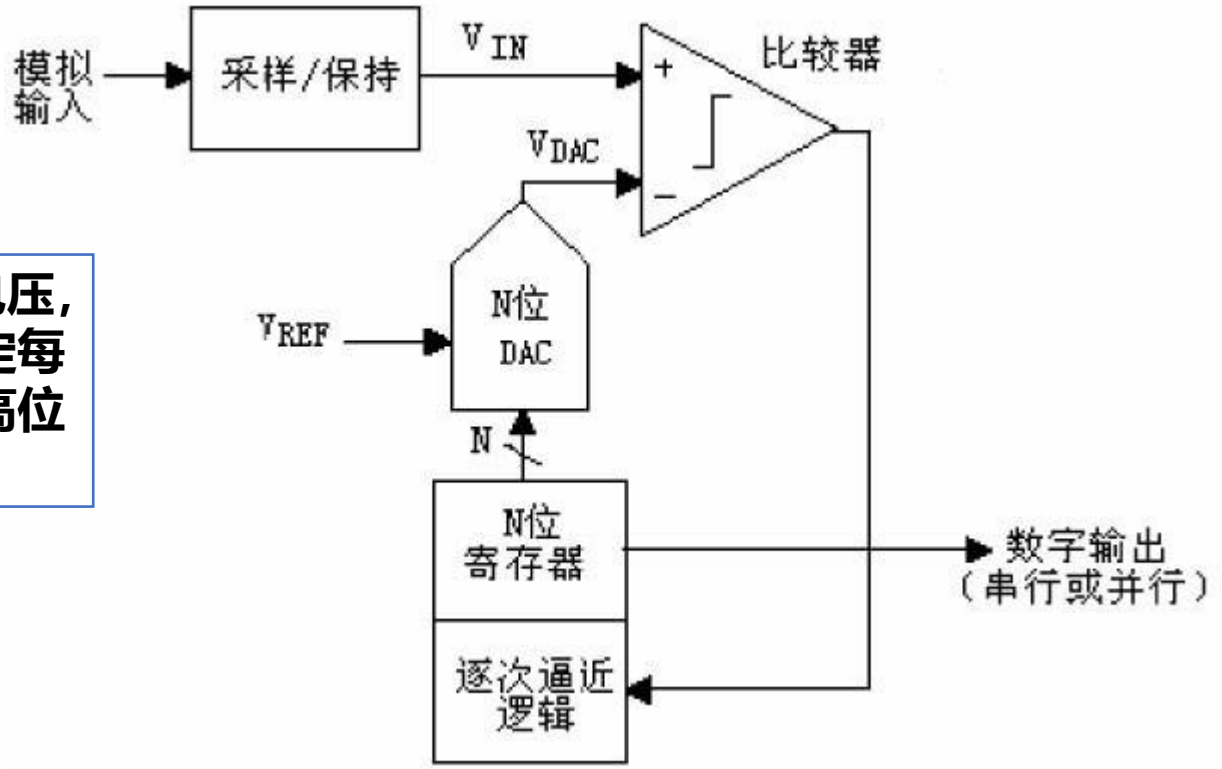
常用的4种ADC

架构类型	特点
逐次逼近型 SAR ADC	中速、中高精度、低功耗
Δ-Σ 型 Delta-Sigma ADC	低速、**超高精度**、低功耗
流水线型 Pipelined ADC	**高速**、中高精度、中高功耗
闪速型 Flash ADC	**超高速**、低精度、高功耗

