STM32Cube高效开发教程——基础篇

第14章　ADC

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

第14章　ADC

14.1 ADC功能概述

14.2 ADC的HAL驱动程序

14.3 示例1：软件启动ADC转换

14.4 示例2：定时器触发ADC转换

14.1 ADC功能概述

14.1.1 ADC的特性

14.1.2 ADC的工作原理

14.1.3 多重ADC模式

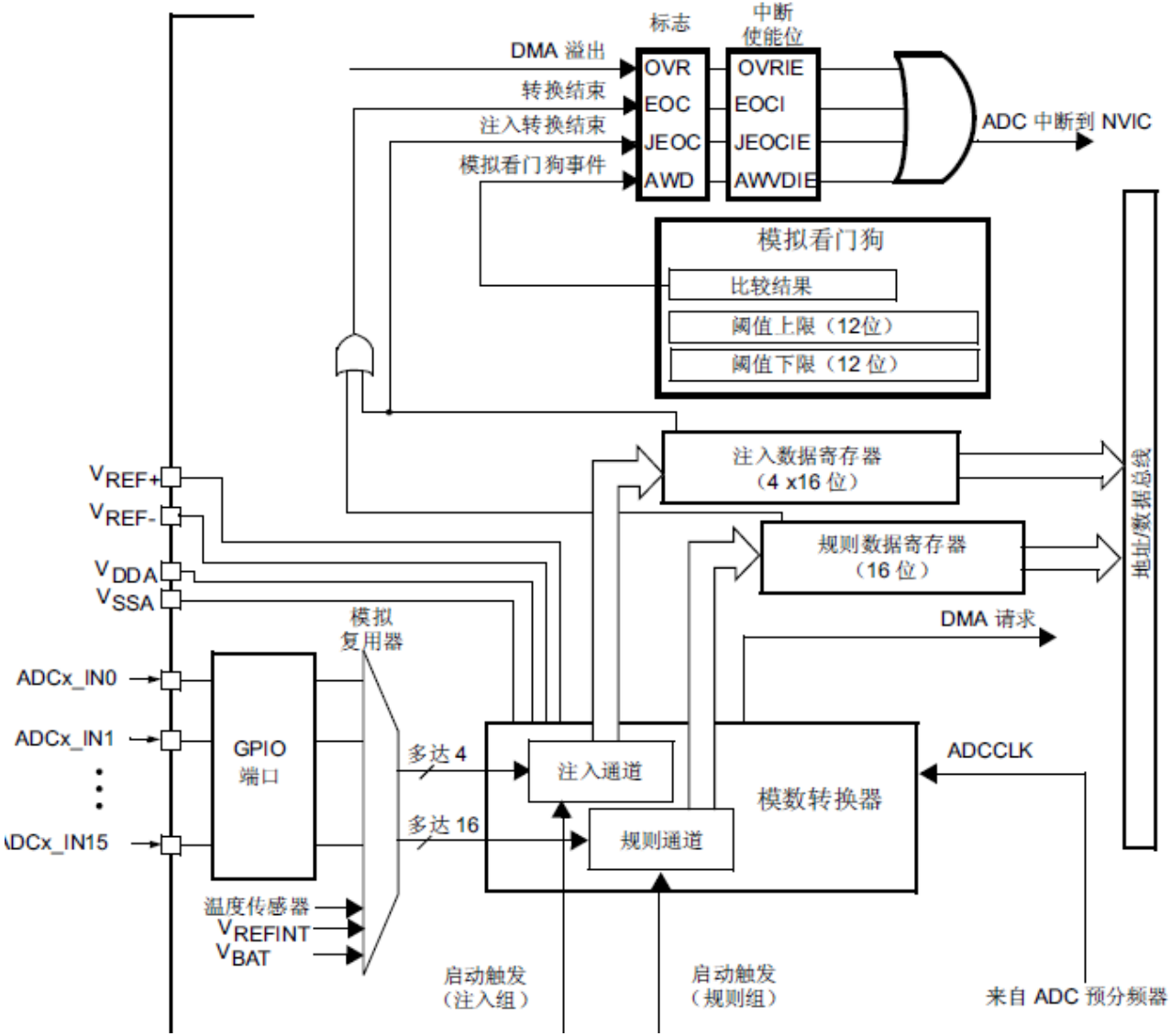
14.1.1 ADC的特性

STM32F407有3个片上ADC单元，最高12位分辨率，最多16个外部通道。ADC的主要特性如下：

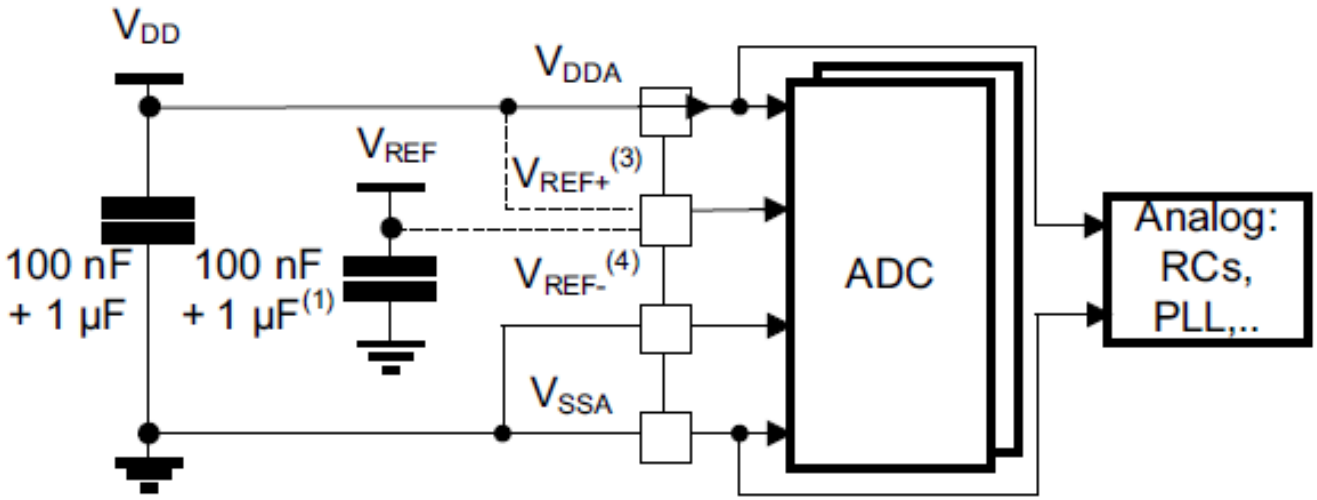
- 可配置为12位、10位、8位或6位分辨率。
- 每个ADC有16个外部输入通道，其中ADC1还有3个内部输入通道，可测量内部温度、内部参考电压和备用电压。
- 多个通道输入时可以划分为规则通道（regular channels）和注入通道（injected channels）。
- 可以单次转换，或连续转换。
- 多通道输入时，具有从通道0到通道n的扫描模式。
- 具有内部和外部触发选项，可由定时器触发或外部中断触发。
- 具有模拟看门狗功能，可以监测电压范围。
- 具有双重（两个ADC工作）或三重（三个ADC工作）模式。

