北京航空航天大學
BEIHANG UNIVERSITY

# 《单片机系统实验》报告一
# ——I/O口及定时器实验

北京航空航天大学研究生院

二〇二五年十月

# 目　录

# 第一章　实验目的

了解32位微控制器（STM32系列）原理及其应用，熟悉单片机的资源，掌握微控制器的最小系统设计及扩展技术，掌握其编程语言。

# 第二章　实验设备

STM32实验系统一套，个人笔记本（Windows操作系统，带USB接口）一台。

# 第三章　实验原理

STM32的GPIO口控制4个发光二极管，了解其硬件连接方式，学会使用STM32的一个定时器，掌握对定时器计时方式的编程。编写程序循环点亮4个发光二极管，控制点亮时间为1秒钟闪烁。

## 3.1　感知执行模块

“感知执行模块”是指系统种的传感器模块和执行器模块。传感器模块就是利用传感器等感知元件来感知环境变量，例如温度、光照度、加速度、颜色等；而执行器模块则是利用声光报警器、电机等执行器来执行一些特定动作。

感知执行模块的主体构造是一致的，处理器采用STM32系列的单片机，只是传感器和执行器的选择不同，有的感知执行元件的安装方式可能也会不同。

图1以“温湿度传感器”为例展示了感知执行模块的主体结构，顶部蓝色元器件为感知元件——温湿度传感器。通常，感知执行模块的感知或执行元件均安装或插接在顶部，传感器或执行器的种类不同，外观和连接方式会有差别。


