** 重庆理工大学**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验课堂表现** | | | **实验报告成绩** | **实验总成绩** |
| **A（ ）** | **B（ ）** | **C（ ）** |  |  |

**实验报告书**

**实验名称**：微处理器嵌入式系统ARM实验

专业班级：

学号：

姓名：

联系电话：

指导老师：

实验时间：

电子与通信工程实验中心

# 实验一：ARM开发环境的安装与配置

## 【实验目的】

1、搭建针对ARM的Keil MDK开发环境；

2、根据实验平台使用的目标ARM芯片类型，安装和配置硬件支持资源库。

## 【实验原理及内容】

1、Keil MDK-ARM 是一个由德国 KEIL 公司开发的集成开发环境（IDE），广泛用于嵌入式系统开发，特别是针对 ARM 处理器（如 Cortex-M 内核处理器）。Keil MDK-ARM 提供了一个强大的开发平台，包含编辑器、编译器、调试器等工具，支持多种 ARM 处理器系列，如 Cortex-M、Cortex-R、ARM7、ARM9、Cortex-A8 等。其中本实验所使用的 STM32F407 单片机属于 Cortex-M4 系列。

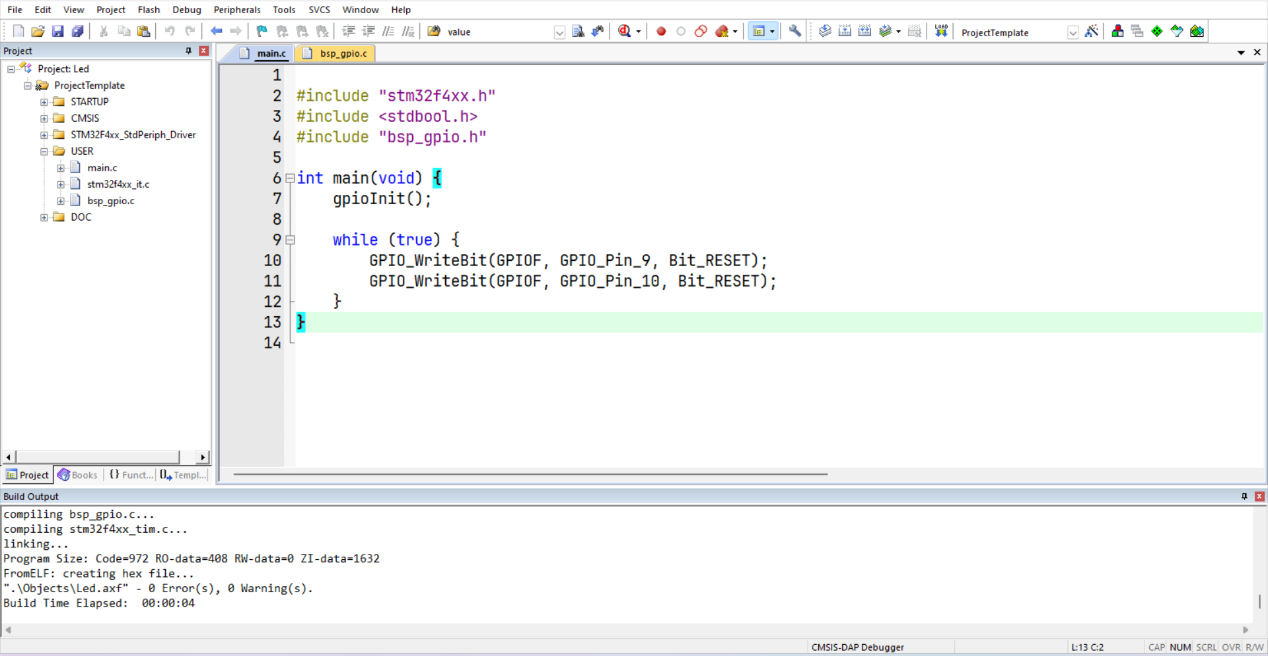
## 【实验步骤】

1. 下载 Keil-MDK 安装包，将其安装至计算机上。
2. 打开 Keil 内的 Pack Installer，在弹出的界面内的左侧菜单中搜索“STM32F407”，然后安装 STM32F4 系列的 DFP 包。
3. 等待硬件资源包安装完成。