14.1.2 ADC的工作原理

单个ADC的内部功能结构



1. 模拟部分供电



ADC转换电压的输入范围范围是$V_{REF-} \le V_{IN} \le V_{REF+}$，由于$V_{REF-}$必须与$V_{SSA}$连接，也就是$V_{REF-}$总是0，所以STM32F407的片上ADC只能转换正电压。

2. 输入通道

每个ADC单元有16个外部输入通道，对应于16个ADC输入复用引脚。ADC1还有3个内部输入使用通道16至18，分别是：

- 温度传感器：芯片内部温度传感器，测温范围-40℃至125℃，精度±1.5℃。
- V_{REFINT}：内部参考电压，实际连接内部1.2V调压器的输出电压。
- V_{BAT}：备用电源电压，因为V_{BAT}电压可能高于V_{DDA}，内部有桥接分压器，实际测量的电压是$V_{BAT}/2$。

3. 规则通道和注入通道

选择的多个模拟输入通道可以分为两组：规则通道和注入通道，每个组的通道构成一个转换序列。

- 规则转换序列最多可设置16个通道，一个规则转换序列规定了多路复用转换时的顺序。
- 注入通道就是可以在规则通道转换过程中插入进行转换的通道，类似于中断的现象。每个注入通道还可以设置一个数据偏移量，每次转换结果自动减去这个偏移量，所以转换结果可以是负数。

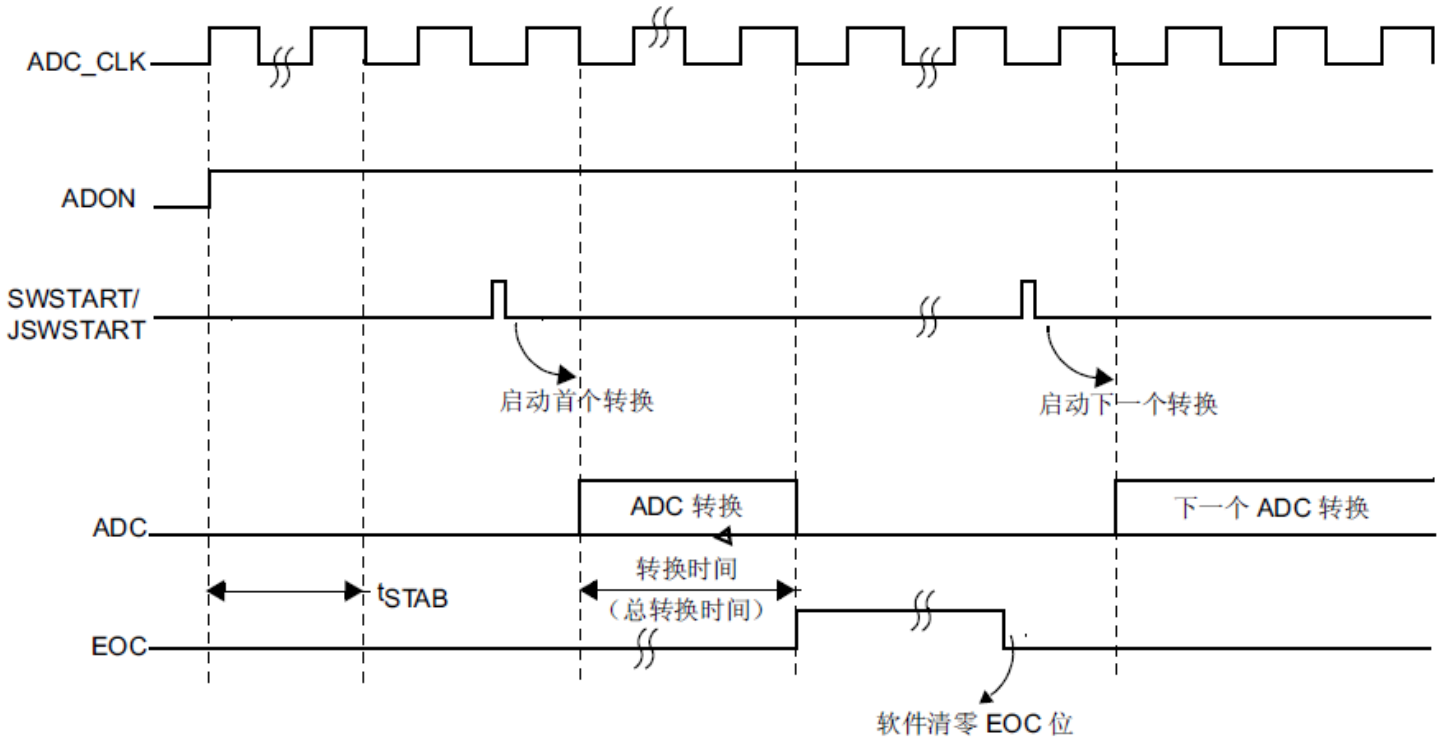
4. 启动或触发转换

规则通道和注入通道有单独的触发源，有三类启动或触发转换的方式：

- 软件启动：这种方式常用于轮询方式的ADC转换。
- 内部定时器触发：可以选择某个定时器的触发输出信号（TRGO）作为触发源，这样定时器每次定时溢出时就启动一次转换。这种方式可用于周期性ADC转换。
- 外部IO触发：可以选择外部中断线EXTI_11或EXTI_15作为规则组或注入组的外部中断触发源。

