本章教程使用CH32V103开发板的ADC1通道1对开发板的VCC引脚和GND引脚进行采样，并将采样结果通过串口调试助手打印显示。

# 1、ADC简介及相关函数介绍

CH32V103的ADC模块包含一个 12 位的逐次逼近型的模拟数字转换器，最高14MHz的输入时钟。支持16个外部通道和2个内部信号源采样源。可完成通道的单次转换、连续转换，通道间自动扫描模式、间断模式、外部触发模式等功能。可以通过模拟看门狗功能监测通道电压是否在阈值范围内。

关于ADC具体信息，可参考CH32V103应用手册。ADC标准库函数具体内容如下：

1. void ADC_DeInit(ADC_TypeDef* ADCx);
2. void ADC_Init(ADC_TypeDef* ADCx, ADC_InitTypeDef* ADC_InitStruct);
3. void ADC_StructInit(ADC_InitTypeDef* ADC_InitStruct);
4. void ADC_Cmd(ADC_TypeDef* ADCx, FunctionalState NewState);
5. void ADC_DMACmd(ADC_TypeDef* ADCx, FunctionalState NewState);
6. void ADC_ITConfig(ADC_TypeDef* ADCx, uint16_t ADC_IT, FunctionalState NewState);
7. void ADC_ResetCalibration(ADC_TypeDef* ADCx);
8. FlagStatus ADC_GetResetCalibrationStatus(ADC_TypeDef* ADCx);
9. void ADC_StartCalibration(ADC_TypeDef* ADCx);
10. FlagStatus ADC_GetCalibrationStatus(ADC_TypeDef* ADCx);
11. void ADC_SoftwareStartConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
12. FlagStatus ADC_GetSoftwareStartConvStatus(ADC_TypeDef* ADCx);
13. void ADC_DiscModeChannelCountConfig(ADC_TypeDef* ADCx, uint8_t Number);
14. void ADC_DiscModeCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
15. void ADC_RegularChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime);

16. void ADC_ExternalTrigConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
17. uint16_t ADC_GetConversionValue(ADC_TypeDef* ADCx);
18. uint32_t ADC_GetDualModeConversionValue(void);
19. void ADC_AutoInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
20. void ADC_InjectedDiscModeCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
21. void ADC_ExternalTrigInjectedConvConfig(ADC_TypeDef* ADCx, uint32_t ADC_ExternalTrigInjecConv);
22. void ADC_ExternalTrigInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
23. void ADC_SoftwareStartInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState);
24. FlagStatus ADC_GetSoftwareStartInjectedConvCmdStatus(ADC_TypeDef* ADCx);
25. void ADC_InjectedChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime);
26. void ADC_InjectedSequencerLengthConfig(ADC_TypeDef* ADCx, uint8_t Length);
27. void ADC_SetInjectedOffset(ADC_TypeDef* ADCx, uint8_t ADC_InjectedChannel, uint16_t Offset);
28. uint16_t ADC_GetInjectedConversionValue(ADC_TypeDef* ADCx, uint8_t ADC_InjectedChannel);
29. void ADC_AnalogWatchdogCmd(ADC_TypeDef* ADCx, uint32_t ADC_AnalogWatchdog);
30. void ADC_AnalogWatchdogThresholdsConfig(ADC_TypeDef* ADCx, uint16_t HighThreshold, uint16_t LowThreshold);
31. void ADC_AnalogWatchdogSingleChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel);
32. void ADC_TempSensorVrefintCmd(FunctionalState NewState);
33. FlagStatus ADC_GetFlagStatus(ADC_TypeDef* ADCx, uint8_t ADC_FLAG);
34. void ADC_ClearFlag(ADC_TypeDef* ADCx, uint8_t ADC_FLAG);
35. ITStatus ADC_GetITStatus(ADC_TypeDef* ADCx, uint16_t ADC_IT);
36. void ADC_ClearITPendingBit(ADC_TypeDef* ADCx, uint16_t ADC_IT);
37. s32 TempSensor_Volt_To_Temper(s32 Value);

复制代码

1.1、void ADC_DeInit(ADC_TypeDef* ADCx)

