浙江大学实验报告

专业：生物医学工程
姓名：
学号：
日期：2025.4.8
地点：东 1B 416

课程名称：微机原理及其应用　　指导老师：张恒义　　实验类型：微机实验
实验名称：交通灯控制仿真系统设计　　成　绩：　　签　名：

装 订 线

一、 实验目的和要求

1. 实验目的

(1) 通过设计和实现十字路口交通灯控制系统的仿真实验，深入理解微处理器在实际应用中的软硬件设计方法。
(2) 学习并掌握软件定时与硬件定时器定时在控制程序中的应用技巧。

2. 实验要求

(1) 利用 proteus 软件，设计一个基于微处理器的简易十字路口交通灯控制系统，该系统能够模拟真实的交通灯变换过程。即模拟十字路口东西南北走向的交通灯状态变化，初始状态为东西南北方向均为红灯，然后按照特定顺序转换状态，包括东西方向绿灯通车（南北红灯）、东西方向黄灯过渡（南北红灯）、南北方向绿灯通车（东西红灯）、南北方向黄灯过渡（东西红灯），并且要循环进行这些状态转换。
(2) 系统需具备两种定时方式：软件循环延迟法和定时器定时法，确保交通灯按照预设的时间间隔准确切换。例如，绿灯持续时间用软件延时获取，黄灯持续时间用定时器定时获取。

二、 实验内容和原理

1. 硬件电路设计

(1) 在 PROTEUS 环境下，交通灯组件 traffic lights 的工作原理是通过对控制线的电平控制来实现灯的亮灭。由于其默认有公共地，当控制线接高电平时，对应的红、黄、绿灯就会被点亮。但由于 I/O 的驱动能力不足，所以要加入上拉排阻 RESPACK - 8 为交通灯组件提供上拉电流，以满足其正常工作的电气特性要求。
(2) 十字路口在东西和南北方向上，相向两个方向上的灯的变化是相同的，可以并联连接，以保证相向两方向上的灯同步工作。

2. 软件定时原理

(1) 软件循环延迟法原理：通过编写程序循环结构，利用指令执行时间来实现时间延迟。例如，在一个循环中执行一定数量的条件转移指令或者其它指令，根据指令的执行周期和循环次数来计算总的延迟时间。这种方法简单直接，但会占用较多的 CPU 资源，并且定时精度相对较低。
(2) 定时器定时法原理：利用微处理器内部的定时器资源。首先需要对定时器进行配置，包括设置定时器的工作模式（如定时模式、计数模式等）、定时初值等。当定时器启动后，它会按照设定的时钟源进行计数，当计数达到设定的定时值时，会产生定时器中断。这种方法定时精度较高，并且不会过多占用 CPU 资源，因为定时器在计数过程中，CPU 可以执行其他任务，当定时器计数完成后再进行相应的处理。