11.1.3 A/D转换分类及原理

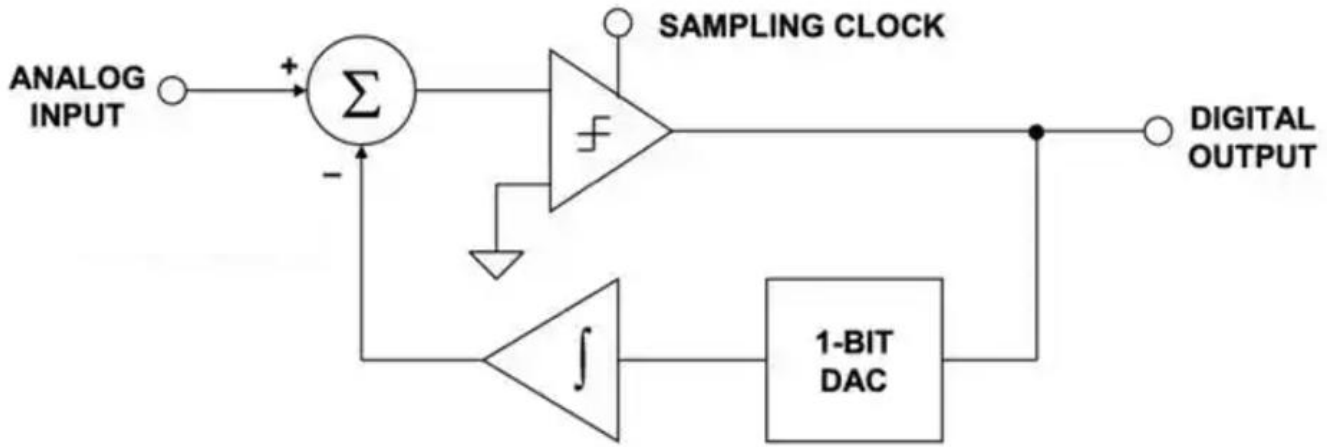
- SAR ADC通过 “逐位比较” 逼近模拟信号。

内部 DAC 生成参考电压，与输入信号比较，确定每一位二进制值（从最高位到最低位）



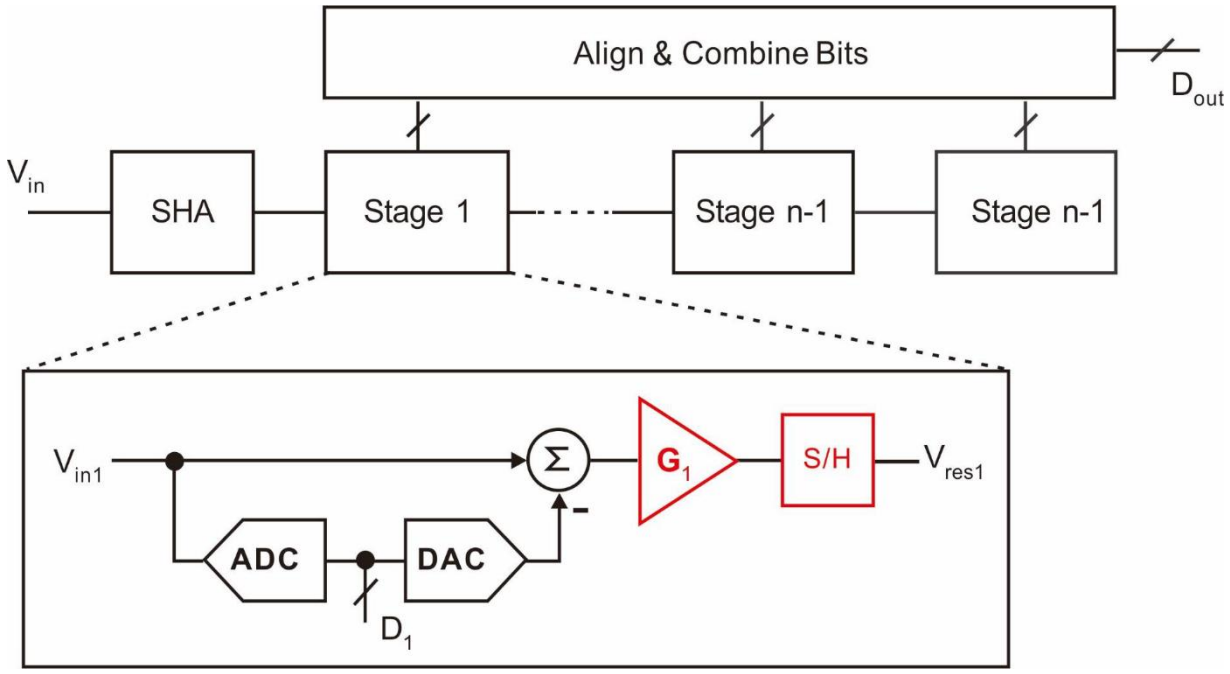
11.1.3 A/D转换分类及原理

- Δ-ΣADC由ΔΣ调制器和数字/抽取滤波器组成，通过过采样和噪声整形技术实现高分辨率转换。



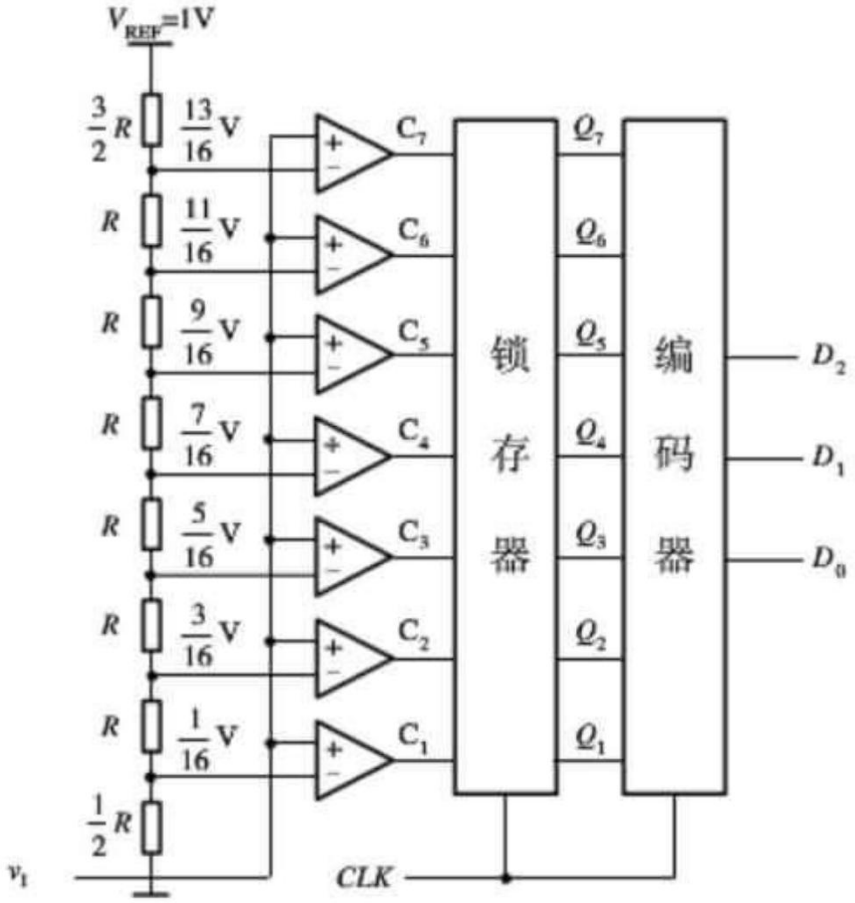
11.1.3 A/D转换分类及原理

- 流水线ADC采用分多级量化，每级完成低精度量化（如2~3 位），通过余数放大器放大误差，传递给下一级，最终拼接结果。



11.1.3 A/D转换分类及原理

- **FLASH ADC是一种基于比较器和多个参考电压的模拟信号转换器。**



使用线性电压阶梯和每个阶梯上的比较器来将输入电压与连续的参考电压进行比较，从而实现ADC转换。

11.1.4 A/D转换输入模式

常用的4种ADC

输入模式	核心定义	抗干扰能力	精度
单端输入 (Single-Ended)	信号一端接 ADC 输入信号，另一端接固定参考地	中等	中低
差分输入 (Differential)	信号两端分别接 ADC 的两个互补通道，测量两者差值	强	高
伪差分输入 (Pseudo-Differential)	信号一端接输入通道，另一端接动态参考电压（非 GND，如传感器共模电压)	中强	中高
全差分输入 (Fully Differential)	差分输入+差分参考电压（参考电压V_{REF-},V_{REF+}）	极强	超高

11.2 STM32的ADC

集成有三个12位逐次逼近型的A/D转换器

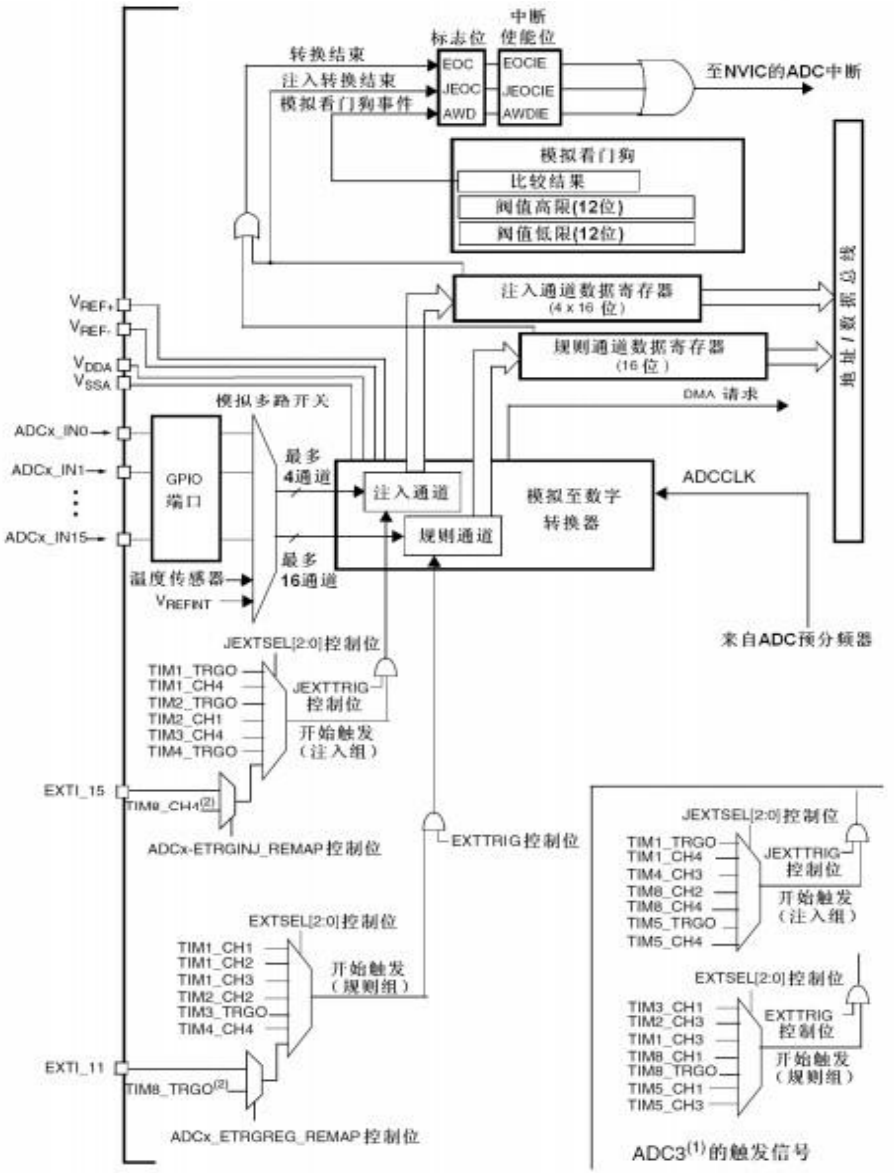
多达18个通道，可实现16个外部模拟输入通道和2个内部信号源的A/D转换

转换结果是一个12位的二进制数，可以以左对齐或右对齐两种方式存储在16位数据寄存器中

各通道的A/D转换可采用单次、连续、扫描或间断模式执行

11.2 STM32的ADC

- ADC
  - ADC引脚
  - 模拟多路开关
  - ADC时钟
  - 注入或规则通道
  - 中断



11.2 STM32的ADC

信号输入	触发信号	A/D转换	数据保存
输入信号经ADC的输入通道ADCx_IN0~ADCx_IN15送入模拟至数字转换器功能模块。	软件触发或EXTI外部触发或定时器触发。将硬件触发源分为两类：规则通道和注入通道。	在ADC时钟ADCCLK的驱动下，对输入通道的信号进行A/D转换，即完成采样—量化—编码等工作。	A/D转换结束后，将转换结果的12位数据以左对齐或右对齐的方式保存到16位的寄存器中。

11.2 STM32的ADC

11.2.1 ADC的引脚

11.2.2 ADC通道选择

11.2.3 ADC中断和DMA请求

11.2.4 ADC转换时间

11.2.5 ADC数据对齐

11.2.6 ADC的转换模式

11.2.7 ADC校准

11.2.1 ADC的引脚

ADCx_IN0~ADCx_IN15为ADC的输入信号通道，即为ADC的输入GPIO引脚，ADC的GPIO引脚均为复用功能，使用时需配置为模拟输入模式。

ADC1还有两个内部通道：温度传感器通道和V_{REFINT}。

通道	ADC1（复用引脚）	ADC2（复用引脚）	ADC3（复用引脚）
通道0(ADC_Channel_0)	PA0	PA0	PA0
通道1(ADC_Channel_1)	PA1	PA1	PA1
通道2(ADC_Channel_2)	PA2	PA2	PA2
通道3(ADC_Channel_3)	PA3	PA3	PA3
通道4(ADC_Channel_4)	PA4	PA4	PF6
通道5(ADC_Channel_5)	PA5	PA5	PF7
通道6(ADC_Channel_6)	PA6	PA6	PF8
通道7(ADC_Channel_7)	PA7	PA7	PF9
通道8(ADC_Channel_8)	PB0	PB0	PF10
通道9(ADC_Channel_9)	PB1	PB1	
通道10(ADC_Channel_10)	PC0	PC0	PC0
通道11(ADC_Channel_11)	PC1	PC1	PC1
通道12(ADC_Channel_12)	PC2	PC2	PC2
通道13(ADC_Channel_13)	PC3	PC3	PC3
通道14(ADC_Channel_14)	PC4	PC4	
通道15(ADC_Channel_15)	PC5	PC5	
通道16(ADC_Channel_16)	温度传感器		
通道17(ADC_Channel_17)	内部参考电压		