5. ADC时钟与转换时间

ADCCLK最高42MHz。一个通道一次ADC转换的总时间是N+12个ADCCLK周期，N是设置的采样次数。

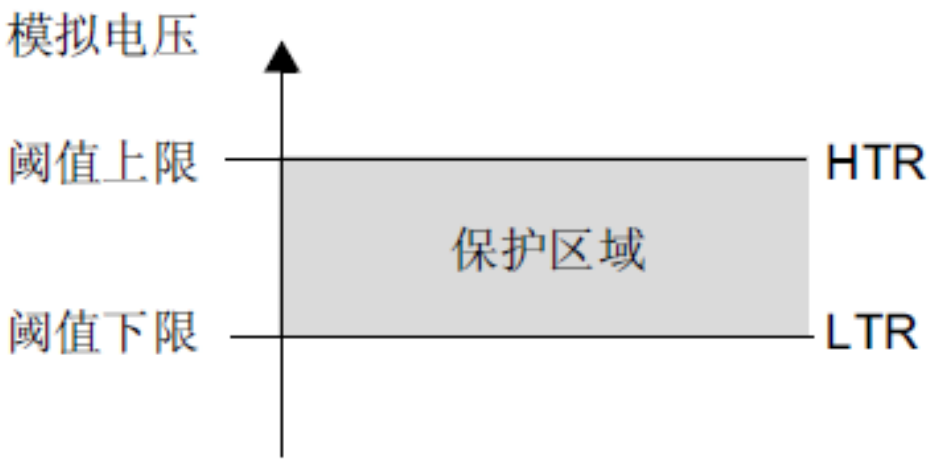


6. 转换结果数据寄存器

ADC完成转换后将结果数据存入数据寄存器，规则通道和注入通道有不同的数据寄存器。

- 规则通道只有一个数据寄存器ADC_DR，只有低16位有效。在多通道时一般在EOC中断里及时读取数据或通过DMA将数据传输到内存里。
- 注入通道有4个数据寄存器，分别对应4个注入通道的转换结果。

7. 模拟看门狗



可以使用模拟看门狗对某一个通道的模拟电压进行监测，设置一个阈值上限和下限（12位数表示的数值，0至4095），当监测的模拟电压ADC结果超出范围时就产生模拟看门狗中断

8. 转换结果电压计算

ADC转换的结果是一个数字量，与实际的模拟电压之间的计算关系由V_{REF+}和转换精度位数确定。

例如转换精度为12位，V_{REF+}=3.3V，ADC转换结果为12位数字量对应的整数X，则实际电压为

$$\text{Voltage} = \frac{3300 * X}{4096}\ mV$$

14.1.3 多重ADC模式

STM32F407有3个ADC，这3个ADC可以独立工作，也可以组成双重或三重工作模式。

- 在多重模式下，ADC1是主器件，必须被使用
- 双重模式就是使用ADC1和ADC2，不能使用ADC1和ADC3
- 三重模式就是3个ADC都使用
- 多重ADC有多种工作模式，可以交替触发，也可以同步触发
- 在双重ADC同步模式下，两个ADC不能转换同一个通道

14.2 ADC的HAL驱动程序

14.2.1 常规通道

14.2.2 注入通道

14.2.3 多重ADC

14.2.1 常规通道

分组	函数名	功能描述
初始化和配置	HAL_ADC_Init()	ADC的初始化，设置ADC的总体参数
	HAL_ADC_MspInit()	ADC初始化的MSP弱函数，在HAL_ADC_Init()里被调用
	HAL_ADC_ConfigChannel()	ADC常规通道配置，一次配置一个通道
	HAL_ADC_AnalogWDGConfig()	模拟看门狗配置
	HAL_ADC_GetState()	返回ADC当前状态
	HAL_ADC_GetError()	返回ADC的错误码
软件启动转换	HAL_ADC_Start()	启动ADC，并开始常规通道的转换
	HAL_ADC_Stop()	停止常规通道的转换，并停止ADC
	HAL_ADC_PollForConversion()	轮询方式等待ADC常规通道转换完成
	HAL_ADC_GetValue()	读取常规通道转换结果寄存器的数据
中断方式转换	HAL_ADC_Start_IT()	开启中断，开始ADC常规通道的转换
	HAL_ADC_Stop_IT()	关闭中断，停止ADC常规通道的转换
	HAL_ADC_IRQHandler()	ADC中断ISR函数里调用的ADC中断通用处理函数
DMA方式转换	HAL_ADC_Start_DMA()	开启ADC的DMA请求，开始ADC常规通道的转换
	HAL_ADC_Stop_DMA()	停止ADC的DMA请求，停止ADC常规通道的转换

软件启动转换

```
HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef* hadc);
 //软件启动转换

HAL_StatusTypeDef HAL_ADC_Stop(ADC_HandleTypeDef* hadc);
 //停止转换

HAL_StatusTypeDef HAL_ADC_PollForConversion(ADC_HandleTypeDef*
                                hadc, uint32_t Timeout);

uint32_t HAL_ADC_GetValue(ADC_HandleTypeDef* hadc);
//读取转换结果寄存器的32位数据
```

其中，参数hadc是ADC外设对象指针，Timeout是超时等待时间，单位是节拍数。

中断方式转换

启动和停止中断方式ADC转换的函数：

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

ADC有4个中断事件源，中断事件类型的宏定义如下：

```
#define ADC_IT_EOC      ((uint32_t)ADC_CR1_EOCIE)
 //规则通道转换结束（EOC）事件

#define ADC_IT_AWD      ((uint32_t)ADC_CR1_AWDIE)
//模拟看门狗触发事件

#define ADC_IT_JEOC     ((uint32_t)ADC_CR1_JEOCIE)
 //注入通道转换结束事件

#define ADC_IT_OVR      ((uint32_t)ADC_CR1_OVRIE)
//数据溢出事件,即转换结果未被及时读出
```

ADC的中断事件类型和对应的回调函数

中断事件类型	中断事件	回调函数
ADC_IT_EOC	规则通道转换结束（EOC）事件	HAL_ADC_ConvCpltCallback()
ADC_IT_AWD	模拟看门狗触发事件	HAL_ADC_LevelOutOfWindowCallback()
ADC_IT_JEOC	注入通道转换结束事件	HAL_ADCEx_InjectedConvCpltCallback()
ADC_IT_OVR	数据溢出事件，即数据寄存器内的数据未被及时读出	HAL_ADC_ErrorCallback()

