目　录

浙江大学实验报告

课程名称：电子电路系统设计与调试实践　　指导老师：李锡华、施红军　　成绩：_____

实验名称：出租车计费器的设计和制作　　同组学生姓名：

一、实验任务和性能指标

1.1 实验任务

设计并制作出租车计费器

1.2 性能指标

①　自动计费器具有行车里程计费、等候时间计费和起步费 3 项客户总费用，3 项费用总和统一用 4 位数码管显示，最大金额为 99.99 元。

②　起步费设为 2 公里 8 元，行车单价设为 1.80 元/公里，等候时间计费设为 1.50 元/5 分钟。要求行车时，计费值每 0.5 公里刷新一次；等候时间每 1 分钟刷新一次；行程不足 0.5 公里或等候时间不足 5 分钟，则忽略不计。

③　设置起步、停车的提示音。

④　具有数据的复位功能。

扩展功能：

⑤　能进行手动修改单价（加、减）。

⑥　时钟显示功能。

二、设计方案

2.1 基本方案

采用单片机控制，用 AT89S51 加上少量的外围电路实现。利用单片机丰富的 IO 端口，及其控制的灵活性，实现基本的里程计价功能和价格调节等功能。本方案有较大的活动空间，不但能实现所要求的功能而且能在很大的程度上扩展功能，而且还可以方便的对系统进行升级，因此我们采用这一种方案。

2.2 系统设计

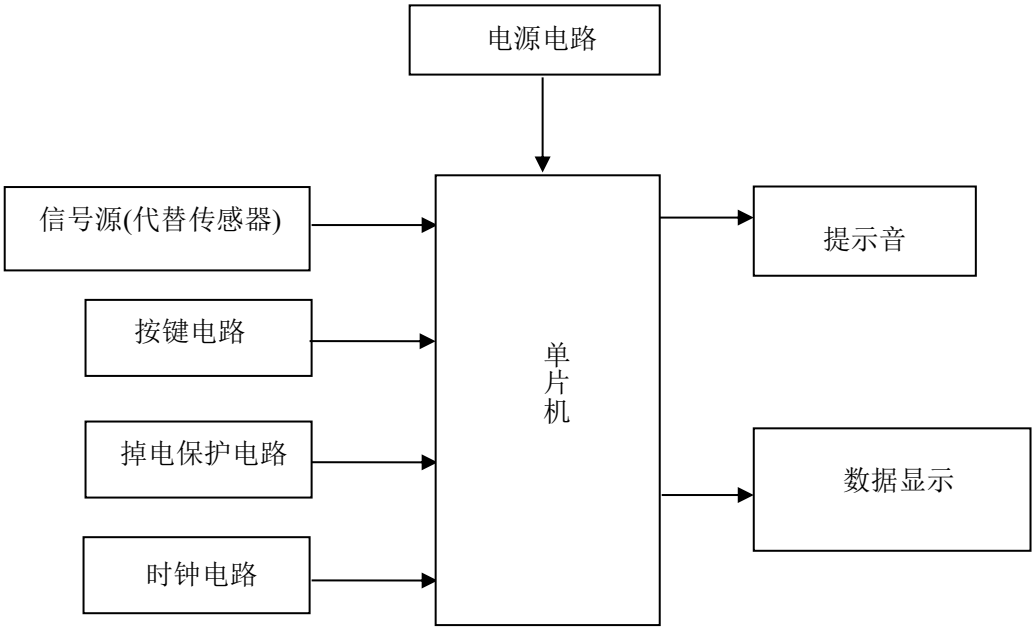


图 2.1 系统原理图

本实验所做的出租车计价器设计由硬件设计和软件设计两部分组成，主要内容包括：

1) 出租车计价器系统的工作原理；
2) 系统的硬件电路设计（单片机主电路、里程与计价单元电路、数据存储电路、显示电路、按键电路、提示音电路、电源电路等）；
3) 系统的软件设计（初始化模块、里程及计价换算模块、数据存储模块、按键处理模块、显示模块、提示音模块等）。

利用 AT89S51 单片机，设计简单的出租车计价器。在出租车计价器的总体设计中，我主要负责出租车计价器硬件设计。其中主要的外围功能电路有：驱动电路，按键控制电路，掉电保护电路，时钟部分，数码管显示电路，提示音电路等。