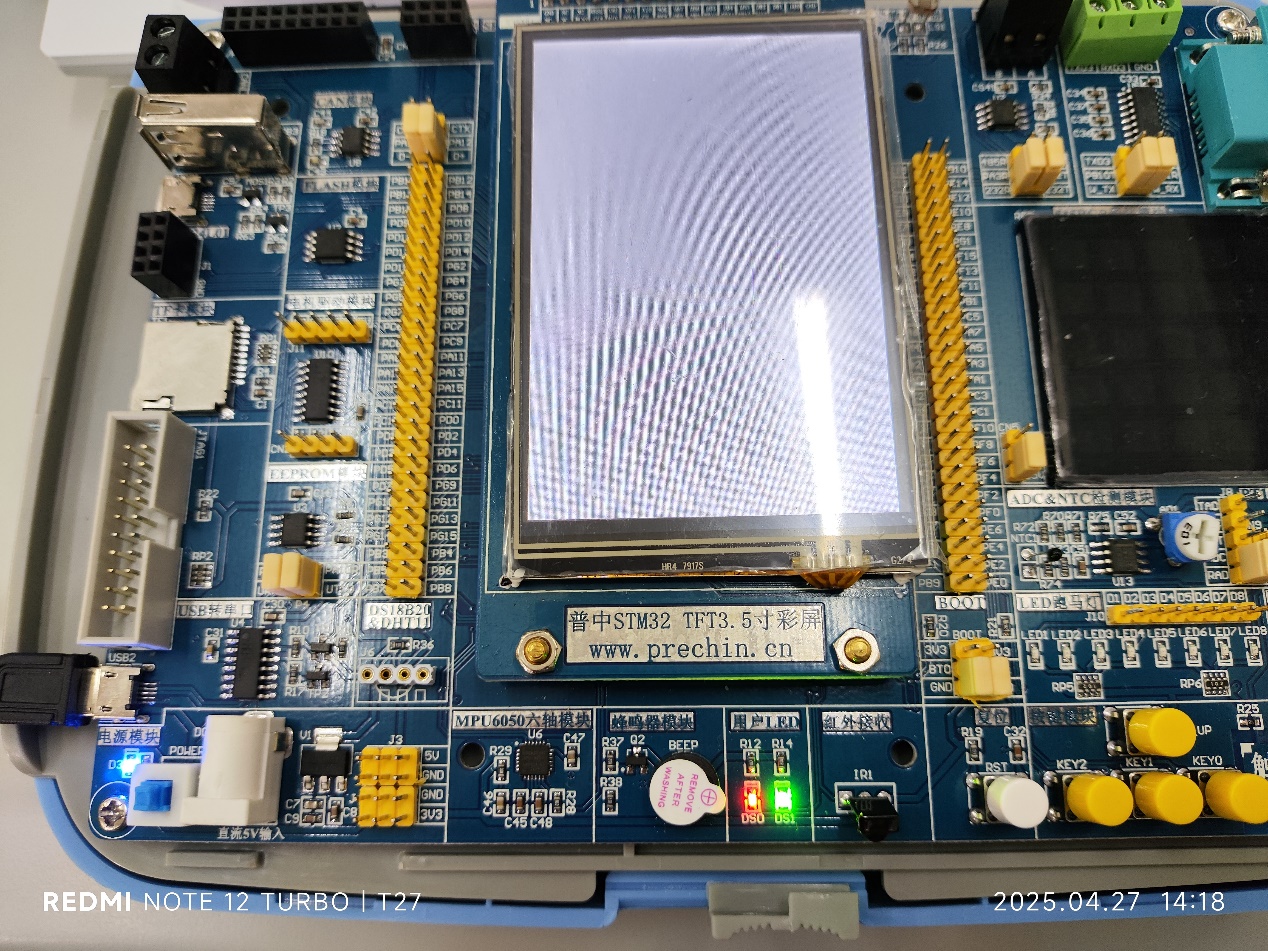
4、打开 Keil，新建一个工程并烧录程序测试。

## 【实验结果】

如下图，已完成 Keil 环境的安装



将程序烧录进入单片机后，运行效果如下：



用户LED已被点亮。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_gpio.h"

int main(void) {

gpioInit();

while (true) {

GPIO\_WriteBit(GPIOF, GPIO\_Pin\_9, Bit\_RESET);

GPIO\_WriteBit(GPIOF, GPIO\_Pin\_10, Bit\_RESET);

}

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

void gpioInit(void);

#endif // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

void gpioInit(void)

{

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9 | GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_Init(GPIOF, &GPIO\_InitStructure);

