《微机原理与接口技术》

实验指导书

实验四 中断和 ADC

“微机原理与接口技术”课程教学团队

北京航空航天大学

仪器科学与光电工程学院

2022 年 11 月

一、实验目的

1.掌握外部中断原理，完成相应实验测试
2.掌握 ADC 的配置和使用，完成芯片内部温度的测量

二、实验设备

1.PC 计算机；
2.Keil for ARM(MDK)开发环境 V5.36 及以上；
3.NUCLEO 或 Disco 实验平台，主要查看相应的开发板电路图。

三、实验内容

1. 外部中断，将用户按键配置为外部中断，按键按下，产生中断，执行相应操作。

2. ADC 实验，芯片内部有测温传感器，通过 ADC 将内部温度测量量转化为数字量，再通过公式计算出真实温度。

3、拓展实验，自由发挥。

四、实验步骤

4.1 外部中断

IO 口外部中断的一般步骤：
1）使能 IO 口时钟。
2） 调用函数 HAL_GPIO_Init 设置 IO 口模式，触发条件，使能 SYSCFG 时钟以及设置 IO
口与中断线的映射关系。
3）配置中断优先级（ NVIC），并使能中断。
4）在中断服务函数中调用外部中断共用入口函数 HAL_GPIO_EXTI_IRQHandler。
5）编写外部中断回调函数 HAL_GPIO_EXTI_Callback 实现控制逻辑。

用户按键连接在 PC13，下述代码功能是检测用户按键是否按下，如果按下则系统调用回调函数完成相应功能。

1. 配置中断初始化

```
void EXTI15_10_IRQHandler_Config(void)
{
  GPIO_InitTypeDef     GPIO_InitStructure;
```

```
    /* Enable GPIOC clock */
    __HAL_RCC_GPIOC_CLK_ENABLE();

    /* Configure PC.13 pin as the EXTI input event line in interrupt mode for both CPU1 and CPU2*/
    GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;      /* current CPU (CM7) config in IT rising */
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Pin = BUTTON_USER_PIN;
    HAL_GPIO_Init(BUTTON_USER_GPIO_PORT, &GPIO_InitStructure);

    /* Enable and set EXTI lines 15 to 10 Interrupt to the lowest priority */
    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 2, 0); //抢占优先级为 2，子优先级为 0
    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);              //使能中断线

    /* Configure the second CPU (CM4) EXTI line for IT*/
    HAL_EXTI_D2_EventInputConfig(EXTI_LINE13 , EXTI_MODE_IT,   ENABLE);
}

//中断服务函数
void EXTI15_10_IRQHandler(void)
{
    HAL_GPIO_EXTI_IRQHandler(BUTTON_USER_PIN); //调用中断处理公用函数
}
```

2. 编写中断处理回调函数，在函数里编写产生中断后执行的代码，比如翻转 LED。

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == BUTTON_USER_PIN)
    {
        //按键按下，产生中断，调用回调函数，编写相应的代码
    }
}
```

3. 主函数程序

主函数中调用 EXTI15_10_IRQHandler_Config 即可。完成回调函数编写（自由发挥），观察中断现象。

4.2 ADC 实验

1. 初始化 ADC

```
ADC_HandleTypeDef hadc3;
```

```
static void MX_ADC3_Init(void)
{
  ADC_ChannelConfTypeDef sConfig = {0};

  hadc3.Instance = ADC3;
  hadc3.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV6;          //6 分频，
  hadc3.Init.Resolution = ADC_RESOLUTION_16B;                //16 位分辨率
  hadc3.Init.ScanConvMode = ADC_SCAN_DISABLE;                // 非扫描模式
  hadc3.Init.EOCSelection = ADC_EOC_SINGLE_CONV;             //关闭 EOC 中断
  hadc3.Init.LowPowerAutoWait = DISABLE;                     // 自动低功耗关闭
  hadc3.Init.ContinuousConvMode = DISABLE;                   //开启单次转换
  hadc3.Init.NbrOfConversion = 1;
 //1 个转换在规则序列中 也就是只转换规则序列 1
  hadc3.Init.DiscontinuousConvMode = DISABLE;                //禁止不连续采样模式
  hadc3.Init.ExternalTrigConv = ADC_SOFTWARE_START;          //软件触发
  hadc3.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;//使用软件触发
  hadc3.Init.ConversionDataManagement = ADC_CONVERSIONDATA_DR;  //规则通道的数据仅仅保存在 DR 寄存器里面
  hadc3.Init.Overrun = ADC_OVR_DATA_PRESERVED;
//保存旧数据
  hadc3.Init.LeftBitShift = ADC_LEFTBITSHIFT_NONE;

  hadc3.Init.OversamplingMode = DISABLE;                     //过采样关闭


  if (HAL_ADC_Init(&hadc3) != HAL_OK)
  {
    Error_Handler();
  }
  sConfig.Channel = ADC_CHANNEL_TEMPSENSOR;    //通道，内部测温通道
  sConfig.Rank = ADC_REGULAR_RANK_1;                 //1 个序列
  sConfig.SamplingTime = ADC_SAMPLETIME_810CYCLES_5;//采样时间
  sConfig.SingleDiff = ADC_SINGLE_ENDED;           //单边采集
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc3, &sConfig) != HAL_OK)
```