11.2.2 ADC通道选择

- **按组进行转换的模式，可以由程序设置实现对多个模拟通道自动地进行逐个采样转换，STM32分为两种组转换模式：规则组和注入组。**

规则通道组最多允许16个规则通道进行转换

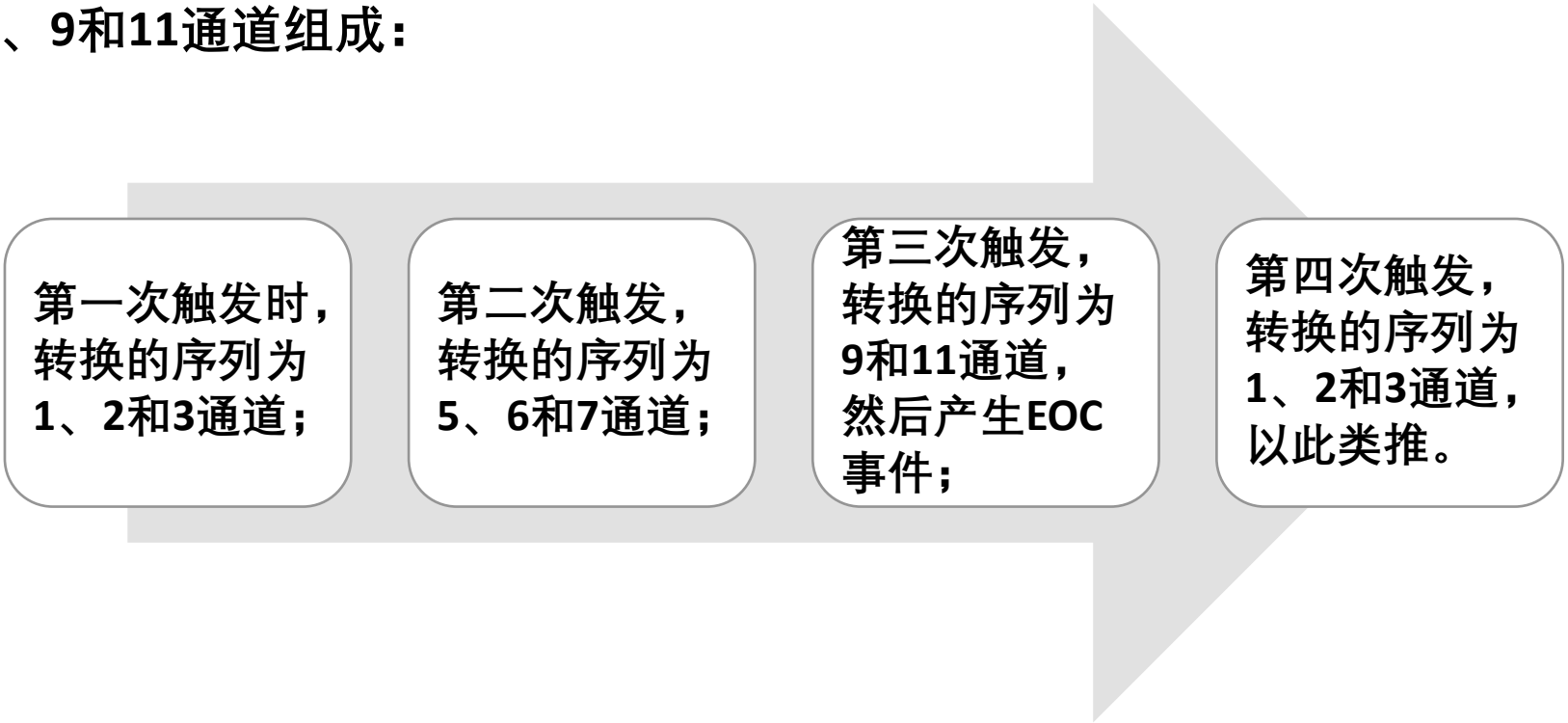
规则组

可通过编程来设定规则通道的数量n，n最多可设定为16。

转换结果保存在同一个16位的规则通道数据寄存器ADC_DR中，同时，EOC标志被置位，产生相应的中断或DMA请求。

11.2.2 ADC通道选择

例如：设定n=3，规则通道组由1、2、3、5、6、7、9和11通道组成：



11.2.2 ADC通道选择

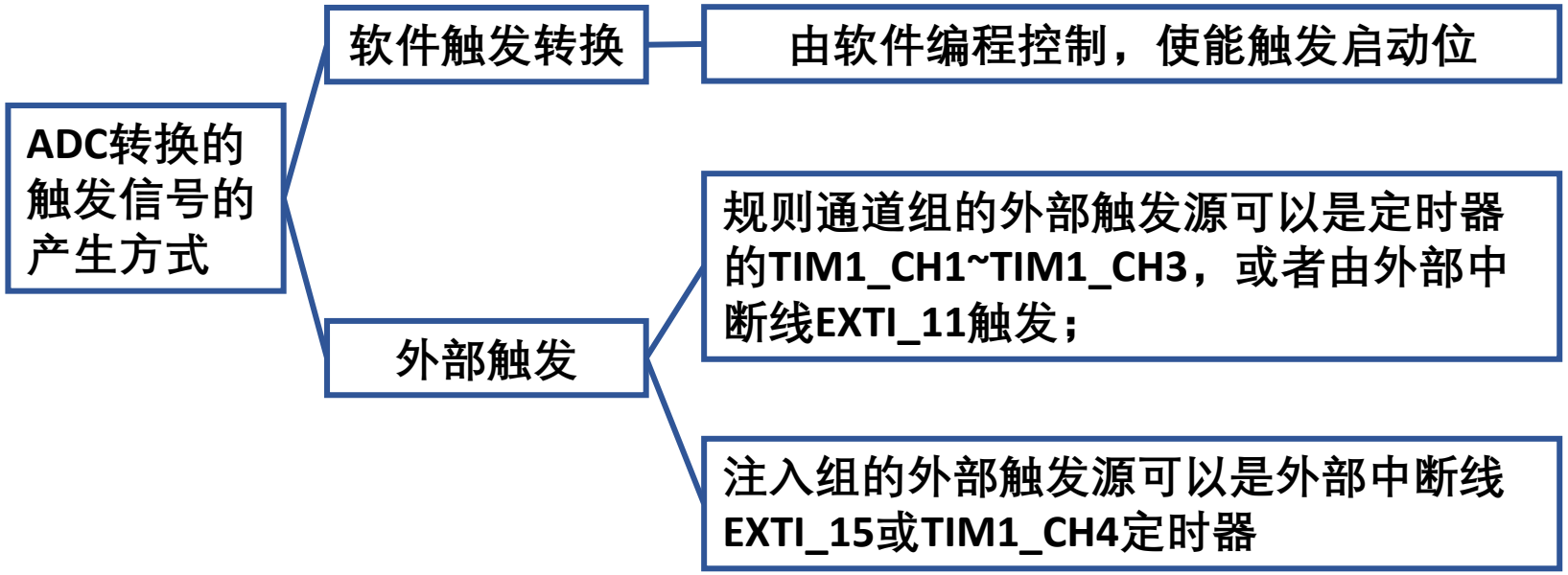
最多允许4个通道进行转换，并且对应有4个注入通道寄存器用来存放注入通道的转换结果，因此注入通道组没有DMA请求。

注入通道组的触发转换序列和规则通道一样

转换结果保存数据寄存器ADC_JDRx中，同时产生ADC注入转换结束事件，即JEOC标志被置位，产生相应的中断。

规则通道组转换好比是程序的正常执行，注入通道组的转换则好比是一个中断处理程序

11.2.3 ADC中断和DMA请求



11.2.3 ADC中断和DMA请求

ADC在每个通道转换完成后，可产生相应的中断请求。

对于规则通道，如果ADC_CR1寄存器的EOCIE位被置“1”，则会产生EOC中断；

对于注入通道，如果ADC_CR1寄存器的JEOCIE位被置“1”，则会产生JEOC中断；

ADC1和ADC3的规则通道转换完成后还可产生DMA请求。

11.2.3 ADC中断和DMA请求

ADC中断事件主要有三个，ADC_IT_EOC中断针对规则通道，ADC_IT_JEOC中断针对注入通道。

中断事件	事件标志	使能控制位
规则组转换结束中断 ADC_IT_EOC	EOC中断 （End of Coversion）	EOCIE
注入组转换结束中断 ADC_IT_JEOC	JEOC中断 （End of injected Coversion）	JEOCIE
模拟看门狗中断 ADC_IT_AWD	AWD中断 （Analog WatchDOG）	AWDIE