}

# 实验二：通用IO端口实验

## 【实验目的】

1、根据电路原理图，了解目标IO端口的硬件电路连接情况；

2、学习ARM通用IO端口的配置流程；

3、编程控制目标IO端口，点亮或熄灭其连接的4个LED灯。

## 【实验原理及内容】

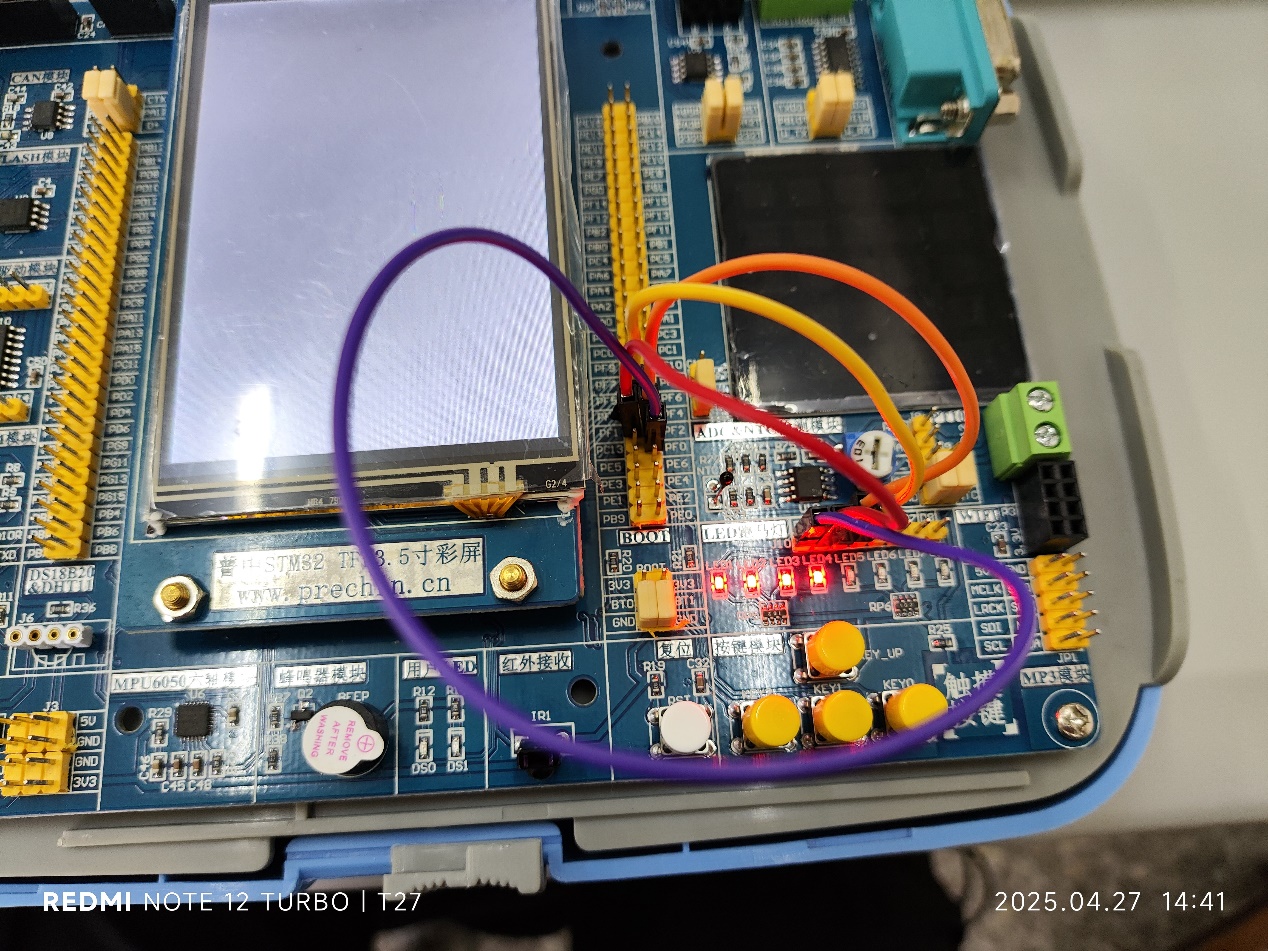
1. 由原理图可知：LED灯在IO口输出低电平的时候会被点亮。
2. 配置GPIO的流程：1）查找手册，使能相应总线的时钟。2）调用相关函数初始化GPIO，使之配置成：输出、低速、推挽输出模式。3）调用相关函数使GPIO输出所需电平。

## 【实验步骤】

1. 根据单片机开发板的原理图连接电路。
2. 初始化GPIO相关外设。
3. 编写程序，使得对应IO口输出低电平以点亮LED。

## 【实验结果】

LED1~4分别连接到单片机的PF0~PF3



可见：LED1~4均被点亮。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_gpio.h"

**int** main(**void**) {

    gpioInit();

**while** (**true**) {

        GPIO\_WriteBit(GPIOF, GPIO\_Pin\_0, Bit\_RESET);

        GPIO\_WriteBit(GPIOF, GPIO\_Pin\_1, Bit\_RESET);

        GPIO\_WriteBit(GPIOF, GPIO\_Pin\_2, Bit\_RESET);

        GPIO\_WriteBit(GPIOF, GPIO\_Pin\_3, Bit\_RESET);

    }

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

**void** gpioInit(**void**);

#endif  // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

**void** gpioInit(**void**)

{

    RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

    GPIO\_InitTypeDef GPIO\_InitStructure;

    GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 | GPIO\_Pin\_3;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

    GPIO\_Init(GPIOF, &GPIO\_InitStructure);