三、 主要仪器设备

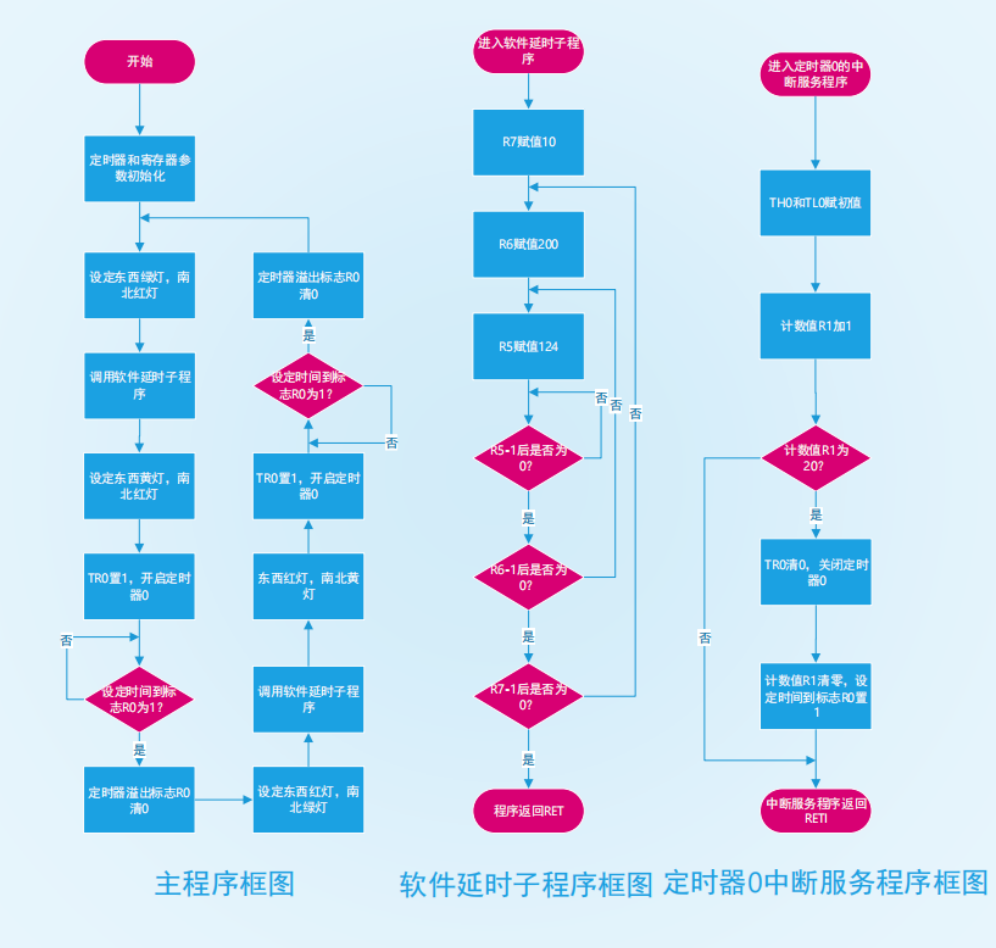
微型计算机、单片机开发工具软件 Keil C51，Proteus

四、 操作方法和实验步骤

1. 硬件电路设计

运用 proteus 的交通信号灯元器件，8051 单片机的 P0 口输出来控制信号灯，注意 P0 口输出时需要外接上拉电阻。详细的部分请见实验结果与分析部分的硬件电路设计图

2. 软件流程图



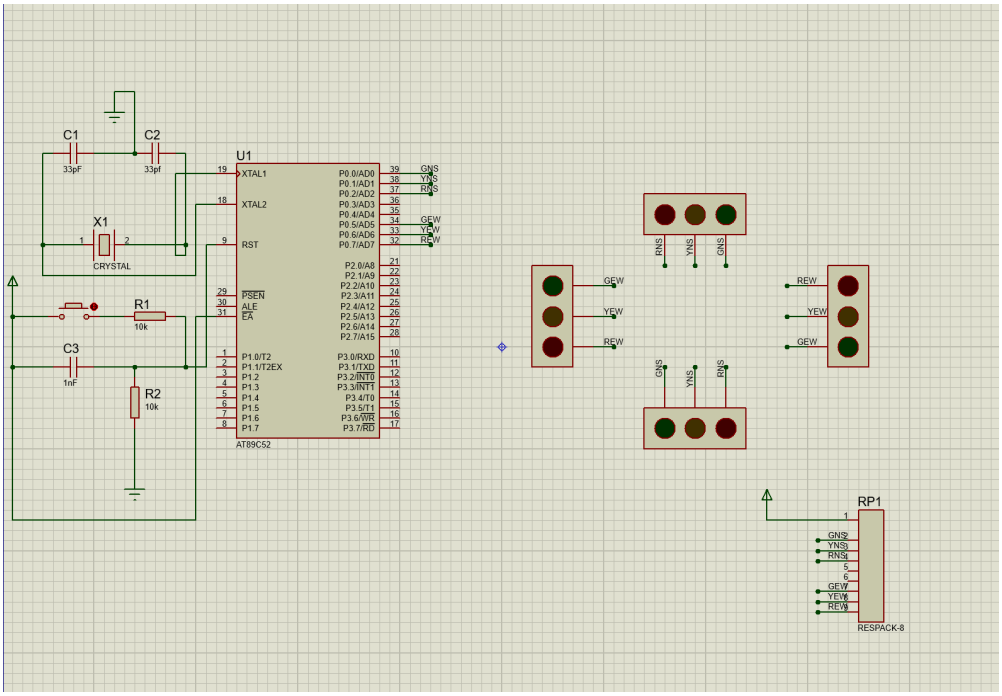
3. 程序代码

```
ORG 0000H
LJMP MAIN           ; 复位后跳转到主程序
ORG 000BH           ; 定时器 0 中断向量地址
LJMP T0Interrupt    ; 跳转到中断服务程序

ORG 0030H           ; 主程序起始地址

MAIN:
    MOV TMOD, #01H  ; 定时器定时 50ms 初始化
    MOV TH0, #3CH
    MOV TL0, #0B0H
    SETB EA         ; 开启总中断
    SETB ET0        ; 允许定时器 0 中断
    MOV R0, #0      ; 初始化 1 秒标志位
    MOV R1, #0      ; 初始化中断计数器
    ; 注意：此时未启动定时器 TR0，仅在需要定时器延时时启动
Start:
    MOV A, #3CH     ; 东西绿灯，南北红灯
    MOV P0, A
    LCALL DELAY     ; 调用 5 次软件延时（共延时 2.5 秒）
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY

    MOV A, #5CH     ; 东西黄灯，南北红灯
    MOV P0, A

    SETB TR0        ; 启动定时器 0
WAIT1:
    CJNE R0, #1, WAIT1 ; 等待 1 秒标志位
    MOV R0, #0      ; 清除标志位
    CLR TR0         ; 关闭定时器

    MOV A, #99H     ; 东西红灯，南北绿灯
    MOV P0, A
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY
    LCALL DELAY

    MOV A, #9AH     ; 东西红灯，南北黄灯
```

```
    MOV P0, A
    SETB TR0        ; 再次启动定时器 0
WAIT2:
    CJNE R0, #1, WAIT2 ; 等待 1 秒标志位
    MOV R0, #0      ; 清除标志位
    CLR TR0         ; 关闭定时器

    SJMP Start      ; 循环主程序

T0Interrupt:
    MOV TH0, #3CH   ; 重新装载定时初值
    MOV TL0, #0B0H
    INC R1          ; 中断计数器 +1
    CJNE R1, #20, CASE ; 定时器时基是 50ms，总定时 1s 需要中断 20 次
    CLR TR0         ; 关闭定时器
    MOV R0, #1      ; 设置 1 秒标志位，定位时间到
    MOV R1, #0      ; 清零计数器
CASE:
    RETI            ; 中断返回

DELAY:
    MOV R7, #10
DE1:
    MOV R6, #200
DE2:
    MOV R5, #124
    DJNZ R5, $
    DJNZ R6, DE2
    DJNZ R7, DE1
    RET
    END
```