11.2.3 ADC中断和DMA请求

DMA请求

适用情况

只有ADC1和ADC3能产生DMA请求，ADC2转换数据可以在双ADC模式中使用ADC1的DMA请求

使用原因

规则通道的转换只有一个数据寄存器ADC_DR，每个通道转换完成后，将覆盖以前的数据，所以，可以使用DMA方式处理数据及时地将已完成转换的数据读出。

应用

在每次产生转换结束事件EOC标志后，DMA控制器会把保存在ADC_DR寄存器中的规则组通道的转换数据传送到SRAM中（用户指定的目标地址）

11.2.4 ADC转换时间

STM32的ADC的采样时间可选择为采样周期的1.5、7.5、13.5、28.5、41.5、55.5、71.5或239.5倍。

ADC总的转换时间可以表示为：

ADC总的转换时间=采样时间+12.5个周期

举例

若采样时间为1.5个周期，PCLK2=72MHz，且ADCCLK为APB2的6分频，即12MHz，则

总的转换时间=（1.5+12.5）/12MHz=1.17us

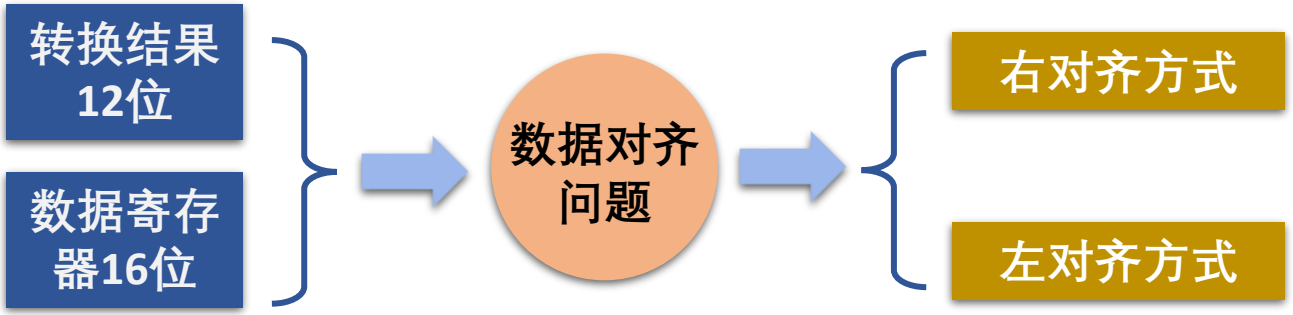
如果ADCCLK=14MHz，采样时间为1.5个周期，则

总的转换时间=（1.5+12.5）/14MHz=1us

注意

ADC的时钟ADCCLK最大不能超过14MHz，因此STM32F103系列微控制器的ADC最短转换时间为1us。

11.2.5 ADC数据对齐



一般建议采用右对齐，因为数据传输一般是从最低位开始（也即右边开始），如果数据位是从高位开始传输的，可以采用左对齐方式。

11.2.6 ADC的转换模式

ADC只执行一次转换

当前ADC转换一结束就会立即启动下一个转换

扫描一组模拟通道：

- 单次模式：扫描完本组所有通道后，ADC自动停止；
- 连续模式：扫描完本组所有通道后，再从第一个通道继续扫描。

用于多个通道的规则组和注入组

11.2.6 ADC的转换模式

间断模式

以规则组为例，若被转换的通道为0、1、2、3、6、7、9、10，n=3，则

第一次触发：进行转换的通道序列为0、1、2；

第二次触发：进行转换的通道序列为3、6、7；

第三次触发：进行转换的通道序列为9、10，并产生EOC事件；

第四次触发：进行转换的通道序列0、1、2。

在间断模式下，转换也有其特殊的规则：

- 当以间断模式转换一个规则组时，转换序列结束后不自动从头开始；
- 当所有子组被转换完成，下一次触发启动第一个子组的转换；
- 规则组和注入组不能同时设置为间断模式。

11.2.7 ADC校准

校准目的

A/D转换过程需要一定的转换时间，对ADC进行校准可以减少因内部电容的变化而导致的误差。

校准实现

STM32的ADC具有内置的自校准模式，ADC在上电后至少2个ADC时钟周期才能开始校准

校准要求

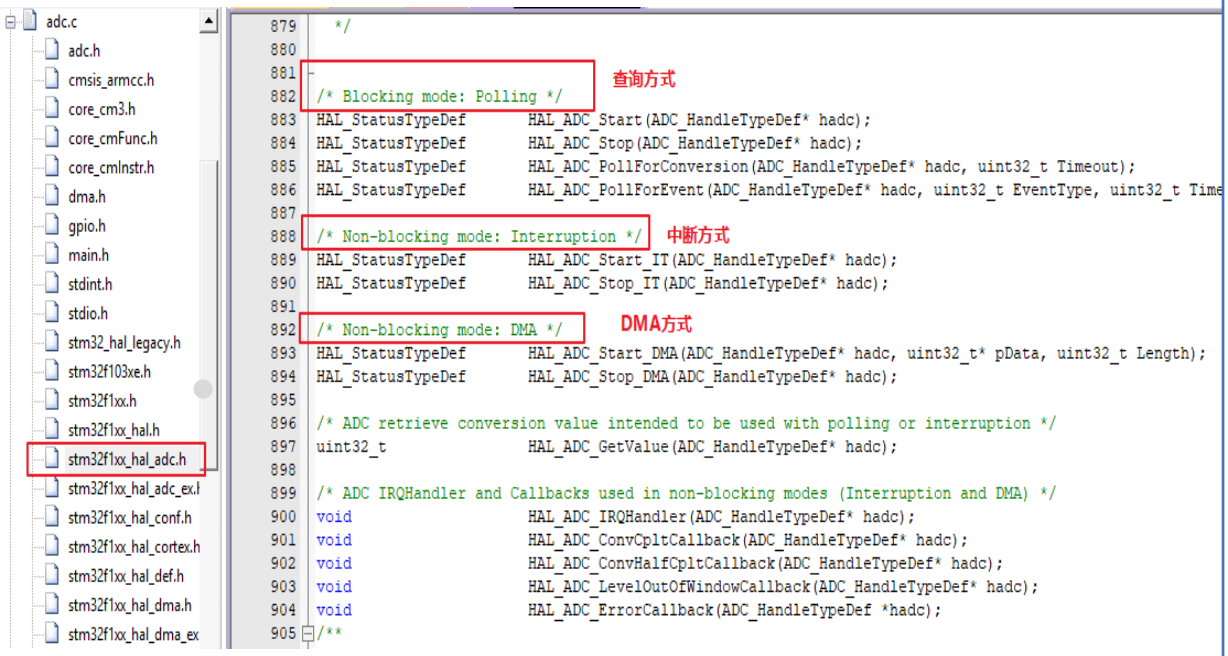
建议每次上电后都执行一次ADC校准，以保证采集数据的准确性。

11.3 ADC模块的HAL库接口函数及应用

11.3.1 ADC的HAL库接口函数

11.3.2 ADC的HAL库应用实例

11.3.1 ADC的HAL库接口函数



- 定义在**stm32f1xx_hal_adc.c**源文件
- 在**stm32f1xx_hal_adc.h**头文件中可以查看HAL库中DMA库函数的声明以及相关的结构体定义和串口一样
- 可以通过轮询、中断和DMA三种方式进行操作。

11.3.1 ADC的HAL库接口函数

ADC的HAL库常用接口函数

11.3.1 ADC的HAL库接口函数

ADC的HAL库常用接口函数

类型		函数及功能描述
引脚功能操作函数	轮询方式	HAL_ADC_Start(ADC_HandleTypeDef* hadc); 功能描述：启动ADC转换
		HAL_ADC_Stop(ADC_HandleTypeDef* hadc); 功能描述：停止ADC转换
		HAL_ADC_PollForConversion(ADC_HandleTypeDef* hadc, uint32_t Timeout); 功能描述：等待转换结束，采用超时管理机制
	中断方式	HAL_ADC_Start_IT(ADC_HandleTypeDef* hadc); 功能描述：中断方式下启动ADC 转换
		HAL_ADC_Stop_IT(ADC_HandleTypeDef* hadc); 功能描述：中断方式下停止ADC 转换
	DMA方式	HAL_ADC_Start_DMA(ADC_HandleTypeDef* hadc, uint32_t* pData, uint32_t Length); 功能描述：DMA方式下启动ADC转换
		HAL_ADC_Stop_DMA(ADC_HandleTypeDef* hadc); 功能描述：DMA方式下停止ADC转换
		uint32_t HAL_ADC_GetValue(ADC_HandleTypeDef* hadc); 功能描述：轮询或中断方式下获取ADC转换值

11.3.1 ADC的HAL库接口函数

ADC的HAL库常用接口函数

类型	函数及功能描述
中断处理函数（回调函数）	void HAL_ADC_IRQHandler(ADC_HandleTypeDef* hadc); 功能描述：ADC中断处理函数
	void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc); 功能描述：ADC中断回调函数，用户在该函数内编写实际的中断处理程序
	void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc); 功能描述：ADC转换一半时调用的中断回调函数
外设配置函数	HAL_ADC_ConfigChannel(ADC_HandleTypeDef* hadc, ADC_ChannelConfTypeDef* sConfig); 功能描述：ADC配置函数
外设状态函数	uint32_t HAL_ADC_GetState(ADC_HandleTypeDef* hadc); 功能描述：获取ADC转换状态函数
	uint32_t HAL_ADC_GetError(ADC_HandleTypeDef *hadc); 功能描述：获取ADC错误函数

11.3.1 ADC的HAL库接口函数

1．启动ADC转换函数HAL_ADC_Start()

函数原型

```
HAL_StatusTypeDef  HAL_ADC_Start(ADC_HandleTypeDef* hadc);
```

使用范例

```
 if(HAL_ADC_Start(&hadc1) != HAL_OK)
{   Error_Handler();
 }
   /*等待转换结束*/
HAL_ADC_PollForConversion(&hadc1, 10);
if(HAL_ADC_GetState(&hadc1) == HAL_ADC_STATE_EOC_REG)
{
/*获取转换结果*/
ADCx_ConvertedValue = HAL_ADC_GetValue(&hadc1);
}
```

11.3.1 ADC的HAL库接口函数

2．启动ADC中断转换函数HAL_ADC_Start_IT()

函数原型

```
HAL_StatusTypeDef  HAL_ADC_Start_IT(ADC_HandleTypeDef* hadc);
```

使用范例

```
 if(HAL_ADC_StartIT(&hadc1) != HAL_OK)
{
    Error_Handler();
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    /* 获取ADC转换结果 */
   ADCx_ConvertedValue = HAL_ADC_GetValue(&hadc1);
}
```

11.3.1 ADC的HAL库接口函数

3．启动ADC的DMA转换函数 HAL_ADC_Start_DMA()

函数原型

```
HAL_StatusTypeDef HAL_ADC_Start_DMA(ADC_HandleTypeDef* hadc,
uint32_t* pData, uint32_t Length);
```

使用范例

```
if(HAL_ADC_Start_DMA(&hadc1,&ADCx_ConvertedValue,1) != HAL_OK)
{ Error_Handler();
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
/* DMA传输结束，指示灯LED亮 */
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_5, GPIO_PIN_RESET);
}
```

11.3.2 ADC的HAL库应用实例

功能描述

内部有一个温度传感器，测量范围为：-40℃~115℃，测量精度范围为±1.5℃，用来测量芯片内部的温度，连接在ADC1_IN16的输入通道上。

本实例将采集到的温度数据通过串口发送到PC机。

Reading the temperature

To use the sensor:

1. Select the ADCx_IN16 input channel.
2. Select a sample time of 17.1 µs
3. Set the TSVREFE bit in the *ADC control register 2 (ADC_CR2)* to wake up the temperature sensor from power down mode.
4. Start the ADC conversion by setting the ADON bit (or by external trigger).
5. Read the resulting V_{SENSE} data in the ADC data register
6. Obtain the temperature using the following formula:

   Temperature (in °C) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25.