}

# 实验三：流水灯实验

## 【实验目的】

1、编写程序，使8个LED灯呈现“流水灯”运行。

## 【实验原理及内容】

1. GPIO的电平控制通过ODR寄存器控制，对于一个GPIO而言，其ODR寄存器的每一位的置位/复位状态对应了相应IO口的电平状态。
2. 利用C语言的位移运算可以实现流水灯。

## 【实验步骤】

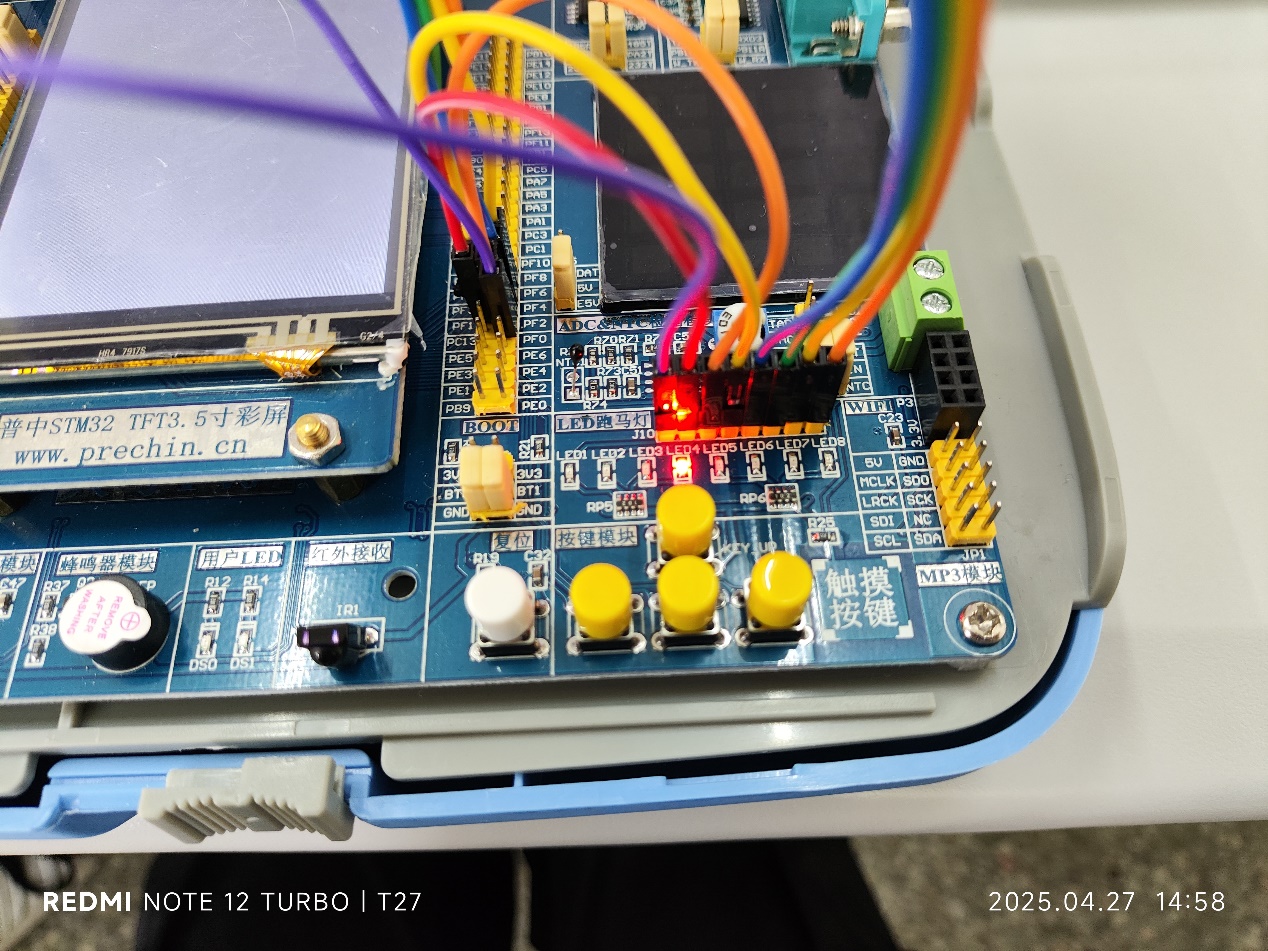
1、连接电路。

2、初始化GPIO及其相关外设。

3、编写程序，观察运行结果。

## 【实验结果】

如下图，LED1~8分别连接单片机的PF0~7



可以看到，LED灯从1号开始逐个亮起，且后一个亮起之后，前一个熄灭。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_gpio.h"

**void** delay(**uint16\_t** time);

**int** main(**void**)

{

    gpioInit();

**while** (**true**) {

**for** (**uint8\_t** i = 0**U**; i < 8**U**; i++) {

            GPIO\_Write(GPIOF, ~(**0x**01**U** << i));

            delay(1000**U**);

        }

    }

}

**void** delay(**uint16\_t** time)

{

**for** (**volatile** **uint16\_t** i = 0; i < time; i++) {

**for** (**volatile** **uint16\_t** j = 0; j < 1000; j++) {

        }

    }

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

**void** gpioInit(**void**);

#endif  // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

**void** gpioInit(**void**)

{

    RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

    GPIO\_InitTypeDef GPIO\_InitStructure;

    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\_Mode = GPIO\_Mode\_OUT;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

    GPIO\_Init(GPIOF, &GPIO\_InitStructure);

