# 系统实验

——交通灯设计

学号：1342401047

姓名：赵煜植

# 系统实验

*——分秒计时器、交通灯设计*

## 一、 实验目的

1、 通过 PCB 板自主画出实验原理图，了解单片机与周围硬件的连接方式及各模块作用及原理。

2、 了解各个芯片的功能及引脚利用。

3、 提高焊接电路的能力。

4、 锻炼 C51 编程能力，实现分秒计时器以及交通灯的功能。

5、 学习硬件烧录、测试过程。

## 二、 实验内容

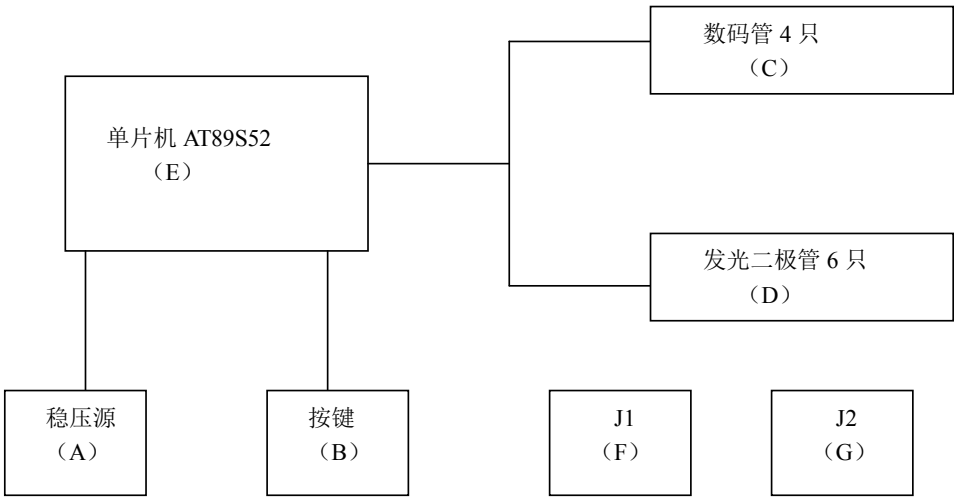
1、 完成编程使整个模块实现：四只数码管低二位 0-59 计数后进位给高二位 0-59 计数（即分秒计时），一只按键控制开始，一只按键控制暂停/重新开始。

2、 完成编程使整个模块实现：灯与数码管相结合，模拟十字路口的交通灯，并在以上功能的基础上实现数码管倒计时显示时间，一只按键控制开始，一只按键控制暂停/重新开始。

3、 完成编程使整个模块实现：分秒计时功能和模拟交通灯功能的切换，即通过一个按键使两功能实现切换。

## 三、 电路模块说明

1、模块整体结构说明：




（A）稳压源部分将输入的交流电转换为稳定的 5V 直流电压；由于目前提供的已经是直流

变压器，所以整流桥可以不用；

（B）作为单片机的输入部分（P1.0~P1.3）起控制作用；

（C）作为单片机的输出部分，P0.0~P0.3 经由 7 段译码器（74LS47）的控制数码管的显示结果（0~9），P0.4~P0.7 数码管的选通控制口，提供数码管工作电源（公共端），P2.7 控制数码管的小数点数否点亮，数码管为共阳极由 74LS46/47 驱动；

