嵌入式平时实验

班级：

姓名：

学号：

目录

嵌入式平时实验 1
实验 6-1 从左到右的流水灯 1
编程结果 1
调试结果 2
分析 2
首先是对连接 LED 灯的 GPIOC 口进行初始化，因为使用的引脚数目比较多，就将 16 个引脚都使能，只用前 8 个。 2
在循环中，主要采用的是给整个 GPIOC 口进行赋值的方式，先确定要点亮的灯，即为低电平引脚，然后进行一定时间的延时，然后确定下个点亮的灯，赋值，进行延时，如此进行 8 次赋值，分别点亮 8 个 LED 灯。 2
实验 6-2 交通灯控制 2
编程结果 2
调试结果 3
分析 3
实验 6-3 LED 条形灯 4
编程结果 4
调试结果 5
分析 6
实验 7-1 按键控制小灯转态 6
编程结果 6
调试结果 8
分析 8
实验 7-2 按键控制多显示 LED 灯 8
编程结果 8
调试结果 11
分析 12
实验 8-1 定时器流水灯 13
编程结果 13
调试结果 15
分析 15
实验 8-2 LED-条形灯 15
编程结果 15
调试结果 17
分析 18
实验 实验板-1 18
编程结果 18
调试结果 19
分析 19
首先是对连接 LED 灯的 GPIOC 口进行初始化，因为使用的引脚数目比较多，就将 16 个引脚都使能，只用前 8 个。 19
在循环中，主要采用的是给整个 GPIOC 口进行赋值的方式，先确定要点亮的灯，即为低电平引脚，然后进行一定时间的延时，然后确定下个点亮的灯，赋值，进行延时，如此

进行 8 次赋值，分别点亮 8 个 LED 灯。 19
实验 实验板-2 19
编程结果 20
调试结果 22
分析 22
实验 实验板-3 22
编程结果 22
调试结果 24
分析 24

实验 6-1 从左到右的流水灯

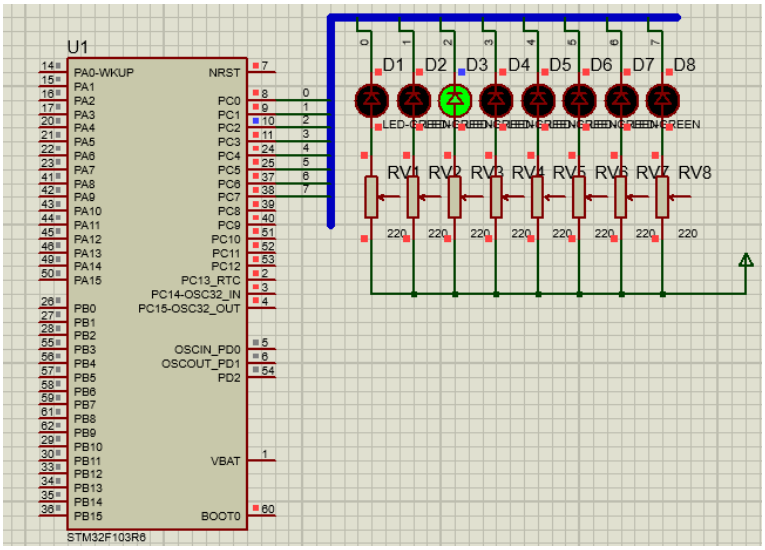
实现从左到右的流水灯，从左到右依次点亮（每次只亮一个）。循环往复。

编程结果

```
#include "stm32f10x.h"                  // Device header
#include "Delay.h"
int main()
{
        GPIO_InitTypeDef    GPIO_InitStructure; //定义 GPIO 口初始化结构体
          RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);//使能 C 口
          GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;//设置为推挽输出
          GPIO_InitStructure.GPIO_Pin=GPIO_Pin_All;//打开所有端口
          GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;//设置频率为 50M_Hz
          GPIO_Init(GPIOC,&GPIO_InitStructure);

          GPIO_Write(GPIOC,0xFFFF);//初始状态使所有灯
    while(1)
    {
          GPIO_Write(GPIOC,0xFFFE);//通过逐个赋值控制引脚电平高低
          Delay(300000);
          GPIO_Write(GPIOC,0xFFFD);
          Delay(300000);
          GPIO_Write(GPIOC,0xFFFB);
          Delay(300000);
          GPIO_Write(GPIOC,0xFFF7);
          Delay(300000);
          GPIO_Write(GPIOC,0xFFEF);
          Delay(300000);
          GPIO_Write(GPIOC,0xFFDF);
          Delay(300000);
          GPIO_Write(GPIOC,0xFFBF);
          Delay(300000);
          GPIO_Write(GPIOC,0xFF7F);
          Delay(300000);
    }
}
```