}

# 实验四：按键计数器实验

## 【实验目的】

实现按键控制8位LED计数显示。

## 【实验原理及内容】

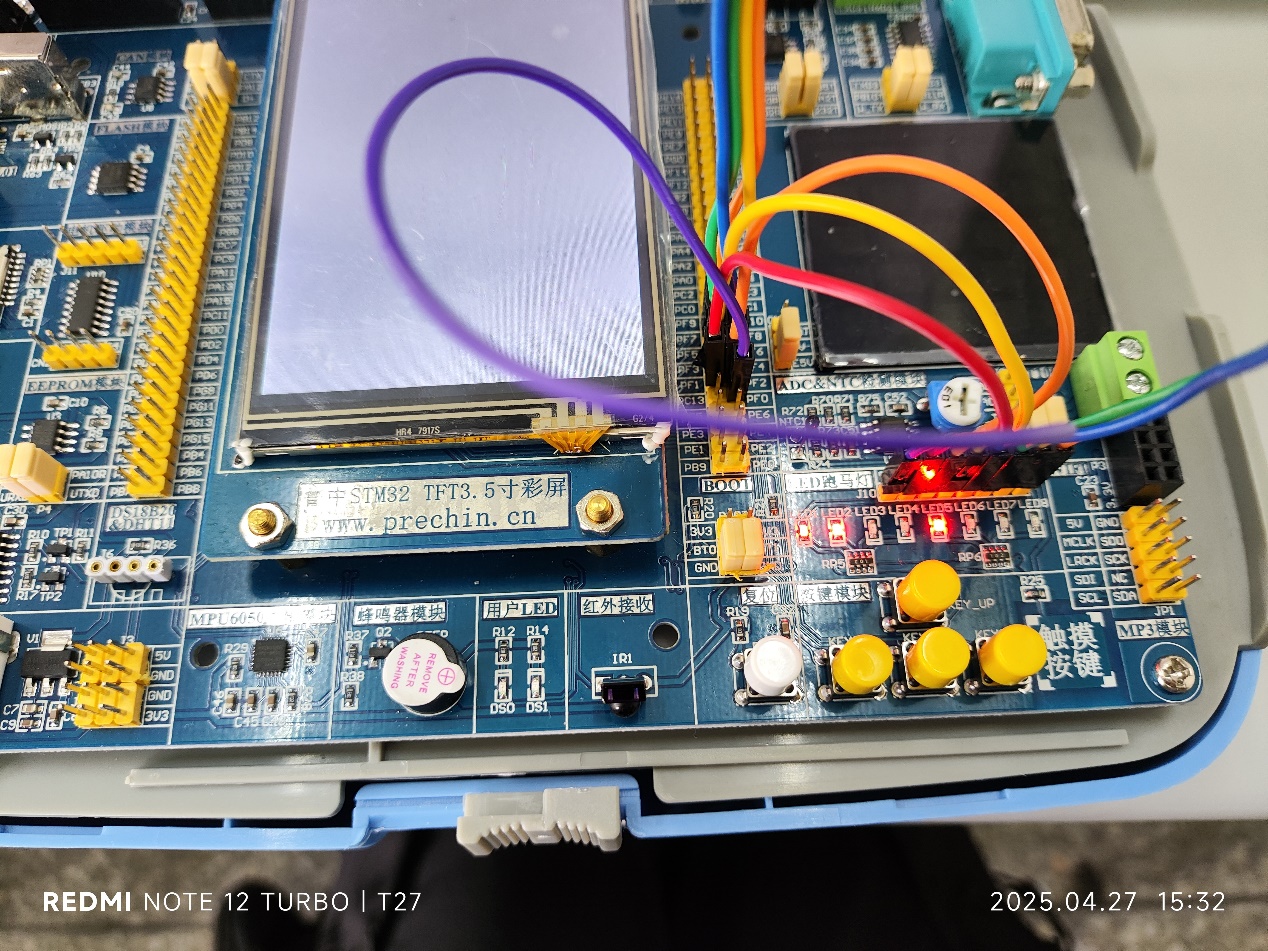
1. GPIO的IDR寄存器可以读取电平状态。按下按键后，电平被拉低，可以被检测到。
2. 将一个uint8\_t类型的数据直接写入GPIO的ODR寄存器后，可以使得LED的电平状态显示二进制数。

## 【实验步骤】

1. 初始化GPIO，配置成输入模式。
2. 编写程序，观察运行结果。

## 【实验结果】

实验现象如图：



8个LED显示一个二进制数（左侧为低位，右侧为高位）。例如图中的LED显示的二进制数就是1100 1000。每按一下左侧的按钮后，计数加一。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_gpio.h"

**void** delay(**uint16\_t** time);

**int** main(**void**)

{

    gpioInit();

**uint8\_t** led\_value = 0**U**;

**while** (**true**) {

**if** (GPIO\_ReadInputDataBit(KEY\_GPIO\_PORT, KEY\_GPIO\_PIN) == Bit\_RESET) {

            delay(500**U**);

**while** (GPIO\_ReadInputDataBit(KEY\_GPIO\_PORT, KEY\_GPIO\_PIN) != Bit\_RESET) {

            }

            led\_value++;

        }

        GPIO\_Write(LED\_GPIO\_PORT, ~led\_value);

        delay(1000**U**);

    }

}

**void** delay(**uint16\_t** time)

{

**for** (**volatile** **uint16\_t** i = 0; i < time; i++) {

**for** (**volatile** **uint16\_t** j = 0; j < 1000; j++) {

        }

    }

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

#define LED\_GPIO\_PORT GPIOF

#define KEY\_GPIO\_PORT GPIOE

#define KEY\_GPIO\_PIN GPIO\_Pin\_4

void gpioInit(void);

#endif // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

#include "bsp\_gpio.h"

**void** gpioInit(**void**)

{

    RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

    GPIO\_InitTypeDef GPIO\_InitStructure;

    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\_Mode = GPIO\_Mode\_OUT;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

    GPIO\_Init(LED\_GPIO\_PORT, &GPIO\_InitStructure);

    GPIO\_InitStructure.GPIO\_Pin = KEY\_GPIO\_PIN;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

    GPIO\_Init(KEY\_GPIO\_PORT, &GPIO\_InitStructure);