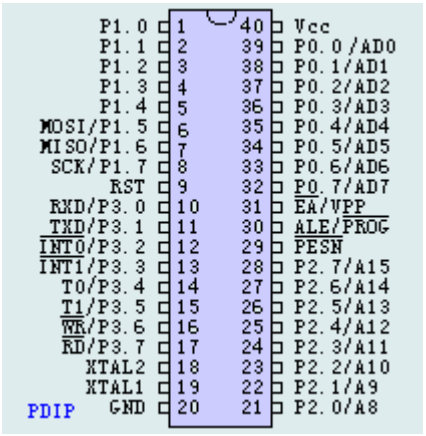
（D）发光二极管：输出部分，由 P2.0~P2.5 口控制，高电平灯亮；

（E）单片机 AT89S52,中心控制器

（F）J1 口留作通信口使用，其中 J1.1（TXD）由 P3.1（11 脚）控制，J1.2（RXD）由 P3.0（10 脚）控制

（G）J2 留作在线烧录程序使用

2、单片机 AT89S51 介绍




AT89S51 是一个低功耗，高性能 CMOS 8 位单片机，片内含 4k Bytes ISP(In-system programmable)的可反复擦写 1000 次的 Flash 只读程序存储器，器件采用 ATMEL 公司的高密度、非易失性存储技术制造，兼容标准 MCS-51 指令系统及 80C51 引脚结构，芯片内集成了通用 8 位中央处理器和 ISP Flash 存储单元。

图一 AT89S51 引脚图

VCC：ATAT89S51 电源正端输入，接+5V。

GND：电源地端。

XTAL1：单芯片系统时钟的反向放大器输入端。

XTAL2：系统时钟的反向放大器输出端，一般在设计上只要在 XTAL1 和 XTAL2 上接上一只石英振荡晶体系统就可以动作了，此外可以在两个引脚与地之间加入一个 20PF 的小电容，可以使系统更稳定， 避免噪声干扰而死机。

RST：AT89S51 的重置引脚，高电平动作，当要对晶片重置时，只要对此引脚电平提升至高电平并保持两个机器周期以上的时间，AT89S51 便能完成系统重置的各项动作，使得内部特殊功能寄存器之内容均被设成已知状态，并且至地址 0000H 处开始读入程序代码而执行程序。

**EA/VPP**："EA"为英文"External Access"的缩写，表示存取外部程序代码之意，低电平动作，也就是说当此引脚接低电平后，系统会取用外部的程序代码（存于外部 EPROM 中）来执行程序。

**ALE/PROG**：ALE 是英文"Address Latch Enable"的缩写，表示地址锁存器启用信号。AT89S51 可以利用这个引脚来触发外部的 8 位锁存器（如 74LS373），将端口 0 的地址总线（A0～A7）锁进锁存器中，因为 ATAT89S51 是以多工的方式送出地址及数据。平时在程序执行时 ALE 引脚的输出频率约是系统工作频率的 1/6，因此可以用来驱动其他周边晶片的时基输入。此外在烧录 8751 程序代码时，此引脚会被当成程序规划的特殊功能来使用。

**PSEN**：此为"Program Store Enable"的缩写，其意为程序储存启用，当 8051 被设成为读取外部程序代码工作模式时（EA=0），会送出此信号以便取得程序代码，通常这支脚是接到 EPROM 的 OE 脚。ATAT89S51 可以利用 PSEN 及 RD 引脚分别启用存在外部的 RAM 与 EPROM，使得数据存储器与程序存储器可以合并在一起而共用 64K 的定址范围。

**PORT0（P0.0～P0.7）**：端口 0 是一个 8 位宽的开路电极（Open Drain）双向输出入端口，共有 8 个位，P0.0 表示位 0，P0.1 表示位 1，依此类推。其他三个 I/O 端口（P1、P2、P3）则不具有此电路组态，而是内部有一提升电路，P0 在当作 I/O 用时可以推动 8 个 LS 的 TTL 负载。如果当 EA 引脚为低电平时（即取用外部程序代码或数据存储器），P0 就以多工方式提供地址总线（A0～A7）及数据总线（D0～D7）。设计者必须外加一个锁存器将端口 0 送出的地址锁住成为 A0～A7，再配合端口 2 所送出的 A8～A15 合成一组完整的 16 位地址总线，而定位地址到 64K 的外部存储器空间。

**PORT2（P2.0～P2.7）**：端口 2 是具有内部提升电路的双向 I/O 端口，每一个引脚可以推动 4 个 LS 的 TTL 负载，若将端口 2 的输出设为高电平时，此端口便能当成输入端口来使用。P2 除了当作一般 I/O 端口使用外，若是在 ATAT89S51 扩充外接程序存储器或数据存储器时，也提供地址总线的高字节 A8～A15，这个时候 P2 便不能当作 I/O 来使用了。