   Where,

   V_{25} = V_{SENSE} value for 25° C and

   Avg_Slope = Average Slope for curve between Temperature vs. V_{SENSE} (given in mV/° C or µV/ °C).

   Refer to the Electrical characteristics section for the actual values of V_{25} and Avg_Slope.

STM32内部温度传感器说明

11.3.2 ADC的HAL库应用实例

5.3.21 Temperature sensor characteristics

Table 64. TS characteristics

Symbol	Parameter	Min	Typ	Max	Unit
T_L	V_{SENSE} linearity with temperature	-	±1	±2	°C
Avg_Slope	Average slope	4.0	4.3	4.6	mV/°C
V_{25}	Voltage at 25 °C	1.34	1.43	1.52	V
$t_{START}^{(1)}$	Startup time	4	-	10	µs
$T_{S\_temp}^{(2)(1)}$	ADC sampling time when reading the temperature	-	-	17.1	µs

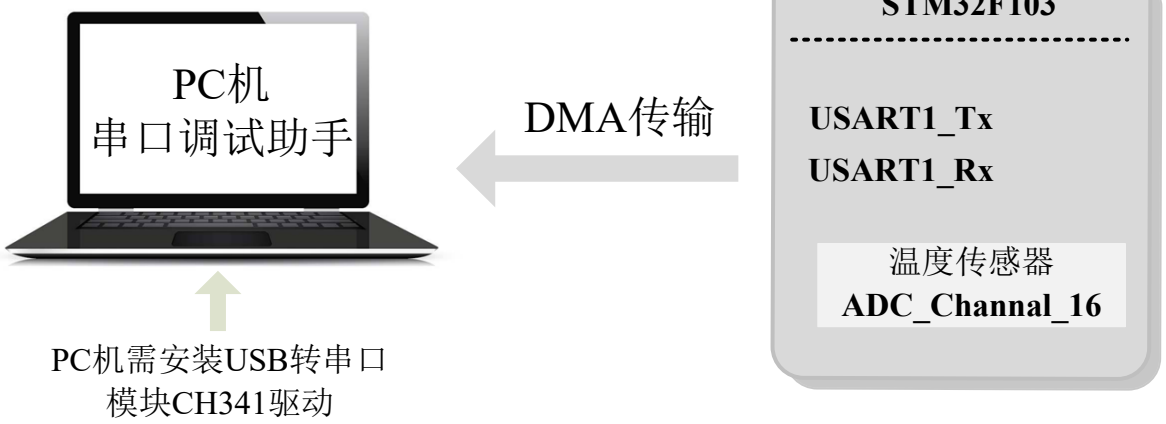
1. Guaranteed by design.
2. Shortest sampling time can be determined in the application by multiple iterations.

