机电控制课程设计论文

# 交通信号灯控制系统

**论文作者** 孙　楠

**学　　号** 5080209271

**小组成员** 张力文 陈相帆 周游

**指导教师** 张银桥

2011 年 6 月

# 交通信号灯控制系统

## 摘　　要

机电控制技术在生产生活中发挥着巨大的作用，尤其是以单片机为核心器件的控制系统因其功耗低、可编程性强、扩展能力强等优点应用日益广泛。交通信号灯是社会维持正常交通秩序的重要工具。信号灯对交通参与者的友好性及其自身的可编程性，在机动车数量迅猛增长的当下凸显其重要性。在机电控制技术课程学习之后，我们尝试利用单片机技术及相关软硬件技术架设一交通信号灯控制系统，整合倒计时、对闯红灯者拍照等功能，并实现红绿灯持续时间的简便可调。本文将对该系统的设计和功能使用做出说明，并讨论一些可能的拓展功能。

由于本人在课程设计过程中主要负责汇编程序的编写，本文也将更为侧重软件部分：程序设计的思路、主要代码的解释、关键问题的探讨等。同时也会较对设计过程中软件设计和硬件设计之间的相互影响做一些阐述。

**关键词：** 单片机　控制　汇编

# 目　　录

# 第一章 设计背景及任务

## 1.1 设计背景

交通信号灯是城市交通行为中重要的环节，它带来有秩序的交通，引导人流车流、缓解拥堵、提高通行效率。

传统的交通信号灯简单地实现红黄绿等的切换，时间间隔难以调节，高峰时间和夜间红绿灯的间隔不变，更不可能根据不同路口的车流量做出调节。单调不变的红灯会加剧人的焦急感，人们要求信号灯能够有显示倒计时的功能。另外，对于闯红灯、超速等交通违规现象的监控，如需要整合到路口信号灯系统中。

要控制具有如上所有功能的交通信号灯系统，单片机因其运行速度快、功耗低、成本低廉、可编程性优越，接口丰富等优点成为控制核心的必然选择。

## 1.2 设计任务

结合设计背景，要求设计一个以 S52 单片机为核心的交通信号灯控制系统，同时对闯红灯车辆进行拍照。

### 1.2.1 基本设计要求

根据交通规则设计

1. 十字路口交通信号灯分为红灯、绿灯和黄灯，交替亮灭，保证车辆安全有序通行;
2. 亮灯时间可以设置;
3. 路口安装摄像监控装置，对于闯红灯车辆进行拍摄，存入录像机;
4. 操作简单。

### 1.2.2 扩展设计要求

在基本设计要求的基础上，结合日常的生活经验、以及我们对单片机技术的掌握，我们认为最终设计的系统还具有以下功能:

1. 可以实时显示两组红绿灯倒计时时间;
2. 相交的两条路应有不同的红绿灯时间;
3. 亮灯时间可以根据当前所处的时段自动调整;
4. 亮灯时间可以通过监控车流量等信息的上位机智能调节;
5. 同一路口的红绿灯之间有互锁控制;
6. 设置界面应友好易用。

# 第二章 硬件设计

本文将主要介绍软件部分的设计，通过对软件设计方案的描述，实现对整个系统功能的介绍。但是硬件部分仍是整个系统的基础，与软件设计也有着密切的关系。本章将对基本的硬件设计做一个浏览。并重点介绍与软件相关最为密切的部分。

## 2.1 硬件设计总体概况

本系统采用 AT89S52 单片机为控制核心，晶振电路产生 11.0592 MHz 的晶振频率，具有复位电路。输入部分有四个地感线圈信号（接 P1 口低四位），五个键盘按键输入信号（接 P2 口高五位）。输出部分驱动六个数码管，六个信号灯，四个拍照驱动信号。图2–1是控制部分电路图，附录B图B–2为其放大版。附录 B 中还有 PCB 版图等。

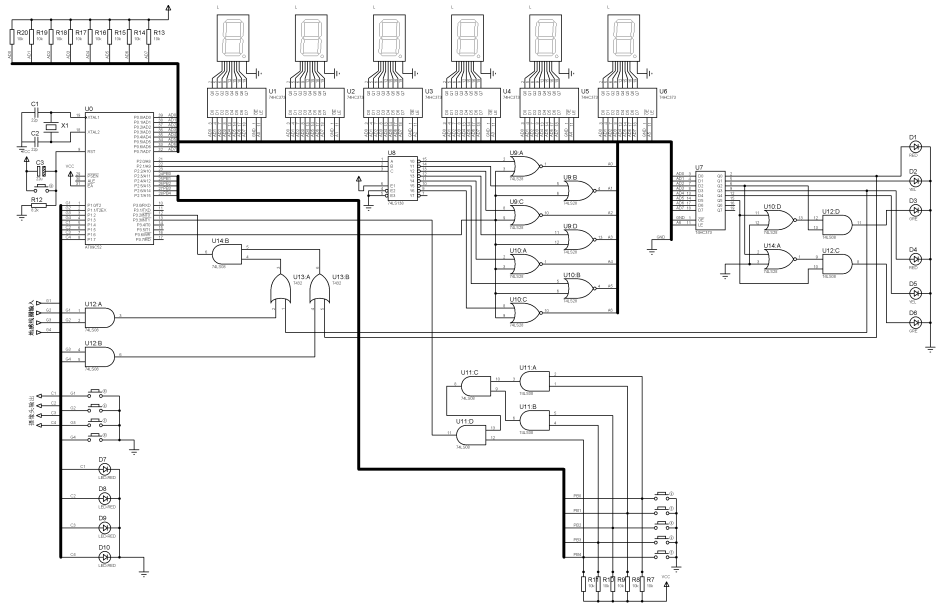

图 2–1 控制电路图

Fig 2–1 Control Circuit Figure

要实现规定的功能目标，通过估算，另外也考虑到 S52 单片机内部存储器足够大（8K），而最终生成的程序为 1.32Kb，RAM 空间也有剩余，故不需要进行存储器的扩展。

地感线圈处的输入信号需通过继电器与控制电路连接，具体的连接参数视地感线圈而定。信号灯与数码管的控制信号需要通过光电耦合器后放大再驱动相应器件。这些电路，在以上的控制电路图中并没有体现，但在完整的系统中，它们是同样重要的。

## 2.2 数码管及信号灯驱动电路

### 2.2.1 数码管

数码管采用共阴数码管。为了提高显示质量，每一个数码管均通过一个 373 锁存器驱动，共六个。这样做可以实现静态显示，数码不会闪烁。软件实现也大为便捷，CPU 占用率也降低许多。图2–2是数码管部分的电路图。

实际系统中，控制信号还需要经过放大耦合才能驱动倒计时显示器

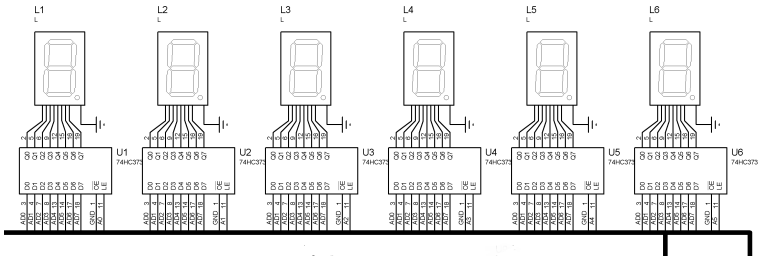


图 2–2 数码管驱动电路

Fig 2–2 Digital Led drive circuit

### 2.2.2 信号灯驱动电路

信号灯驱动电路如图2–3所示。因为相对的路的信号灯内容相同，所以两条路且每条路三盏灯，共六盏灯，通过一个 373 锁存器驱动。图中可见互锁电路由两个或非门（非门）和两个与门组成，则两个绿灯不能同时点亮。尽管软件中避免了两盏绿灯同时点亮的情况的出现，但硬件互锁还是必不可少的，毕竟软件在运行过程中可能会受到干扰而混乱，可能出现不可预料的情况。

图中所绘的小灯为演示所用，实际应将输出信号放大后，再驱动高电压的信号灯。附录B图B–3为信号后部放大驱动部分电路图。

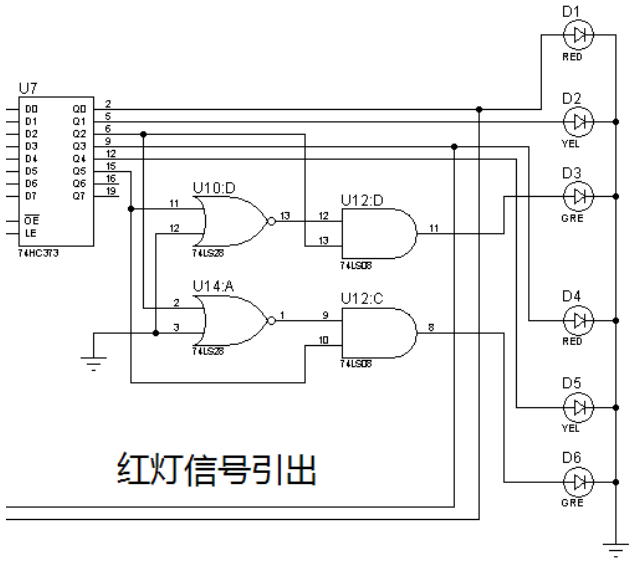


图 2–3 信号灯驱动电路

Fig 2–3 Signal light drive circuit

图中还可以看到两个红灯处引出了两路信号，这样做的目的将在2.4节详细阐述。

### 2.2.3　数码管及信号灯的选通

P0 口作为数据总线和七个 373 锁存器相连，P1 口的低三位作为地址线，再经过一个 38 译码器选通各锁存器。单片机的 $\overline{\mathbf{WR}}$ 信号与 38 译码器的七个输出端分别或非之后再选通锁存器。在对方案的重新评估中，我们发现也可以令 WR 信号直接选通 138 译码器，但考虑到译码器的输出端仍需通过非门，器件数量并未减少太多，故仍保留最初的设计。