**PORT1（P1.0～P1.7）**：端口 1 也是具有内部提升电路的双向 I/O 端口，其输出缓冲器可以推动 4 个 LS TTL 负载，同样地，若将端口 1 的输出设为高电平，便是由此端口来输入数据。如果是使用 8052 或是 8032 的话，P1.0 又当作定时器 2 的外部脉冲输入脚，而 P1.1 可以有 T2EX 功能，可以做外部

中断输入的触发引脚。

PORT3（P3.0～P3.7）：端口 3 也具有内部提升电路的双向 I/O 端口，其输出缓冲器可以推动 4 个 TTL 负载，同时还多工具有其他的额外特殊功能，包括串行通信、外部中断控制、计时计数控制及外部数据存储器内容的读取或写入控制等功能。 其引脚分配如下： P3.0：RXD，串行通信输入。

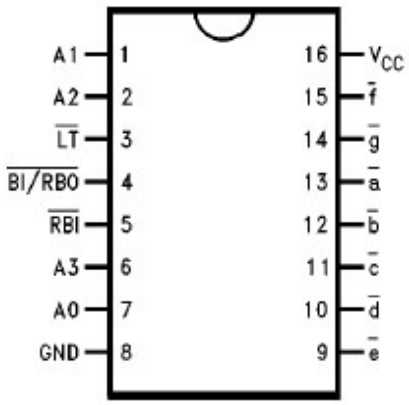
P3.1：TXD，串行通信输出。 P3.2：INT0，外部中断 0 输入。

P3.3：INT1，外部中断 1 输入。 P3.4：T0，计时计数器 0 输入。

P3.5：T1，计时计数器 1 输入。 P3.6：WR：外部数据存储器的写入信号。

P3.7：RD，外部数据存储器的读取信号。