调试结果

流水灯，从左到右依次亮起，往复循环。



分析

首先是对连接 LED 灯的 GPIOC 口进行初始化，因为使用的引脚数目比较多，就将 16 个引脚都使能，只用前 8 个。

在循环中，主要采用的是给整个 GPIOC 口进行赋值的方式，先确定要点亮的灯，即为低电平引脚，然后进行一定时间的延时，然后确定下个点亮的灯，赋值，进行延时，如此进行 8 次赋值，分别点亮 8 个 LED 灯。

实验 6-2 交通灯控制

编程结果

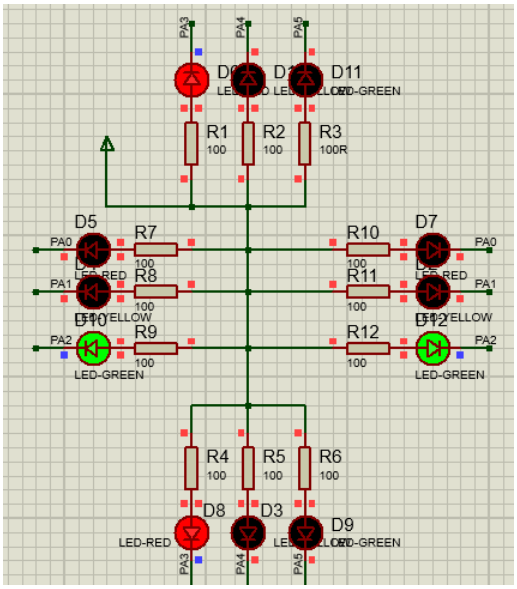
```
#include "stm32f10x.h"                  // Device header
#include "Delay.h"
int main()
{
    GPIO_InitTypeDef    GPIO_InitStructure;//定义 GPIO 口初始化结构体
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);//使能 A 口
      GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;//设置为推挽输出
      GPIO_InitStructure.GPIO_Pin=GPIO_Pin_All;//打开所有端口
      GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;//设置频率为 50M_Hz
```

```
GPIO_Init(GPIOA,&GPIO_InitStructure);
```

```
        GPIO_Write(GPIOA,0x00ED);
        Delay(10000);
    while(1)
    {
            GPIO_Write(GPIOA,0x00DE);//红绿灯
            Delay_100(30000);
            GPIO_Write(GPIOA,0x00ED);//黄灯
            Delay_100(30000);
            GPIO_Write(GPIOA,0x00F3);//红绿灯
            Delay_100(30000);
    }
}
```

调试结果

红黄绿三种灯按照交通规则点亮。



分析

现实生活中，在南北向红灯亮起时，东西向的绿灯亮起，而黄灯是同时亮起。如上图所示，同一个方向上的同一种颜色的灯是由同一个引脚控制的，再根据现实中的交通规则，容易得到 PA2 和 PA3 电平一致，PA0 和 PA5 电平一致，PA1 和 PA4 电平一致。因为一共只有三种状态，程序中采用了最直接的赋值方法，为红绿灯和黄灯时单独赋值，状态切换中进行一段时间的延时。