本电路设计的计价器不但能实现基本的计价，而且还能通过按键，根据白天、黑夜不同情况来调节单价，能在掉电的情况下存储数据，防止外界干扰。

2.3 总电路

2.3.1 芯片介绍

AT89S51 具有如下特点：40 个引脚，4k Bytes Flash 片内程序存储器，128 bytes 的随机存取数据存储器（RAM），32 个外部双向输入/输出（I/O）口，5 个中断优先级 2 层中断嵌套，2 个 16 位可编程定时计数器，2 个全双工串行通信口，看门狗（WDT）电路，片内时钟振荡器。

P0 口有二个功能：

1、外部扩展存储器时，当做数据/地址总线。

2、不扩展时，可做一般的 I/O 使用，但内部无上拉电阻，作为输入或输出时应在外部接上拉电阻。

P1 口只做 I/O 口使用：其内部有上拉电阻。

P2 口有两个功能：

1、扩展外部存储器时，当作地址总线使用。

2、做一般 I/O 口使用，其内部有上拉电阻。

P3 口有两个功能：除了作为 I/O 使用外（其内部有上拉电阻），还有一些特殊功能，由特殊寄存器来设置。

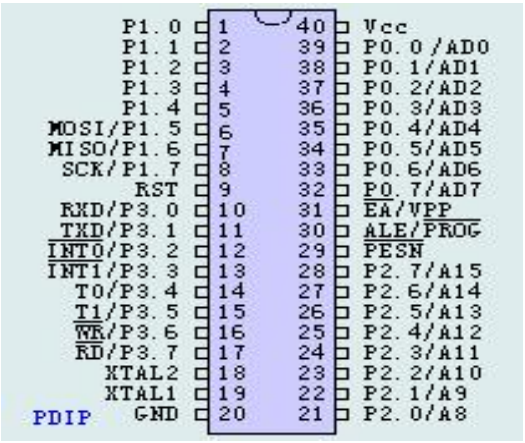


图 2.2　AT89S51 引脚图

2.3.2 电路

设计中用到的单片机各管脚(图 2.2)功能介绍如下：

VCC:接+5V 电源。

GND:接地。

XTAL1、XTAL2：接 12MHz 晶振和 30PF 的电容，构成时钟电路。它可以使单片机稳定可靠的运行。

RST:复位电路，高电平有效。

P1.0:外接提示音电路（蜂鸣器）

P1.1:启动/停止按钮——START，控制计价

P1.2:时钟显示按钮——S2，显示当前时间

P1.3:里程单价+0.1 元

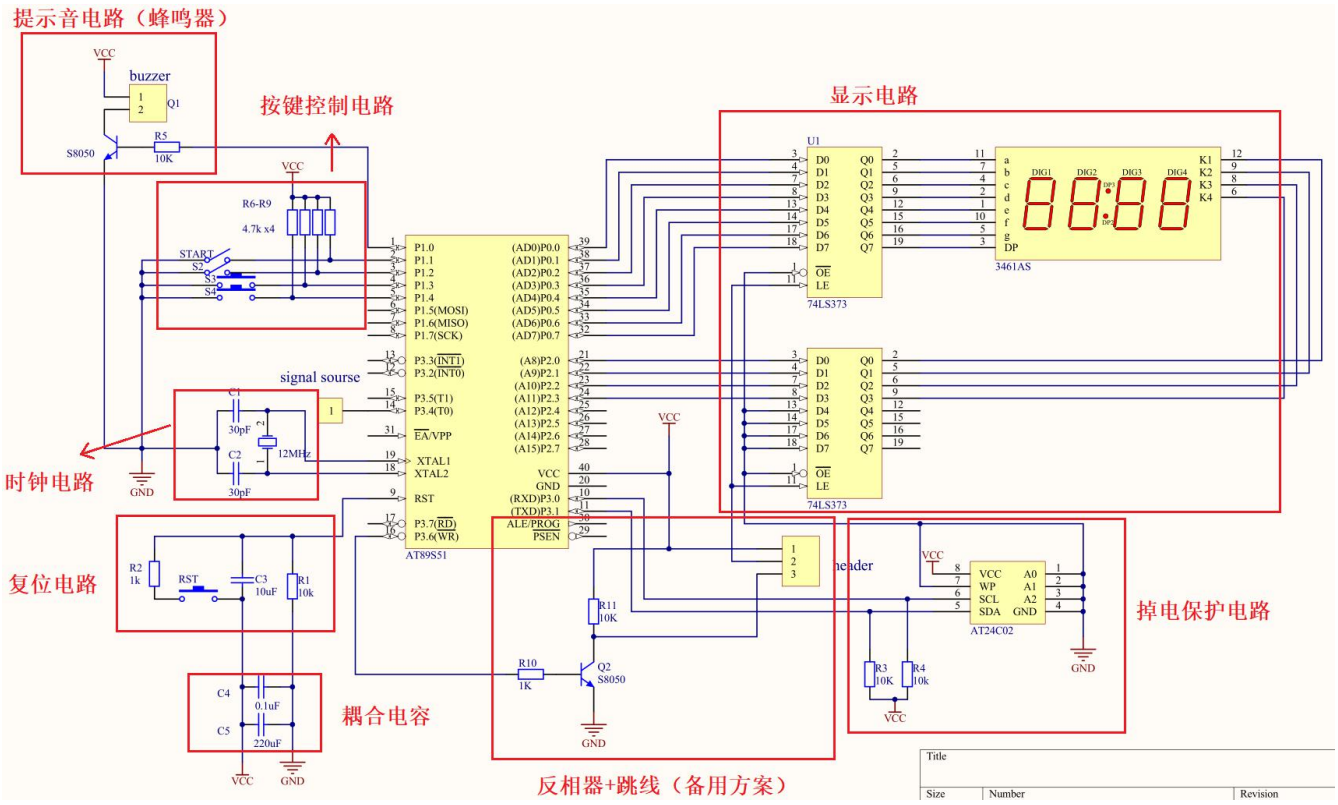
P1.4:里程单价-0.1 元

P0：接驱动芯片 74LS373，数码管段选端

