

# 零死角玩转STM32

与野火同行 乐意惬意无边



原创教程，完全开源。



由浅入深，结合实操。



通俗易懂，详尽解读。



配套板子，全面玩转。



强强联合，不断更新。



野火团队 Wild Fire Team



## 0、友情提示

《零死角玩转 STM32》系列教程由初级篇、中级篇、高级篇、系统篇、四个部分组成，根据野火 STM32 开发板旧版教程升级而来，且经过重新深入编写，重新排版，更适合初学者，步步为营，从入门到精通，从裸奔到系统，让您零死角玩转 STM32。M3 的世界，与野火同行，乐意惬意无边。

另外，野火团队历时一年精心打造的《STM32 库开发实战指南》将于今年 10 月份由机械工业出版社出版，该书的排版更适于纸质书本阅读以及更有利于查阅资料。内容上会给你带来更多的惊喜。是一本学习 STM32 必备的工具书。敬请期待！



### 3、Temperature（芯片温度）

#### 3.1 实验描述及工程文件清单

实验描述	串口 1(USART1)向电脑的超级终端以 1s 为时间间隔打印当前 STM32F103VET6 芯片内部的温度值。
硬件连接	温度传感器在芯片内部和 ADCx_IN16 输入通道相连接
用到的库文件	startup/start_stm32f10x_hd.c CMSIS/core_cm3.c CMSIS/system_stm32f10x.c FWlib/stm32f10x_gpio.c FWlib/stm32f10x_rcc.c FWlib/stm32f10x_usart.c FWlib/stm32f10x_adc.c FWlib/stm32f10x_dma.c FWlib/stm32f10x_flash.c
用户编写的文件	USER/main.c USER/stm32f10x_it.c USER/usart1.c USER/adc.c

#### 3.2 ADC 及内部温度传感器简介

STM32F103xC、STM32F103xD 和 STM32F103xE 增强型产品，内嵌 3 个 12 位的模拟/数字转换器(ADC)，每个 ADC 共用多达 21 个外部通道，可以实现单次或多次扫描转换。STM32 开发板用的是 STM32F103VET6，属于增强型的 CPU。它有 18 个通道，可测量 16 个外部和 2 个内部信号源，分别是 ADCx\_IN16（温度传感器）和 ADCx\_IN173(VREFINT)。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储

在 16 位数据寄存器中。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高/低阈值。

STM32 内部的温度传感器和 ADCx\_IN16 输入通道相连接，此通道把传感器输出的电压值转换成数字值。STM 内部的温度传感器支持的温度范围：-40 到 125 摄氏度。精度较差，误差为+（-）1.5 度左右，听起来有点蛋疼。

ADC 可以使用 DMA(direct memory access)方式操作。

本实验用的是 ADC1 的通道 16，采用 DMA 的方式操作。

内部温度传感器的基本操作步骤：（STM32 参考手册）

1. 选择 ADCx\_IN16 输入通道
2. 选择采样时间大于  $2.2 \mu s$  （推荐值为 17.1us）
3. 设置 ADC 控制寄存器 2(ADC\_CR2)的 TSVREFE 位，以唤醒关电模式下的温感器
4. 通过设置 ADON 位启动 ADC 转换(或用外部触发)
5. 读 ADC 数据寄存器上的 VSENSE 数据结果
6. 利用下列公式得出温度

$$\text{温度} (^{\circ} C) = \{ (V25 - V_{\text{SENSE}}) / \text{Avg\_Slope} \} + 25$$

式中 V25 是 VSENSE 在 25 摄氏度时的数值（典型值为 1.42V）

Avg\_Slope 是温度与 VSENSE 曲线的平均斜率（典型值为 4.3mV/C）

PS： 对于 12 位的 AD，3.3V 的 AD 值为 0xfff；1.42V 对应的 AD 值为：0x6E2；4.3mV 对应的 AD 值为：0x05（用系统自带计算器可轻易算得）这些是计算温度值的时候用得到的，也可以用其他方法计算。详情请参考 STM32 手册。