## 2.3　键盘

图2–4是键盘的电路图。因为 P2 口正好还有五个 IO 口可用，五个键也正好可以实现设置的要求，所以采用的独立键盘。这样对于编程工作也是很大的便利。各键的定义在3.3.1节介绍。

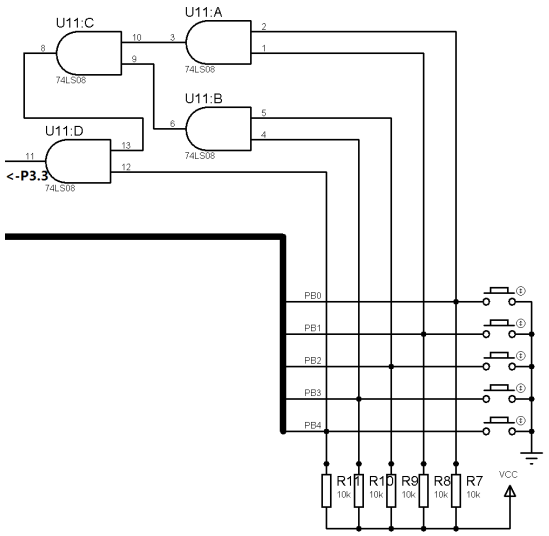


图 2–4 键盘电路

Fig 2–4 Keyboard circuit

按键触发后 P2 口有下降沿信号，这个信号全部相与后接 P3.3（$\overline{\textbf{INT1}}$），可产生中断信号，在软件中再通过轮询确定具体按下的键位。

## 2.4 地感线圈输入与中断触发

地感线圈输入信号送入 P1 口低三位，并触发 $\overline{\textbf{INT0}}$ 中断。在电路图中地感线圈的输入以开关代替，实际信号也是一开关量，但由地感线圈转换至单片机控制电路输入信号还需要一些处理。

由图2–5可见，此电路与普通的外部中断扩展电路稍有不同，在2.2.2节已提到，两路红灯信号由右侧引入，分别与另一条路的地感线圈输入信号取或运算，然后在送入 $\overline{\textbf{INT0}}$ 口。这是为了在某一路为绿灯时，屏蔽其地感线圈的信号，使之不能产生中断。只有在某一路为红灯时，闯红灯的车辆才能触发地感线圈，继而触发单片机中断。这样大大减少了中断的产生大大降低了 CPU 的工作量，提高了反映速度。编程难度也有所减轻。

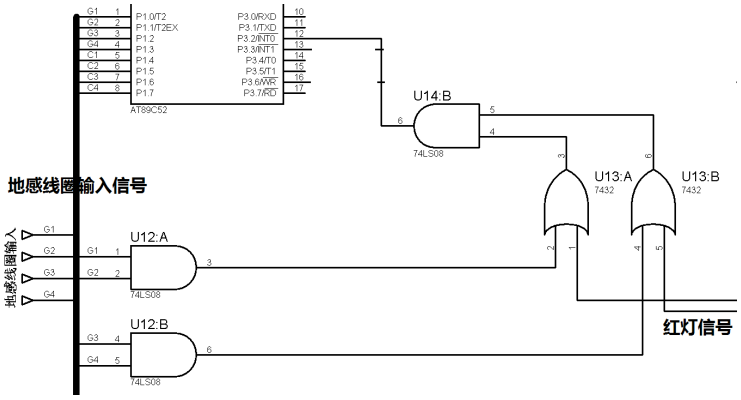


图 2–5 地感线圈输入电路

Fig 2–5 Ground wire input

## 2.5　小结

除了以上介绍的主要输入输出电路外，我们还设计了上电复位电路，晶振频率为 11.0952MHz 的时钟电路，P0 口加入了上拉电阻。由于这些电路都较为常规，在此不再详述。

通过以上的分析可见，我们的设计对单片机 IO 口的利用非常充分 P0，P1，P2 口均已占满。通过充分利用 IO 口实现了数码管和信号灯的静态显示、独立键盘、独立的地感线圈输入，减少了元器件的使用，减小了编程难度。

硬件电路设计中也通过一些逻辑电路的组合，实现了互锁、条件屏蔽等功能。

# 第三章　系统功能设计

本章起到一个功能说明书的作用，将抛开底层的硬件和具体的软件实现，面向普通用户，介绍本系统的主要功能、工作流程，以及设置界面的使用方法。本章也作为下一章软件设计的先导，软件设计都是围绕着以下功能的实现而进行的。

合理的硬件设计也是以下功能实现的重要基础。

## 3.1　信号灯交替流程

红黄绿三色信号灯按照图3–1所示的工作顺序交替闪亮。由于相交的两条路交通状况可能相差很大，其红灯持续时间是不同的。两个红灯时间 T1 和 T2 可以分别设置[1]，黄灯时间默认为 3 秒。由图可见一个亮灯循环只需要两个独立参数（T1 和 T2）就可以确定，绿灯时间比相应红灯时间短 3 秒。

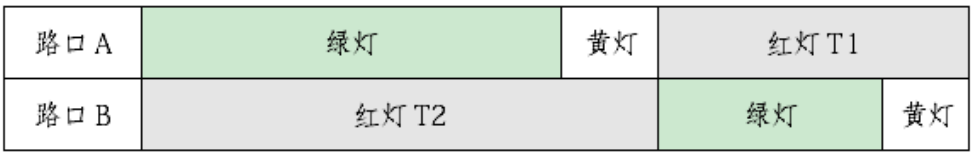


图 3–1 亮灯顺序循环图

Fig 3–1 Signal light changing pattern

各个路口均悬挂有倒计时显示器，两组相对的路口显示内容分别相同。倒计时的内容为当前亮的灯将要持续的时间。当显示完 **00**之后，灯的状态改变，倒计时显示屏也重新开始计数。

## 3.2　可调的亮灯时间

所谓亮灯时间的可调性，就是指图3–1中的循环参数 T1 和 T2 是可变的，这是为了适应不同的交通状况，车流量大时缓解拥堵，车流量小时减少等待。调节亮灯时间的方式有以下三种。

[1]下文将称这两个参数为亮灯时间参数

### 3.2.1 预置数据表调节

在单片机的 ROM 中预先存放了一数据表，一天中的每小时[2]都有存储了两个数据在表中，分别代表 A 路口和 B 路口的红灯时间。则此表格共包含 48 个数据。单片机会根据当前的时间段[3]，自动切换亮灯时间。

### 3.2.2 用户自定义亮灯时间

本系统还提供了一人机界面（包括 5 个按键和 2 位数显屏）供用户随时调节亮灯时间。用户可以调节任一时间段的亮灯时间，也可以重新定义当前所处的时间段。若用户调节了当前时间段的亮灯时间，则在当前红灯倒计时变为 0 后立即生效。3.3节会详细介绍该方式的使用方法。

时间参数存储在 RAM 中，单片机复位后会全部清零。用户定义的时间参数优先级比内置参数表高。

### 3.2.3 双机通信调节

单片机系统中留出了接口，上位机可以通过串行接口向单片机发出指令，直接改变当前时间段的亮灯时间参数。上位机可以是自动监控车流量的单片机，也可以是交通管理网络中的计算机。这种调节方式和通过键盘的调节方式，后设置的参数会覆盖之前覆盖的参数，所以它们的优先级是一致的。

## 3.3 键盘屏显接口使用方法

用户通过键盘可以实现两个功能。

1. 设定任一时段两路口的红灯时间参数，该参数存入 RAM 中。
2. 当用户设定的数据为两个 00，则更改目前的时间区段为用户设定的值，这一功能可用于在单片机复位后初始化当前时区。

在设定过程中，数字显示的相应位会闪烁，提示用户目前正在操作的位的位置。

[2]一天中划分为 24 个时间段，从 0 到 23，分别对应一天中的第一到第 24 个小时。

[3]单片机复位后，用户可通过键盘设定初始时区为当前时间段，如图3–2，之后单片机即从此时区开始自动累加

### 3.3.1 键位定义

表3–1详细列出了各个按键的定义，了解了这张表基本就可以尝试操作了。但是由于屏显只有两位数码管，刚开始也许还会有些费解，还需要参考3.3.2节给出的一个完整的设置流程图。

表 3–1 键位定义

Table 3–1 keyboard defination

| 键名 | | 端口 | 功能 |
|---|---|---|---|
| 开关 | on/off | p2.3 | 开关显示屏。开屏则首先显示当前时间段代码。关屏则重置键盘接口，清空所有寄存器。 |
| 确认键 | ok | p2.4 | 确认当前选择，并前进到下一选择界面 |
| 选位键 | <> | p2.5 | 切换选择高低位。在设定数码时，用于在个位十位间切换，被选择的位会闪烁。设定时间段时，该键不起作用。 |
| 加一键 | + | p2.6 | 使屏显的内容加一。在红灯时间调节状态下，对闪烁的位加一。若超出最大值，则变为最小值。 |
| 减一键 | - | p2.7 | 使屏显的内容减一。在红灯时间调节状态下，对闪烁的位减一。若小于最小值，则变为最大值。 |