P2：接驱动芯片 74LS373，数码管位选端

P3.4(T0):外部信号源输入端，替代了出租车计价器中的霍尔传感器。

P3.1、P3.0：掉电保护电路。防止断电后重新开机数据丢失。



三、系统硬件设计

3.1 单片机的最小系统

AT89S51 单片机芯片采用 40 引脚的双列直插封装方式。40 条引脚如下：
1、主电源引脚 Vss 和 Vcc
2、外接晶振引脚 XTAL1 和 XTAL2
3、控制或与其它电源复用引脚 RST/VPD，ALE/，和/Vpp
4、输入/输出引脚 P0.0 - P0.7，P1.0 - P1.7，P2.0 - P2.7，P3.0 - P3.7。

单片机的最小系统由电源供电模块、复位电路模块、晶体振荡电路模块组成。

3.1.1 电源电路

因为出租车上的电压是 12V，而芯片所需供电电压是 5V，本次直接采用 5V 的直流输入电压。

3.1.2 复位电路

单片机的复位是由外部的复位电路实现的，复位电路通常采用上电自动复位和按钮复位两种方式。主要由电源、电容、独立开关键以及 10KΩ 电阻组成。其中：

1) 上电复位电路：

a) 电路原理：VCC 上电时，使电容 C 充电，在 10K 电阻上出现高电位电压，使得单片机复位；几个毫秒后，C 充满，10K 电阻上电流降为 0，电压也为 0，使得单片机进入工作状态。工作期间，按下 RST，C 放电，在 10K 电阻上出现电压，使得单片机复位。RST 松手，C 又充电，几个毫秒后，单片机进入工作状态。

b) 功能实现：10KΩ电阻与 10uF 电容构成 RC 充放电电路，RERST 引脚的高电平只要能保持足够的时间（2 个机器周期），单片机就可以进行复位操作。

2) 按钮复位电路：通过 RST 端经电阻与电源 VCC 接通而实现的。只需按下 RST 就能完成复位。其中电平复位是通过 RST 端经电阻与电源 VCC 接通而实现的。按键复位电路模块如图 3.1 所示。

