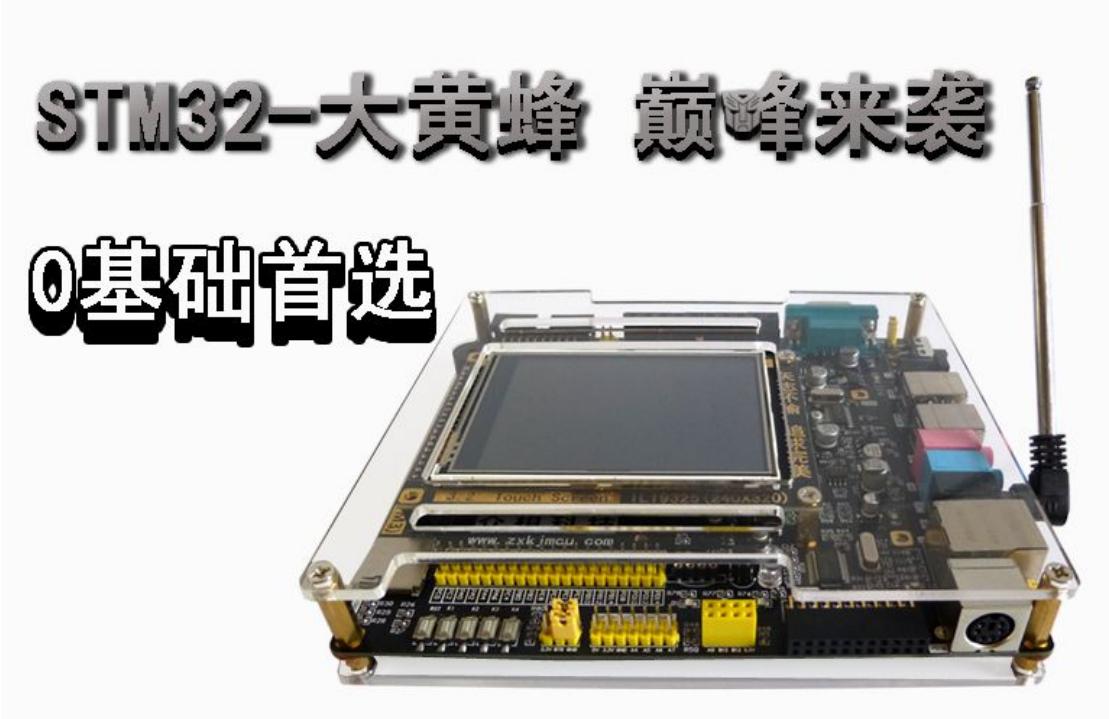
学 ARM 从 STM32 开始

STM32 开发板库函数教程--实战篇



官方网站：http://www.zxkjmcu.com
官方店铺：http://zxkjmcu.taobao.com
官方论坛：http://bbs.zxkjmcu.com
刘洋课堂：http://school.zxkjmcu.com

4.16 STM32 内部温度传感器工作原理及程序设计

4.16.1 温度传感器概述

4.16.1.1 温度传感器定义

传感器是一种能把物理量或化学量转变成便于利用的电信号的器件。国际电工委员会(IEC:International Electrotechnical Committee)的定义为：“传感器是测量系统中的一种前置部件，它将输入变量转换成可供测量的信号”。按照通用说法是:“传感器是包括承载体和电路连接的敏感元件”，而“传感器系统则是组合有某种信息处理(模拟或数字)能力的系统”。传感器是传感系统的一个组成部分，它是被测量信号输入的第一道关口。

4.16.2 STM32 温度传感器

4.16.2.1 STM32 内部温度传感器可以用来测量器件周围的温度(TA)。 温度传感器在内部和 ADC1_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器模拟输入推荐采样时间是 17.1μs。 当没有被使用时，传感器可以设置于关电模式。

注意： 必须设置 TSVREFE 位激活内部通道：ADC1_IN16(温度传感器)和 ADC1_IN17(VREFINT)的转换。

温度传感器输出电压随温度线性变化，由于生产过程的变化，温度变化曲线的偏移在不同芯片上会有不同(最多相差 45° C，所以 STM32 内部集成的温度传感器精度不高)。因而内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。下图是温度传感器的方框图。