### 3.3.2 键盘设定操作流程

图3–2 是键盘相关的完整流程图。从图中可以明确看到，通过键盘设定可以完成两个功能。

需要补充的一点是，在成功设置某一时段的时间参数后，程序将检查设置的是否就是当前时段，如果是的话，会重新载入该时间常数。故更改会在当前红灯倒计时完毕后立即生效。

### 3.3.3 强壮性设计

如果用户向信号灯系统定义了两个 0 为时间参数，显然这样的定义是没有意义的，通过软件已经将这样的输入转换为对当前时间段的重定义。本节就将

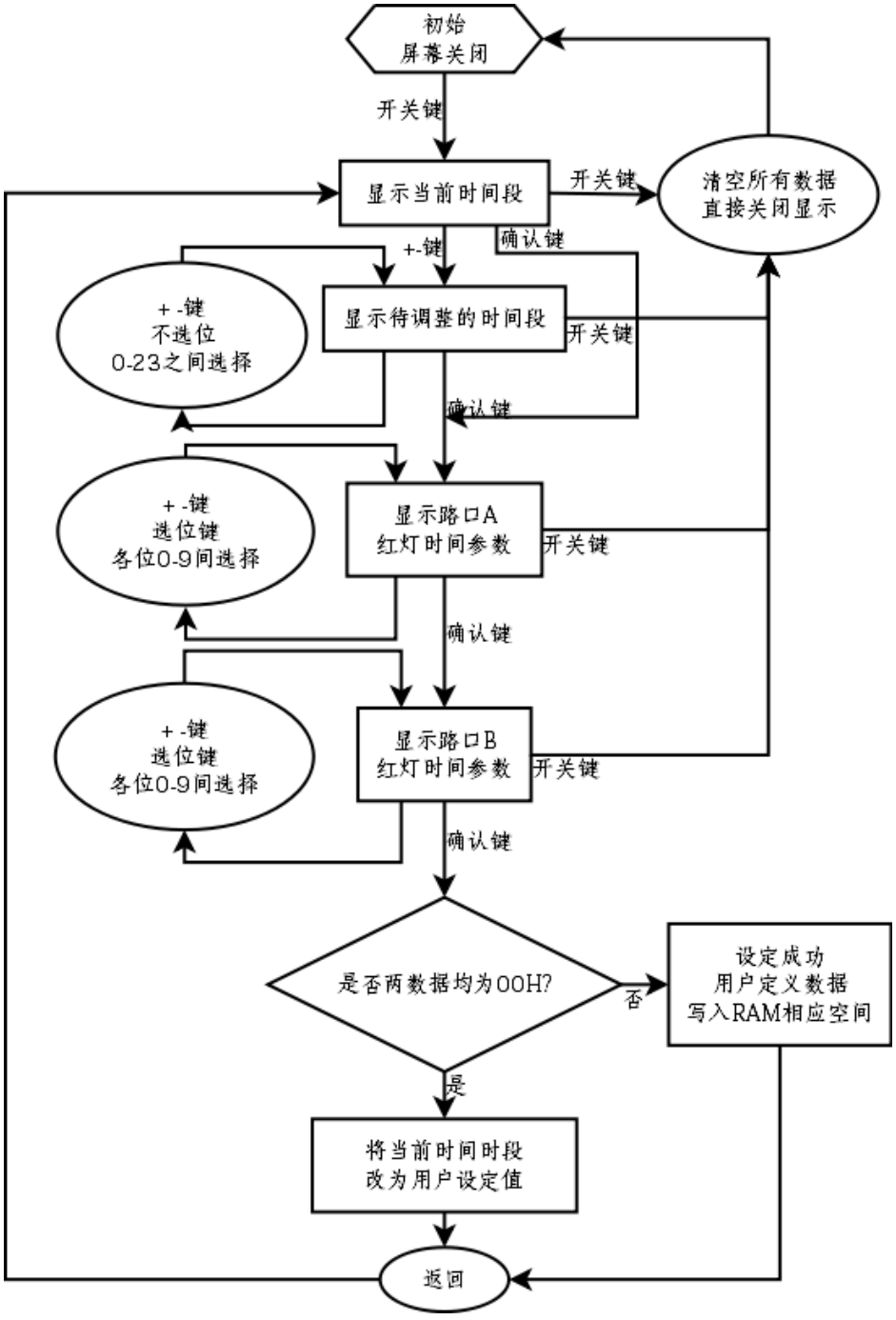


图 3–2 键盘操作流程图

Fig 3–2 keyboard control flow figure

讨论用户对用户各种输入数据的处理。

因为黄灯时间需要持续三秒，所以红灯时间必须大于 3 秒，任何小于三秒的定义也是无效的。

表 3–2 键盘输入处理

Table 3–2 keyboard input interpretation

| 用户输入 | | 相应处理 | |
|---|---|---|---|
| T1 | T2 | | |
| 0 | 0 | 设置当前时间段 | |
| $\geq 0, < 3$ | $\geq 0, < 3$ | 无效输入，不操作 | |
| $\geq 0, < 3$ | 有效输入 T2 | T2 写入 A 路口对应 RAM | T2 写入 A 路口对应 RAM |
| 有效输入 T1 | $\geq 0, < 3$ | 写入 T1 | 写入 T1 |
| 有效输入 T1 | 有效输入 T2 | 写入 T1 | 写入 T2 |

表3–2列出了软件中对于各种输入情况的处理。由表可见，程序会自动消除用户的错误输入。默认的修正原则是：如果两个数据都错误则抛弃；如果其中有一个数据有效，则另一个路口的时间参数取相同的值。

## 3.4 闯红灯拍照功能

本系统还具有对闯红灯的车辆拍照的功能。需要说明的是，四个路口分别布置了地感线圈，只有在该路为红灯状态时，地感线圈发出的信号才有效。系统可以响应两个相对的路口同时闯红灯的状况，虽然这种情况的几率并不大。

# 第四章 软件设计及问题研讨

本章是本篇论文的重点，因为本人在本次课程设计工作中主要做的的就是程序设计，对于这一部分有许多的了解最为深刻。首先我将阐述程序设计的基本思路，对重要的变量和子程序做解释。另外还将就程序设计中一些比较关键的问题做分析。

## 4.1 程序概览

本程序用全部用汇编语言编写，整体大约六百四十余行，编译后约占1.3Kb。附录A提供了完整的源代码。

### 4.1.1 红灯时间参数的存储和调用

本程序中最重要的参数就是两个路口的两个红灯时间参数，倒计时、切换灯的状态、设置时间参数等都是围绕这两个参数进行的。要解释清楚该程序，首先就需要解释这个参数的不同形式，和它们的存储和调用方式。表4–1列出了相关的参数作用和存储方式。在这些存储空间中，数据都是以 BCD 码的形式存储的，这是为了用户输入和显示的方便[1]。

这个表中数据是从底部行流向顶部行描述的存储空间的。而最后两行展示了 ROM 表与 RAM 表之间的优先级关系。

### 4.1.2 中断的应用

为了提高程序响应的速度，减少 CPU 的工作量，所有功能都移入中断处理程序中，主函数只起到初始化的作用。表4–2列出了中断的应用方式。中断优先级的设置将在4.2.2中讨论。此外，若要实现双机多机通讯，串口中断自然也需要应用。

[1]但是本程序中标志时区的量都是以十六进制的形式存储的，这是因为时区参数与查表有关，而表格时连续的，16 进制的时区参数对于查表是便捷的。

表 4–1 时间参数表

Table 4–1 Time registers

| 存储形式 | 参数功能 |
| --- | --- |
| R2 R3 | 分别存储当前两个倒计时数码管组显示的内容，R2 为红灯，R3 为绿灯 (或黄灯) |
| 30H 31H | 存储当前时段的两个红灯时间参数，倒计时到 0 后，R2 和 R3 会从这两个字节重新载入倒计时时间。三种情况改变这两个字节的值:<br>• 当一个小时结束时;<br>• 用户通过键盘定义了当前时段的时间参数时，立即刷新，立即生效;<br>• 上位机发送包要求改变这两个位时。 |
| ROM 中的表 #TAB_LIGHT _TIME | 是预置在 ROM 中的时间参数表，按小时排列，每小时两个，共 48 个字节。当一小时计数到时，30H 和 31H 从这个表中载入新的数据。 |
| RAM 中 50H 到 7FH | 这 48 个字节存储了用户通过键盘定义的时间参数数据。它的优先级比 ROM 中的表要高。但如果这个表中的数据为 0。则视为未定义，程序仍从 ROM 表中读取数据存入 30H 和 31H。 |

表 4–2 中断应用表

Table 4–2 Interrupt vectors

| 中断优先级 | 中断名 | 中断处理程序 | 中断功能 |
|---|---|---|---|
| 1 | 计时器 0 | INT_T0 | 计时 $\frac{1}{144}$ 秒，清零看门狗，取反计数器 2 的输入端 |
| 2 | 计数器 1 | INT_C1 | 计数 36，即 0.5 秒，中断处理程序完成倒计时等工作 |
| 3 | 外部中断 0 | INT_EX0 | 响应地感线圈输入信号 |
| 4 | 外部中断 1 | INT_EX1 | 响应按键输入信号 |