}

# 实验五：频率可控的LED计数器实验

## 【实验目的】

1、根据电路原理图，了解目标IO端口的硬件电路连接情况；

2、使用Cortex-M3自带的SysTick定时器或自行编写延时函数，实现频率可控的计数器；

3、编程控制目标IO端口，使其连接的LED灯显示计数器的低4位取值变化。

## 【实验原理及内容】

1、SysTick是单片机自带的一个定时器，通过系统时间戳可以获得准确的延时。

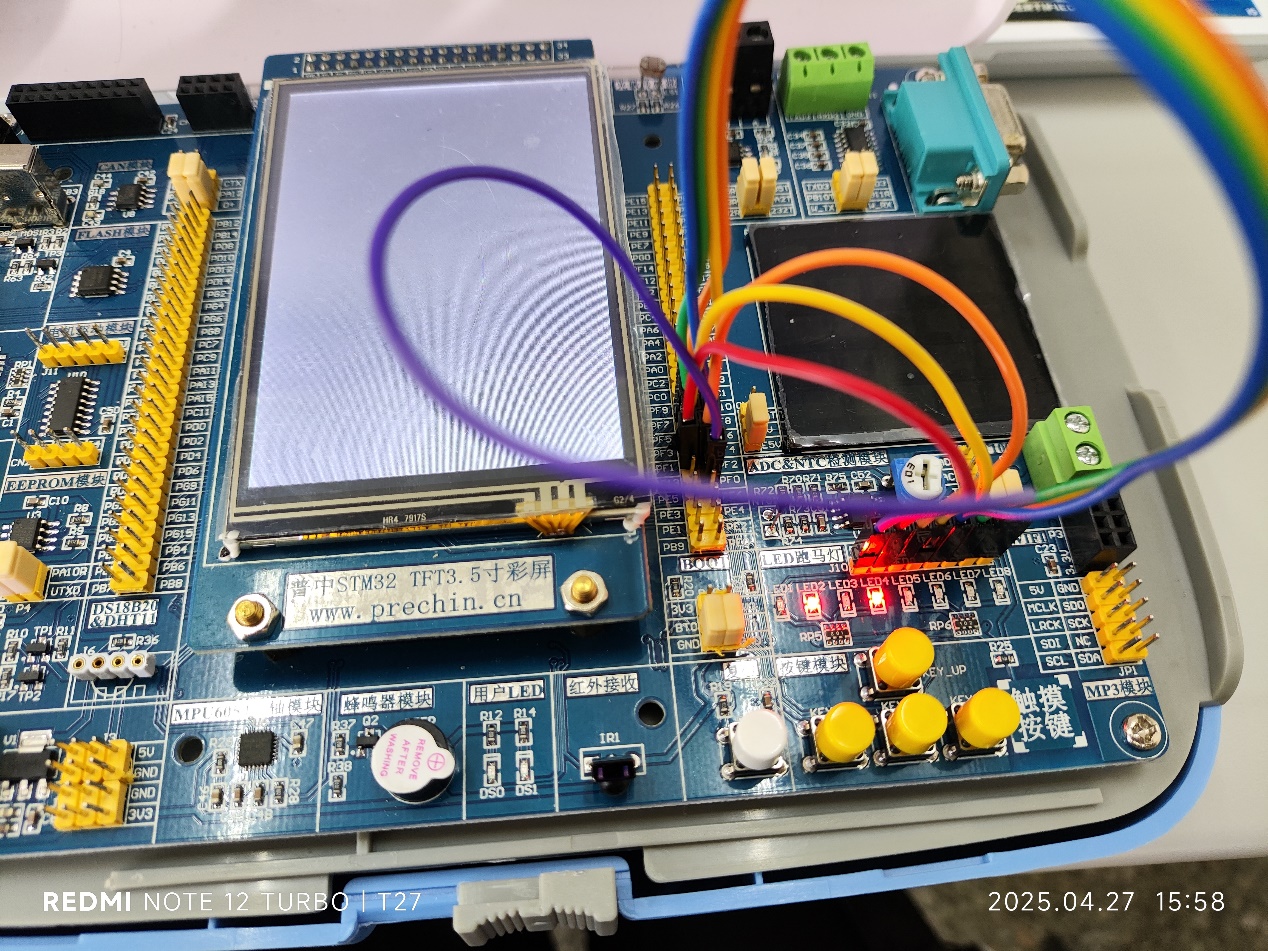
## 【实验步骤】

1、初始化GPIO与SysTick。

2、编写程序，观察运行结果。

## 【实验结果】

现象如下图：



LED计数器的频周期可以通过调整主函数中的Delya\_ms参数进行改变。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_gpio.h"

#include "bsp\_systick.h"

**int** main(**void**)

{

    gpioInit();

**uint8\_t** led\_value = 0**U**;

**while** (**true**) {

        GPIO\_Write(LED\_GPIO\_PORT, ~led\_value);

        led\_value++;

        Delay\_ms(1000U); // 1 秒延时

    }

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

#define LED\_GPIO\_PORT GPIOF

void gpioInit(void);

#endif // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

#include "bsp\_gpio.h"

**void** gpioInit(**void**)

{

    RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

    GPIO\_InitTypeDef GPIO\_InitStructure;

    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\_Mode = GPIO\_Mode\_OUT;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

    GPIO\_Init(LED\_GPIO\_PORT, &GPIO\_InitStructure);

}

// bsp\_systick.c

#include "stm32f4xx.h"

#include "bsp\_systick.h"

#include <stdbool.h>

**volatile** **uint32\_t** TimeCountDown = 0**U**;

**void** Delay\_ms(**uint32\_t** time\_ms) {

    SysTick\_Config(SystemCoreClock / 1000);

**for** (**uint32\_t** i = 0**U**; i < time\_ms; i++) {

        // 当定时器清零的时候，SysTick->CTRL 的第十六位将置1，此时跳出阻塞循环

        // 置1后，再次读取将会自动置0

**while** (!(SysTick->CTRL & (1 << 16))) {

            // 进入阻塞

        }

    }

    // 关闭定时器

    SysTick->CTRL &= ~SysTick\_CTRL\_ENABLE\_Msk;

}

// bsp\_systick.h

#ifndef \_\_BSP\_SYSTICK\_H\_\_

#define \_\_BSP\_SYSTICK\_H\_\_

**void** Delay\_ms(**volatile** **uint32\_t** time\_ms);

#endif  // !\_\_BSP\_SYSTICK\_H\_\_

// stm32f4xx\_it.c

// ……

/\*\*

  \* **@brief**  This function handles SysTick Handler.