功 能：将ADCx外围寄存器初始化为其默认重置值。

输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.2、void ADC_Init(ADC_TypeDef* ADCx, ADC_InitTypeDef* ADC_InitStruct)

功 能：根据ADC_InitStruct中指定的参数初始化ADCx外围设备。

输 入：ADCx：其中x可以是1以选择ADC外围设备；

ADC_InitStruct：指向包含指定ADC外围设备的配置信息的ADC_InitTypeDef结构的指针。

1.3、void ADC_StructInit(ADC_InitTypeDef* ADC_InitStruct)
功 能：用默认值填充每个ADC_InitStruct成员。
输 入：ADC_InitStruct：指向包含指定ADC外围设备的配置信息的ADC_InitTypeDef结构的指针。

1.4、void ADC_Cmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：启用或禁用指定的ADC外围设备。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.5、void ADC_DMACmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：启用或禁用指定的ADC DMA请求。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.6、void ADC_ITConfig(ADC_TypeDef* ADCx, uint16_t ADC_IT, FunctionalState NewState)
功 能：启用或禁用指定的ADC中断。
输 入：ADCx：其中x可以是1以选择ADC外围设备；ADC_IT:指定要启用或禁用的ADC中断源。NewState:启用或禁用。

1.7、void ADC_ResetCalibration(ADC_TypeDef* ADCx)
功 能：重置所选ADC校准寄存器。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.8、FlagStatus ADC_GetResetCalibrationStatus(ADC_TypeDef* ADCx)
功 能：获取所选ADC重置校准寄存器状态。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.9、void ADC_StartCalibration(ADC_TypeDef* ADCx)
功 能：启动所选ADC校准过程。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.10、FlagStatus ADC_GetCalibrationStatus(ADC_TypeDef* ADCx)
功 能：获取所选ADC校准状态。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.11、void ADC_SoftwareStartConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：启用或禁用所选ADC软件启动转换。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.12、FlagStatus ADC_GetSoftwareStartConvStatus(ADC_TypeDef* ADCx)
功 能：获取所选ADC软件开始转换状态。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.13、void ADC_DiscModeChannelCountConfig(ADC_TypeDef* ADCx, uint8_t Number)
功 能：为所选ADC常规组通道配置不连续模式。
输 入：ADCx：其中x可以是1以选择ADC外围设备；Number：指定不连续模式常规通道计数值。

1.14、void ADC_DiscModeCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：为指定的ADC启用或禁用常规组通道上的不连续模式。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.15、void ADC_RegularChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)
功 能：为所选ADC常规通道配置其在序列器中的相应列组及其采样时间。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_Channel：要配置的ADC信道；Rank：常规组序列器中的等级；ADC_SampleTime：要为所选通道设置的采样时间值。

1.16、void ADC_ExternalTrigConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：通过外部触发器启用或禁用ADCx转换。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.17、uint16_t ADC_GetConversionValue(ADC_TypeDef* ADCx)
功 能：返回常规通道的最后一个ADCx转换结果数据。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.18、uint32_t ADC_GetDualModeConversionValue(void)
功 能：以双模式返回最后一个ADC1和ADC2转换结果数据。
输 入：无。

1.19、void ADC_AutoInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：启用或禁用所选ADC在常规转换后自动注入组转换。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.20、void ADC_InjectedDiscModeCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：为指定的ADC启用或禁用注入组通道的不连续模式。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.21、void ADC_ExternalTrigInjectedConvConfig(ADC_TypeDef* ADCx, uint32_t ADC_ExternalTrigInjecConv)
功 能：为注入通道转换配置ADCx外部触发器。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_ExternalTrigInjecConv：指定开始注入转换的ADC触发器。

1.22、void ADC_ExternalTrigInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：通过外部触发器启用或禁用ADCx注入通道转换。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.23、void ADC_SoftwareStartInjectedConvCmd(ADC_TypeDef* ADCx, FunctionalState NewState)
功 能：启用或禁用注入通道转换的所选ADC启动。
输 入：ADCx：其中x可以是1以选择ADC外围设备；NewState:启用或禁用。