V25为温度传感器在25℃时的输出电压，典型值为1.43V；

Avg_Slope为温度传感器输出电压与温度曲线的平均斜率的关联参数，典型值为4.3mV/℃

11.3.2 ADC的HAL库应用实例

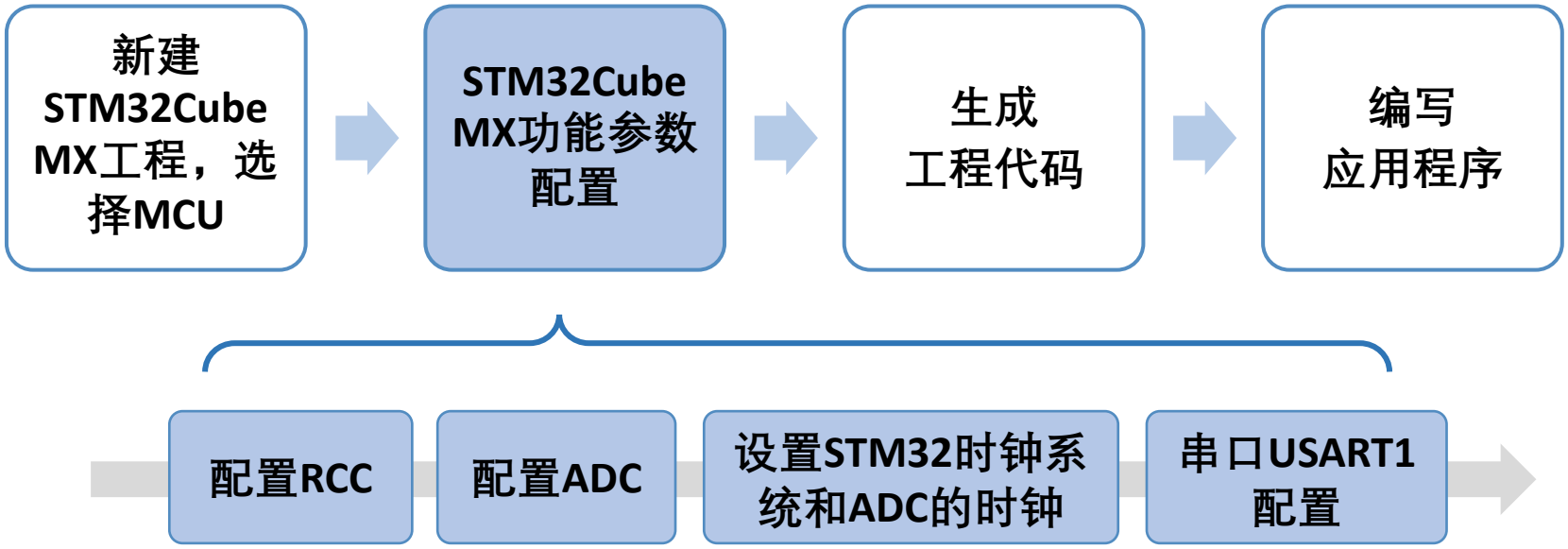
硬件设计



- 通过连接在ADC1的通道16上的STM32F103内部温度传感器获取芯片内部的温度；
- 利用DMA传输方式通过串口将采集到的温度数据显示出来。

11.3.2 ADC的HAL库应用实例

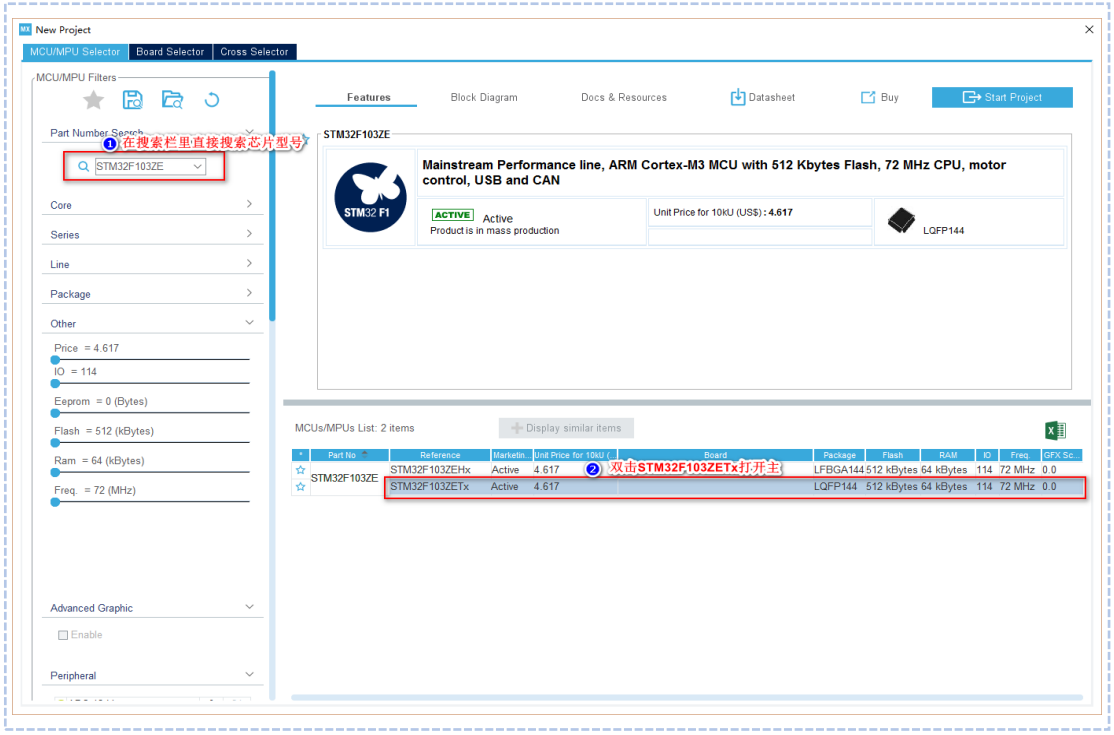
软件设计



11.3.2 ADC的HAL库应用实例

软件设计——新建STM32CubeMX工程，选择MCU

新建STM32CubeMx工程，选择MCU，这里选择STM32F103ZETx芯片，读者可根据自己的目标板选择相应的芯片

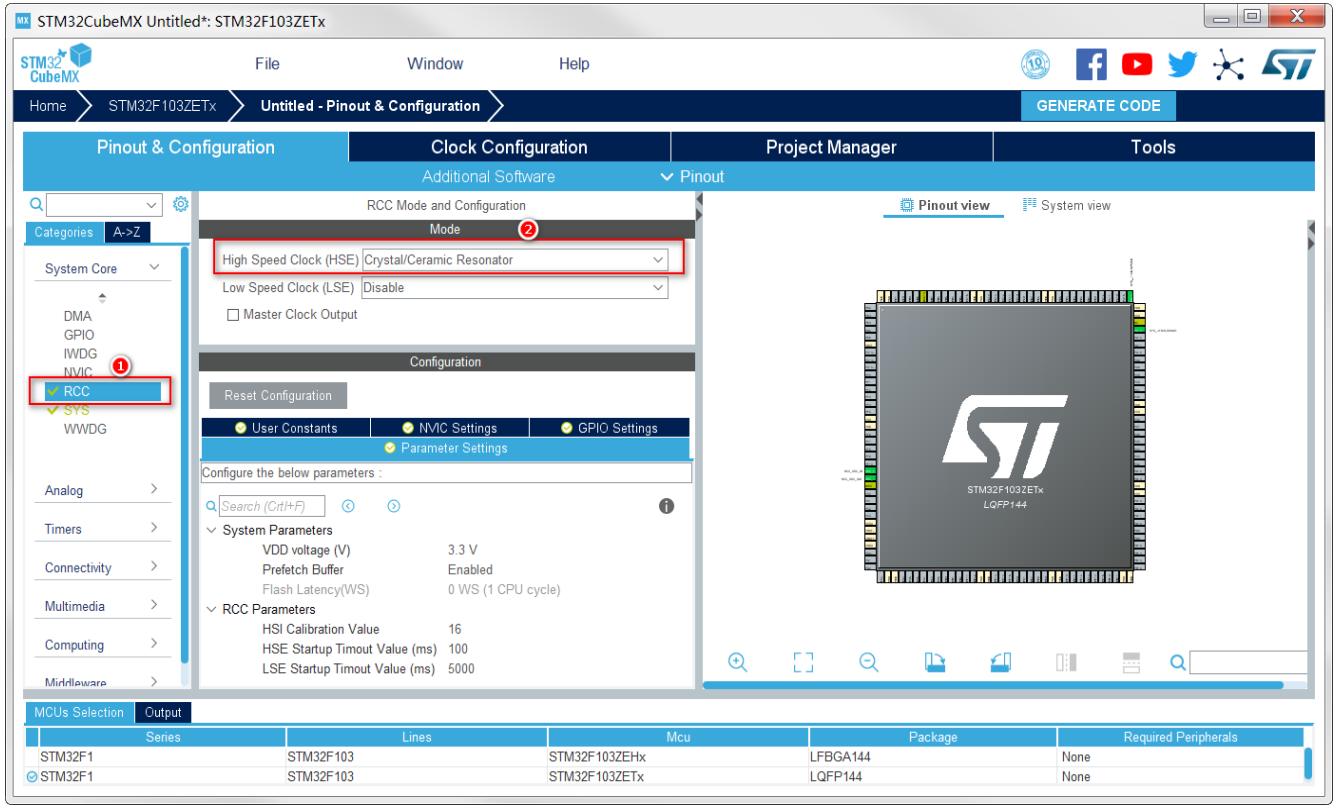


11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

RCC配置

HSE选择“Crystal/Ceramic Resonator”（晶振/陶瓷谐振器），LSE选择“Disable”

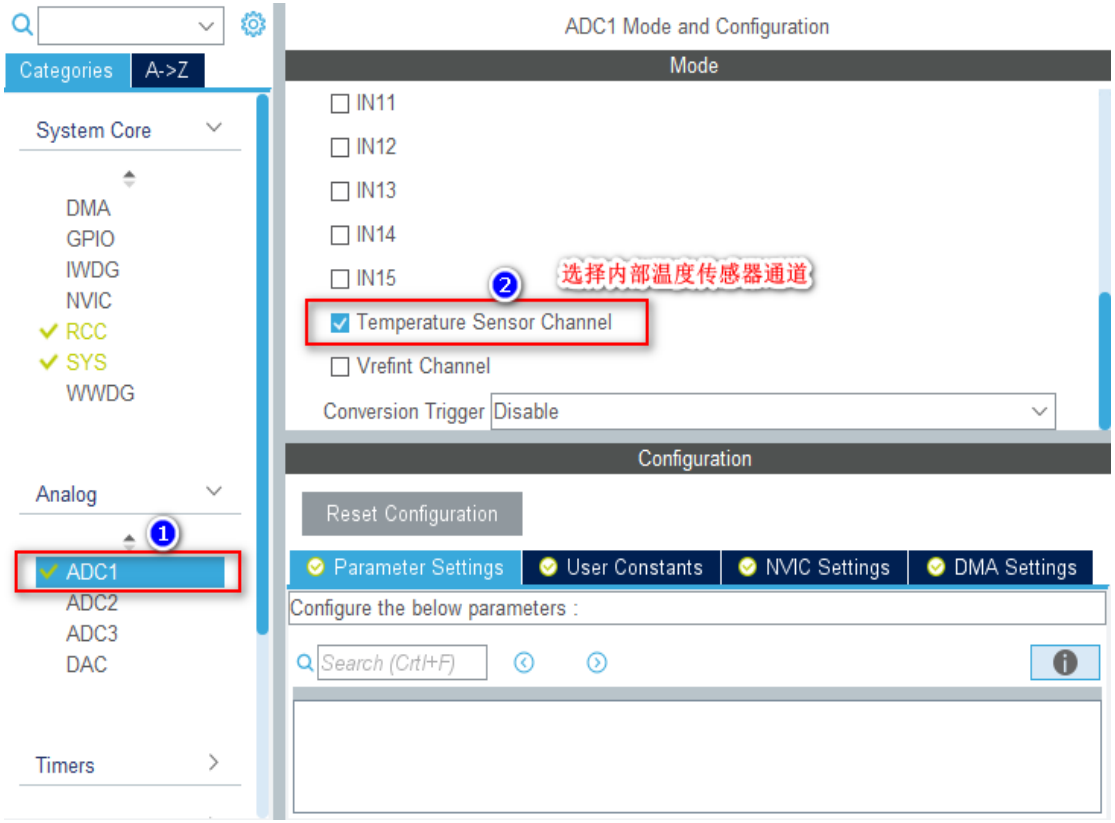


11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

本实例采用ADC1的“Temperature Sensor Channel”通道（即连接在ADC1_IN16）

ADC配置——设置ADC1的通道

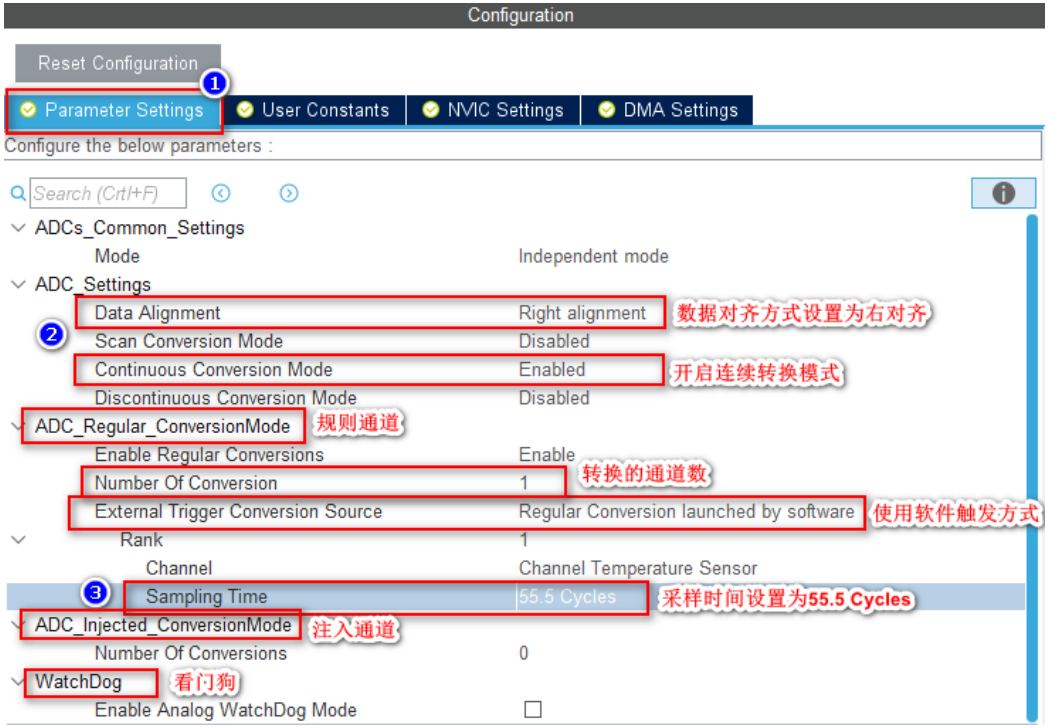


11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

ADC配置——配置ADC1的参数

在ADC1参数配置“Configuration”选项配置页的“Parameter Settings”中对ADC1的相关参数进行设置，本例配置ADC1为独立模式，设置数据对齐方式为右对齐，禁止扫描转换模式，使能连续转换模式，在“Rank”项中设置采样周期为55.5Cycles