DMA方式转换

ADC只有一个DMA请求，方向是外设到存储器。

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

其中，参数hadc的ADC外设对象指针；参数pData是uint32_t类型缓存区指针，因为ADC转换结果寄存器是32位的，所以DMA数据宽度是32位；参数Length是缓存区长度，单位是字（4个字节）

DMA流中断事件类型和关联的回调函数

DMA中断事件类型	DMA中断事件类型	关联的回调函数名称
DMA_IT_TC	传输完成中断	HAL_ADC_ConvCpltCallback()
DMA_IT_HT	传输半完成中断	HAL_ADC_ConvHalfCpltCallback()
DMA_IT_TE	传输错误中断	HAL_ADC_ErrorCallback()

14.2.2 注入通道

ADC的注入通道相关函数

分组	函数名	功能描述
通道配置	HAL_ADCEx_InjectedConfigChannel()	注入通道配置
软件启动转换	HAL_ADCEx_InjectedStart()	软件方式启动注入通道的转换
	HAL_ADCEx_InjectedStop()	软件方式停止注入通道的转换
	HAL_ADCEx_InjectedPollForConversion()	查询注入通道转换是否完成
	HAL_ADCEx_InjectedGetValue()	读取注入通道的转换结果数据寄存器
中断方式转换	HAL_ADCEx_InjectedStart_IT()	开启注入通道的中断方式转换
	HAL_ADCEx_InjectedStop_IT()	停止注入通道的中断方式转换
	HAL_ADCEx_InjectedConvCpltCallback()	注入通道转换结束中断事件（ADC_IT_JEOC）的回调函数

14.2.3 多重ADC

多重ADC就是两个或三个ADC同步或交错使用

函数名	功能描述
HAL_ADCEx_MultiModeConfigChannel()	多重模式的通道配置
HAL_ADCEx_MultiModeStart_DMA()	以DMA方式启动多重ADC
HAL_ADCEx_MultiModeStop_DMA()	停止多重ADC的DMA方式传输
HAL_ADCEx_MultiModeGetValue()	停止多重ADC后，读取最后一次转换结果数据

第14章　ADC

14.1 ADC功能概述

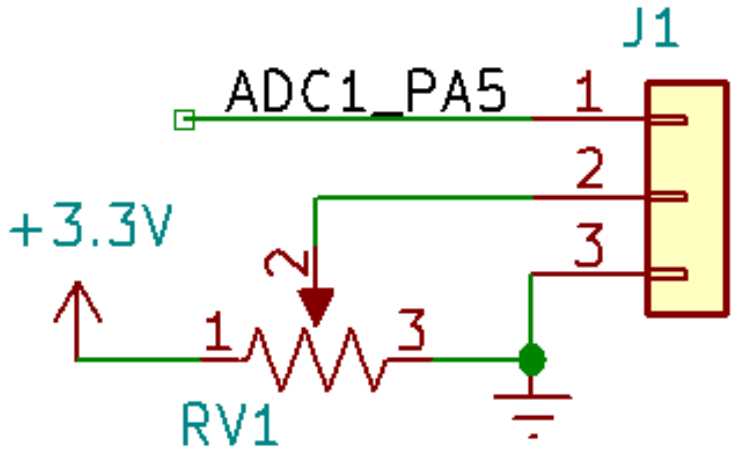
14.2 ADC的HAL驱动程序

14.3 示例1：软件启动ADC转换

14.4 示例2：定时器触发ADC转换

14.3.1 电路和示例功能

示例Demo14_1ADC_Poll，使用ADC1的IN5通道采集电位器的电压并在LCD上显示。采用软件方式启动ADC转换，在main()函数的while循环里每隔约500ms转换一次。