**图1　感知执行模块正面布局**

模块的底部有两个接口，如图2所示。其中USB3.0调试烧写口，可用来给模块供电、下载程序、在线调试及串口通讯等；RJ11-485通信接口，用来与其他模块进行

485通信。

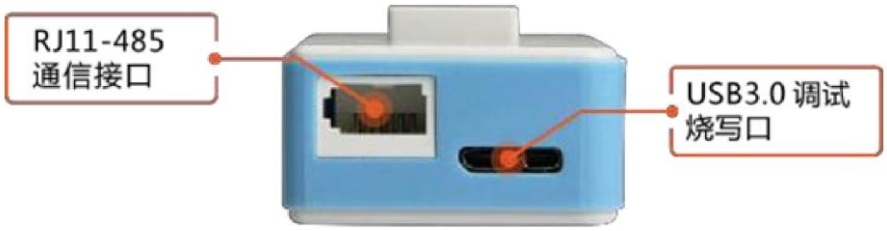



**图2　模块底部接口**

传感器模块接口用来插接传感器模块，而无线节点模块接口用来插接无线节点模块，此次实验中没有用到无线节点。感知执行模块与主控模块的连接如图3所示。

**图3　主控模块与感知执行模块、无线模块连接示意图**

## 3.2　STM调试下载器

STM下载器用于给控制芯片为STM32芯片的节点下载或调试程序。


**图4　ST-LINK调试器**

从顶部看，ST-LINK调试器的接口如下图所示：

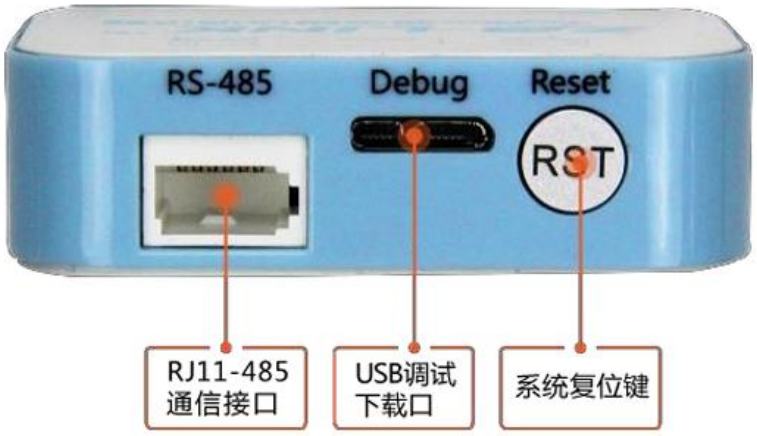



**图5　ST-LINK调试器接口示意图（顶部）**

（1）RJ11-485通信接口：连接其他模块进行485通信；

（2）USB 调试烧写口：连接感知执行模块，以对其进行调试或串口监测；系统复位键：用于系统复位。

从底部看，ST-LINK调试器的接口如下图所示：

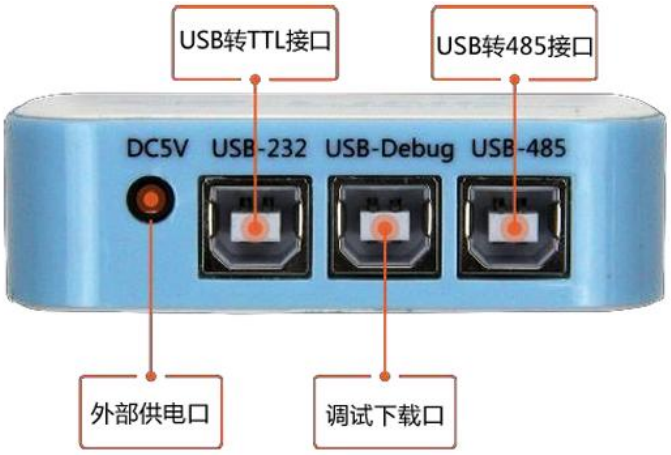



**图6　ST-LINK调试器接口示意图（底部）**

（1）外部供电口：提供直流5V电源；

（2）USB转TTL接口：PC机与感知执行模块进行串口通讯时，该接口通过USB A口转B口线与PC机相连；

（3）调试下载口：在调试感知执行模块或为其下载程序时，该接口通过USB A口转B口线与PC机相连；

（4）USB转485接口：PC机与感知执行模块进行485通讯时，该接口通过USB A口转B口线与PC机相连。

# 第四章　实验内容与步骤

## 4.1　实验内容

选择一个传感器或控制器进行相关信号的采集或控制，能在调试环境中观察到相关实验数据的变化或在

（1）学会使用Keil编译链接调试环境，熟悉有关STM32使用到的库，并能顺利建立包含各种库文件的工程。（1学时）

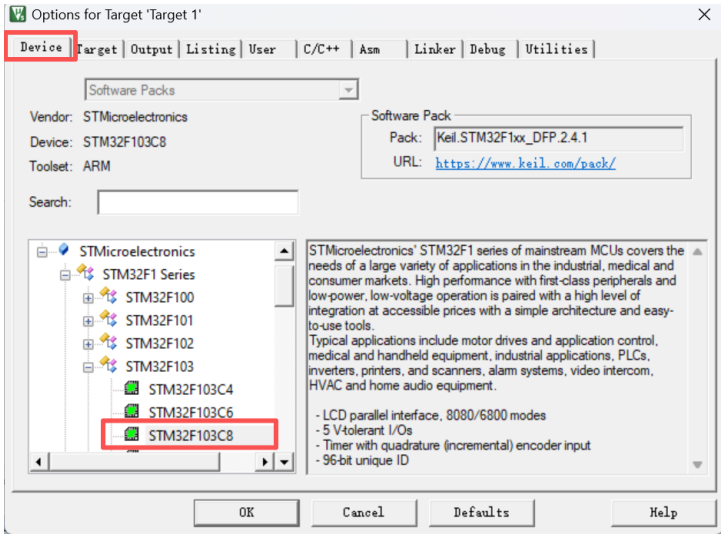
（2）I/O口实验：在建立工程的基础上能点亮发光二极管。（1学时）

（3）定时器实验：循环定时点亮4个灯，即每1秒闪烁点亮一个灯，循环往复（或叫跑马灯实验）。（2学时）

## 4.2　实验步骤

（1）打开工程文件：在Project文件夹中，双击打开绿色图标的.uvprojx工程文件 实验例程.uvprojx，进入Keil开发环境。

（2）检查芯片与调试驱动：点击工具栏中的Option for Target，在弹出的选项中，在Device选项卡中选择芯片型号为STM32F103C8，在Debug选项卡中选择调试驱动为ST-Link Dubugger，点击在Settings按钮，确认ARM核心是否成功连接，如图7所示。