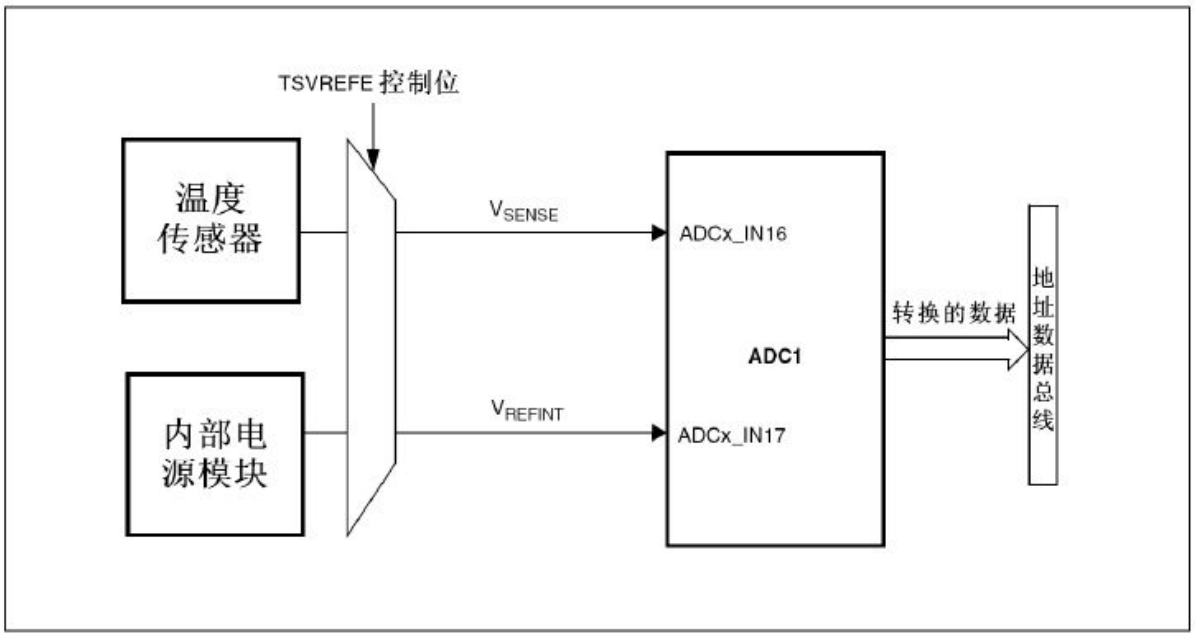


图 1 STM32 内部温度传感器方框图

4.16.2.2 读温度

为使用传感器,应按照下列步骤：

1. 选择 ADC1_IN16 输入通道

2. 选择采样时间为 17.1 μs

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度传感器

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

温度(° C) = {(V25 - VSENSE) / Avg_Slope} + 25

这里： V25 = VSENSE 在 25° C 时的数值

V25 最小=1.34V，典型=1.43，最大=1.52，单位=mV/° C

Avg_Slope = 温度与 VSENSE 曲线的平均斜率(单位为

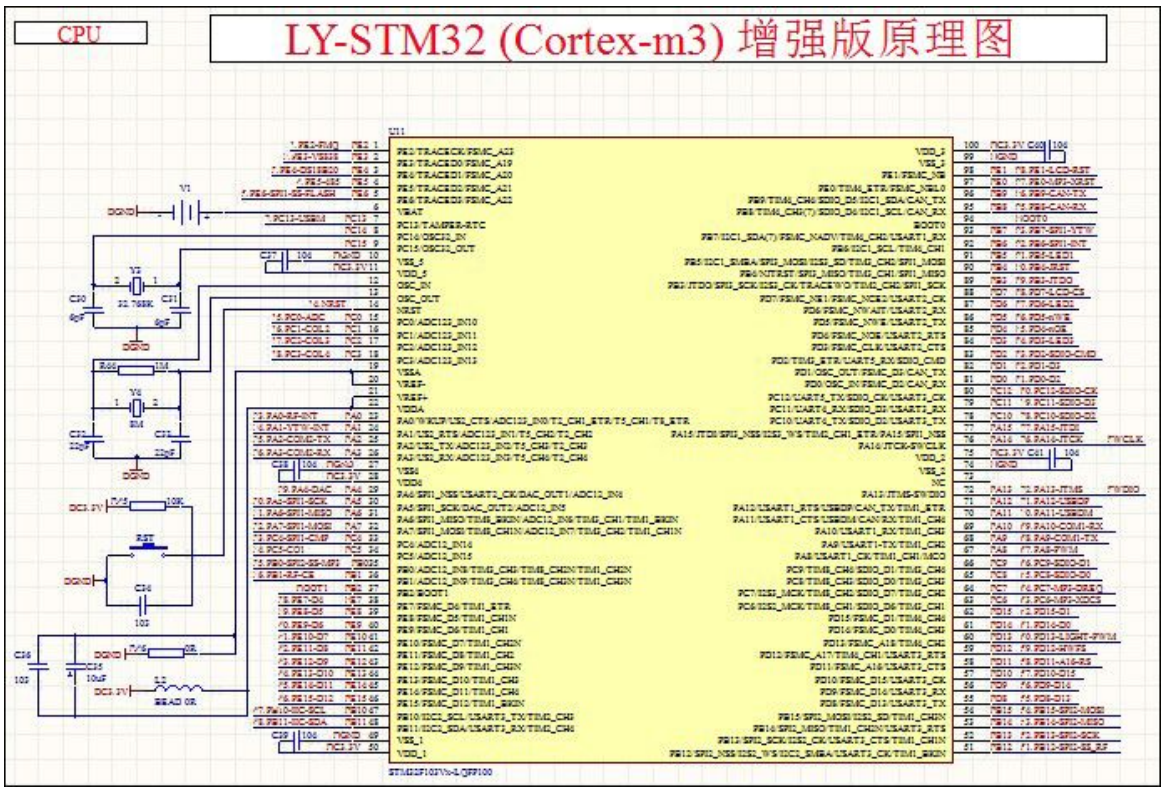
mV/° C 或 μV/° C)，最小=4.0、典型=4.3，最大=4.6，单位是 mV/° C

注意： 传感器从关电模式唤醒后到可以输出正确水平的 VSENSE 前，有一个建立时间。ADC 在上电后也有一个建立时间，因此为了缩短延时，应该同时设置 ADON 和 TSVREFE 位。

4.16.3 STM32 低功耗实验目的

编写 C 语言程序，通过模拟量采集，通过内部温度传感器把外部温度输出打印到屏幕。