### 3.3 代码分析

首先要添加用的库文件，在工程文件夹下 Fwlib 下我们需添加以下库文件：

```
1. stm32f10x_gpio.c
2. stm32f10x_rcc.c
3. stm32f10x_usart.c
4. stm32f10x_adc.c
```



```
5. stm32f10x_dma.c
6. stm32f10x_flash.c
```

还要在 [stm32f10x\\_conf.h](#) 中将相应头文件的注释去掉：

```
1. /* Uncomment the line below to enable peripheral header file inclusion */
2. #include "stm32f10x_adc.h"
3. /* #include "stm32f10x_bkp.h" */
4. /* #include "stm32f10x_can.h" */
5. /* #include "stm32f10x_crc.h" */
6. /* #include "stm32f10x_dac.h" */
7. /* #include "stm32f10x_dbgmcu.h" */
8. #include "stm32f10x_dma.h"
9. /* #include "stm32f10x_exti.h" */
10. #include "stm32f10x_flash.h"
11. /* #include "stm32f10x_fsmc.h" */
12. #include "stm32f10x_gpio.h"
13. /* #include "stm32f10x_i2c.h" */
14. /* #include "stm32f10x_iwdg.h" */
15. /* #include "stm32f10x_pwr.h" */
16. #include "stm32f10x_rcc.h"
17. /* #include "stm32f10x_rtc.h" */
18. /* #include "stm32f10x_sdio.h" */
19. /* #include "stm32f10x_spi.h" */
20. /* #include "stm32f10x_tim.h" */
21. #include "stm32f10x_usart.h"
22. /* #include "stm32f10x_wwdg.h" */
23. /*#include "misc.h"*/ /* High level functions for NVIC and SysTick (add-
    on to CMSIS functions) */
```

配置好所需的库文件之后，我们就从 [main](#) 函数开始分析：

```
1. /**
2.  * @brief Main program.
3.  * @param None
4.  * @retval : None
5.  */
6.
7. int main(void)
8. {
9.     /* config the sysclock to 72M */
10.    SystemInit();
11.
12.    /* USART1 config */
13.    USART1_Config();
14.}
```



```

15.      /* enable adc1 and config adc1 to dma mode */
16.      Temp_ADC1_Init();
17.
18.      printf("\r\n Print current Temperature \r\n");
19.
20.      while (1)
21.      {
22.          ADC_tempValueLocal= ADC_ConvertedValue;    // 读取转换的 AD 值
23.          Delay(0xffffee);                          // 延时
24.          Current_Temp=(V25-ADC_tempValueLocal)/Avg_Slope+25;    //计算方法 2
25.
26.          printf("\r\n The current temperature = %02d C\r\n", Current_Temp); //10 进制示
27.      }

```

系统库函数 `SystemInit()`；将系统时钟设置为 72M，`USART1_Config()`；配置串口，关于这两个函数的具体讲解可以参考前面的教程，这里不再详述。

`Temp_ADC1_Init()`；函数使能了 **ADC1**，并使 **ADC1** 工作于 **DMA** 方式。

`Temp_ADC1_Init()`；这个函数是由我们用户在 `tempad.c` 文件中实现的应用程序：

```

1.  /*
2.   * 函数名:Temp_ADC1_Init();
3.   * 描述   : 无
4.   * 输入   : 无
5.   * 输出   : 无
6.   * 调用   : 外部调用
7.   */
8.  void Temp_ADC1_Init(void)
9.  {
10.     ADC1_GPIO_Config();
11.     ADC1_Mode_Config();
12. }

```

`Temp_ADC1_Init()`；调用了 `ADC1_GPIO_Config()`；和 `ADC1_Mode_Config()`；这两个函数的作用分别是配置好 **ADC** 和 **DMA** 的时钟、配置它的工作模式为 **MDA** 模式。

```

1.  /*
2.   * 函数名: ADC1_GPIO_Config
3.   * 描述   : 使能 ADC1 和 DMA1 的时钟
4.   * 输入   : 无
5.   * 输出   : 无
6.   * 调用   : 内部调用
7.   */
8.  static void ADC1_GPIO_Config(void)
9.  {
10.     GPIO_InitTypeDef GPIO_InitStructure;
11.     /* Enable DMA clock */
12.     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
13.
14.     /* Enable ADC1 and GPIOC clock */
15.     RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
16.
17. }