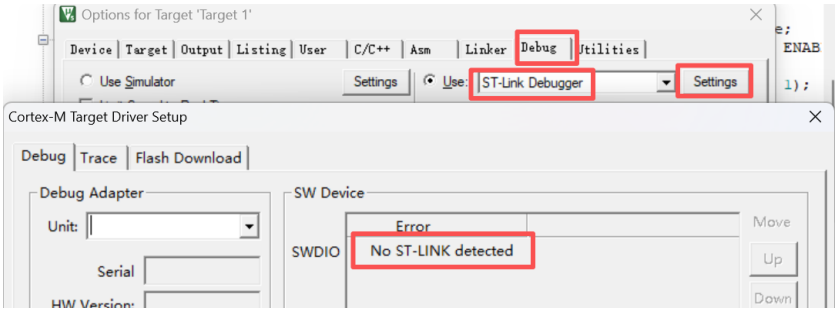



**图7　芯片与调试器设置**

（3）在文件组中加入与 I/O 口相关的库文件：在左侧的工程管理器中，右键Libraries文件组，选择Add Existing Files to Group ‘Libraries’，在src文件夹中选择stm32f10x_gpio.c并添加，在inc文件夹中选择stm32f10x_gpio.h并添加，如图8所示。

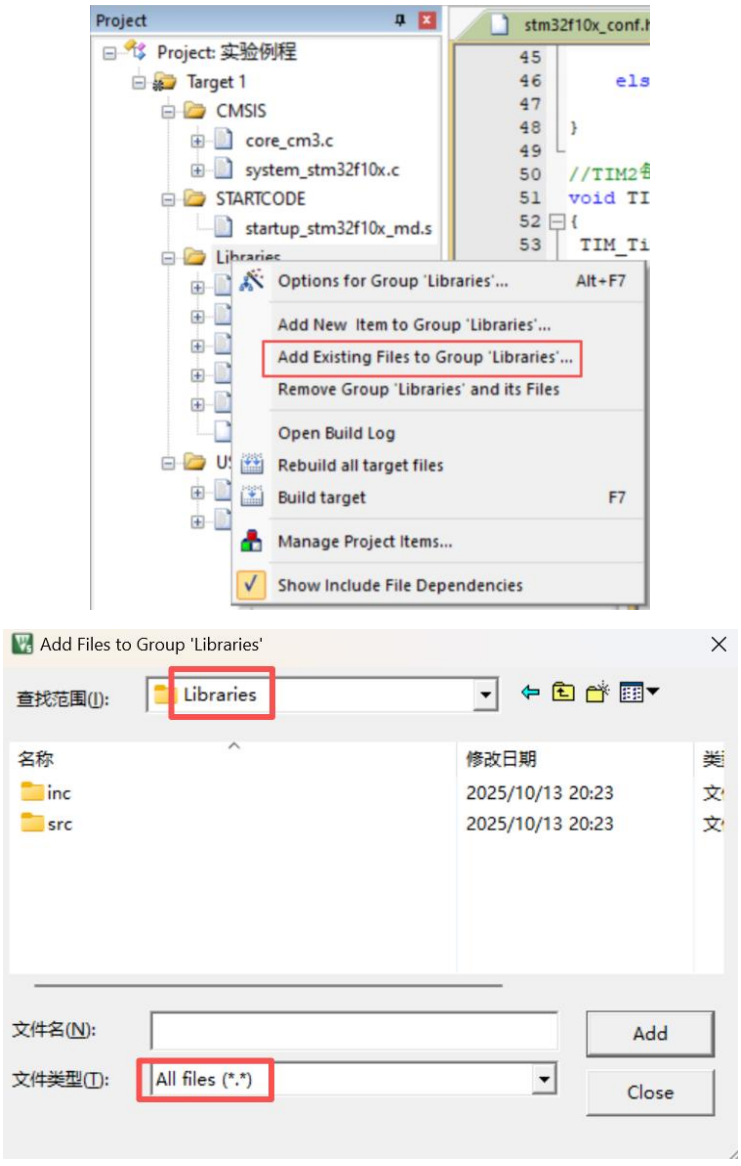



**图8　添加GPIO库文件**

（4）接线：使用一根蓝色数据线将电脑的USB接口与ST-LINK底部的USB-

Debug-调试下载口连接（见图6）；再使用一根白色数据线，将ST-LINK顶部的Debug-USB调试下载口（见图5）连接至感知执行模块底部的USB3.0调试烧写口（见图2）。