4.16.4 硬件设计



4.16.5 温度传感器温度采集软件设计

4.16.5.1 STM32 库函数文件

```
stm32f10x_gpio.c
stm32f10x_rcc.c
Misc.c              // 中断控制字（优先级设置）库函数
stm32f10x_exti.c // 外部中断库处理函数
stm32f10x_tim.c  // 定时器库处理函数
```

```
stm32f10x_usart.c  // 串口通讯函数
stm32f10x_adc.c  // ADC 模拟量输入函数
```

本节实验及以后的实验我们都是用到库文件，其中 stm32f10x_gpio.h 头文件包含了 GPIO 端口的定义。stm32f10x_rcc.h 头文件包含了系统时钟配置函数以及相关的外设时钟使能函数，所以我们要把这两个头文件对应的 stm32f10x_gpio.c 和 stm32f10x_rcc.c 加到工程中；Misc.c 库函数主要包含了中断优先级的设置，stm32f10x_exti.c 库函数主要包含了外部中断设置参数，tm32f10x_tim.c 库函数主要包含定时器设置，tm32f10x_usart.c 库函数主要包含串行通讯设置，tm32f10x_adc.c 库函数主要包含模拟量输入相关设置，这些函数也要添加到函数库中。以上库文件包含了本次实验所有要用到的函数使用功能。