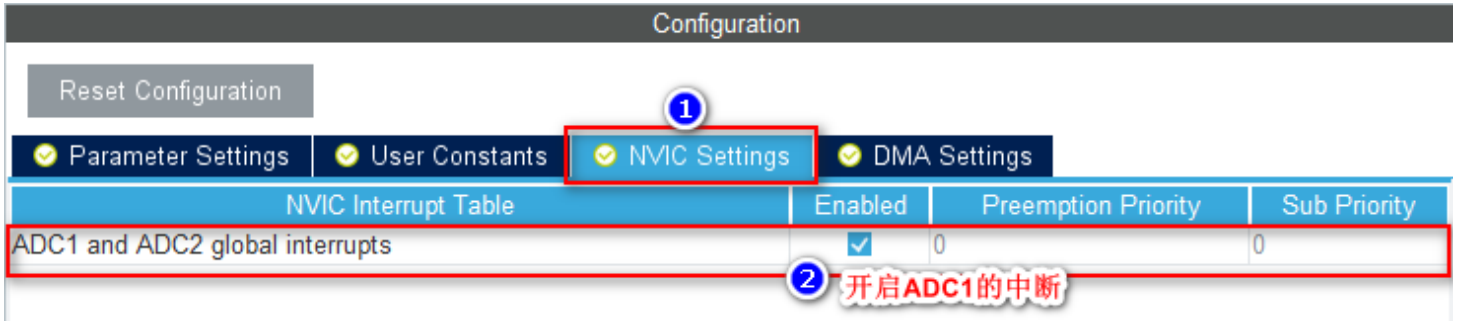


11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

ADC配置——开启ADC1的中断

在ADC1参数配置“Configuration”选项配置页的“NVIC Settings”中开启ADC1的全局中断

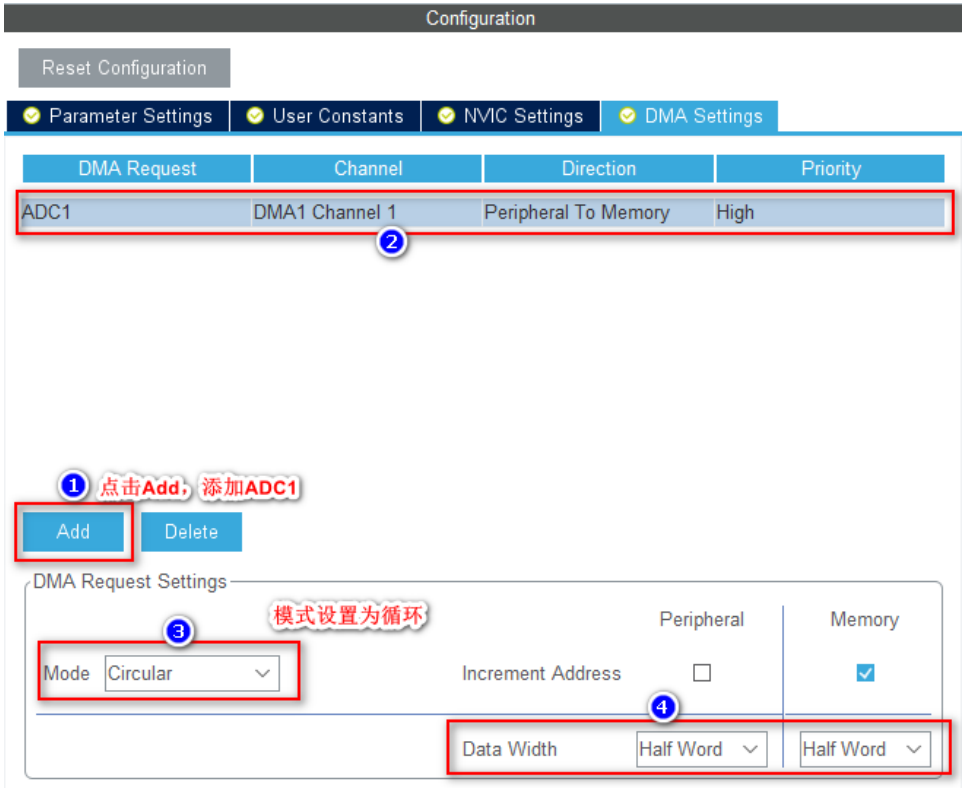


11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

ADC配置——添加ADC1的DMA传输

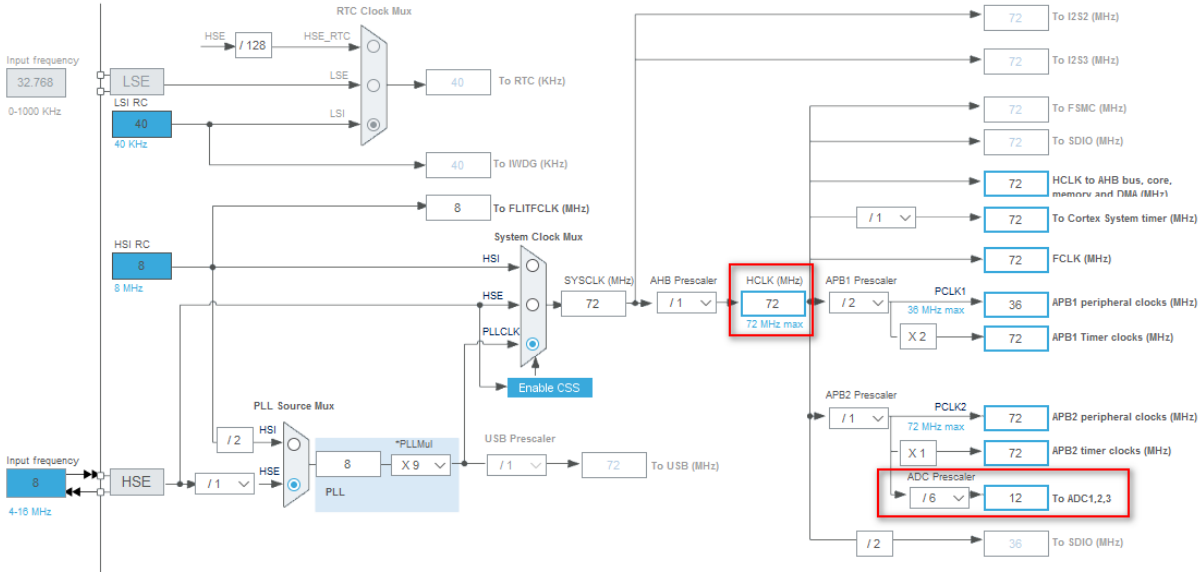
在ADC1参数配置“Configuration”选项配置页的“DMA Settings”中，单击“Add”，添加ADC1的DMA传输方式，设置其优先级“Priority”为“High”，“Mode”选择为“Circular”，“Data Width”设置为“Half Word”



11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

设置STM32时钟系统和ADC的时钟



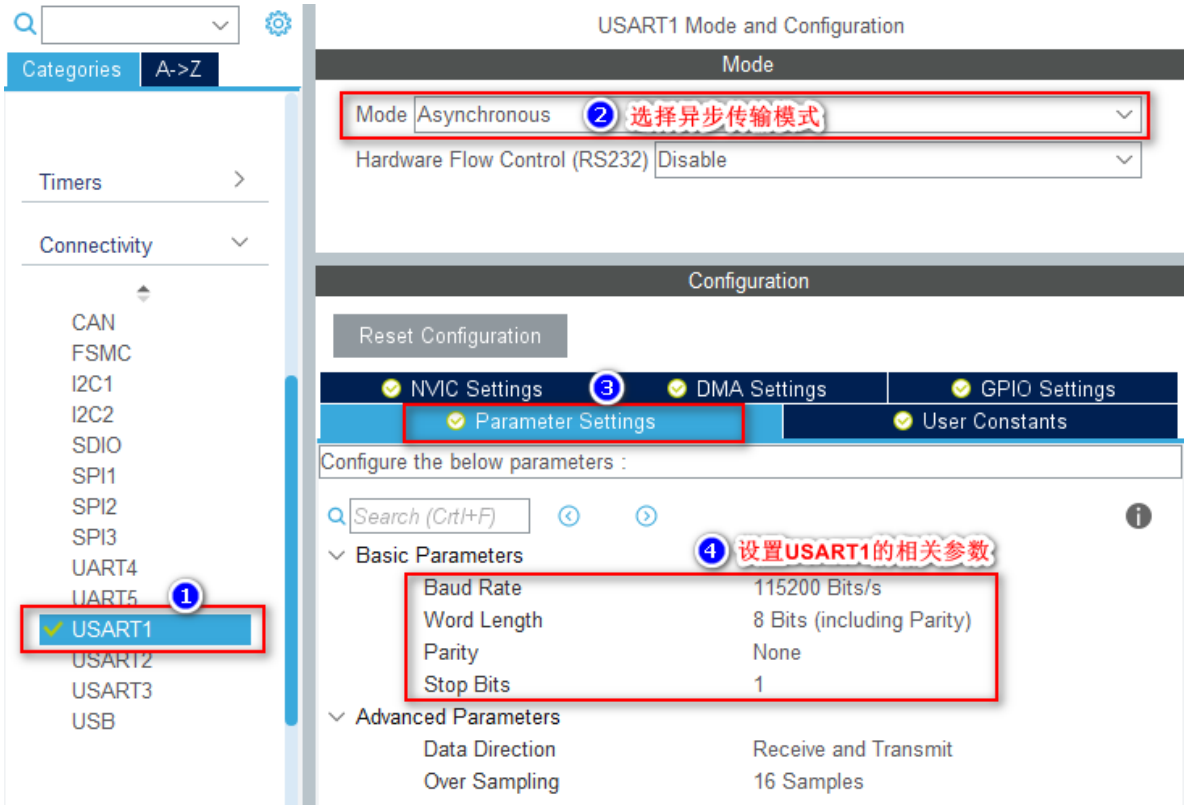
- 使用的外部晶振为8MHz
- 配置系统时钟为72MHz
- APB2为72MHz
- APB1为36MHz
- ADC为12MHz