```
    {
      Error_Handler();
    }
  }

  void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
  {
      __HAL_RCC_ADC3_CLK_ENABLE();              //使能 ADC3 时钟
        __HAL_RCC_ADC_CONFIG(RCC_ADCCLKSOURCE_CLKP); //ADC 外设时钟选择

  }
```

2. 主函数程序

```
      MX_ADC3_Init();

      HAL_ADCEx_Calibration_Start(&hadc3,ADC_CALIB_OFFSET,ADC_SINGLE_ENDED);// ADC 校准
  int adc_v;
    while (1)
    {
          HAL_ADC_Start(&hadc3);
          adc_v = HAL_ADC_GetValue(&hadc3);
          //在这编写计算真实温度代码，具体公式下面有写！
          HAL_Delay(1000);
    }
```

3. 计算真实温度

上述循环中 adc_v 中的值并不是真实温度值，芯片内部存储着 30 度和 110 度时的校准值：

```
      ts_cal1=*(volatile unsigned short*)(0X1FF1E820);
      ts_cal2=*(volatile unsigned short*)(0X1FF1E840);
```

根据公式可求得真实的温度值

$$T(^{\circ}C)=\frac{110-30}{ts\_cal2-ts\_cal1}*(adc\_v-ts\_cal1)+30$$

上式中：

ts_cal1 是温度传感器在 30℃时的校准值，固定保存在芯片内部的： 0X1FF1 E820 ~0X1FF1 E821 这两个地址（ 16 位）。

ts_cal2 是温度传感器在 110℃时的校准值，固定保存在芯片内部的： 0X1FF1 E840 ~0X1FF1 E841 这两个地址（ 16 位）。

根据上述公式，完成 while 循环的代码，计算出真实温度。完成仿真中的测试如图：

main	0x080049C0	int f()
timeout	0x0000FFFF	auto - int
temp	0.0192771088	auto - float
temperate	28.4963856	auto - float
ts_cal1	0x00003043	auto - int
ts_cal2	0x00004079	auto - int
adc_v	0x00002FF5	auto - int
tmpreg	<not in scope>	auto - uint

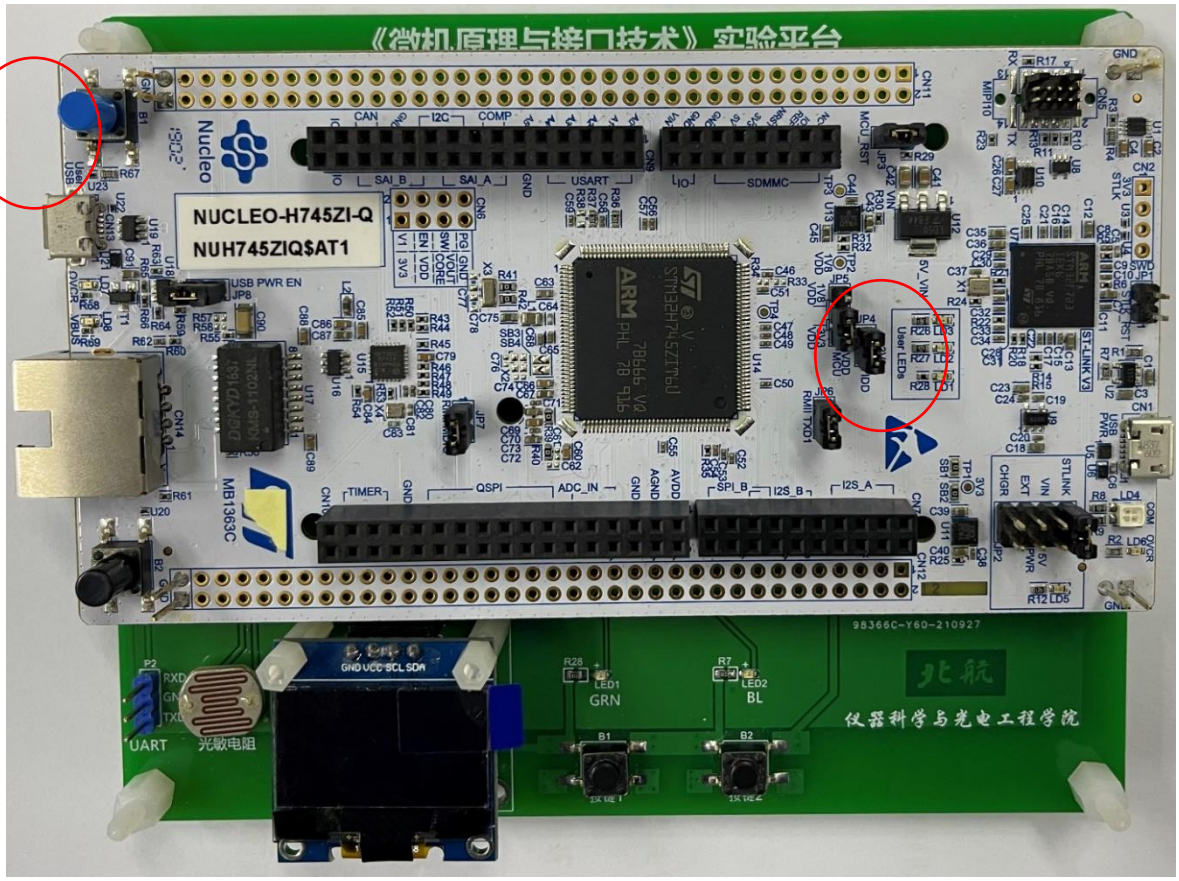
提示：如果想观测温度变化，可拿风扇吹 CPU，再进行仿真会发现温度降低。不要拿火烤 CPU 来观测升温现象!!

4.5 调试程序

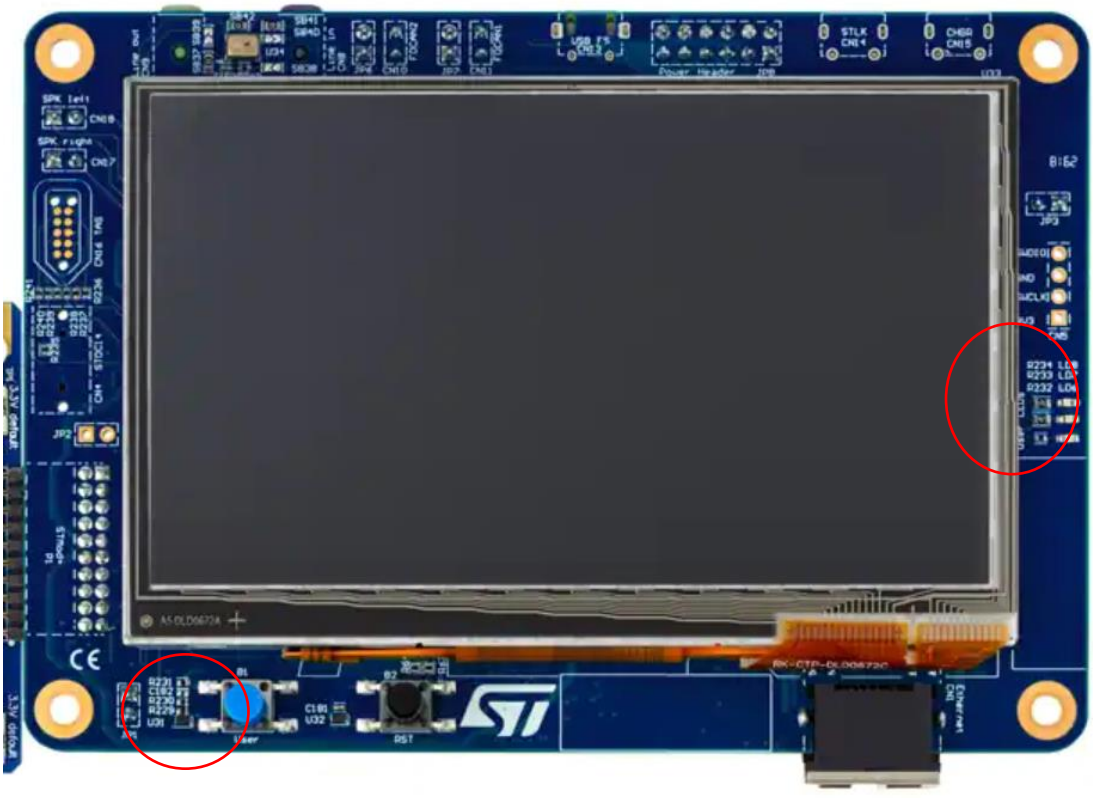
调试程序，并完成实验

五、参考程序

见之前模板。



实验平台实物照片（Nucleo）



实验平台实物照片（Disco 版）