3、74LS47 引脚结构




图二 74LS47 引脚

是 BCD-7 段译码器/驱动器，数字集成电路，用于将 BCD 码转化成数码块中的数字，然后我们就能看到从 0-9 的数字。

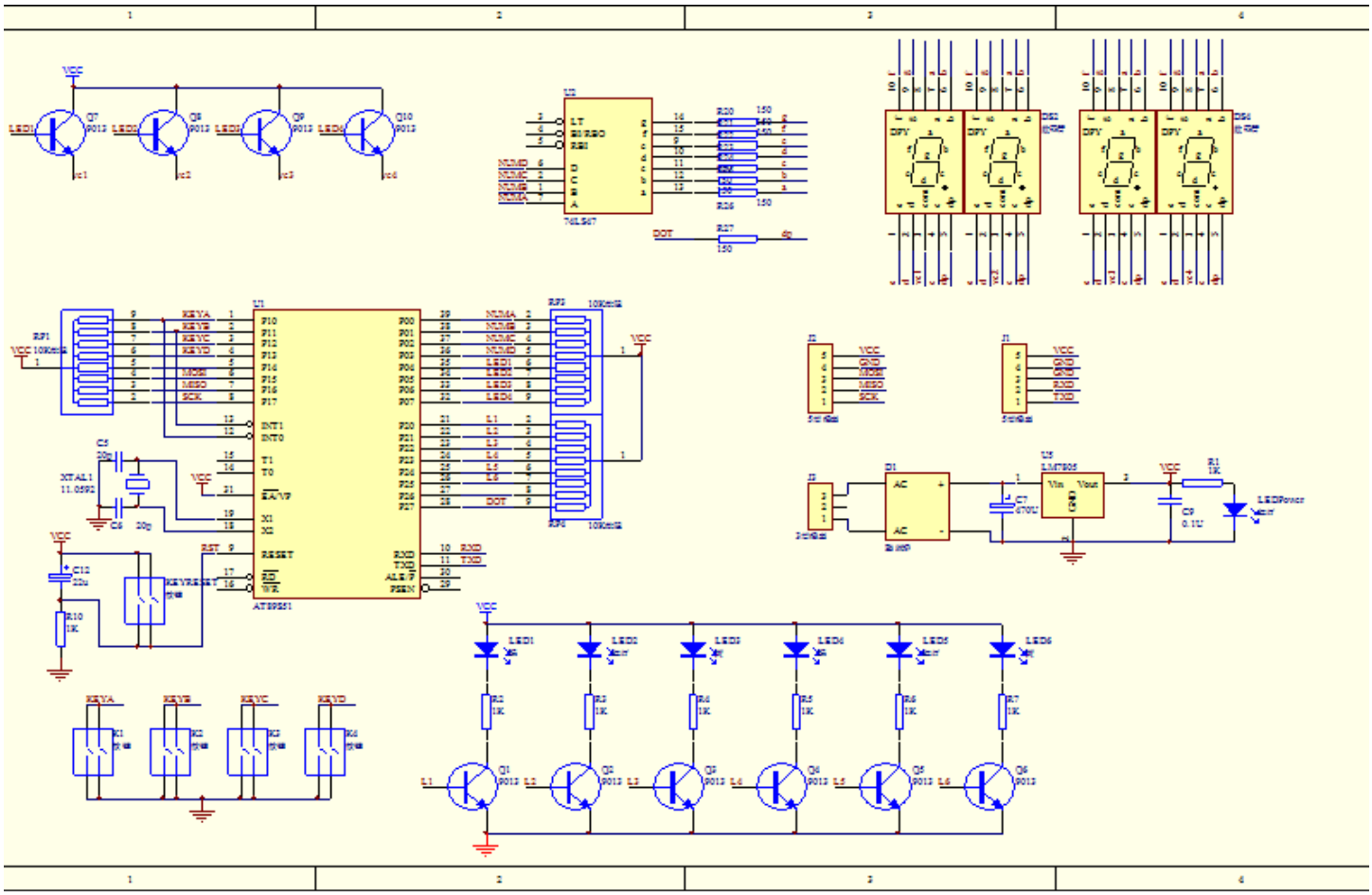
(1)LT：试灯输入，是为了检查数码管各段是否能正常发光而设置的。当 LT=0 时，无论输入 A3 ，A2 ，A1 ，A0 为何种状态，译码器输出均为低电平，也就是七段将全亮,若驱动的数码管正常，是显示 8。

(2)BI：灭灯输入，是为控制多位数码显示的灭灯所设置的。当 BI=0 时，不论 LT 和输入 A3 ，A2 ，A1，A0 为何种状态，译码器输出均为高电平，使共阳极数码管熄灭。

(3)RBI：灭零输入，它是为使不希望显示的 0 熄灭而设定的。当对每一位 A3= A2 =A1 =A0=0 时，本应显示 0，但是在 RBI=0 作用下，使译码器输出全为高电平。其结果和加入灭灯信号的结果一样，将 0 熄灭。

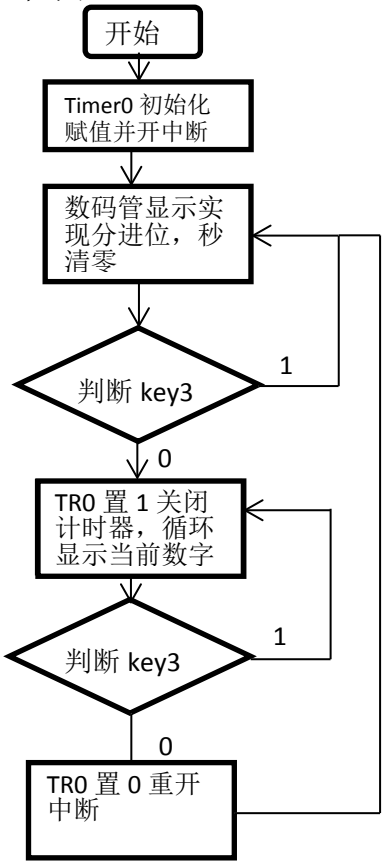
(4)RBO：灭零输出，它和灭灯输入 BI 共用一端，两者配合使用，可以实现多位数码显示的灭零控制。