  \* **@param**  None

  \* **@retval** None

  \*/

**void** SysTick\_Handler(**void**)

{

    // 执行 SysTick\_Config() 之后，定时器中断每1ms执行此函数

**if** (TimeCountDown != 0) {

        TimeCountDown--;

    }

}

// ……

# 实验六：定时器中断实验

## 【实验目的】

通过定时器中断实现周期地执行代码

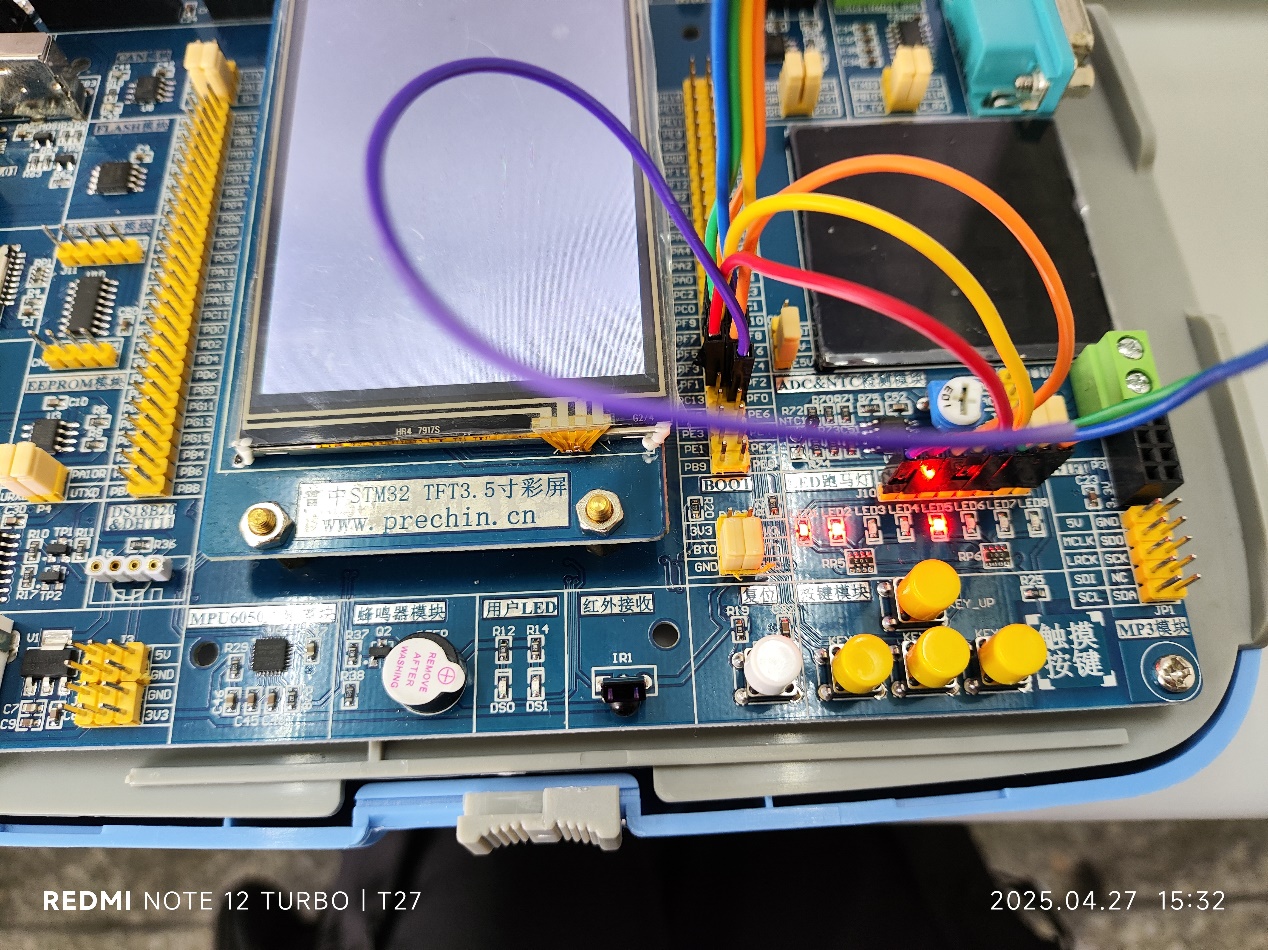
## 【实验原理及内容】

1、定时器根据主频与分频系数，每隔一定时间使计数加一，当计数达到最大计数周期后，清零并产生中断。

## 【实验步骤】

1. 初始化GPIO。
2. 初始化定时器。
3. 将GPIO操作放于定时器中断中。

## 【实验结果】



每间隔

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_timer.h"

#include "bsp\_gpio.h"

uint8\_t count = 0U;

int main(void) {

Timer\_Config();

gpioInit();

while (true) {

GPIO\_Write(LED\_GPIO\_PORT, ~count);

}

}

// bsp\_timer.h

#ifndef \_\_BSP\_TIMER\_H\_\_

#define \_\_BSP\_TIMER\_H\_\_

#include "stm32f10x.h"

void Timer\_Config(void);

#endif // !\_\_BSP\_TIMER\_H\_\_

// bsp\_timer.c

void Timer\_Config(void) {

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);

TIM\_InternalClockConfig(TIM2);

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStruct;

