计算机科学与技术学院

嵌入式系统实验报告

（十）

姓　　名：Banban

专　　业：计 算 机 科 学 与 技 术

班　　级：20 级

学　　号：

指 导 教 师：

2023 年 5 月 15 日

一、任务要求

1、 跑通并理解 ADC（查询）项目

2、 完成下列任务之一：

a) 任务 1：以 ADC 子板上的可调电位器作为信号来源，用 STM32 内置 ADC 采集转换，将数字量送至数码管显示

b) 任务 2：以 ADC 子板上的可调电位器作为信号来源，用 STM32 内置 ADC 采集转换，设定一个上限值和一个下限值，改变电位器，当采集值高于上限值，点亮一个红色 led，低于下限值，点亮一个黄色 led，位于下限和上线之间时，点亮一个绿色 led。

二、实验报告要求

1、 任务 1/2 中自编程序的源代码（加上注释）

2、 能说明软件仿真结果的截图、反映硬件电路连接和硬件验证结果的图片或视频。

三、实验过程

一. 任务一：跑通并理解 ADC（查询）项目

1. 代码

```
// main.c
// ------------------------------------------------------------
int main(void) {
    int ADC_ConvertedValue;
    float voltage;
    /* USART1 config */
    USART1_Config();
    /* enable adc1 and config adc1 to dma mode */
    ADC1_Init();
    printf("ADC1 转换结果\r\n");
    while (1) {
        /* 由于没有采用外部触发，所以使用软件触发 ADC 转换 */
```

```
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);
        /* 等待转换结束 */
        while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC))
            ;
        ADC_ConvertedValue = ADC_GetConversionValue(ADC1);
        voltage = (float)ADC_ConvertedValue / 4096 * 3.3;  // 读取转换的 AD 值
        printf("\r\n 当前电压数字量：0x%04X \r\n", ADC_ConvertedValue);
        printf("\r\n 当前电压模拟值：%f V \r\n", voltage);
        // Delay(0xffffee);          // 延时
    }
}

// adc.c
// ----------------------------------------------------------
static void ADC1_GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 | RCC_APB2Periph_GPIOC,
ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);  // PC1,输入时不用设置速率
}

// 配置 ADC1 的工作模式为 MDA 模式
static void ADC1_Mode_Config(void) {
    ADC_InitTypeDef ADC_InitStructure;
    /* ADC1 configuration */
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;  // 独立 ADC 模式
    ADC_InitStructure.ADC_ScanConvMode =  DISABLE;  // 禁止扫描模式，扫描模式
用于多通道采集
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;  // 单次转换模式
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;  //
不使用外部触发转换
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;  // 采集数据右对齐
    ADC_InitStructure.ADC_NbrOfChannel = 1;  // 要转换的通道数目 1
```

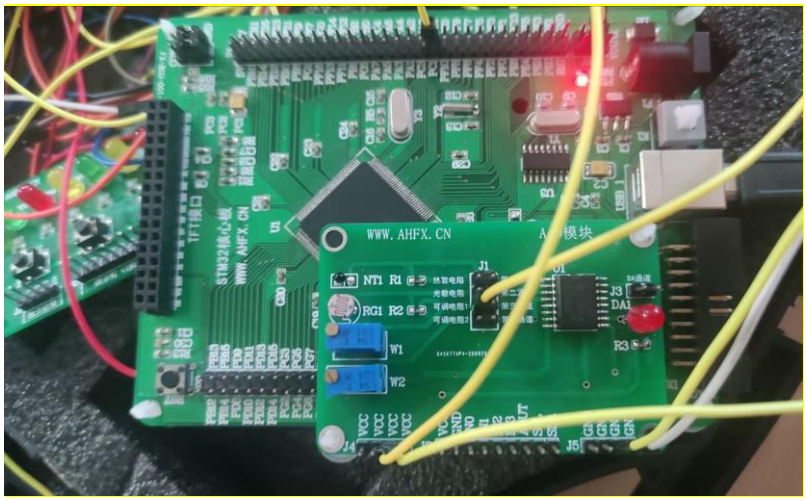
```
    ADC_Init(ADC1, &ADC_InitStructure);

    RCC_ADCCLKConfig(RCC_PCLK2_Div8); /*配置 ADC 时钟为 PCLK2 的 8 分频，即 9Hz*/
    /*配置 ADC1 的通道 11 为 55.  5 个采样周期，序列为 1 */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1,
ADC_SampleTime_55Cycles5);
    ADC_Cmd(ADC1, ENABLE);      /* Enable ADC1 */
    ADC_ResetCalibration(ADC1); /*复位校准寄存器 */
    while (ADC_GetResetCalibrationStatus(ADC1));  /*等待校准寄存器复位完成 */
    ADC_StartCalibration(ADC1); /* ADC 校准 */
    while (ADC_GetCalibrationStatus(ADC1)); /* 等待校准完成*/
}
```