五、 实验结果与分析

1. 硬件电路设计图



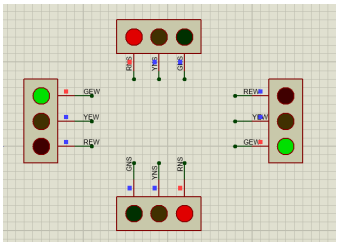
2. 软件编写经历

本次实验我基本是在实验说明文件的例程基础上完善的。阅读实验文件对预期结果说明后，其实本次实验只需要考虑 P0 口输出四个交通信号灯对应的状态，然后分别用软件延时和定时器延时两种方式延时对应的时间即可。我采用了绿灯亮时采用软件延时的方法，例程已经给出了软件延时的代码，老师上课也讲解过具体的原理，直接用 LCALL 调用即可，调用 5 次，共延时大约 2.5 秒。然后黄灯亮时采用定时器延时，定时器延时的原理代码我一开始对定时器的初值并不是特别理解，后来查阅了相关资料之后得知这是使得过了对应的时间后定时器能溢出，然后进入中断的响应，初值决定了定时器中断延时的时间。在调用时，只需要先启动定时器，然后写一个循环程序等待 R0 标志位置 1 即可，并重新将 R0 标志位清零方便下一次中断。然后还需要注意的是每一次定时器中断都要设置定时器的初值已确保延时固定的时间。

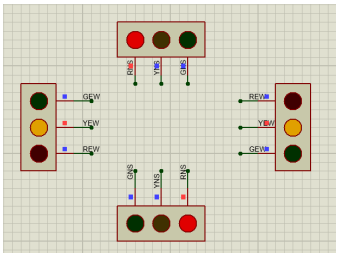
3. 软件仿真运行结果

最后实现的仿真结果是，先实现东西绿灯，南北红灯，经过约 2.5 秒后，转换为东西黄灯，南北红灯，经过 1 秒后，转换为东西红灯，南北绿灯，再经过约 2.5 秒后，转换为东西红灯，南北黄灯，经过 1 秒后又重新回到东西绿灯，南北红灯的初状态，如此循环。

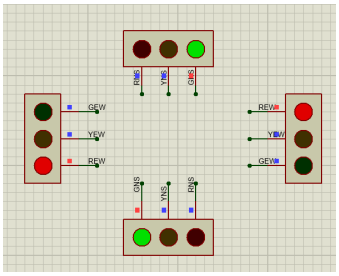
东西绿灯，南北红灯：



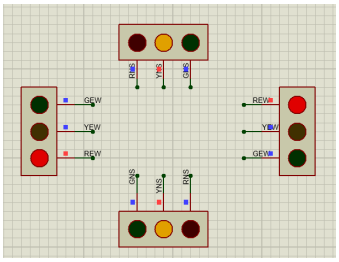
东西黄灯，南北红灯：



东西红灯，南北绿灯：



东西红灯，南北黄灯



六、 讨论、心得

感觉本次实验的代码部分与上次相比还是比较简单的，逻辑结构比较清晰。通过本次实验，我较好地掌握了定时器延时和软件延时的方法，在编写定时器延时的代码过程中，对中断的理解也加深了。最后成功通过单片机实现了一个简单的交通信号灯控制系统。