（5）启动头文件宏定义：打开头文件stm32f10x_conf.h，找到以下语句并取消注释：

```
#include "stm32f10x_tim.h"
```

（6）引用头文件。

```
#include"stm32f10x_conf.h"
#include"gpio.h"
#include <stdint.h>
```

（7）在源文件main.c中编写主函数，完成系统初始化与定时器配置。

```
int main(void)
{
    RCC_Configuration();
    led_gpio_init();
 TIM2_Configuration(); //TIM2 定时配置
 TIM2_NVIC_Configuration(); //TIM2 中断优先级设置
 while(1)
 {
  /主循环中什么都没有，一切在中断服务程序中完成
 }
}
```

（8）在源文件main.c中编写中断配置函数，设置中断周期。

```
//TIM2 每一秒产生一次中断
void TIM2_Configuration(void)
{
 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2 , ENABLE);
 TIM_DeInit(TIM2);
 TIM_TimeBaseStructure.TIM_Prescaler= (7200 - 1);
 //时钟预分频数为 7200，即单位时间为 7200/72M=0.1ms
 TIM_TimeBaseStructure.TIM_Period=(5000-1);
 //累计 5000 个始终后产生一个更新或者中断，即周期 5000*0.1ms=0.5s
 TIM_TimeBaseStructure.TIM_CounterMode=TIM_CounterMode_Up;
 TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
 TIM_ClearFlag(TIM2, TIM_FLAG_Update);
 TIM_ITConfig(TIM2,TIM_IT_Update,ENABLE);
 TIM_Cmd(TIM2, ENABLE);
}
```

（9）在源文件main.c中编写中断优先级配置函数。

```
//TIM2 中断优先级配置
void TIM2_NVIC_Configuration(void)
{
 NVIC_InitTypeDef NVIC_InitStructure;
 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
 NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
 NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC_Init(&NVIC_InitStructure);
}
```

（10）在源文件main.c中编写中断服务函数，实现LED跑马灯逻辑。

```
//中断服务程序
void TIM2_IRQHandler(void)
{
 if ( TIM_GetITStatus(TIM2 , TIM_IT_Update) != RESET )
 {
  static int led_state = 0;//静态局部变量，只初始化一次
    led_state++;
  switch (led_state)
  {
   case 1: //状态 1：点亮 Status
   led_set(4,1);
   break;
     case 2: //状态 2：灭掉 Status
   led_set(4,0);
   break;
   case 3: //状态 3：点亮 RS485-R
   led_set(5,1);
     break;
     case 4: //状态 4：灭掉 RS485-R
   led_set(5,0);
   break;
   case 5: //状态 5：点亮 RS485-T
   led_set(6,1);
   break;
     case 6: //状态 6：灭掉 RS485-T
   led_set(6,0);
   break;
   case 7: //状态 7：点亮 User1
   led_set(7,1);
   break;
     case 8: //状态 8：灭掉 User1
   led_set(7,0);
   led_state=0; //led_state 置 0
   break;
   default:
   break;
  }
  TIM_ClearITPendingBit(TIM2 , TIM_FLAG_Update);
 }
}
```

（11）在源文件main.c中编写参数函数，设置LED指示灯的亮灭状态。

```
void led_set(uint8_t led_num, int state)
{
    if(state)
          LEDOn(led_num);
    else
            LEDOff(led_num);
}
```

（12）编译与下载程序：点击工具栏依次执行Translate  、Build  、Download  ，烧录完成后，LED将以设定的定时周期依次闪烁，实现跑马灯效果。

# 第五章　注意事项

（1）通过查看单片机底板电路图可知，指示灯分别连接在 PB4、PB5、PB6、PB7 四个引脚上；

（2）在编写中断服务函数时，定义led_set()函数用于管理灯光状态，函数内部

使用静态局部变量led_state来记录当前灯光状态，并通过循环结构实现LED的顺序轮转显示，每个亮灯状态都必须配有相应的灭灯操作，程序初始化时，应确保所有指示灯均处于灭灯状态，以避免上电瞬间误亮。

（3）在进行Debug调试时，应尽量避免将断点设置在break语句处，以防止程序在中断流程中异常暂停。