4、原理图



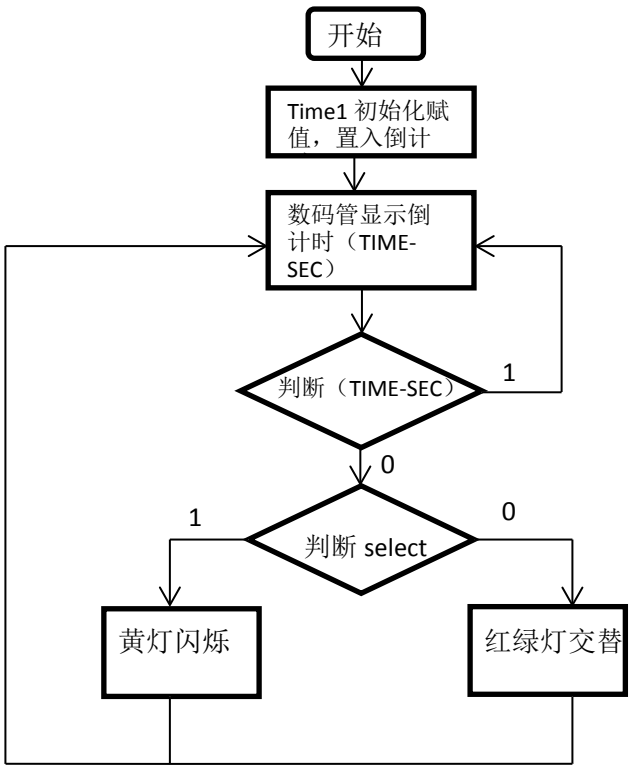
图三原理图

5、分秒计时器原理框图




图四分秒计时器原理框图

6、交通灯流程




图五交通灯流程图

## 四、调试过程

1、制作分秒计时器时，最初将计数的 count 定义为了 unsinged char,所以位数不够在两分钟之后程序有可能会跑飞，最后将 count 定义为 unsigned int,之后程序不会跑飞执行比较稳定。

2、在在执行程序的时候，系统报错 synatx error 在 void 附近，经过检查发现是函数没有事先声明，而在它之前的函数中却进行了引用，所以出现错误。

3、在制作交通灯的过程中，程序开始不计时（数码管维持 20 秒），灯也不能正常点亮，检查程序后发现定时器 1 的工作方式即 TOMD 配置出错，应置入 0x01，而非 0x10，导致定时器不能正常工作，更改后定时器开始工作。

4、在制作交通灯的过程中，开始程序计时直接从 20 秒跳到黄灯倒计时的 5 秒开始，灯也从绿灯或者红灯直接跳到绿灯执行，检查程序后发现在控制灯和数码管的执行程序中，有一个大括号将一个 ELSE 里的 IF 没有括入大括号导致一段程序没有执行，所以红绿灯点亮和倒计时没有进行，导致程序出错。

## 五、实现功能

1、分秒计时功能

2、交通灯及数码管倒计时功能

## 六、实验小结

经过本次实验，首先通过画原理图复习了认读 pcb 板并且了解了整个模块的芯片及各个器件的连接关系，之后通过焊接电路又让我们更加熟悉了焊接的操作，我发现在焊接电路的过程中一定要细心认真，就例如排组等器件一定要注意电源方向，而且多引脚的器件一旦焊错很难拆卸；在编写程序的过程中首先需要设计程序的整体框架然后分步完成功能，在此期间产生过各种错误，分秒计时程序比较顺利，但是交通灯比较费时，主要是在控制显示的程序里出现问题，总之，经过本次编程使我对整个编程语言以及规则更加了解，收获颇丰。