### 4.1.3 子程序一览

| 序号 | 子程序名 | 主要功能 | 主要参量 | 包含子程序 |
|---|---|---|---|---|
| 1 | INT_T0 | 清零看门狗，取反计数器 2 的输入端 | | |
| 2 | INT_C1 | 倒计时减一显示，切换灯状态，计时，重载时间参数 | R0 灯状态，R6 秒计数，R7 分计数 | SUBBCD, CLOSEDIG, DIS-PLAY_NUMBER, CHANGE-LIGHT, GET_TIME_LIGHT 等 |
| 3 | INT_EX0 | 响应地感线圈输入信号，向摄像头输出开关量控制信号 | | DELAY10 |
| 4 | INT_EX1 | 响应按键输入信号，完成键盘操作流程所要求的功能。 | R4 存储当前屏幕状态（显示内容的标志），3AH 当前选择的时间段，3BH 路口 A 的正在被设定的数据，3CH 相应的路口 B 的数据位 7F 标志当前调整的是高位还是低位 | DELAY10, DELAY4MS5, HEX2BCD, DIS-PLAY_NUMBER, GET_TIME_LIGHT, BCDINC, BCD-DEC |
| 5 | BCDINC | 根据选位标志 7F 对高位或低位做加一运算，对 9 加一得到 0 | 3DH 传递被处理的数据 | |

| | | | | |
|---|---|---|---|---|
| 6 | BCDDEC | 根据选位标志 7F 对高位或低位做减一运算，对 0 减一得到 9 | 3DH 传递被处理的数据 | |
| 7 | CLOSEDIG | 根据选位标志，关闭某一位的显示 | 3EH 传递被处理的数据 | 通过调用 DISPLAY _NUMBER 送显示 |
| 8 | HEX2BCD | 将 16 进制数转为 BCD 码，主要用于时区的显示 | 38H 传递被处理的数据 | |
| 9 | GET TIME LIGHT | 根据当前的时区，刷新 30H 和 31H 中的数据 | 3DH 传递被处理的数据 | |
| 10 | CHANGE LIGHT | 根据目前的灯状态，选通灯锁存器，点亮相应灯 | R0 指向存储亮灯数值的单元 (32H31H) | |
| 11 | DISPLAY NUMBER | 将 BCD 码显示到数码管上，一次显示相邻的两位 | 38H 存放 BCD 码，39H 存放低位数码管的地址 | GETDIGIT, DISDIGIT |
| 12 | GETDIGIT | 查表取数码管段码 | 取得段码存入 38H | |
| 13 | DISDIGIT | 将 38H 中的段码送入数码管显示 | | |
| 14 | SUBBCD | BCD 码减一 | 对 36H 中的 BCD 码操作 | |

## 4.2 问题研讨及程序分析

### 4.2.1 定时程序的设计

本程序中利用 T0 和 T1 串联产生 0.5 秒的定时。T0 定时到后，取反 P3.5 位，即送给 T1 一个二分频的计数信号。主程序放在 T1 中断处理程序中执行，提高了响应速度和计时准确性。理论上 T0 的定时时间应越长越好，这样与 T1 同时触发中断的概率越小，但是考虑到 S52 单片机看门狗需要每隔 8192 个机器周期就会产生复位信号，而 T0 的中断处理程序中清零看门狗，故 T0 的计数周期必须小于 8192 个机器周期。实际设定 T0 每 6400 个机器周期产生中断，计数器 T1 的计数个数为 36。

若按照以上设定，则 T0 的时间常数为 #E700H。但是考虑到本程序要在一天中累计小时数，计时精度要求较高。仿真测试显示该时间常数还需要修正。经估算和仿真测试发现，T0 的中断处理程序约有 9 个周期的滞后，故修正时间常数为 #E709H。

计数器 1 的中断处理程序长度远小于 6400×2 个机器周期，故与计时器 0 的中断不会发生冲突。

代码 4.1 时间常数设置

```
1    MOV TMOD,#61H  ;初始化计时器定时器 0 方式 1 计数器 1 方式 2
2    MOV TH0,#0E7H
3    MOV TL0,#09H
4    MOV TH1,#0DCH ;256-36
5    MOV TL1,#0DCH
```

经仿真测试，按照如代码4.1设置后，运行时间如表4–4所示，每秒偏差小于 $5\mu s$，每天的偏差约为 0.4s 还是比较精确的。代码中 T0 中断中还需先插入一 nop 语句。

表 4–4 定时偏差

Table 4–4 Time deviation

| 每次 T1 中断时的时刻 (ms) | | | | |
|---|---|---|---|---|
| 493.44 | 993.44 | 1493.43 | 1993.43 | 2493.43 |
| 2993.43 | 3493.43 | 3993.42 | 4492.42 | 4492.42 |

### 4.2.2 中断优先级的设计

代码 4.2 中断优先级设置

```
1    MOV IP,#00001010B
2    ; EA | - | PT2 | PS | PT1 | PX1 | PT0 | PX0
3    ; 1  | 0 |  0  |  0 |  1  |  0  |  1  |  0
```

表4–2已经给出了中断优先级的排列顺序。T0 中断的优先级最高，因为它是精确定时的关键。T1 计数器次之，设为高优先级是防止其被持续时间较长的按键中断阻断。T1 中断程序关乎倒计时的正确显示，而键盘中断可能长达数百毫秒，若将 T1 阻断，则倒计时也将延迟数百毫秒，这是不能接受的。两个外部中断都处于低优先级，除了它们耗时较长不能阻断计时中断的原因，它们本身也不需要非常迅速的响应，用户的的反应在百毫秒级别，闯红灯拍照计时延迟一秒之久，也仍能拍到肇事车辆。

### 4.2.3 看门狗程序

根据文献 [1], 设计看门狗程序如代码所示。S52 单片机内置看门狗硬件，它的作用是防止程序跑飞。但也存在一个问题：复位后用户定义的数据将被清空。

代码 4.3 看门狗程序

```
1    ;主程序中喂狗
2    MOV 0A6H,#01EH ;激活看门狗
3    MOV 0A6H,#0E1H
4
5    ;INT_T0 中断处理程序中清零看门狗
6    INT_TO:                          ;计时器 0 中断处理程序
7    nop
8    MOV TH0,#0E7H
9    MOV TL0,#09H
10   MOV 0A6H,#01EH          ;清零看门狗
11   MOV 0A6H,#0E1H
12   CPL P3.5
13   RETI
```

### 4.2.4 双机通讯

双机通讯由于难以仿真，故没有放到最终的源程序中。但我们考虑到了这个功能，并设计了它的接口程序。

代码 4.4 双机通训代码

```
;主机从机设置相同的波特率，为 1/64 f_osc
MOV PCON #80H
MOV SCON #80H
;从机还需要设置 SETB SM2
;从机中断处理程序
CLR RI
如果是SBUF#，则FFHCLR SM2 开始接收数据
接受一帧地址a
接受一帧红灯时间，直接存入a
如果是SBUF#，则FEHSETB SM2 结束接受数据
;主机发送包程序
SETB TB8
发送 #0FFH
CLR TB8
发送一帧地址
发送一帧数据
发送#0FEH 结束发送
```

### 4.2.5 按键去抖

在按键中断处理程序中，调用 DELAY10 延时 10ms 进行前沿去抖。考虑到键盘处理程序较长，且最后总要送数码管显示，所以去后沿抖动时仅延时 4.5ms。而且去抖是通过循环判断是否已撤键，避免对同一次按键的多次响应。这也要求延时程序不能耗时太长，避免过长的延迟。

### 4.2.6 BCD 码的处理

在本程序中 BCD 码的应用很多。

通过 HEX2BCD 程序将 16 进制数转换为 BCD 码，主要思路是除以 10。

BCD 码的加一可通过以下命令简单实现，汇编中的 DA 命令必须紧跟在 ADD 或 ADDC 命令之后。

代码 4.5 BCD 码加一，结果存入 ACC

```
1        MOV A, 码BCD
2        ADD A,#1
3        DA A
```

但是汇编中对减一没有相应的命令，本程序中有子程序 SUBBC 可以实现此功能。主要思路是如果出现借位，个位会出现 F，则再减 6 将个位变为 9 即可。

代码 4.6 SUBBCD 程序，BCD 码减一

```
1        SUBBCD:                    ;BCD 码减一, 对 36H 中的数做 BCD 码减 1
2        PUSH ACC
3        PUSH PSW
4        DEC 36H
5        MOV A,36H
6        ANL A,#0FH
7        CJNE A,#0FH,SUBBCD_EXIT
8        CLR CY
9        MOV A,36H
10       SUBB A,#06
11       MOV 36H,A
12   SUBBCD_EXIT:
13       POP PSW
14       POP ACC
15       RET
```

本程序中另外写有子程序 BCDINC 和 BCDDEC，它们是为了配合键盘操作中对屏显数字做加减操作而设计的。除了实现个位和十位的加减，它还考虑到了 0-9 之间循环的问题。由于键盘操作的要求比较复杂，故直接编写了这两个新子程序，而没有重用 SUBBCD 程序。而且 SUBBCD 程序调用的频率很高，比键盘操作的频率高的多，所以程序不宜写的太臃肿，运行效率是考虑的一个重要因素。

# 全文总结

本文对本次课程设计工作作了总结。首先介绍了硬件设计部分设计。有了硬件设计的基础之后，功能实现成为可能。第三章阐述了信号灯系统具体的功能，并对各种调节时间参数的方式和键盘操作了详细介绍。第四章则主要介绍了软件程序的结构和设计思路，对一些关键问题如定时参数、中断优先级等作了讨论。

总体来看，我们设计的系统达到了预先设定的要求，也完成了我们自己定下的扩展要求。仿真效果良好，反应迅速，定时精确，操作便捷。但是我们也认识到，将这个系统投入实际应用，硬件调试仍是必须的。

**待增加的功能**

以下这些功能主要也是软件实现的问题，难度并不大，但是由于时间紧迫，目前的程序并未包含，今后如有机会继续改进，可以考虑增加这些功能。

1. 通过键盘切换入紧急状态下，此状态下所有路口均显示红灯，倒计时不工作。
2. 通过键盘切换入夜间模式，此状态下所有路口均有闪烁的黄灯。
3. 加入黄灯闪烁的功能。
4. …