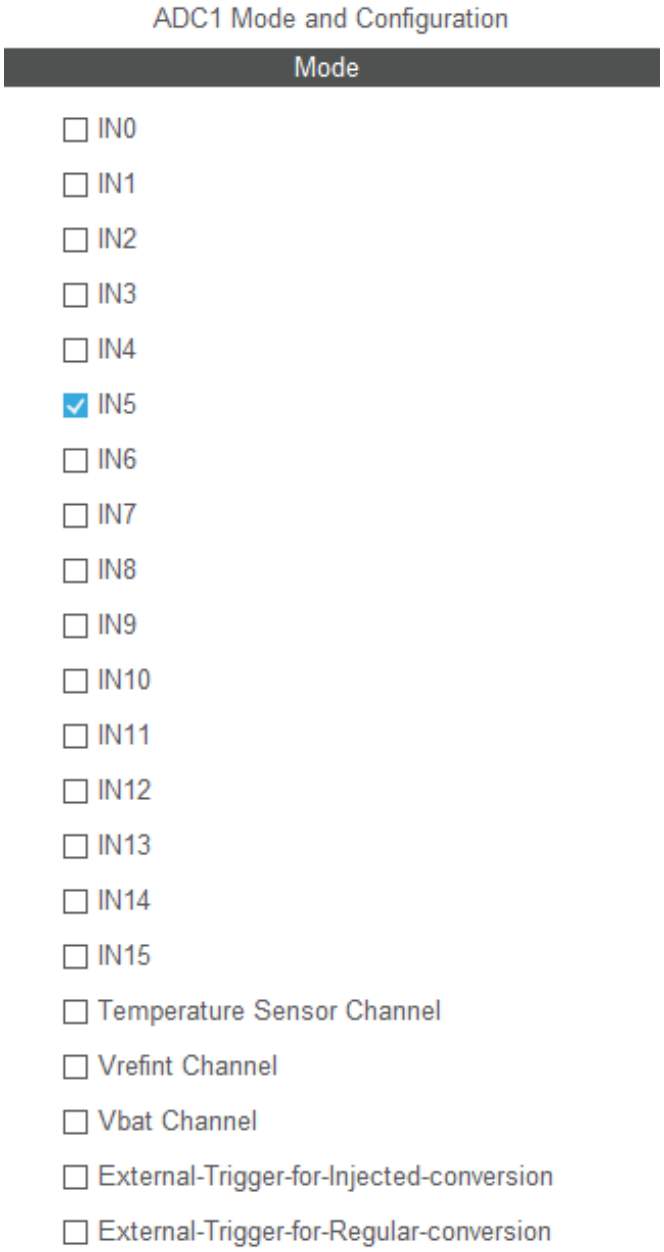


开发板上的可调电位器

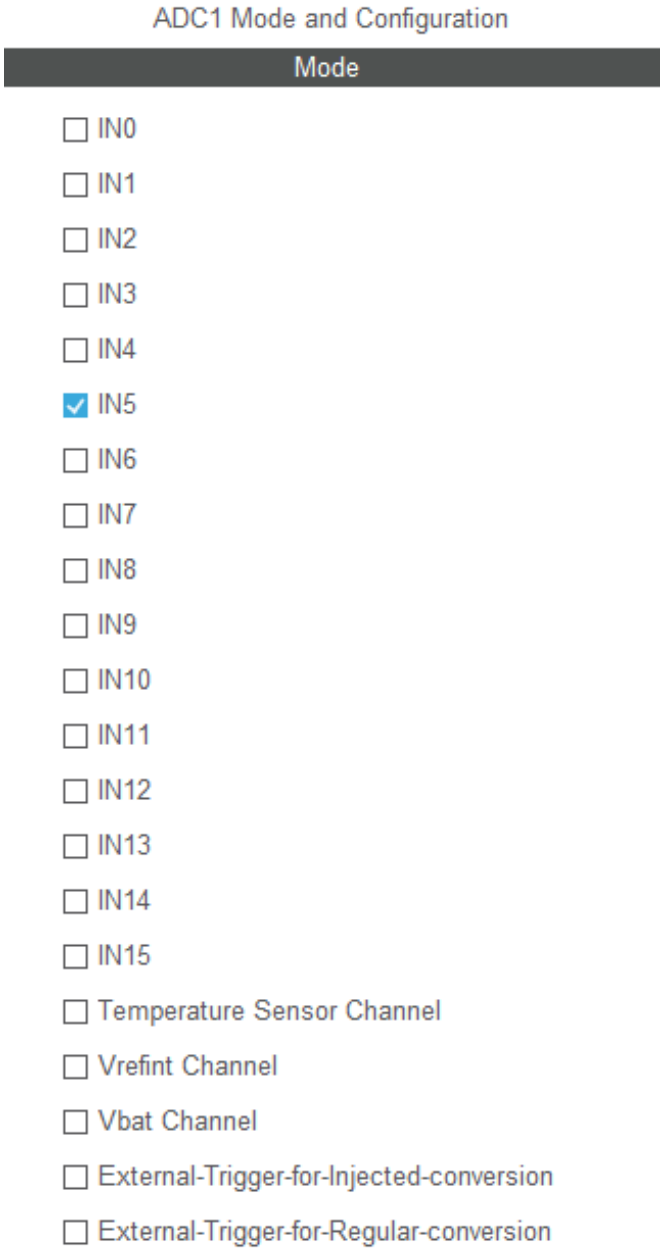
14.3.2 CubeMX项目设置

ADC1的模式设置

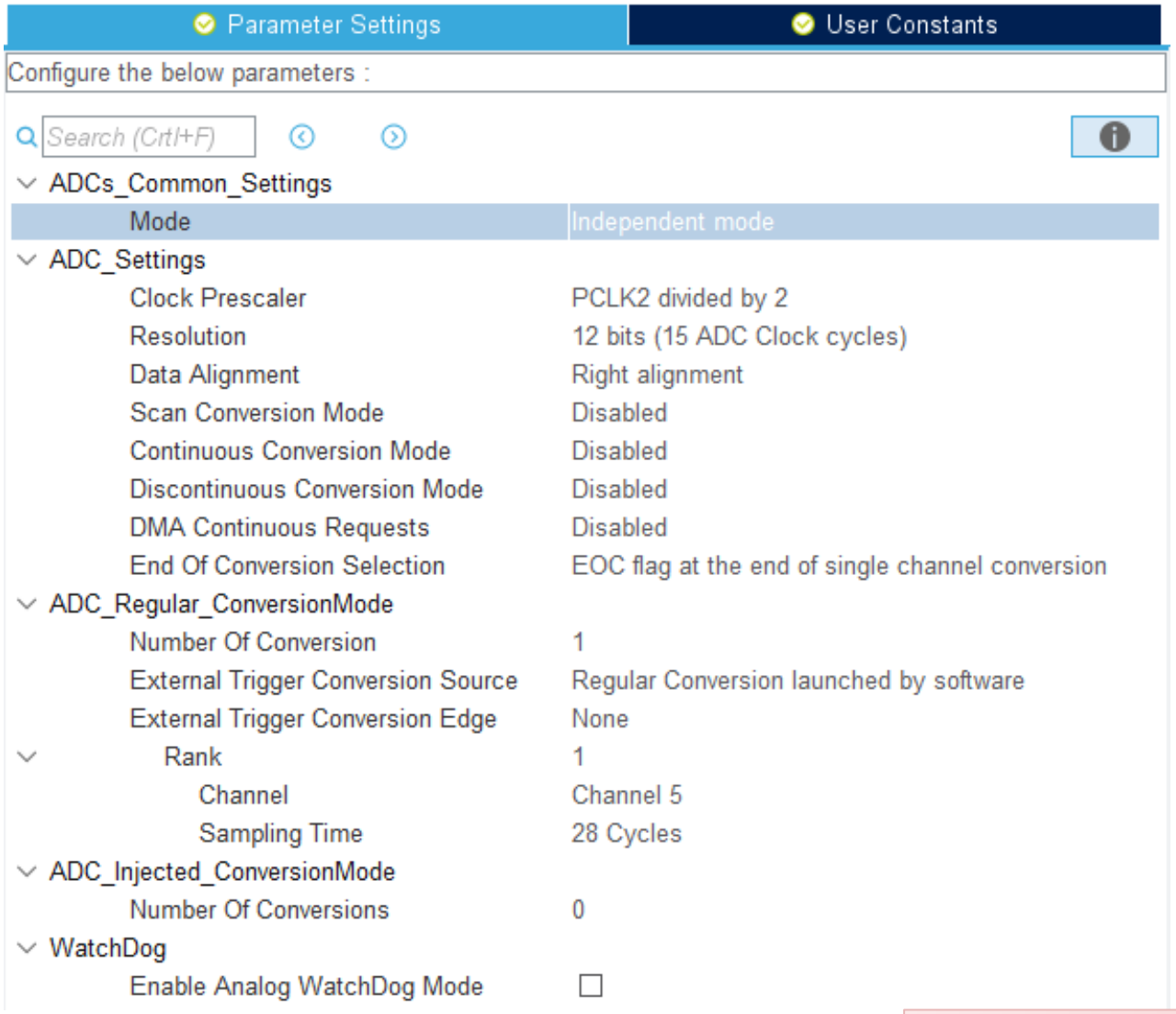
- IN0至IN15，是ADC1的16个外部输入通道。开发板电位器的输出连接的是ADC1的IN5通道。
- Temperature Sensor Channel，内部的温度传感器通道，连接ADC1的IN16通道。
- Vrefint Channel，内部参考电压通道，连接ADC1的IN17通道。