1.24、FlagStatus ADC_GetSoftwareStartInjectedConvCmdStatus(ADC_TypeDef* ADCx)
功 能：获取所选ADC软件开始注入转换状态。
输 入：ADCx：其中x可以是1以选择ADC外围设备。

1.25、void ADC_InjectedChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)
功 能：为所选ADC注入通道配置其在序列器中的相应秩及其采样时间。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_Channel：要配置的ADC信道；Rank：注入组序列器中的秩；
ADC_SampleTime：要为所选通道设置的采样时间值。

1.26、void ADC_InjectedSequencerLengthConfig(ADC_TypeDef* ADCx, uint8_t Length)
功 能：配置注入通道的序列器长度。
输 入：ADCx：其中x可以是1以选择ADC外围设备；Length：序列器的长度。

1.27、void ADC_SetInjectedOffset(ADC_TypeDef* ADCx, uint8_t ADC_InjectedChannel, uint16_t Offset)
功 能：设置注入通道转换值偏移。
输 入：ADCx：其中x可以是1以选择ADC外围设备；Offset：所选ADC注入通道的偏移值。

1.28、uint16_t ADC_GetInjectedConversionValue(ADC_TypeDef* ADCx, uint8_t ADC_InjectedChannel)
功 能：ADC返回注入通道的结果。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_InjectedChannel：转换后的ADC注入通道。

1.29、void ADC_AnalogWatchdogCmd(ADC_TypeDef* ADCx, uint32_t ADC_AnalogWatchdog)
功 能：启用或禁用单个/所有常规或注入通道上的模拟看门狗。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_AnalogWatchdog：ADC模拟看门狗配置。

1.30、void ADC_AnalogWatchdogThresholdsConfig(ADC_TypeDef* ADCx, uint16_t HighThreshold,uint16_t LowThreshold)
功 能：配置模拟看门狗的高阈值和低阈值。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
HighThreshold：ADC模拟看门狗高阈值;LowThreshold：ADC模拟看门狗低阈值。

1.31、void ADC_AnalogWatchdogSingleChannelConfig(ADC_TypeDef* ADCx, uint8_t ADC_Channel)
功 能：配置模拟看门狗保护的单通道。
输 入：ADCx：其中x可以是1以选择ADC外围设备；
ADC_Channel：为模拟看门狗配置的ADC信道。

1.32、void ADC_TempSensorVrefintCmd(FunctionalState NewState)
功 能：启用或禁用温度传感器和Vrefint通道。
输 入：NewState:启用或禁用。

1.33、FlagStatus ADC_GetFlagStatus(ADC_TypeDef* ADCx, uint8_t ADC_FLAG)
功 能：检查是否设置了指定的ADC标志。
输 入：ADCx：其中x可以是1以选择ADC外围设备；ADC_FLAG：指定要检查的标志。

1.34、void ADC_ClearFlag(ADC_TypeDef* ADCx, uint8_t ADC_FLAG)
功 能：清除ADCx的挂起标志。
输 入：ADCx：其中x可以是1以选择ADC外围设备；ADC_FLAG：指定要清除的标志。

1.35、ITStatus ADC_GetITStatus(ADC_TypeDef* ADCx, uint16_t ADC_IT)
功 能：检查指定的ADC中断是否已发生。
输 入：ADCx：其中x可以是1以选择ADC外围设备；ADC_IT：指定要检查的ADC中断源。

1.36、void ADC_ClearITPendingBit(ADC_TypeDef* ADCx, uint16_t ADC_IT)
功 能：清除ADCx的中断挂起位。
输 入：ADCx：其中x可以是1以选择ADC外围设备；ADC_IT：指定要清除的ADC中断挂起位。

1.37、s32 TempSensor_Volt_To_Temper(s32 Value)
功 能：内部温度传感器电压与温度之间的关系。
输 入：Value：电压值。

以上函数均为库函数内部函数，在进行使用时只需在程序中进行调用即可。

# 2、硬件设计

本章教程通过ADC1通道1读取开发板VCC引脚和GND引脚ADC值，并通过串口调试助手打印显示出来。

# 3、软件设计

CH32V103C8T6的ADC_IN1在PC1和PA1引脚，本文使用PA1，通过PA1读取开发板VCC引脚ADC值和GND引脚ADC值，具体程序如下：
adc.h文件

```
1. #ifndef __ADC_H
2. #define __ADC_H
3.
```

```
#include "ch32v10x_conf.h"

void adc_Init(void);
u16 get_adc(u8 ch);

#endif
```