# 附录 A 源程序

```
;;————————————
;;TrafLight v2.0
;;by SN icetiny@gmail.com
;;2011-6-16
;;————————————
;;R0      | R1        | R2           | R3
;;灯状态 | 临时使用 | 2 红灯倒计时 | 1 绿灯倒计时
;; R6        |  R7   |  R5
;;秒状态数码 | 分状态 | 空
;;    R4
;;屏状态（0 开 1 选段 2 调路口 A3 调路口 B）
;;————————————
;;30H   |  31H  |  32H-35H   |  36H   |  37H
;;红灯 A | 红灯 B | 灯状态地址 | 减一用 | 当前小时状态码
;;————————————
;; 38H       |  39H             |  3AH        |  3BH    |  3CH
;;LED 显示寄存器 | 数码管地址高八位 | 屏段（HEX 码）| 屏红灯 A | 屏红灯 B
;;————————————
;; 位 7F   |  3DH                          |  3EH
;;选位标志 | BCDINC 和 BCDDEC 子程序的操作位 | 闪烁关灯程序数据传递位
;;————————————
;;50H-7FH        |  40H 41H            |  位 7E
;;用户定义灯状态 | delay10 中用到的变量 | 区分一秒中的上下半秒
;;————————————
;; 键盘
;;1      |  2      |  3     | 4 | 5
;;on/off | ok/next | <> 选位 | + | -
;;————————————

        ORG 0000H
        AJMP MAIN
;————————-INTERRUPT VECTORS————————————
        ORG 0003H ;外部中断 0
        LJMP INT_EX0
        ORG 000BH
        LJMP INT_TO
```

```
37          ORG 0013H ;外部中断 1
38          LJMP INT_EX1
39          ORG 001BH
40          LJMP INT_C1
41  ;————————DEFINE CONSTANT VALUE————————————
42          LDD1 EQU 0f8H ;数码管地址
43          LDD2 EQU 0f9H
44          LDD3 EQU 0faH
45          LDD4 EQU 0fbH
46          LDD5 EQU 0fcH
47          LDD6 EQU 0fdH
48          LED  EQU 0feH ;灯地址
49          G_R1 EQU 1100B
50          Y_R2 EQU 1010B
51          R_G3 EQU 100001B
52          R_Y4 EQU 10001B
53          SECS_PER_MIN EQU 4  ;每分中的秒数，调试用
54          MINS_PER_HOUR EQU 5 ;每小时中的分数，调试用
55  ;————————————————————————————————
56  ;;————————MAIN PROGRAM BEGIN————————————
57          ORG 0030H
58  MAIN:
59          MOV SP,#10H
60          MOV P1,#0FH
61          MOV 32H,#G_R1
62          MOV 33H,#Y_R2
63          MOV 34H,#R_G3
64          MOV 35H,#R_Y4
65          MOV TMOD,#61H   ;初始化计时器定时器 0 方式 1 计数器 1 方式 2
66          MOV TH0,#0E7H
67          MOV TL0,#09H    ;2^16 − 6400 = 59136 = E700, E700 + 7 = E707
68          ;12 ∗ 2 ∗ 72 ∗ 6400 = 11.0592MHz，取得较小是为了照顾 WDT 看门狗
69          MOV TH1,#0DCH
70          MOV TL1,#0DCH   ;36. 0.5S ;
71          MOV R6,#SECS_PER_MIN ;每分中的秒数，调试用
72          MOV R7,#MINS_PER_HOUR ;每小时中的分数，调试用
73          MOV 37H,#8      ;初始化当前时间（小时状态）
74          LCALL GET_LIGHT_TIME ;获取倒计时时间存入 30H,31H
75          MOV R0,#32H ;R0 记录信号灯寄存器状态
76          MOV R2,30H
77          MOV R3,30H
78          MOV 36H,R3
```

```
79          LCALL SUBBCD
80          LCALL SUBBCD
81          LCALL SUBBCD
82          MOV R3,36H
83          LCALL CHANGE_LIGHT ;开信号灯
84          MOV 38H,R3    ;送显示，开数码管
85          MOV 39H,#LDD2
86          LCALL DISPLAY_NUMBER
87          MOV 38H,R2
88          MOV 39H,#LDD4
89          LCALL DISPLAY_NUMBER
90          ;中断优先级处理
91          MOV IP,#00001010B   ;优先级依次为 t0 t1 x0 x1
92          MOV IE,#10001111B ; EA | - | ET2 | ES | ET1 | EX1 | ET0 | EX0
93                          ; 1  | 0 |  0  |  0 |  1  |  1  |  1  |  1
94          MOV TCON,#05H      ;IT0=1 IT1=1
95          SETB TR0           ;开定时器
96          SETB TR1
97          MOV 0A6H,#01EH ;激活看门狗
98          MOV 0A6H,#0E1H
99          ;初始化完毕，开始计时
100         SJMP $
101 ;;————— END OF MAIN —————
102 INT_TO:                          ;计时器 0 中断处理程序
103         nop
104         MOV TH0,#0E7H
105         MOV TL0,#09H
106         MOV 0A6H,#01EH           ;清零看门狗
107         MOV 0A6H,#0E1H
108         CPL P3.5
109         RETI
110
111 INT_C1:                          ;计数器 1 中断处理程序
112         PUSH ACC
113         CPL 7EH
114         JB 7EH,INT_C1_NORMAL
115 INT_C1_BLINK:
116         MOV A,R4
117         JZ INT_C1_BLINK_EXIT
118         MOV A,R4
119         CLR C
120         SUBB A,#2
```

```
121         JBC CY,INT_C1_BLINK_EXIT
122         JNZ INT_C1_BLINK_RED
123         MOV 3EH,3BH
124         LCALL CLOSEDIG
125         LJMP INT_C1_EXIT1
126 INT_C1_BLINK_RED:
127         MOV 3EH,3CH
128         LCALL CLOSEDIG
129 INT_C1_BLINK_EXIT:
130         LJMP INT_C1_EXIT1
131 INT_C1_NORMAL:
132         MOV A,R4
133         JZ INT_C1_NORMAL1
134         MOV A,R4
135         CLR C
136         SUBB A,#2
137         JBC CY,INT_C1_NORMAL1
138         JNZ INT_C1_UNBLINK_RED
139         MOV 38H,3BH
140         MOV 39H,#LDD6
141         LCALL DISPLAY_NUMBER
142         LJMP INT_C1_NORMAL1
143 INT_C1_UNBLINK_RED:
144         MOV 38H,3CH
145         MOV 39H,#LDD6
146         LCALL DISPLAY_NUMBER
147 INT_C1_NORMAL1:    MOV 36H,R3
148         LCALL SUBBCD
149         MOV R3,36H
150         MOV 36H,R2
151         LCALL SUBBCD
152         MOV R2,36H
153         CJNE R2,#0F9H,INT_C1_NEXT2
154         INC R0
155         CJNE R0,#36H,INT_C1_NEXT0
156         MOV R0,#32H
157         LCALL CHANGE_LIGHT
158         MOV R2,30H
159         MOV R3,30H    ;绿灯时间比红灯时间短 3 秒
160         MOV 36H,R3
161         LCALL SUBBCD
162         LCALL SUBBCD
```

```
163         LCALL SUBBCD
164         MOV R3,36H
165         SJMP INT_C1_EXIT
166 INT_C1_NEXT0:
167         CJNE R0,#34H,INT_C1_NEXT1
168         LCALL CHANGE_LIGHT
169         MOV R2,31H
170         MOV R3,31H    ;绿灯时间比红灯时间短 3 秒
171         MOV 36H,R3
172         LCALL SUBBCD
173         LCALL SUBBCD
174         LCALL SUBBCD
175         MOV R3,36H
176         SJMP INT_C1_EXIT
177 INT_C1_NEXT1:
178         CJNE R0,#35H,INT_C1_NEXT2
179         LCALL CHANGE_LIGHT
180         MOV A,R3
181         XCH A,R2
182         SJMP INT_C1_EXIT
183 INT_C1_NEXT2:
184         CJNE R3,#0F9H,INT_C1_EXIT
185         INC R0
186         LCALL CHANGE_LIGHT
187         MOV A,R2
188         XCH A,R3
189 INT_C1_EXIT:
190         MOV A,R0
191         CLR CY
192         SUBB A,#33H
193         JNB CY,INT_C1_REVERSEDISPLAY
194         MOV 38H,R3    ;送显示
195         MOV 39H,#LDD2
196         LCALL DISPLAY_NUMBER
197         MOV 38H,R2
198         MOV 39H,#LDD4
199         LCALL DISPLAY_NUMBER
200         LJMP INT_C1_EXIT00
201 INT_C1_REVERSEDISPLAY: ;调换显示的内容（原来显示红灯倒计时，现在则显示绿
        灯倒计时）
202         MOV 38H,R2
203         MOV 39H,#LDD2
```

```
204         LCALL DISPLAY_NUMBER
205         MOV 38H,R3
206         MOV 39H,#LDD4
207         LCALL DISPLAY_NUMBER
208 INT_C1_EXIT00:
209         DJNZ R6,INT_C1_EXIT1
210         MOV R6,#SECS_PER_MIN