2. 图片效果

原始

顺时针扭动



二. 任务二：以 ADC 子板上的可调电位器作为信号来源，用 STM32 内置 ADC 采集转换，将数字量送至数码管显示

1. 代码

```
// main.c
// ------------------------------------------------------------
int main(void) {
    int ADC_ConvertedValue;
    float voltage;
    int i, j;
    uint8_t a[8];
    int b[8] = {GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_2, GPIO_Pin_3, GPIO_Pin_4,
                GPIO_Pin_5, GPIO_Pin_6, GPIO_Pin_7};  // 数码管位选端口
    led_duan_init();
    led_wei_init();
    LED_GPIO_Config();
    /* USART1 config */
    USART1_Config();
    /* enable adc1 and config adc1 to dma mode */
    ADC1_Init();
    printf("ADC1 转换结果\r\n");
    while (1) {
        /* 由于没有采用外部触发，所以使用软件触发 ADC 转换 */
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);
```

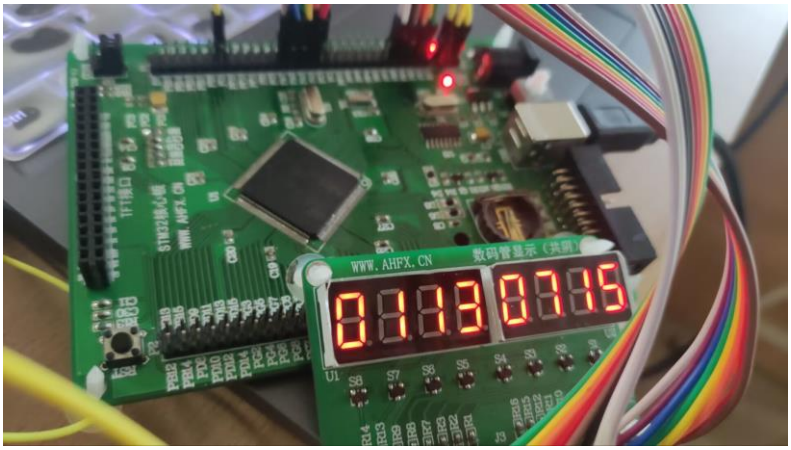
```
        /* 等待转换结束 */
        while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
        ADC_ConvertedValue = ADC_GetConversionValue(ADC1);
        voltage = (float)ADC_ConvertedValue / 4096 * 3.3;  // 读取转换的 AD 值
        a[0] = (int)voltage;
        a[1] = 10;
        a[2] = ((int)(voltage * 10)) % 10;
        a[3] = ((int)(voltage * 100)) % 10;
        a[4] = ((int)(voltage * 1000)) % 10;
        a[5] = ((int)(voltage * 10000)) % 10;
        a[6] = ((int)(voltage * 100000)) % 10;
        a[7] = ((int)(voltage * 1000000)) % 10;
        for (j = 0; j < 1000; j++) {
            for (i = 0; i < 8; i++) {
                Display(a[i]);                // 写入段选端口
                GPIO_ResetBits(GPIOD, b[i]);  // 第 i 位亮
                Delay(0x0000FF);              // 短暂延时
                GPIO_SetBits(GPIOD, b[i]);    // 第 i 位灭
            }
            i = 0;
        }
        printf("\r\n 当前电压数字量：0x%04X \r\n", ADC_ConvertedValue);
        printf("\r\n 当前电压模拟值：%f V \r\n", voltage);
    }
}

// smg.c
// -----------------------------------------------------------
#include "smg.h"
u8 seg_tab[11] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80,
0x90, 0x7F};
void led_duan_init(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);  // 使能 PA
    // 初始化 PC0~PC7(控制数码管 LED)
```

```
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  // 推挽输出
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
                                  GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 |
                                  GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);

    GPIO_SetBits(GPIOE, GPIO_Pin_All);  // 默认输出高电平
}
void led_wei_init(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);  // 使能 PE 端口
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  // 推挽输出
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |
                                  GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 |
                                  GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
    GPIO_SetBits(GPIOD, GPIO_Pin_All);  // 默认输出高电平，各位置不亮
}
void Display(u8 index) {
    GPIO_Write(GPIOE, seg_tab[index]);  // 将译码写到 GPIOA 的端口
}
```

2. 图片效果



四、总结与分析

本次实验我们使用 STM32 的 USART1 模块，通过串口向电脑的超级终端输出当前 ADC1 的转换电压值。我们需要了解 USART1 模块的原理和使用方法，同时需要了解如何配置 ADC1 模块并读取其转换值。首先，我们需要对 USART1 模块进行初始化。我们需要配置 USART1 的时钟源，设置波特率、数据位、校验位和停止位等参数，以及使能 USART1 的发送功能。

我们需要配置 ADC1 模块并读取它的转换值。我们需要使能 ADC1 时钟和 DMA 传输，以及配置 ADC1 的采样时间、分辨率和转换模式。通过使用 DMA 传输，我们能够实现 ADC1 的单次和连续转换，并将转换值存储到指定的存储器区域。

最后，我们需要注意一些细节问题，例如 USART1 的发送和接收状态、电压值的数据类型和精度、电路连线的正确性等。我们需要仔细检查电路连线是否正确，以及 USART1 输出的电压值是否正确，并在需要时进行调试和修正。

通过本次实验，我不仅了解了如何配置 USART1 和 ADC1 模块，并进行数据的读取和处理，还巩固了 C 语言的基本知识，并加深了我对嵌入式系统和实时应用的理解。我相信这些经验和技巧对我的未来学习和工作都将非常有用。