实验 6-3 LED 条形灯

实现 LED 条形灯从上至下，依次点亮直至全亮，然后从下至上依次熄灭直至全灭。循环往复。

编程结果

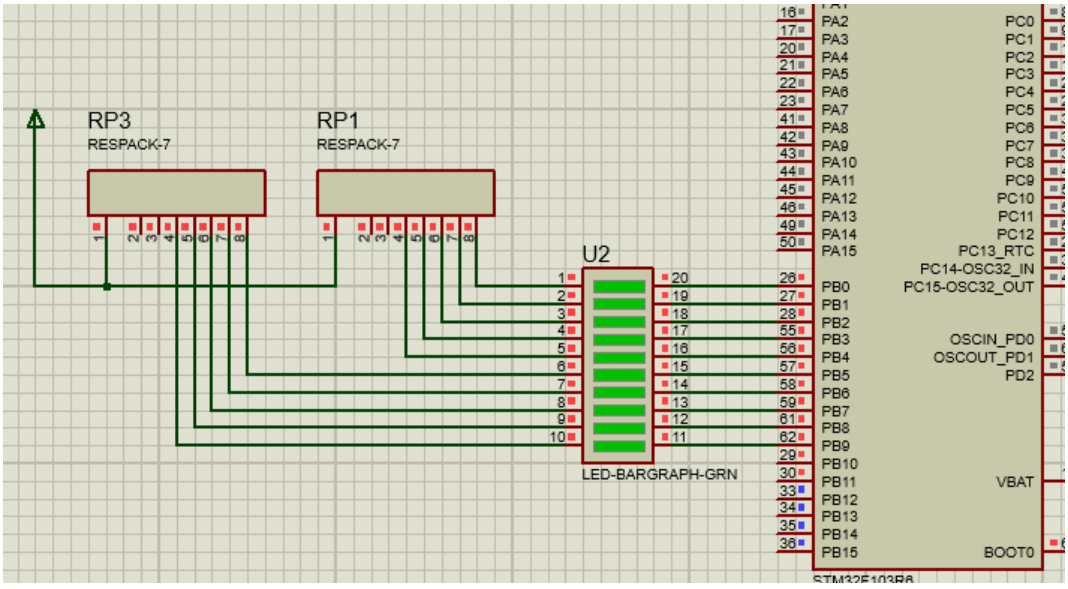
```
#include "stm32f10x.h"                  // Device header
#include "Delay.h"
int main()
{
    GPIO_InitTypeDef    GPIO_InitStructure;
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);//和上面的差不多
  //initial
      GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
      GPIO_InitStructure.GPIO_Pin=GPIO_Pin_All;
      GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
      GPIO_Init(GPIOB,&GPIO_InitStructure);

      GPIO_Write(GPIOB,0x0FFF);
  while(1)
  {
          GPIO_WriteBit(GPIOB,GPIO_Pin_0,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_1,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_2,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_3,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_4,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_5,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_6,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_7,Bit_RESET);
          Delay_100(3000);
          GPIO_WriteBit(GPIOB,GPIO_Pin_8,Bit_RESET);
          Delay_100(3000);
```

```
GPIO_WriteBit(GPIOB,GPIO_Pin_9,Bit_RESET);//以上，从上到下点亮
```

```
            Delay_100(10000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_9,Bit_SET);//以下，从下到上熄灭
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_8,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_7,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_6,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_5,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_4,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_3,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_2,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_1,Bit_SET);
            Delay_100(3000);
            GPIO_WriteBit(GPIOB,GPIO_Pin_0,Bit_SET);
            Delay_100(10000);           //在一次完成后进行一次较长的延时
    }
}
```

调试结果

从上至下，依次点亮直至全亮，然后从下至上依次熄灭直至全灭。循环往复。



分析

按照要求，依次点亮灯泡，每次采取点亮一个灯的方式，因而采用位操作的函数GPIO_WriteBit，对单个引脚进行操作，从 GPIO_Pin_0~GPIO_Pin_9 点亮一个灯后，延迟一段时间后点亮下一个灯。熄灭是也采取同样的逻辑，从 GPIO_Pin_9~GPIO_Pin_0 采用位操作函数依次熄灭。

实验 7-1 按键控制小灯转态

按下按键并弹起后，小灯状态反转。按键采用中断方式实现。

编程结果

LED 初始化

```
#include "stm32f10x.h"                  // Device header
void LEDInit()
{
	GPIO_InitTypeDef    GPIO_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);

	GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Pin=GPIO_Pin_0;
	GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;//和上面差不多
	GPIO_Init(GPIOB,&GPIO_InitStructure);
}
```

EXIT 初始化

```
#include "stm32f10x.h"                  // Device header
void ExtiInit()
{
    GPIO_InitTypeDef GPIO_InitStructure;
    EXTI_InitTypeDef EXTI_Initstructure;
    NVIC_InitTypeDef NVIC_Structure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);//复用为中断输入口
```

```
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IPU;//上拉输入
```

```
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_0;//初始化 A 口
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource0);//将 A 口配置为中断口


    EXTI_Initstructure.EXTI_Line=EXTI_Line0 ;//中断线路选择
    EXTI_Initstructure.EXTI_LineCmd=ENABLE;//使能
    EXTI_Initstructure.EXTI_Mode=EXTI_Mode_Interrupt;//中断还是事件
    EXTI_Initstructure.EXTI_Trigger=EXTI_Trigger_Falling;//上边沿还是下降沿触发
    EXTI_Init(&EXTI_Initstructure);

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//配置 NVIC
    NVIC_Structure.NVIC_IRQChannel=EXTI0_IRQn;//配置对应的中断服务路线
    NVIC_Structure.NVIC_IRQChannelCmd=ENABLE;
    NVIC_Structure.NVIC_IRQChannelPreemptionPriority=1;//设置中断优先极
    NVIC_Structure.NVIC_IRQChannelSubPriority=1;
    NVIC_Init(&NVIC_Structure);
}
```