211         DJNZ R7,INT_C1_EXIT1
212         MOV R7,#MINS_PER_HOUR
213         INC 37H
214         MOV A,37H
215         CJNE A,#24,INT_C1_EXIT0
216         MOV 37H,#0
217 INT_C1_EXIT0:
218         LCALL GET_LIGHT_TIME
219 INT_C1_EXIT1:
220         POP ACC
221         RETI
222 ;————————-地感线圈中断处理 ————————-
223 INT_EX0:
224         PUSH ACC
225         MOV A,P1
226                     CJNE R0,#32H,INT_EX0_NEXT1
227                     LJMP INT_EX0_SHOOT2
228 INT_EX0_NEXT1:    CJNE R0,#33H,INT_EX0_NEXT2
229                     LJMP INT_EX0_SHOOT2
230 INT_EX0_NEXT2:    CJNE R0,#34H,INT_EX0_NEXT3
231                     LJMP INT_EX0_SHOOT1
232 INT_EX0_NEXT3:    CJNE R0,#35H,INT_EX0_EXIT
233 INT_EX0_SHOOT1:   JB ACC.0,INT_EX0_SHOOT11
234                     SETB P1.4
235 INT_EX0_SHOOT11:
236                     JB ACC.1,INT_EX0_EXIT
237                     SETB P1.5
238                     LJMP INT_EX0_EXIT
239 INT_EX0_SHOOT2:   JB ACC.2,INT_EX0_SHOOT22
240                     SETB P1.6
241 INT_EX0_SHOOT22:
242                     JB ACC.3,INT_EX0_EXIT
243                     SETB P1.7
244 INT_EX0_EXIT:
245         MOV A,#30
```

```
246 INT_EX0_EXIT0:
247         LCALL DELAY10
248         DJNZ ACC,INT_EX0_EXIT0
249 INT_EX0_EXIT1:
250         MOV P1,#0FH
251         POP ACC
252         RETI
253 ;————————按键中断处理程序 ————————-
254 INT_EX1:                                ;外部中断 1 处理，键盘
255         PUSH PSW
256         PUSH ACC
257         MOV A,P2   ;判断是否有键按下
258         ANL A,#0F8H
259         XRL A,#0F8H
260         JZ INT_EX1_EXIT
261         LCALL DELAY10
262         MOV A,P2   ;再次判断是否有键按下, 消除前沿抖动
263         ANL A,#0F8H
264         XRL A,#0F8H
265         JZ INT_EX1_EXIT
266         JB ACC.3,KEY01
267         JB ACC.4,KEY02
268         JB ACC.5,KEY03
269         JB ACC.6,KEY04
270         JB ACC.7,KEY05
271         KEY01:LJMP KEY1
272         KEY02:LJMP KEY2
273         KEY03:LJMP KEY3
274         KEY04:LJMP KEY4
275         KEY05:LJMP KEY5
276 INT_EX1_EXIT:
277         MOV A,P2   ;判断是否有键按下，消除后沿抖动
278         ANL A,#0F8H
279         XRL A,#0F8H
280         JZ INT_EX1_EXIT_1
281         LCALL DELAY4ms5
282         SJMP INT_EX1_EXIT
283 INT_EX1_EXIT_1:
284         POP ACC
285         POP PSW
286         RETI
287
```

```
KEY1: ;开关键
        CJNE R4,#00,KEY1_NEXT1
        MOV R4,#1       ;开启屏
        MOV 3AH,37H
        MOV 38H,3AH
        LCALL HEX2BCD
        MOV 39H,#LDD6
        LCALL DISPLAY_NUMBER
        LJMP INT_EX1_EXIT
KEY1_NEXT1:
        MOV R4,#0       ;关闭屏
        MOV 3AH,#0
        MOV 3BH,#0
        MOV 3CH,#0
        MOV 38H,#0FFH
        MOV 39H,#LDD6
        LCALL DISPLAY_NUMBER
        LJMP INT_EX1_EXIT

KEY2:  ;确认键，OK 键
        MOV A,R4
        MOV R1,A
        CJNE R1,#0,KEY2_R01
        LJMP INT_EX1_EXIT
KEY2_R01:     DJNZ R1,KEY2_R02
              LJMP KEY2_R1
KEY2_R02:     DJNZ R1,KEY2_R03
              LJMP KEY2_R2
KEY2_R03:     DJNZ R1,INT_EX1_EXIT
              LJMP KEY2_R3
KEY2_R1:
        INC R4
        MOV 38H,3BH
        MOV 39H,#LDD6
        LCALL DISPLAY_NUMBER
        LJMP INT_EX1_EXIT
KEY2_R2:
        INC R4
        MOV 38H,3CH
        MOV 39H,#LDD6
        LCALL DISPLAY_NUMBER
        LJMP INT_EX1_EXIT
```

```
KEY2_R3:
        MOV R4,#1
        MOV 38H,3AH
        LCALL HEX2BCD
        MOV 39H,#LDD6
        LCALL DISPLAY_NUMBER
        MOV A,3BH
        ORL A,3CH
        JNZ KEY2_R3_NORMAL ;如果用户设定全为零，则更改当前时段
        MOV 37H,3AH
        MOV 3BH,#00H
        MOV 3CH,#00H
        LJMP KEY2_R3_NEXT1_NEW
KEY2_R3_NORMAL:
        MOV A,3AH
        RL A
        ADD A,#50H
        XCH A,R1
        MOV A,3BH
        CLR CY
        SUBB A,#3           ;如果数据大于等于三，才存入 RAM
        JNB CY, KEY2_R3_NEXT1
        MOV 3BH,#00H
KEY2_R3_NEXT1:
        MOV @R1,3BH
        INC R1
        MOV A,3CH
        CLR CY
        SUBB A,#3
        JNB CY, KEY2_R3_NEXT2
        MOV 3CH,#00H
KEY2_R3_NEXT2:
        MOV @R1,3CH
        MOV 3BH,#00H
        MOV 3CH,#00H
        MOV R1,3AH  ;如果更改的为当前时间段，则立即调用 GET_LIGHT_TIME 重
          载倒计时时间
        MOV A,37H
        CLR CY
        SUBB A,R1
        CJNE A,#00H,KEY2_R3_EXIT
KEY2_R3_NEXT1_NEW:
```

```
371         LCALL GET_LIGHT_TIME
372 KEY2_R3_EXIT:
373         LJMP INT_EX1_EXIT
374
375 KEY3:  CPL 7FH                    ;选位数
376         LJMP INT_EX1_EXIT
377
378 KEY4:                            ;加一
379         MOV A,R4
380         MOV R1,A
381         CJNE R1,#0,KEY4_R01
382         LJMP INT_EX1_EXIT
383 KEY4_R01:      DJNZ R1,KEY4_R02
384                 INC 3AH
385                 MOV A,3AH
386                 CJNE A,#24,KEY4_R01_next
387                 MOV 3AH,#0
388 KEY4_R01_next:  MOV 38H,3AH
389                 LCALL HEX2BCD
390                 MOV 39H,#LDD6
391                 LCALL DISPLAY_NUMBER
392                 LJMP INT_EX1_EXIT
393 KEY4_R02:      DJNZ R1,KEY4_R03
394                 MOV 3DH,3BH
395                 LCALL BCDINC
396                 MOV 3BH,3DH
397                 MOV 38H,3BH
398                 MOV 39H,#LDD6
399                 LCALL DISPLAY_NUMBER
400                 LJMP INT_EX1_EXIT
401 KEY4_R03:      DJNZ R1,KEY4_EXIT
402                 MOV 3DH,3CH
403                 LCALL BCDINC
404                 MOV 3CH,3DH
405                 MOV 38H,3CH
406                 MOV 39H,#LDD6
407                 LCALL DISPLAY_NUMBER
408 KEY4_EXIT:     LJMP INT_EX1_EXIT
409
410 KEY5:           MOV A,R4        ;减一
411                 MOV R1,A
412                 CJNE R1,#0,KEY5_R01
```

```
413                 LJMP INT_EX1_EXIT
414 KEY5_R01:       DJNZ R1,KEY5_R02
415                 MOV A,3AH
416                 CJNE A,#0,KEY5_R01_next
417                 MOV 3AH,#24
418 KEY5_R01_next:  DEC 3AH
419                 MOV 38H,3AH
420                 LCALL HEX2BCD
421                 MOV 39H,#LDD6
422                 LCALL DISPLAY_NUMBER
423                 LJMP INT_EX1_EXIT
424 KEY5_R02:       DJNZ R1,KEY5_R03
425                 MOV 3DH,3BH
426                 LCALL BCDDEC
427                 MOV 3BH,3DH
428                 MOV 38H,3BH
429                 MOV 39H,#LDD6
430                 LCALL DISPLAY_NUMBER
431                 LJMP INT_EX1_EXIT