4.16.5.2 自定义头文件

```
pbdata.h
pbdata.c
```

同时我们自己也创建了两个公共的文件，这两个文件主要存放我们自定义的公共函数和全局变量，以方便以后每个功能模块之间传递参数。

4.16.5.3 pbdata.h 文件里的内容是

```
#ifndef _pbdata_H
#define _pbdata_H

#include "stm32f10x.h"
#include "misc.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_dma.h"
#include "stdio.h"

//定义变量
extern u8 dt;
```

```
//定义函数
void RCC_HSE_Configuration(void);
void delay(u32 nCount);
void delay_us(u32 nus);
void delay_ms(u16 nms);

#endif
```

语句 #ifndef、#endif 是为了防止 pbdata.h 文件被多个文件调用时出现错误提示。如果不加这两条语句，当两个文件同时调用 pbdata 文件时，会提示重复调用错误。

4.16.5.4 pbdata.c 文件里的内容是

```
#include "pbdata.h"

u8 dt=0;

void RCC_HSE_Configuration(void) //HSE 作为 PLL 时钟，PLL 作为 SYSCLK
{
   RCC_DeInit(); /*将外设 RCC 寄存器重设为缺省值 */
   RCC_HSEConfig(RCC_HSE_ON);    /*设置外部高速晶振（HSE） HSE 晶振打开(ON)*/

   if(RCC_WaitForHSEStartUp() == SUCCESS) {  /*等待 HSE 起振,  SUCCESS：HSE 晶振稳定且就绪*/

   RCC_HCLKConfig(RCC_SYSCLK_Div1);/*设置 AHB 时钟(HCLK)RCC_SYSCLK_Div1——AHB 时钟 = 系统时*/
   RCC_PCLK2Config(RCC_HCLK_Div1); /*设置高速 AHB 时钟(PCLK2)RCC_HCLK_Div1——APB2 时钟 = HCLK*/
   RCC_PCLK1Config(RCC_HCLK_Div2); /*设置低速 AHB 时钟(PCLK1)RCC_HCLK_Div2——APB1 时钟 = HCLK / 2*/

   RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);/*设置 PLL 时钟源及倍频系数*/
   RCC_PLLCmd(ENABLE);    /*使能 PLL */
   while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) ; /*检查指定的 RCC 标志位(PLL 准备好标志)设置与否*/

   RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);  /*设置系统时钟（SYSCLK） */
   while(RCC_GetSYSCLKSource() != 0x08);      /*0x08：PLL 作为系统时钟 */
 }
```

```
}

void delay(u32 nCount)
{
   for(;nCount!=0;nCount--);
}
/*******************************************************************************
* 名    称：delay_us(u32 nus)
* 功    能：微秒延时函数
* 入口参数：u32  nus
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void delay_us(u32 nus)
{
    u32 temp;
    SysTick->LOAD = 9*nus;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达

    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}

/*******************************************************************************
* 名    称：delay_ms(u16 nms)
* 功    能：毫秒延时函数
* 入口参数：u16 nms
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void delay_ms(u16 nms)
{
    //注意 delay_ms 函数输入范围是 1-1864
    //所以最大延时为 1.8 秒

    u32 temp;
    SysTick->LOAD = 9000*nms;
```

```
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}
```

4.16.6 STM32 系统时钟配置 SystemInit()

每个工程都必须在开始时配置并启动 STM32 系统时钟。

4.16.7 GPIO 引脚时钟使能

```
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);

      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
      RCC_ADCCLKConfig(RCC_PCLK2_Div6);//12M  最大 14M
```

本节实验用到了 PA 端口，所以要把 PA 的时钟打开；串口 1 时钟打开；因为要与外部芯片通讯，所以要打开功能复用时钟；要采集芯片外部温度，所以模拟量输入功能使能；还要 6 分频，使 ADC 采集频率不能超过 14M。