```

代码非常简单，大家就自己花点心思看看吧。这两个函数都用 **static** 关键字修饰了，都属于内部调用，我们只向其他用户提供了这个 `Temp_ADC1_Init()`；接口，使得更方便简洁。



```
1.  /* 函数名: ADC1_Mode_Config
2.   * 描述   : 配置 ADC1 的工作模式为 MDA 模式
3.   * 输入   : 无
4.   * 输出   : 无
5.   * 调用   : 内部调用
6.   */
7.  static void ADC1_Mode_Config(void)
8.  {
9.      DMA_InitTypeDef DMA_InitStructure;
10.     ADC_InitTypeDef ADC_InitStructure;
11.
12.     /* DMA channel1 configuration */
13.     DMA_DeInit(DMA1_Channel1);
14.     DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_Address;
15.     DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&ADC_ConvertedValue;
16.     DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
17.     DMA_InitStructure.DMA_BufferSize = 1;
18.     DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
19.     DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
20.     DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
21.
22.     DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
23.     DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
24.     DMA_InitStructure.DMA_Priority = DMA_Priority_High;
25.     DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
26.     DMA_Init(DMA1_Channel1, &DMA_InitStructure);
27.
28.     /* Enable DMA channel1 */
29.     DMA_Cmd(DMA1_Channel1, ENABLE);
30.
31.     /* ADC1 configuration */
32.     ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
33.     ADC_InitStructure.ADC_ScanConvMode = ENABLE;
34.     ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
35.     ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
36.     ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
37.     ADC_InitStructure.ADC_NbrOfChannel = 1;
38.     ADC_Init(ADC1, &ADC_InitStructure);
39.
40.     /* ADC1 regular channel16 configuration */
41.     ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 1, ADC_SampleTime_55Cycles5);
42.
43.     /*Enable TempSensorVrefintCmd*/
44.     ADC_TempSensorVrefintCmd(ENABLE);
45.
46.     /* Enable ADC1 DMA */
47.     ADC_DMAcmd(ADC1, ENABLE);
48.
49.     /* Enable ADC1 */
50.     ADC_Cmd(ADC1, ENABLE);
51.
52.     /* Enable ADC1 reset calibration register */
53.     ADC_ResetCalibration(ADC1);
54.     /* Check the end of ADC1 reset calibration register */
55.     while(ADC_GetResetCalibrationStatus(ADC1));
56.
57.     /* Start ADC1 calibration */
58.     ADC_StartCalibration(ADC1);
59.     /* Check the end of ADC1 calibration */
60.     while(ADC_GetCalibrationStatus(ADC1));
61.
62.     /* Start ADC1 Software Conversion */
63.     ADC_SoftwareStartConvCmd(ADC1, ENABLE);
64. }
```

Temp\_ADC1\_Init() 里面重要的一步是使能内部的温度传感器，因为内部的温度传感器默认关闭的。

现在我们来认识几个变量：

```
1.  // ADC1 转换的电压值通过 MDA 方式传到 flash
2.  extern  IO u16 ADC_ConvertedValue;
3.  
```

```
4. // 局部变量，用于存从 flash 读到的电压值
5. __IO u16 ADC_tempValueLocal;

6. //存放计算后的温度值

7. __IO u16 Current_Temp;

8.

9. //温度为 25 摄氏度时的电压值

10. __IO u16 V25=0x6E2;

11.

12. //温度、电压对应的的斜率 每摄氏度 4.35mV 对应每摄氏度 0x05

13. __IO u16 Avg_Slope=0x05;
```