432 KEY5_R03:       DJNZ R1,KEY5_EXIT
433                 MOV 3DH,3CH
434                 LCALL BCDDEC
435                 MOV 3CH,3DH
436                 MOV 38H,3CH
437                 MOV 39H,#LDD6
438                 LCALL DISPLAY_NUMBER
439 KEY5_EXIT:      LJMP INT_EX1_EXIT
440
441 BCDINC: PUSH ACC          ;根据选位标志，对高位或低位做加一
442         MOV A,3DH                   ;对 9 加一得到 0
443         JB 7FH,BCDINC_HIGH
444         ANL A,#0FH
445         CJNE A,#09H,BCDINC_LOW1
446         ANL 3DH,#0F0H
447         LJMP BCDINC_EXIT
448 BCDINC_LOW1:
449         INC 3DH
450         LJMP BCDINC_EXIT
451 BCDINC_HIGH:
452         ANL A,#0F0H
453         CJNE A,#90H,BCDINC_HIGH1
454         ANL 3DH,#0FH
```

```
455         LJMP BCDINC_EXIT
456 BCDINC_HIGH1:
457         MOV A,3DH
458         ADD A,#10H
459         MOV 3DH,A
460 BCDINC_EXIT:
461         POP ACC
462         RET
463
464 BCDDEC: PUSH ACC   ;根据选位标志，对高位或低位做减一
465         MOV A,3DH     ;对 0 减一得到 9
466         JB 7FH,BCDDEC_HIGH
467         ANL A,#0FH
468         CJNE A,#00H,BCDDEC_LOW1
469         ORL 3DH,#09H
470         LJMP BCDDEC_EXIT
471 BCDDEC_LOW1:
472         DEC 3DH
473         LJMP BCDDEC_EXIT
474 BCDDEC_HIGH:
475         ANL A,#0F0H
476         CJNE A,#00H,BCDDEC_HIGH1
477         ORL 3DH,#90H
478         LJMP BCDDEC_EXIT
479 BCDDEC_HIGH1:
480         MOV A,3DH
481         SUBB A,#10H
482         MOV 3DH,A
483 BCDDEC_EXIT:
484         POP ACC
485         RET
486
487 CLOSEDIG:              ;根据选位标志，关闭某一位的显示，实现闪烁功能
488         PUSH ACC
489         MOV A,3EH
490         JB 7FH,CLOSEDIG_HIGH
491         ORL A,#0FH
492         LJMP CLOSEDIG_EXIT
493 CLOSEDIG_HIGH:
494         ORL A,#0F0H
495 CLOSEDIG_EXIT:
496         MOV 38H,A
```

```
497         MOV 39H,#LDD6
498         LCALL DISPLAY_NUMBER
499         POP ACC
500         RET
501 ;————————END OF 按键中断处理程序 ————————
502
503 HEX2BCD:             ;将 38H 中的 16 进制数转为 BCD 码
504         PUSH ACC
505         MOV A,38H
506         MOV B,#10
507         DIV AB
508         SWAP A
509         ORL A,B
510         MOV 38H,A
511         POP ACC
512         RET
513 ;————————————————
514 GET_LIGHT_TIME:                  ;获取当前红灯时间子程序，存入 30h 和 31h
515         PUSH DPH                 ;默认红灯时间大于 3 秒
516         PUSH DPL
517         PUSH ACC
518         PUSH PSW
519         SETB RS0
520         MOV DPTR,#TAB_LIGHT_TIME ;读入信号灯延时信息,30H 为红灯 A，31H 为
              红灯 B
521         MOV A,37H
522         RL A
523         MOVC A,@A+DPTR
524         MOV 30H,A
525         MOV A,37H
526         RL A
527         INC A
528         MOVC A,@A+DPTR
529         MOV 31H,A
530         ;检查是否有用户自定义数据
531         MOV A,37H
532         RL A
533         ADD A,#50H
534         MOV R0,A
535         INC A
536         MOV R1,A
537         MOV A,@R0
```

```
538         ORL A,@R1
539         JZ GET_LIGHT_TIME_EXIT ;若全为零，说明无用户定义数据，直接跳出
540         CJNE @R0,#00H,GET_LIGHT_TIME_NEXT1
541         MOV 30H,@R1          ;若只有一个为 0，则使两路口数字相等
542         AJMP GET_LIGHT_TIME_NEXT2
543 GET_LIGHT_TIME_NEXT1:
544         MOV 30H,@R0
545 GET_LIGHT_TIME_NEXT2:
546         CJNE @R1,#00H,GET_LIGHT_TIME_NEXT3
547         MOV 31H,@R0
548         AJMP GET_LIGHT_TIME_EXIT
549 GET_LIGHT_TIME_NEXT3:
550         MOV 31H,@R1
551 GET_LIGHT_TIME_EXIT:
552         CLR RS0
553         POP PSW
554         POP ACC
555         POP DPL
556         POP DPH
557         RET
558 ;————————
559 CHANGE_LIGHT:            ;开信号灯子程序
560         PUSH DPH
561         PUSH ACC
562         MOV DPH,#LED
563         MOV A,@R0
564         MOVX @DPTR,A
565         POP ACC
566         POP DPH
567         RET
568 ;————————————
569 DISPLAY_NUMBER:          ;显示倒计时数字子程序，显示 38H 中的数字到 39H 指定
              的地址中先显示低位，再显示高位
570         PUSH ACC
571         MOV A,38H
572         ANL 38H,#0FH
573         LCALL GETDIGIT
574         MOV DPH,39H
575         LCALL DISDIGIT
576         SWAP A
577         MOV 38H,A
578         ANL 38H,#0FH
```

```
579         LCALL GETDIGIT
580         DEC 39H
581         MOV DPH,39H
582         LCALL DISDIGIT
583         POP ACC
584         RET
585 ;————————————
586 GETDIGIT:                ;取段码子程序
587         PUSH DPH
588         PUSH DPL
589         PUSH ACC
590         MOV DPTR,#DIGIT
591         MOV A,38H
592         MOVC A,@A+DPTR
593         XCH A,38H
594         POP ACC
595         POP DPL
596         POP DPH
597         RET
598 ;————————————
599 DISDIGIT:                ;送数码管显示子程序
600         PUSH ACC
601         MOV A,38H
602         MOVX @DPTR,A
603         POP ACC
604         RET
605 ;————————————
606 SUBBCD:                  ;BCD 码减一, 对 36H 中的数做 BCD 码减 1
607         PUSH ACC
608         PUSH PSW
609         DEC 36H
610         MOV A,36H
611         ANL A,#0FH
612         CJNE A,#0FH,SUBBCD_EXIT
613         CLR CY
614         MOV A,36H
615         SUBB A,#06
616         MOV 36H,A
617 SUBBCD_EXIT:
618         POP PSW
619         POP ACC
620         RET
```

```
621
622 DELAY10:
623         NOP
624         MOV 40H,#9
625 DL10_1: MOV 41H,#255
626 DL10_2: NOP
627         NOP
628         DJNZ 41H,DL10_2
629         DJNZ 40H,DL10_1
630         NOP
631         RET
632
633 DELAY4ms5:
634         MOV 40H,#4
635 DL45_1: MOV 41H,#210
636 DL45_2: NOP
637         NOP
638         NOP
639         DJNZ 41H,DL45_2
640         DJNZ 40H,DL45_1
641         NOP
642         RET
643 ;;————TABLES————-
644 DIGIT:                          ;LED 数码管段码表
645         DB 3FH,06H,5BH,4FH,66H
646         DB 6DH,7DH,07H,7FH,6FH
647         DB 77H,7CH,39H,5EH,79H,00H
648 TAB_LIGHT_TIME:                 ;预设的信号灯时间常数，共 24 行，48 个值
649         DB 03H, 04H
650         DB 04H, 05H
651         DB 05H, 06H
652         DB 06H, 07H
653         DB 07H, 08H
654         DB 08H, 09H
655         DB 09H, 10H
656         DB 10H, 11H
657         DB 11H, 12H
658         DB 12H, 13H
659         DB 13H, 14H
660         DB 14H, 15H
661         DB 15H, 16H
662         DB 16H, 17H
```

```
663         DB 17H, 18H
664         DB 18H, 19H
665         DB 19H, 20H
666         DB 20H, 21H
667         DB 21H, 22H
668         DB 22H, 23H
669         DB 23H, 24H
670         DB 24H, 25H
671         DB 25H, 26H
672         DB 26H, 27H
673 ;————— END ——————————
674 END
```