4.16.8 GPIO 管脚电平控制函数

在主程序中采用 while(1)循环语句，采用查询的方式等待 ADC 模拟量转换完毕，初始化完成以后要在主程序中采集模拟量，加入滤波、取平均值等措施然后转换送出打印。下面是 while(1)语句中详细的内容。

```
   while(1)
  {
    ad=0;
   for(i=0;i<50;i++)
    {
      ADC_SoftwareStartConvCmd(ADC1,ENABLE);
      while(!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));
```

```
        ad=ad+ADC_GetConversionValue(ADC1);
    }

     ad=ad/50;

     //printf("ad =%f\r\n",3.3/4095*ad);//实际电压值

     printf("温度 =%f\r\n",(1.43-3.3/4095*ad)/0.0043+25);

     delay_ms(1000);
     delay_ms(1000);
     delay_ms(1000);
  }
```

4.16.9 stm32f10x_it.c 文件里的内容是

在中断处理 stm32f10x_it.c 文件里中串口 1 子函数非空，进入中断处理函数后，打开串口 1，和外部设备联络好。

```
#include "stm32f10x_it.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#include "pbdata.h"

void NMI_Handler(void)
{
}

void USART1_IRQHandler(void)
{
   if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
   {
       USART_SendData(USART1,USART_ReceiveData(USART1));
       while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
   }
}
```

4.16.10 main.c 文件里的内容是

大家都知道 prinif 重定向是把需要显示的数据打印到显示器上。在这个试验中把温度采集的执行过程通过 prinif 重定向打印到串口精灵上。也

就是送到显示器显示出来，方便与观察实验结果。

```
#include "pbdata.h"

void RCC_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void USART_Configuration(void);
void ADC_Configuration(void);

int fputc(int ch,FILE *f)
{
    USART_SendData(USART1,(u8)ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return ch;
}

int main(void)
{
   u32 ad=0;
   u8 i=0;

   RCC_Configuration(); //系统时钟初始化
   GPIO_Configuration();//端口初始化
   USART_Configuration();
   NVIC_Configuration();
   ADC_Configuration();

   while(1)
   {
     ad=0;
     for(i=0;i<50;i++)
     {
       ADC_SoftwareStartConvCmd(ADC1,ENABLE);
       while(!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));
       ad=ad+ADC_GetConversionValue(ADC1);
     }

      ad=ad/50;
      printf("温度 =%f\r\n",(1.43-3.3/4095*ad)/0.0043+25);//根据计算公式编写

      delay_ms(1000);
      delay_ms(1000);
```

```
      delay_ms(1000);
   }
}

void RCC_Configuration(void)
{
    SystemInit();//72m
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);

   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);//温度传感器必须链接
到 ADC1 通道
   RCC_ADCCLKConfig(RCC_PCLK2_Div6);//12M  最大 14M

}

void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
   //LED
   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;//TX
   GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
   GPIO_Init(GPIOA,&GPIO_InitStructure);

   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;//RX
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA,&GPIO_InitStructure);

   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_0;//AD
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AIN;
   GPIO_Init(GPIOC,&GPIO_InitStructure);
}

void NVIC_Configuration(void)
{
   NVIC_InitTypeDef NVIC_InitStructure;

   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

   NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
```

```
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void USART_Configuration(void)
{
    USART_InitTypeDef  USART_InitStructure;

    USART_InitStructure.USART_BaudRate=9600;
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;
    USART_InitStructure.USART_StopBits=USART_StopBits_1;
    USART_InitStructure.USART_Parity=USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowControl_
None;
    USART_InitStructure.USART_Mode=USART_Mode_Rx|USART_Mode_Tx;

    USART_Init(USART1,&USART_InitStructure);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
    USART_Cmd(USART1,ENABLE);
    USART_ClearFlag(USART1,USART_FLAG_TC);
}

void ADC_Configuration(void)
{
    ADC_InitTypeDef ADC_InitStructure;

    ADC_InitStructure.ADC_Mode=ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode=DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode=DISABLE;
    ADC_InitStructure.ADC_ExternalTrigConv=ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign=ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel=1;

    ADC_Init(ADC1,&ADC_InitStructure);

    ADC_RegularChannelConfig(ADC1,ADC_Channel_16,1,ADC_SampleTime_239Cycles5)
;//温度传感器必须使用 16 通道，这是固定的
    ADC_TempSensorVrefintCmd(ENABLE);//温度传感器使能

    ADC_Cmd(ADC1,ENABLE);

    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
```

```
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));

    ADC_SoftwareStartConvCmd(ADC1,ENABLE);

}
```