ADC\_ConvertedValue 在 `adc.c` 中定义，ADC\_tempValueLocal 在 `main.c` 中定义，关于这两个变量的作用可看代码的描述。这里要注意一点的是，这两个变量都要用 `volatile` 关键字来修饰，为的是不让编译器优化这个变量，这样每次用到这两个变量时都要回到相应变量的内存中去取值，因为这两个变量的值随时都是可变的，而 `volatile` 字面意思就是“可变的，不确定的”。有关 `volatile` 关键字的详细用法大家可去查与 C 语言有关的书，这里推荐一个 C 语言的小册子《C 语言深度解剖-陈正冲编著》，里面对 `volatile` 这个关键字的讲解就非常好，还有其他有关 C 语言的知识也讲得非常好，挺值得大家看看。

主函数的最后以一个无限循环不断地往串口打印检测到芯片的内部温度值：

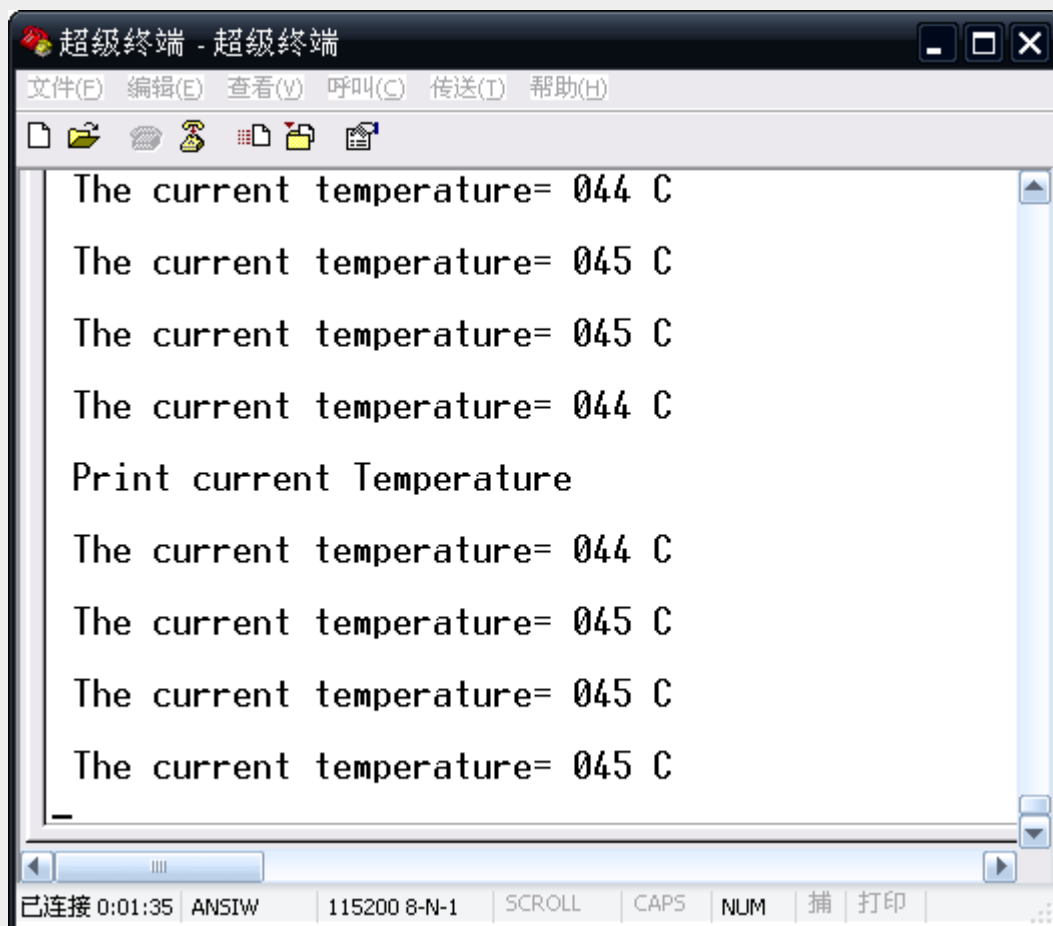
```
1. while (1)
2. {
3.     ADC_tempValueLocal = ADC_ConvertedValue; // 读取转换的 AD 值
4.     Delay(0xffffee); // 延时
5.     Current_Temp=(V25-ADC_tempValueLocal)/Avg_Slope+25; //温度转换计算
6.     printf("\r\n The current temperature = %03d C\r\n", Current_Temp); //10 进制
    显示
7. }
```