- Vbat Channel，备用电源V_{BAT}的通道，连接ADC1的IN18通道。
- Enternal-Trigger-for-Injected-conversion，为注入转换使用外部触发。
- Enternal-Trigger-for-Regular-conversion，为规则转换使用外部触发。



ADC1的参数设置，独立转换模式，12位精度，右对齐



参数的详细解释见教材

14.3.3 程序功能实现

1. 主程序

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_FSMC_Init();
    MX_ADC1_Init();   //ADC1初始化

    /*  USER CODE BEGIN 2 */
    TFTLCD_Init();                  //LCD软件初始化
     //省略中间代码….
    /* USER CODE END 2 */
```

在mian()函数的while循环里，每500毫秒以软件触发方式进行一次ADC转换方式

打开源代码进行解释

```
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        HAL_ADC_Start(&hadc1);          //必须每次启动转换
        if (HAL_ADC_PollForConversion(&hadc1,200)==HAL_OK)
        {
            uint32_t val=HAL_ADC_GetValue(&hadc1);    //读取转换结果
            LCD_ShowUintX(orgX,orgY,val, 5);      //5位显示，前端补空格

            uint32_t Volt=3300*val;         //以mV为单位
            Volt=Volt>>12;                  //除以2^12
            LCD_ShowUintX(voltX,voltY,Volt, 4);    //4位显示，前端补空格
        }
        //      HAL_ADC_Stop(&hadc1);       //无需每次都停止
        HAL_Delay(500);
        /* USER CODE END WHILE */
    }
}
```

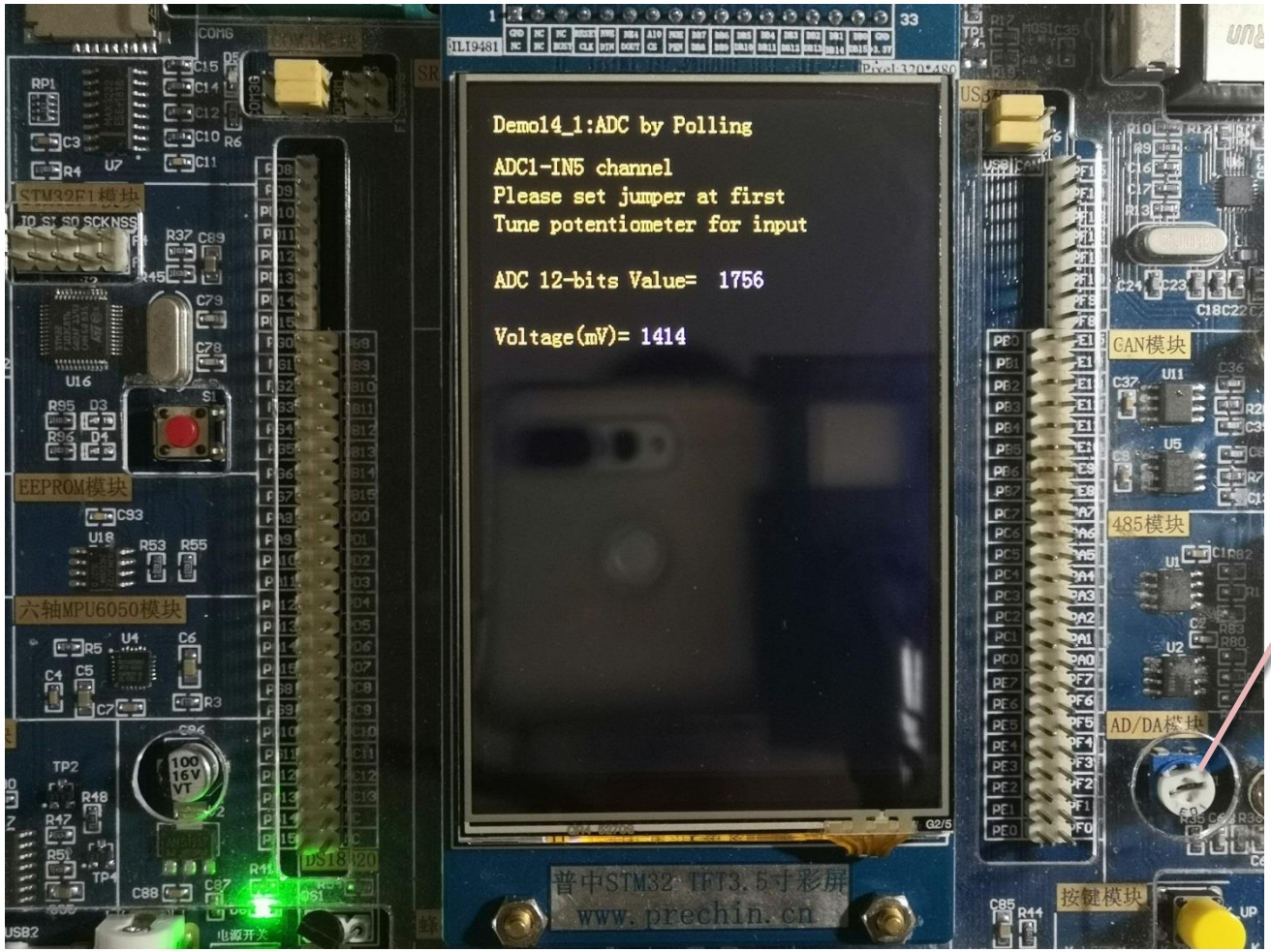
2. ADC1初始化

文件adc.c中的函数MX_ADC1_Init()对ADC1初始化