11.3.2 ADC的HAL库应用实例

软件设计——STM32CubeMX功能参数配置

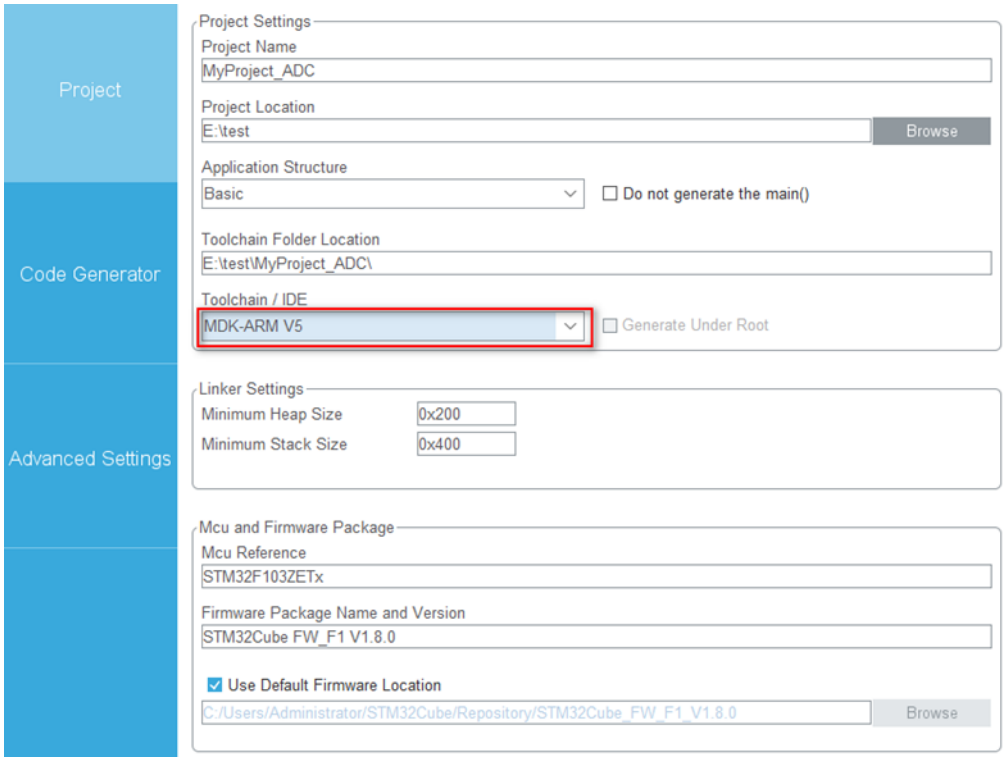
串口USART1配置

- 设置USART1为“Asychronous”模式
- 波特率设置为115200，数据位数为8
- 校验位选择无校验
- 停止位为1位



11.3.2 ADC的HAL库应用实例

软件设计——生成工程代码

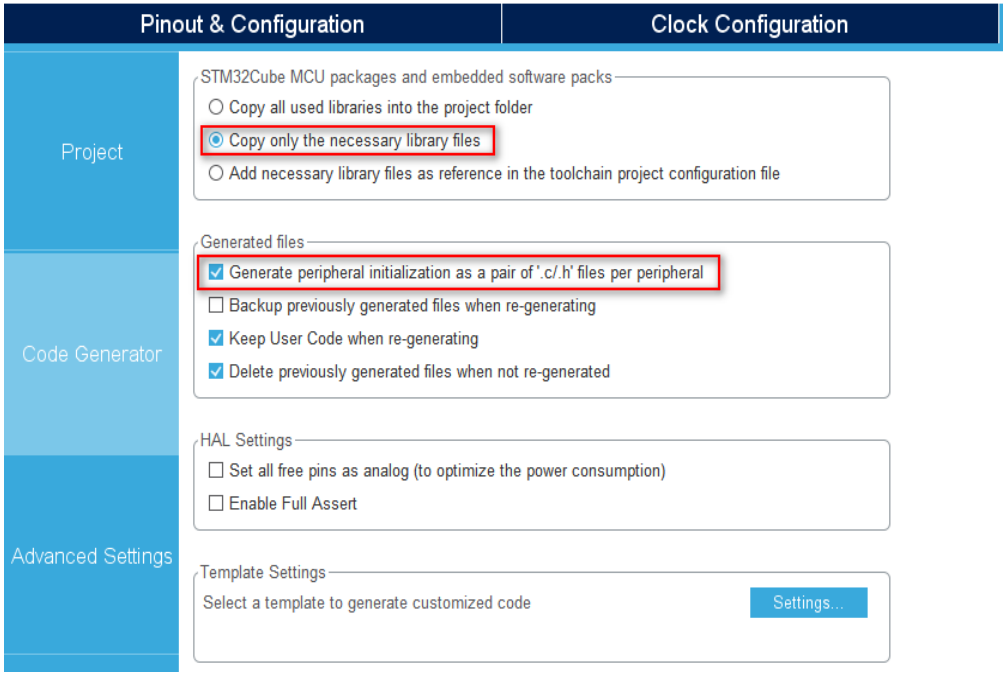


配置keil工程名称和存放位置

配置工程名称和存放位置，输入“Project Name”为“MyProject_ADC”，“Toolchain/IDE”选择“MDK-ARM V5”

11.3.2 ADC的HAL库应用实例

软件设计——生成工程代码



在“Code Generator”选项栏中找到“Generated files”框，勾选“Generate peripheral initialization as a pair of ’.c/.h’ files per IP”，将外设初始化的代码设置生成为独立的.c源文件和.h头文件。

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

- 在main.c文件中的添加代码。先定义两个数组DMA_RxBuffer和DMA_TxBuffe，分别用于存放接收数据和发送数据，将此代码放在main.c文件中的/* USER CODE BEGIN PV */和/* USER CODE END PV */之间。

```
/* USER CODE BEGIN PV */
//定义V25和Avg_Slope两个宏
#define  V25  1.43
#define  Avg_Slope  0.0043

uint16_t  ADC_Converted_Value;
//用于存放AD转换的结果
uint16_t  MCU_Temperature;
 //用于存放芯片内部的温度值
/* USER CODE END PV */
```

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

- 然后在int main(void)函数中的/* USER CODE BEGIN 2 */和/* USER CODE END 2 */之间添加代码：

```
/* USER CODE BEGIN 2 */
/* ADC1校准*/
HAL_ADCEx_Calibration_Start(&hadc1);
/*启动ADC1的DMA转换 */
HAL_ADC_Start_DMA(&hadc1,(uint32_t*)&
ADC_Converted_Value,sizeof(&ADC_Conver
ted_Value));
/* USER CODE END 2 */
```

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

- 在while语句中的/* USER CODE BEGIN 3 */和/* USER CODE END 3 */之间添加代码

```
while (1)
{
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
  HAL_Delay(1000);
  printf("----ADC+DMA+USART实验 -----\n");
  MCU_Temperature = (V25-
ADC_Converted_Value*3.3/4095)/Avg_Slope+25;
  printf("\r\n The MCU temperature = %3d \r\n
",MCU_Temperature);
}
/* USER CODE END 3 */
```

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

- 在usart.c的/* USER CODE BEGIN 1 */和/* USER CODE END 1 */添加代码：

```
/* USER CODE BEGIN 1 */
//重定向fputc函数
int fputc(int ch, FILE *f)
{  HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xffff);
  return ch;
}
//重定向fgetc函数
int fgetc(FILE * f)
{  uint8_t ch = 0;
    HAL_UART_Receive(&huart1,&ch, 1, 0xffff);
    return ch;
}
/* USER CODE END 1 */
```

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

采用轮询方式通过ADC1将STM32F103芯片内部的温度传感器采集到的温度数据通过串口发送到PC

添加全局变量

```
#define  V25  1.43
#define  Avg_Slope  0.0043

uint16_t  ADC_Converted_Value  = 0;
    //用于存放A/D转换的结果
```

11.3.2 ADC的HAL库应用实例

软件设计——编写应用程序

在while(1)中添加代码

```
/* USER CODE BEGIN 3 */
HAL_ADC_Start(&hadc1);//启动ADC装换
HAL_ADC_PollForConversion(&hadc1, 50);
    //等待转换结束，第二个参数为超时时间，单位为ms
    //判断转换完成标志位是否设置，HAL_ADC_STATE_REG_EOC为转换完成标志位
if(HAL_IS_BIT_SET(HAL_ADC_GetState(&hadc1), HAL_ADC_STATE_REG_EOC))
{  ADC_Converted_Value = HAL_ADC_GetValue(&hadc1);
     //读取ADC转换数据，数据为12位
  printf("MCU Temperature : %.1f度\r\n",( (V25-ADC_Converted_Value*3.3/
4095)/Avg_Slope+25);
}
HAL_Delay(1000);
/* USER CODE END 3 */
```

本章小结

ADC基础理论知识
- 转换过程
- 主要技术参数

ADC模块的HAL库接口函数及应用
- 接口函数
- 应用实例

STM32的ADC模块
- 引脚、通道选择、中断、DMA请求、转换时间、数据对齐、转换模式、校准

The End!