复制代码

adc.h文件主要是函数的声明.

adc.c文件

```
#include "adc.h"

void adc_Init(void)
{
    ADC_InitTypeDef ADC_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |RCC_APB2Periph_ADC1 , ENABLE ); //使能GPIOA时钟和ADC

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    RCC_ADCCLKConfig(RCC_PCLK2_Div6); //设置ADC时钟分频为6分频
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;   //配置ADC为独立模式
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;        //设置在单通道模式下执行转换
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;  //设置在单次模式下执行转换
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None; //设置转换不是由外部触发启动
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; //设置ADC数据右对齐
    ADC_InitStructure.ADC_NbrOfChannel = 1;              //顺序进行规则转换的ADC通道的数目
    ADC_Init(ADC1, &ADC_InitStructure);                  //根据ADC_InitStructure中指定的参数初始化ADC1寄存器

    ADC_Cmd(ADC1, ENABLE); //使能ADC1

    ADC_ResetCalibration(ADC1); //重置ADC1校准寄存器。

    while(ADC_GetResetCalibrationStatus(ADC1)); //等待复位校准结束

    ADC_StartCalibration(ADC1); //开启AD校准

```

```
    while(ADC_GetCalibrationStatus(ADC1));     //等待校准结束
}

u16 get_adc(u8 ch)
{
    ADC_RegularChannelConfig(ADC1, ch, 1, ADC_SampleTime_239Cycles5 );//为所选ADC常规通道配置其在序列器中的相应列组及其采样时间。

    ADC_SoftwareStartConvCmd(ADC1, ENABLE); //使能ADC1软件启动转换

    while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC )); //等待转换结束

    return ADC_GetConversionValue(ADC1); //返回常规通道的最后一个ADC1转换结果数据
}
```

复制代码

adc.c文件主要是对ADC的相关配置以及获取ADC_IN1的值，具体配置流程如下：

1、使能ADC及GPIOA

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |RCC_APB2Periph_ADC1 , ENABLE ); //使能GPIOA时钟和ADC
```

复制代码

2、初始化PA1引脚，设置GPIO模式为模拟输入模式

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```

复制代码

3、设置ADC分频及工作模式配置

```
    RCC_ADCCLKConfig(RCC_PCLK2_Div6); //设置ADC时钟分频为6分频
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;  //配置ADC为独立模式
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;       //设置在单通道模式下执行转换
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;  //设置在单次模式下执行转换
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None; //设置转换不是由外部触发启动
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; //设置ADC数据右对齐
```

```
7.    ADC_InitStructure.ADC_NbrOfChannel = 1;                //顺序进行规则转换的ADC通道的数目
8.    ADC_Init(ADC1, &ADC_InitStructure);                    //根据ADC_InitStructure中指定的参数初始化ADC1寄存器
```

复制代码

## 4、使能ADC及进行ADC校准

```
1.    ADC_Cmd(ADC1, ENABLE); //使能ADC1
2.
3.    ADC_ResetCalibration(ADC1); //重置ADC1校准寄存器。
4.
5.    while(ADC_GetResetCalibrationStatus(ADC1)); //等待复位校准结束
6.
7.    ADC_StartCalibration(ADC1); //开启AD校准
8.
9.    while(ADC_GetCalibrationStatus(ADC1));      //等待校准结束
```

复制代码

## 5、编写ADC值获取函数

```
1. u16 get_adc(u8 ch)
2. {
3.    ADC_RegularChannelConfig(ADC1, ch, 1, ADC_SampleTime_239Cycles5 );//为所选ADC常规通道配置其在序列器中的相应列组及其采样时间。
4.
5.    ADC_SoftwareStartConvCmd(ADC1, ENABLE); //使能ADC1软件启动转换
6.
7.    while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC )); //等待转换结束
8.
9.    return ADC_GetConversionValue(ADC1); //返回常规通道的最后一个ADC1转换结果数据
10. }
```

复制代码

通过以上配置，即可对VCC引脚及GND引脚ADC值进行读取。

## main.c文件