# 附录 B 电路图

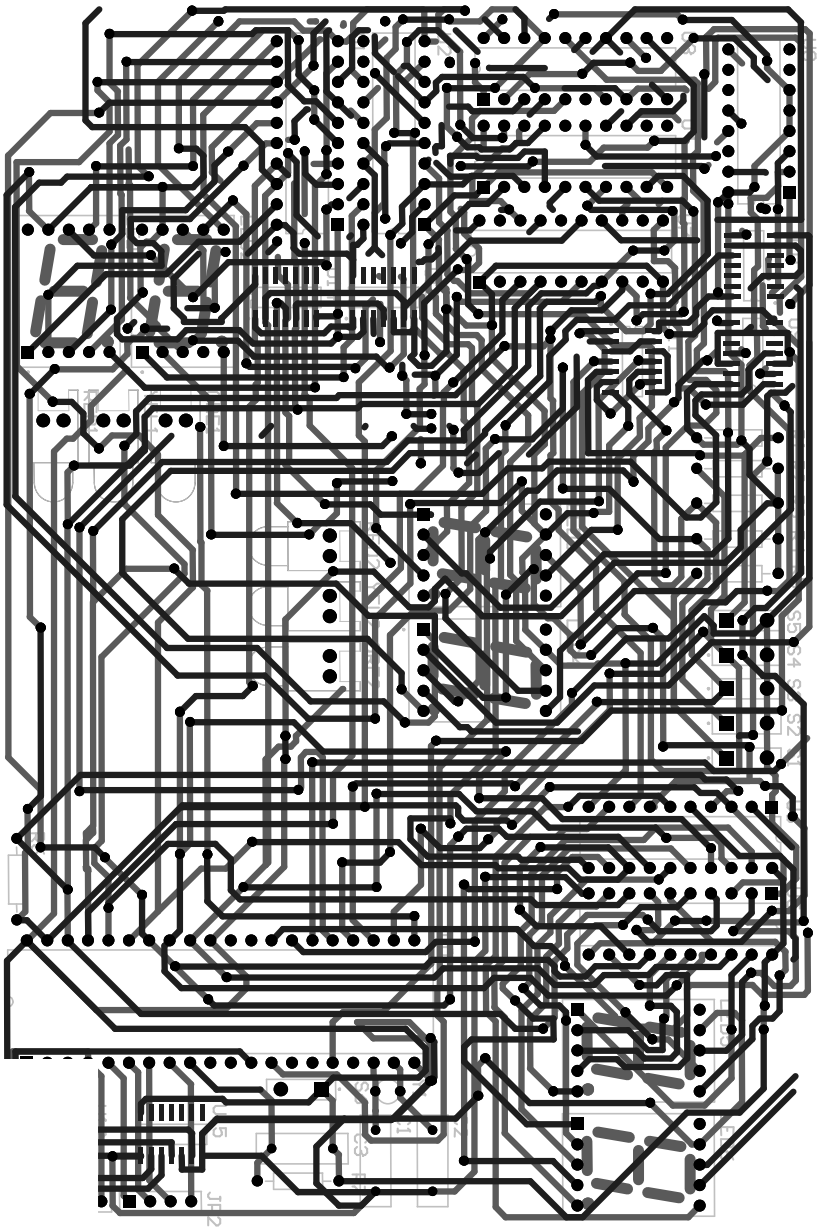

图 B–1 PCB 版图

Fig B–1 PCB Map

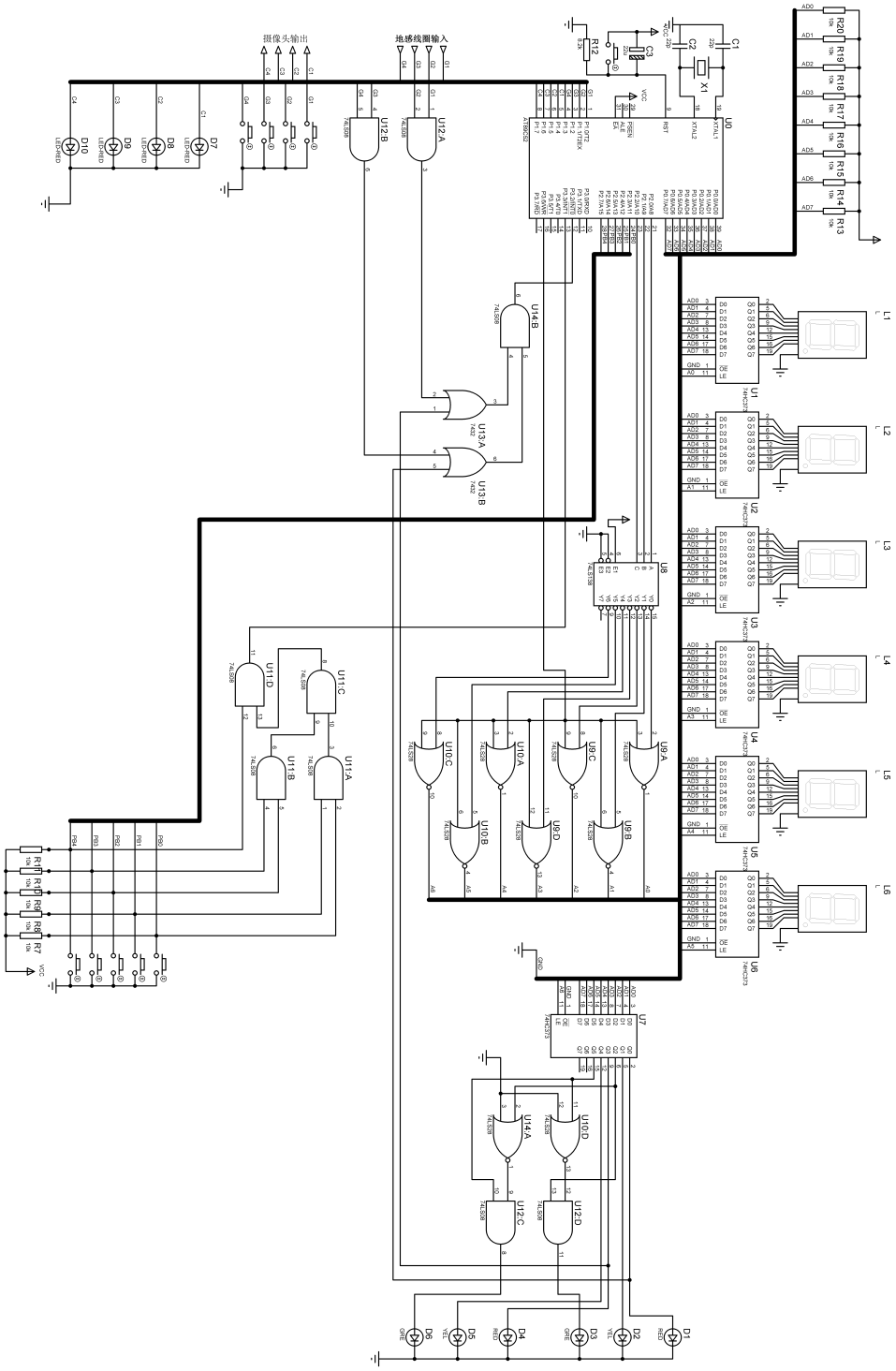


图 B–2 电路图

Fig B–2 Circuit Map

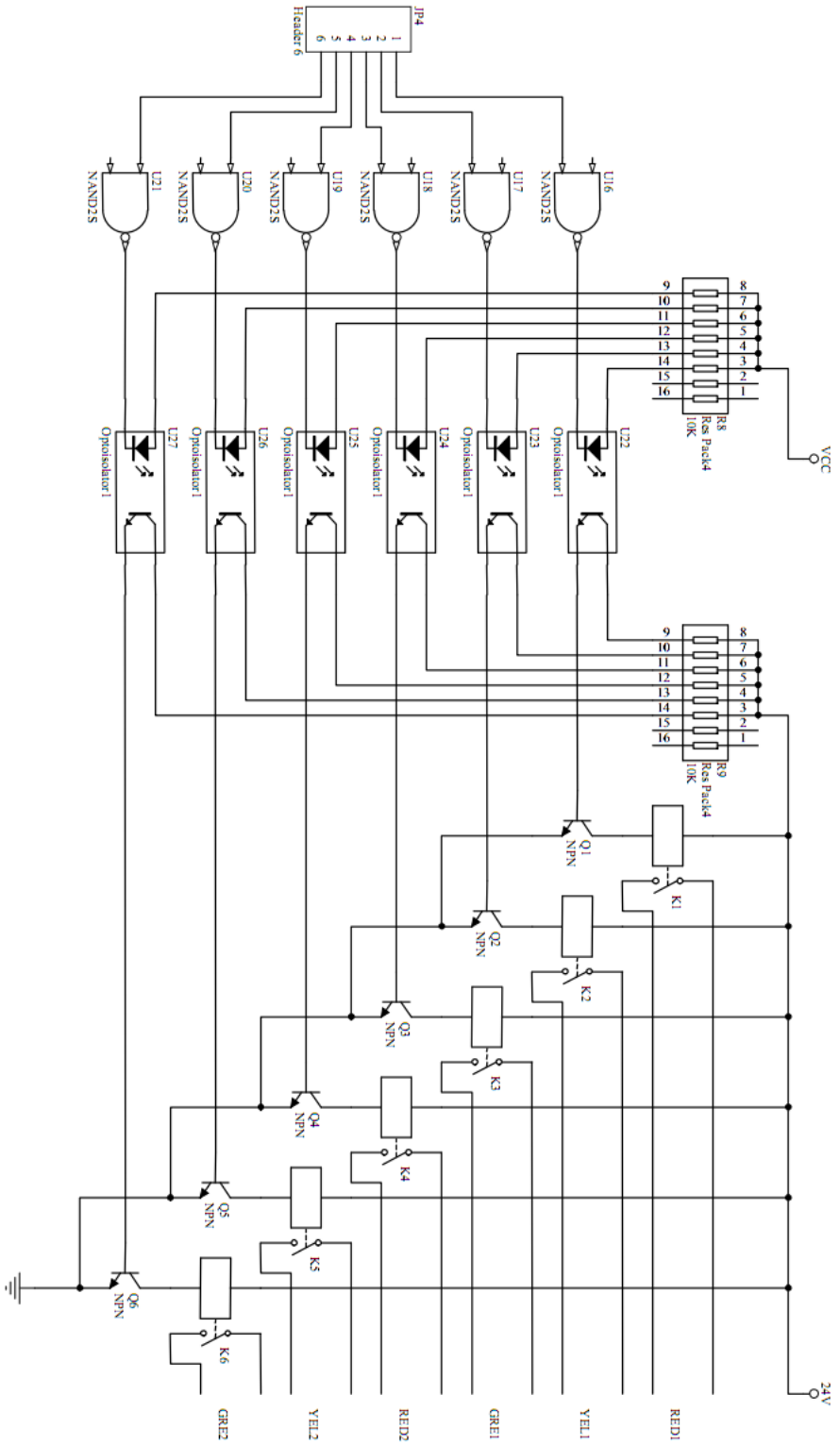


图 B–3 信号灯输出放大电路

Fig B–3 output convert circuit

# 表格索引

# 插图索引

# 参考文献

[1] 杨汝清, 张伟军. 机电控制技术 [M]. 北京: 科学出版社, 2009.

[2] 张迎新. 单片机初级教程：单片机基础 [M]. 北京: 北京航空航天大学出版社, 2008.

[3] 张毅刚. 新编 MCS-51 单片机应用设计 [M]. 哈尔滨: 哈尔滨工业大学出版社, 2003.

[4] 张鑫. 单片机原理及应用 [M]. 北京: 电子工业出版社, 2005.

[5] 郭速学, 朱承彦, 郭楠. 图解单片机功能与应用 [M]. 北京: 中国电力出版社, 2005.

# 致 谢

感谢张银桥老师的精彩授课和您在课程设计过程中给予的热心指导！

感谢张力文、陈相帆、周游同学在课设中给我的帮助和启发，和你们合作是本次课设工作圆满完成的关键！

感谢文献 [1–5] 著者的辛勤劳动和真知灼见。

感谢 Keil, PROTEUS 等软件的开发者，这些软件使得开发工作的难度和工作量大为降低。

感谢 LaTeX 给文档编写工作带来的便利，也感谢 William Wang 同学对 LaTeX 模板移植做出的巨大贡献。