```
#include "adc.h"
ADC_HandleTypeDef  hadc1;          //ADC1外设对象变量

/*  ADC1 初始化函数  */
void MX_ADC1_Init(void)
{
      ADC_ChannelConfTypeDef sConfig = {0};
      /** 配置ADC的全局特性(时钟, 分辨率, 数据对其方式,转换个数等) */
      hadc1.Instance = ADC1;          //寄存器基址
      hadc1.Init.Resolution = ADC_RESOLUTION_12B; //分辨率12位
      if (HAL_ADC_Init(&hadc1) != HAL_OK)                  //ADC1模块初始化
            Error_Handler();

      /** 配置规则转换组里每个Rank，配置通道和采样点数  */
      sConfig.Channel = ADC_CHANNEL_5;  //输入通道
      sConfig.Rank = 1;                 //Rank 序号
      sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES;
      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
            Error_Handler();
}
```

打开源代码进行解释

运行测试



调节电位器，改变ADC的输入值

第14章　ADC

14.1 ADC功能概述

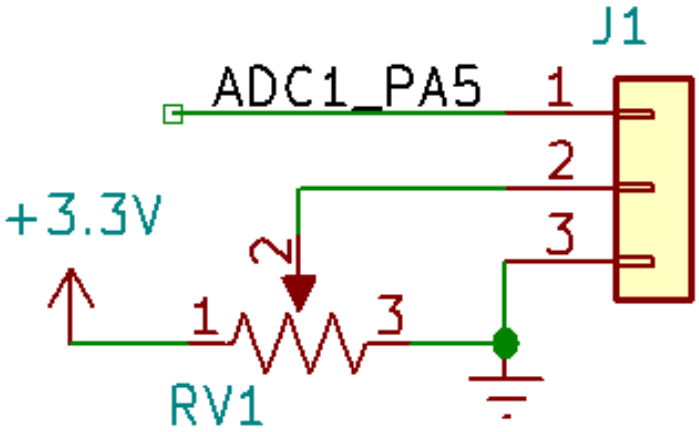
14.2 ADC的HAL驱动程序

14.3 示例1：软件启动ADC转换

14.4 示例2：定时器触发ADC转换

14.4.1 示例功能和CubeMX项目设置

示例Demo14_2TimTrigger，使用TIM3的TRGO信号作为ADC1的外部触发信号，TIM3定时周期为500ms，ADC1以中断模式启动转换，在ADC的转换完成中断里读取转换结果数据



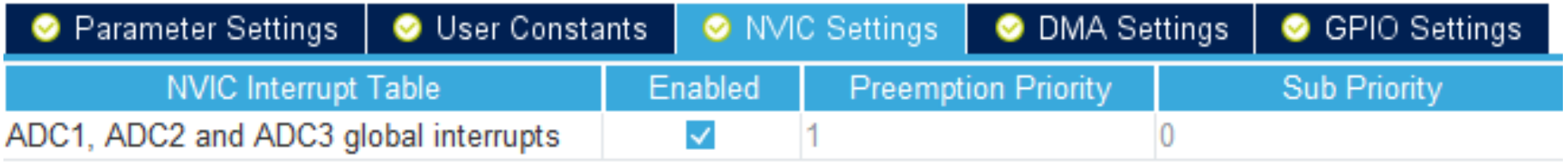
开发板上的可调电位器

1. ADC1的设置

- External Trigger Conversion Source，设置启动ADC转换的外部触发信号源，这里选择Timer 3 Trigger Out Event，也就是定时器TIM3的TRGO信号
- External Trigger Conversion Edge，设置触发转换的跳变沿，这里选择上跳沿，因为TRGO是一个短时正脉冲信号

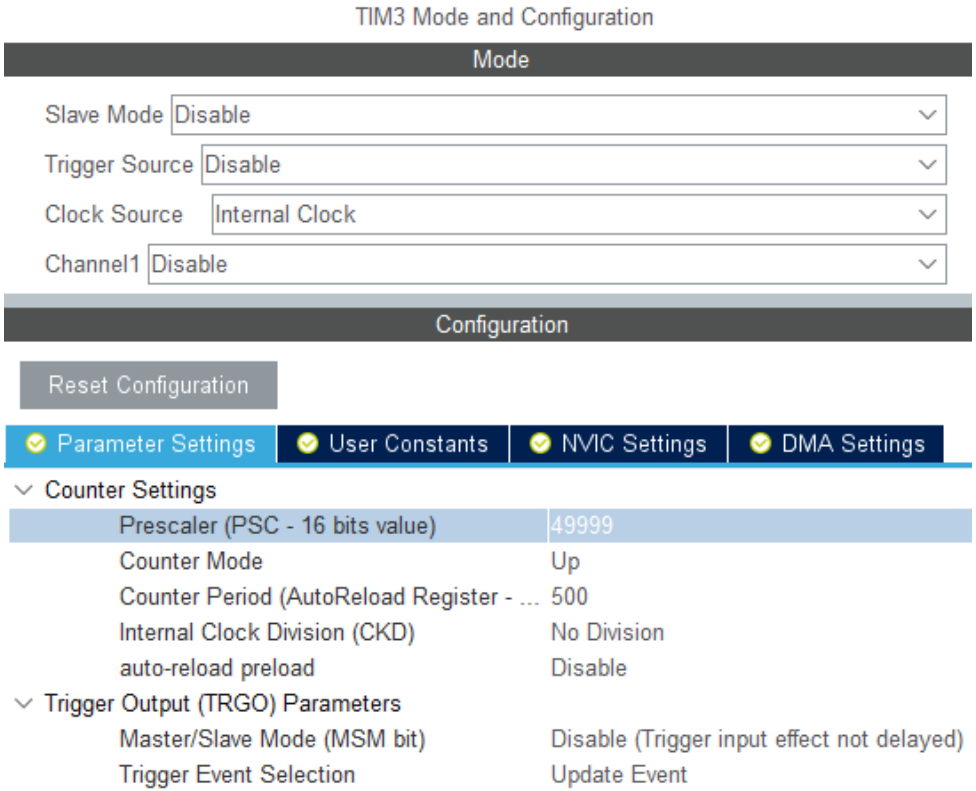
ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion Source	Timer 3 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Channel	Channel 5
Sampling Time	15 Cycles

开启ADC1的全局中断，在转换完成事件（ADC_IT_EOC）中断里读取转换结果。



2. TIM3的设置

- 使TIM3定时周期为500ms
- 触发事件选择（Trigger Event Selection）设置为Update Event，也就是以UEV事件信号作为TRGO信号



14.4.2 程序功能实现

1. 主程序

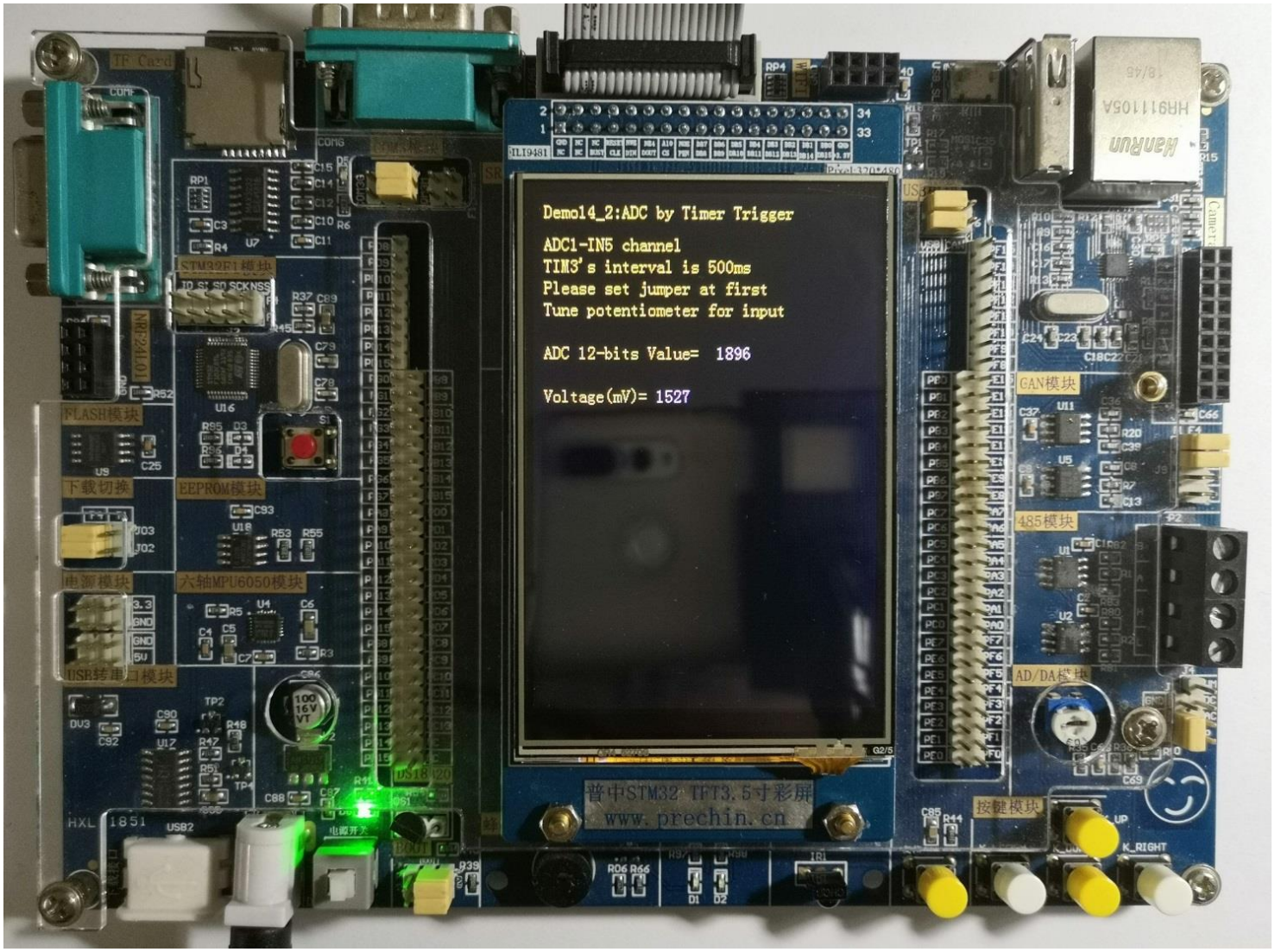
打开源代码进行解释