```
1. int main(void)
2. {
3.       u16 adc;
4.
5.        Delay_Init();
6.        USART_Printf_Init(115200);
7.        adc_Init();
8.
9.        printf("SystemClk:%d\r\n",SystemCoreClock);
```

```
10.
11.         while(1)
12.         {
13.             adc=get_adc(ADC_Channel_1);
14.             printf("adc_value:%d\r\n",adc);
15.             Delay_Ms(250);
16.         }
17. }
```

复制代码

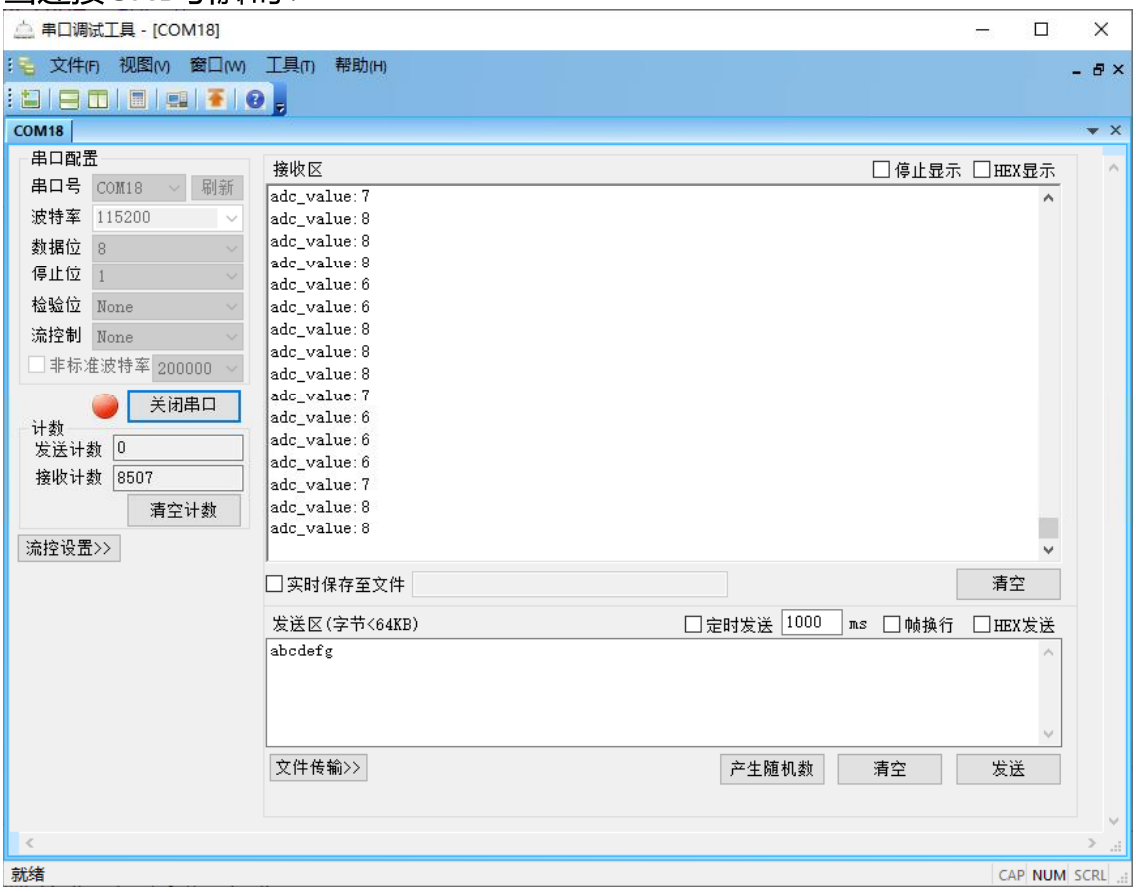
main.c文件主要进行相关函数的初始化以及打印输出通过ADC_IN1读取到的ADC值。