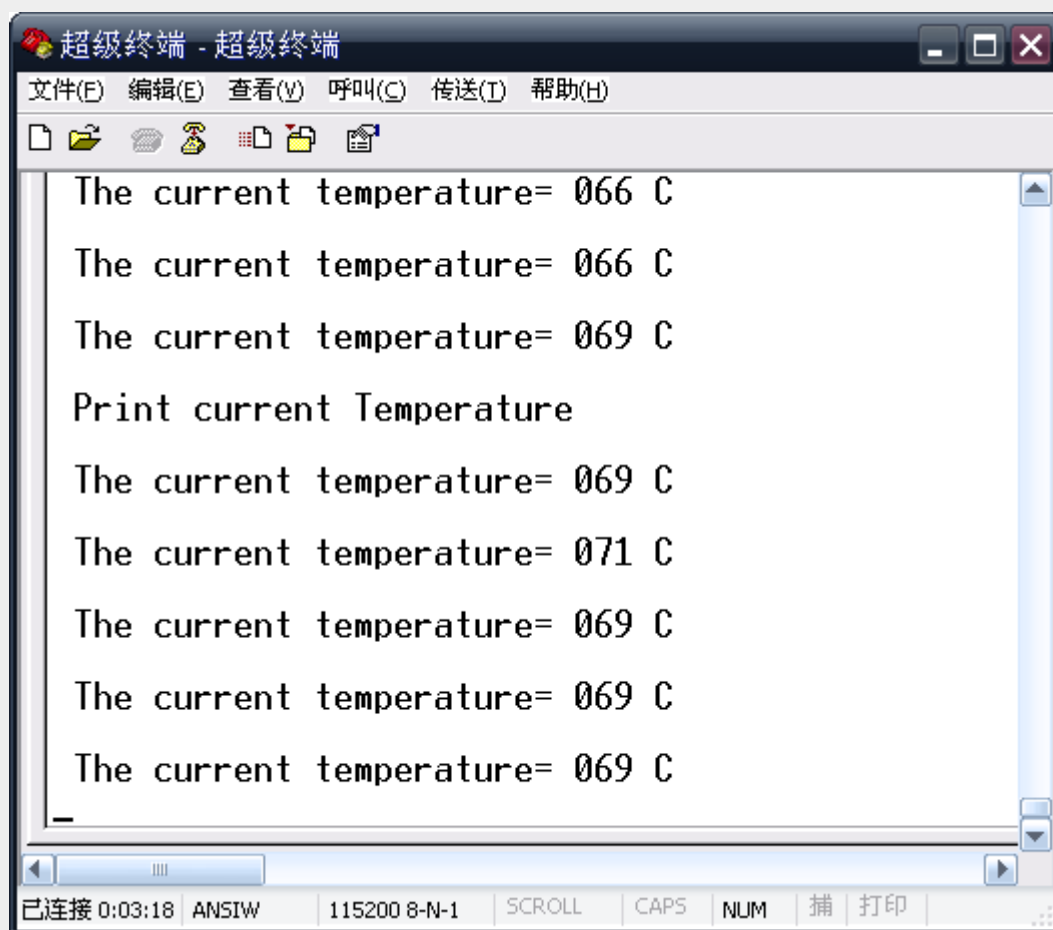




### 3.4 实验想象

将野火 STM32 开发板供电(DC5V)，插上 JLINK，插上串口线(两头都是母的交叉线)，打开超级终端，配置超级终端为 115200 8-N-1，将编译好的程序下载到开发板，即可看到超级终端打印出如下信息：（检测到的是芯片的内部温度）





```
超级终端 - 超级终端
文件(E) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
The current temperature= 066 C
The current temperature= 066 C
The current temperature= 069 C
Print current Temperature
The current temperature= 069 C
The current temperature= 071 C
The current temperature= 069 C
The current temperature= 069 C
The current temperature= 069 C
已连接 0:03:18 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

当用加热的电烙铁轻轻地放在我们的芯片上（或者用热风筒吹芯片）时会看到温度有明显的变化。