4.16.11 应用函数

查 STM32 固件使用手册列出了这节课程主要的应用函数，方便大家学习。

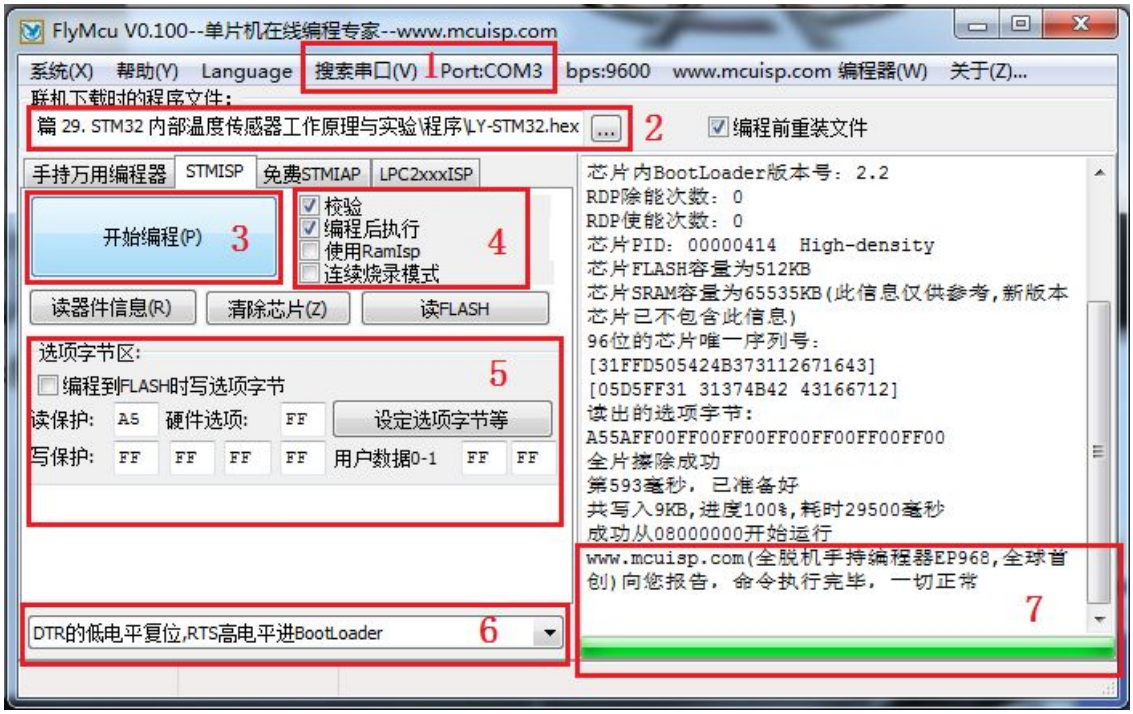
4.16.11.1 ADC_TampSensorVrefintCmd

表 2 描述了函数 ADC_TampSensorVrefintCmd

函数名称	ADC_TampSensorVrefintCmd
函数原型	void ADC_TempSensorVrefintCmd(FunctionalState NewState)
功能描述	使能或者失能温度传感器和内部参考电压通道
输入参数	NewState：温度传感器和内部参考电压通道的新状态 这个参数可以取：ENABLE 或者 DISABLE
输出参数	无
返回值	无
先决条件	无
被调用函数	无