# 4、下载验证

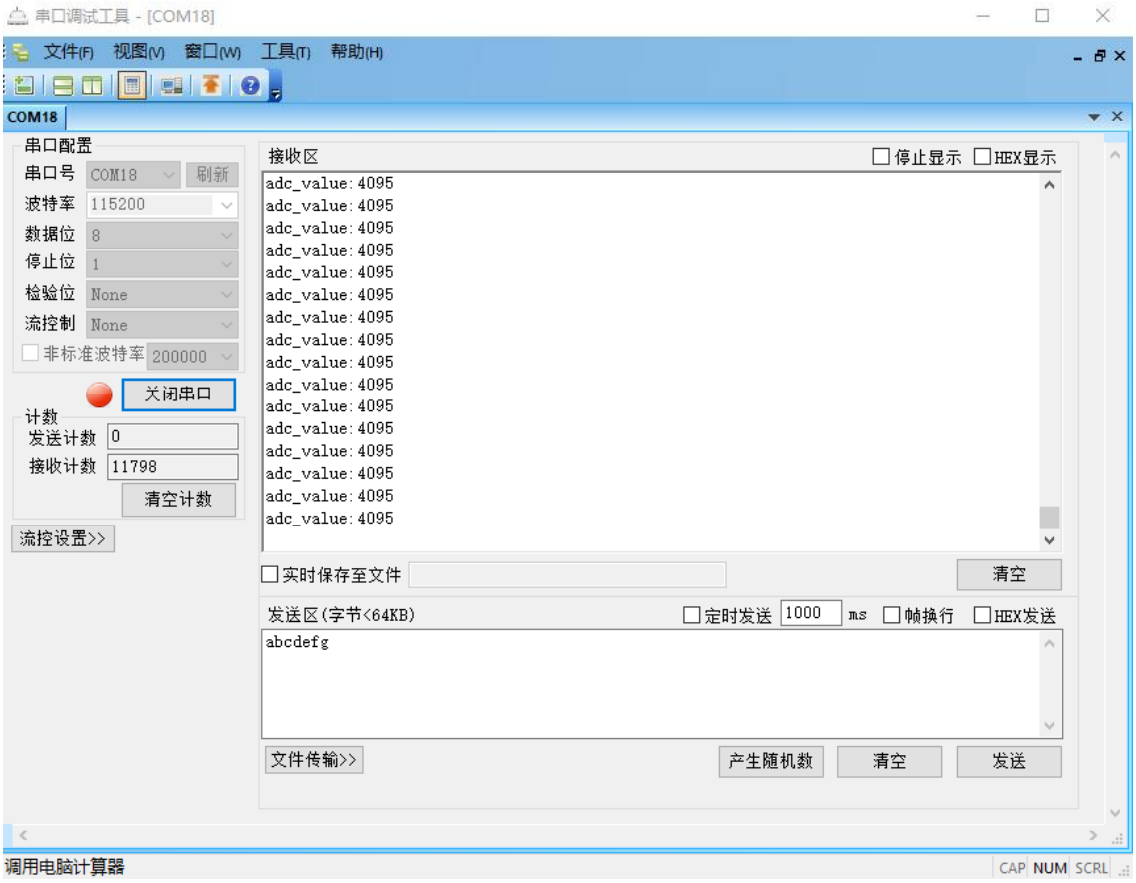
将编译好的程序下载到开发板并复位，用杜邦线将PA1引脚分别与VCC引脚和GND引脚连接，读取ADC值。注意，此处VCC引脚连接3.3V。

串口打印情况具体如下：

当连接GND引脚时：




当连接VCC（3.3V）引脚时：


串口调试工具 - [COM18]
文件(F) 视图(V) 窗口(W) 工具(T) 帮助(H)
COM18
串口配置
串口号 COM18 刷新
波特率 115200
数据位 8
停止位 1
检验位 None
流控制 None
非标准波特率 200000
关闭串口
计数
发送计数 0
接收计数 11798
清空计数
流控设置>>
接收区
停止显示 HEX显示
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
adc_value:4095
实时保存至文件
清空
发送区(字节<64KB)
定时发送 1000 ms 帧换行 HEX发送
abcdefg
文件传输>>
产生随机数
清空
发送
调用电脑计算器
CAP NUM SCRL