TIM\_TimeBaseInitStruct.TIM\_ClockDivision = TIM\_CKD\_DIV1; // 选用1分频

TIM\_TimeBaseInitStruct.TIM\_CounterMode = TIM\_CounterMode\_Up; // 选择为向上计数

TIM\_TimeBaseInitStruct.TIM\_Period = 10000 - 1; // 设置计时周期为10000

TIM\_TimeBaseInitStruct.TIM\_Prescaler = 7200 - 1; // 设置定时器分频值为7200

TIM\_TimeBaseInitStruct.TIM\_RepetitionCounter = 0;

TIM\_TimeBaseInit(TIM2, &TIM\_TimeBaseInitStruct);

// 清除一次中断标志位，防止在初始化结束后直接进入中断

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);

TIM\_ITConfig(TIM2, TIM\_IT\_Update, ENABLE); // 使能 ITM2 的中断

NVIC\_SetPriorityGrouping(NVIC\_PriorityGroup\_2);

NVIC\_InitTypeDef NVIC\_InitStruct;

NVIC\_InitStruct.NVIC\_IRQChannel = TIM2\_IRQn;

NVIC\_InitStruct.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority = 2;

NVIC\_Init(&NVIC\_InitStruct);

TIM\_Cmd(TIM2, ENABLE);

}

// bsp\_gpio.h

#ifndef \_\_BSP\_GPIO\_H\_\_

#define \_\_BSP\_GPIO\_H\_\_

#define LED\_GPIO\_PORT GPIOF

void gpioInit(void);

#endif // !\_\_BSP\_GPIO\_H\_\_

// bsp\_gpio.c

#include "stm32f4xx.h"

#include "bsp\_gpio.h"

**void** gpioInit(**void**)

{

    RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

    GPIO\_InitTypeDef GPIO\_InitStructure;

    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\_Mode = GPIO\_Mode\_OUT;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_25MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

    GPIO\_Init(LED\_GPIO\_PORT, &GPIO\_InitStructure);

}

// stm32f4xx\_it.c

// ……

void TIM2\_IRQHandler(void) {

if (TIM\_GetITStatus(TIM2, TIM\_IT\_Update) == SET) {

count++;

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);

}

}

// ……

# 实验七：蜂鸣器实验

## 【实验目的】

使蜂鸣器响

## 【实验原理及内容】

通过定时器的输出比较功能产生PWM波。

## 【实验步骤】

1. 连接电路。
2. 初始化定时器。

## 【实验结果】

能听到蜂鸣器响

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_pwm.h"

int main(void)

{

PWM\_Config();

while (true) {

TIM\_SetCompare1(TIM13, 50);

}

}

// bsp\_pwm.c

#include "bsp\_pwm.h"

void PWM\_Config(void) {

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13 ,ENABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_8;

GPIO\_InitStruct.GPIO\_Speed = GPIO\_High\_Speed;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

TIM\_InternalClockConfig(TIM2);

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStruct;

TIM\_TimeBaseInitStruct.TIM\_ClockDivision = TIM\_CKD\_DIV1;

TIM\_TimeBaseInitStruct.TIM\_CounterMode = TIM\_CounterMode\_Up;

TIM\_TimeBaseInitStruct.TIM\_Period = 20000 - 1;

TIM\_TimeBaseInitStruct.TIM\_Prescaler = 168 - 1; TIM\_TimeBaseInitStruct.TIM\_RepetitionCounter = 0;

TIM\_TimeBaseInit(TIM2, &TIM\_TimeBaseInitStruct);

TIM\_OCInitTypeDef TIM\_OCInitStruct;

TIM\_OCStructInit(&TIM\_OCInitStruct);

TIM\_OCInitStruct.TIM\_OCMode = TIM\_OCMode\_PWM1;

TIM\_OCInitStruct.TIM\_OCPolarity = TIM\_OCPolarity\_High;

TIM\_OCInitStruct.TIM\_OutputState = TIM\_OutputState\_Enable;

TIM\_OCInitStruct.TIM\_Pulse = 0;

TIM\_OC1Init(TIM2, &TIM\_OCInitStruct);

TIM\_Cmd(TIM2, ENABLE);

}

# 实验八：串口通信实验

## 【实验目的】

使用串口从单片机向电脑发送数据。

## 【实验原理及内容】

使用USB转TTL连接电脑与单片机，可以让单片机发送的串口信号被电脑接收。

## 【实验步骤】

将PA9连接到USB转TTL的RXD接口，PA10连接到TXD接口，然后将单片机上的GND连接到USB转TTL的GND接口上。

## 【实验结果】

能够从电脑上的串口助手工具中收到单片机发送的消息。

## 【主程序代码】

// main.c

#include "stm32f4xx.h"

#include <stdbool.h>

#include "bsp\_usart.h"

int main(void) {

USART\_Config();

while (true) {

USART1\_Send(0x14);

}

}

// bsp\_usart.c

void USART1\_Send(uint8\_t data) {

USART\_SendData(USART1, data);

while (USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET) {

}

}

void USART\_Config(void) {

// 打开 USART1 和 GPIOA 的时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);

// 设置 GPIO

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_InitStruct.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_IPU;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

USART\_InitTypeDef USART\_InitStruct;

USART\_InitStruct.USART\_Mode = USART\_Mode\_Tx;

USART\_InitStruct.USART\_BaudRate = 9600;

USART\_InitStruct.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStruct.USART\_Parity = USART\_Parity\_No;

USART\_InitStruct.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStruct.USART\_WordLength = USART\_WordLength\_8b;

USART\_Init(USART1, &USART\_InitStruct);

USART\_Cmd(USART1, ENABLE);

}