4.16.12 程序下载

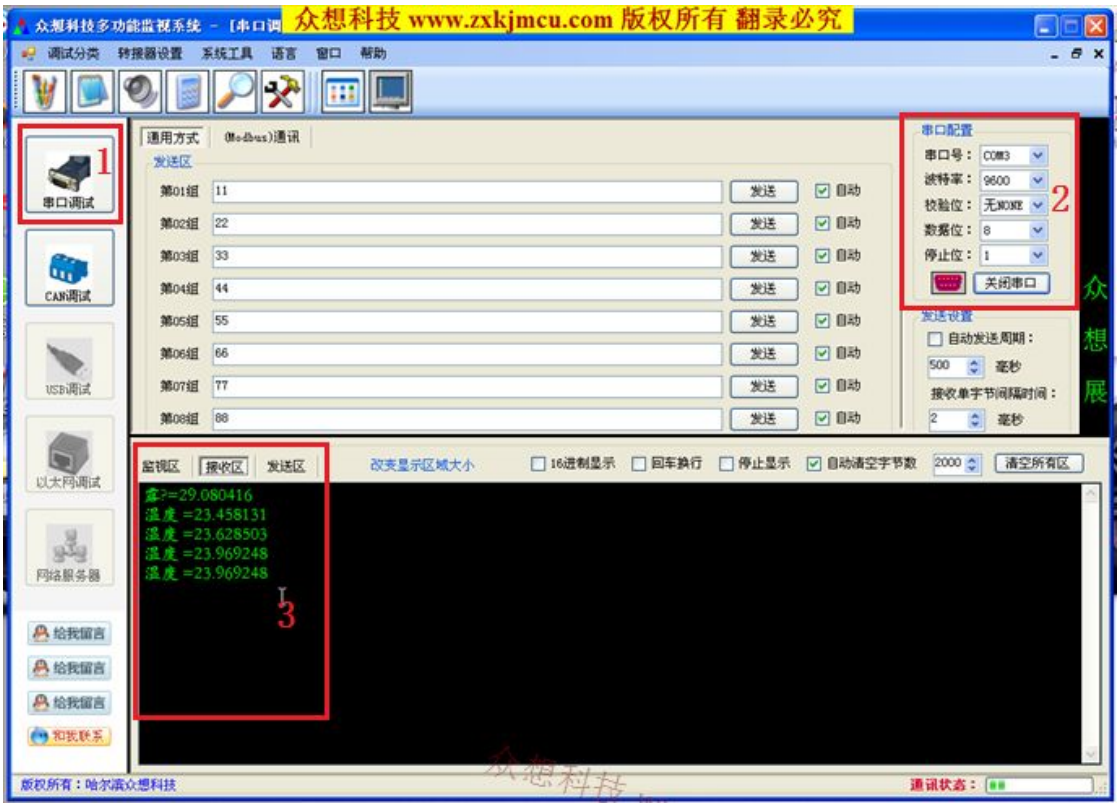
请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM3。（2）号区域请自己选择程序所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。



本节实验的源代码在光盘中：（LY-STM32 光盘资料\1.课程\1,基础篇\基础篇 29. STM32 内部温度传感器工作原理和实验\程序）

4.16.13 实验效果图

程序写入实验板后，使用公司开发的多功能监视系统，选择串口调试（1#框图区域），打开串口（2#框图区域），会在接收区（3#框图区域）有数据被接收到，显示“温度=23.89645”。请大家注意下面的实验截图。数据会间隔 3 秒不停的刷新出来。


众想科技 www.zxkjmcu.com 版权所有 翻录必究
众想科技多功能监视系统
调试分类
转接器设置
系统工具
语言
窗口
帮助
串口调试
CAN调试
USB调试
以太网调试
网络服务器
给我留言
给我留言
给我留言
和我联系
通用方式
发送区
第01组 11
第02组 22
第03组 33
第04组 44
第05组 55
第06组 66
第07组 77
第08组 88
发送
自动
串口配置
串口号：COM3
波特率：9600
校验位：无NONE
数据位：8
停止位：1
关闭串口
发送设置
自动发送周期：
500
毫秒
接收单字节间隔时间：
2
毫秒
监视区
接收区
发送区
改变显示区域大小
16进制显示
回车换行
停止显示
自动清空字节数
2000
清空所有区
温度 =29.080416
温度 =23.458131
温度 =23.628503
温度 =23.969248
温度 =23.969248
1
2
3
众想展
版权所有：哈尔滨众想科技
通讯状态：