中断服务函数

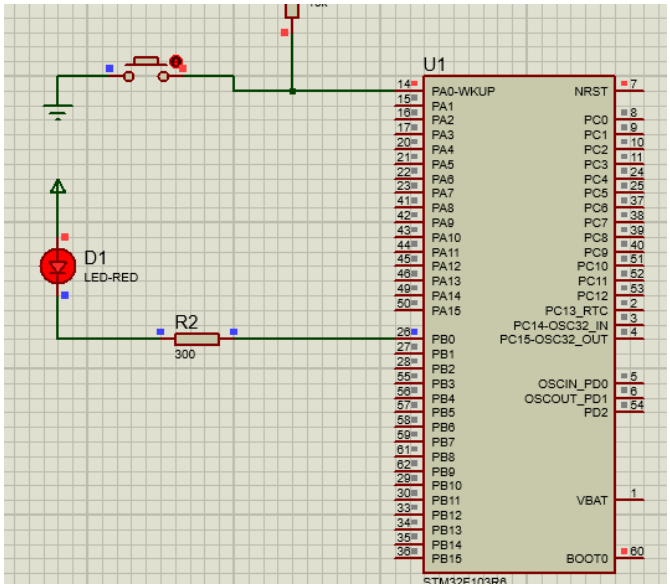
```
void EXTI0_IRQHandler(void)
{
    if(EXTI_GetFlagStatus(EXTI_Line0)!=RESET)
    {
        GPIOB ->ODR ^=GPIO_Pin_0; //对此位进行取反
        EXTI_ClearITPendingBit(EXTI_Line0);//清除中断标记位
    }
}
```

Main 函数

```
#include "stm32f10x.h"                  // Device header
#include "Delay.h"
#include "ExitInit.h"
#include "LEDInit.h"
int main()
{
    ExtiInit();//这两个函数已经写到对应的 C 文件和 H 文件中
    LEDInit();
    GPIO_WriteBit(GPIOB,GPIO_Pin_0,Bit_RESET);//设置小灯的初始状态
    while(1)
    {
    }
}
```

调试结果

按下按键后，小灯的状态进行反转。



分析

按照课上所学与例程进行中断口和 GPIO 口的配置，配置完成后必须要注意中断优先级是否配置或者冲突，中断服务函数是否采用的程序中提供的专门的函数。对小灯的状态取反采取了对寄存器某一位进行操作的方法，每次按下按钮进入中断后，对相应位进行取反。

实验 7-2 按键控制多显示 LED 灯

第一次按下并释放按键后，小灯全亮。第二次按下并释放按键后，小灯全灭。再次按下按键并释放后，小灯从上往下依次点亮（流水灯）。然后再次按下按键并释放后，小灯从下往上依次点亮（流水灯）。之后按键效果循环往复。按键采用中断方式实现。

编程结果

LED 初始化

```
void LEDInit()
{
```

```
GPIO_InitTypeDef    GPIO_InitStructure;
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOC,&GPIO_InitStructure);
}
```

EXIT 初始化

```
void ExtiInit()
{
    GPIO_InitTypeDef GPIO_InitStructure; //定义初始化结构体，GPIO,中断，中断优先级
    EXTI_InitTypeDef EXTI_Initstructure;
    NVIC_InitTypeDef NVIC_Structure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);//A 口作为中断口
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);


    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource1);//选择引脚口 PA1


    EXTI_Initstructure.EXTI_Line=EXTI_Line1 ;//中断线路选择
    EXTI_Initstructure.EXTI_LineCmd=ENABLE;//使能
    EXTI_Initstructure.EXTI_Mode=EXTI_Mode_Interrupt;//中断还是事件
    EXTI_Initstructure.EXTI_Trigger=EXTI_Trigger_Falling;//上边沿还是下降沿触发
    EXTI_Init(&EXTI_Initstructure);

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//配置 NVIC
    NVIC_Structure.NVIC_IRQChannel=EXTI1_IRQn;
    NVIC_Structure.NVIC_IRQChannelCmd=ENABLE;
    NVIC_Structure.NVIC_IRQChannelPreemptionPriority=1;//中断优先极
    NVIC_Structure.NVIC_IRQChannelSubPriority=1;
    NVIC_Init(&NVIC_Structure);
}
```