```
uint16_t orgX, orgY;                    //LCD上的显示位置，显示ADC原始值
uint16_t voltX,voltY;                   //LCD上的显示位置，显示电压 mV
int main(void)
{
    HAL_Init();
    // ……
    LCD_ShowStr(10,LCD_CurY+LCD_SP20, (uint8_t *)"ADC 12-bits Value= ");
    orgX=LCD_CurX;                      //记录LCD显示位置
    orgY=LCD_CurY;
    LCD_ShowStr(10,LCD_CurY+LCD_SP20, (uint8_t *)"Voltage(mV)= ");
    voltX=LCD_CurX;                     //记录LCD显示位置
    voltY=LCD_CurY;

    HAL_ADC_Start_IT(&hadc1);           //启动ADC,中断模式
    HAL_TIM_Base_Start(&htim3);         //启动定时器

    while (1)
    {
    }
}
```

ADC中断回调函数，就在main.c中重新实现，在回调函数里读取转换结果，并显示原始值和mV电压值

```
/* USER CODE BEGIN 4 */
/*   ADC的转换完成事件(ADC_IT_EOC)中断回调函数    */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if (hadc->Instance == ADC1)
    {
        uint32_t val=HAL_ADC_GetValue(hadc);//读取转换结果
        LCD_ShowUintX(orgX,orgY,val, 5); //5位显示，前端补空格
        uint32_t Volt=3300*val;        //以mV为单位
        Volt=Volt>>12;                 //除以2^12
        LCD_ShowUintX(voltX,voltY,Volt, 4);    //4位显示，前端补空格
    }
}
/* USER CODE END 4 */
```

运行测试


Demo14_2:ADC by Timer Trigger
ADC1-IN5 channel
TIM3's interval is 500ms
Please set jumper at first
Tune potentiometer for input
ADC 12-bits Value= 1896
Voltage(mV)= 1527
普中STM32 TFT3.5寸彩屏
www.prechin.cn
STM32F1模块
FLASH模块
EEPROM模块
六轴MPU6050模块
GAN模块
485模块
AD/DA模块
按键模块
K_RIGHT

练习任务

1. 自学教材的14.5和14.6节，掌握ADC的DMA方式数据传输，以及双ADC同步转换的方法，编程测试。