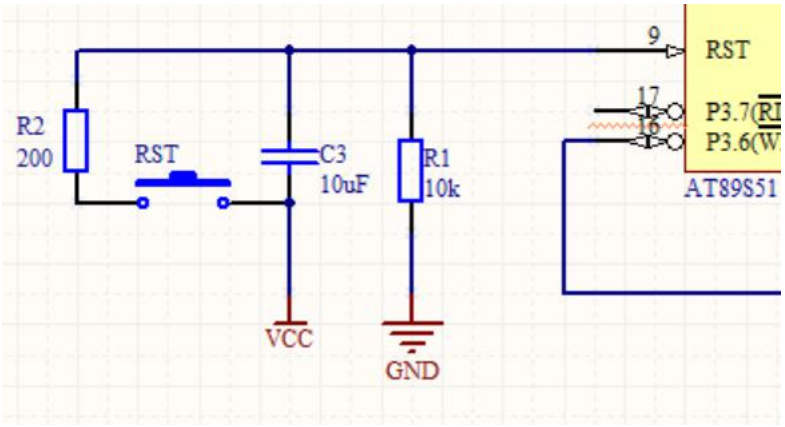


图 3.1.1 复位电路

3.1.3 时钟电路

1) 电路原理：

单片机各功能部件都是以时钟控制信号为基准，内部电路在时钟信号的控制下，严格地按时序执行指令进行工作，单片机本身如同一个复杂的同步时序电路，为了保证其各个部分同步工作，电路要在唯一的时钟信号控制下，严格地按照时序进行工作。其实只需在时钟引脚连接上外围的定时控制元件，就可以构成一个稳定的自激振荡器。为更好地保证振荡器稳定可靠地工作，谐振器和电容应尽可能安装得与单片机芯片靠近。

2) 功能实现：

本设计中使用的振荡电路，由 12MHz(实验室的晶振约 11MHz）晶体振荡器和两个约 30pF 的电容组成，在 XTAL1 和 XTAL2 两端跨接晶体，电容的大小不会影响振荡频率的高低。在整个系统中为系统各个部分提供基准频率，以防因其工作频率不稳定而造成相关设备的工作频率不稳定，晶振可以在电路中产生振荡电流，发出时钟信号。如图 3.2 所示。

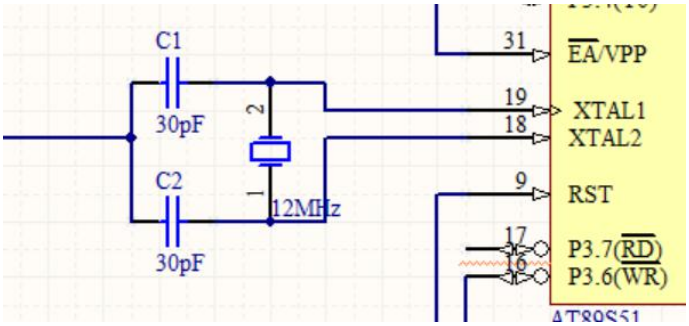


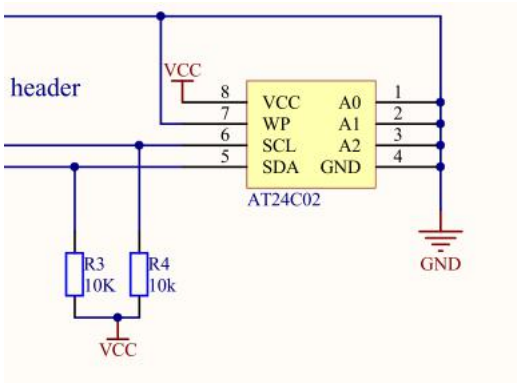
图 3.1.2 时钟电路

3.2 掉电保护模块

出租车司机每次在启动出租车计价器的时候，计价器的价格、时间等信息都会被重置，给出租车司机带来了很多麻烦。对计价器系统添加一个掉电存储模块，之前设置好的数据在掉电时候就会保存，故避免了每次上电需要重新设置的麻烦。

功能实现：

1） 掉电保护电路



2） 芯片 AT24C02

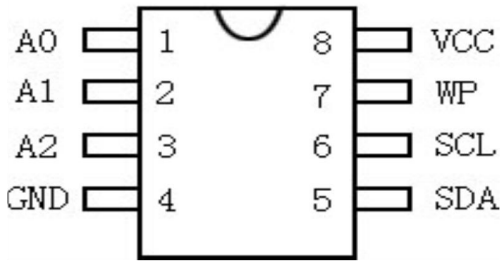


图 3.2.1 AT24C02 引脚图

AT24C02 是一个 CMOS 标准的 EEPROM 存储器，它的内部采用两线串行的总线和单片机进行通信。在不受损坏的情况下，其保存的资料可以保存 40 年以上。掉电保护电路如图 3.4 所示。

引脚功能介绍如下：

A0~A3：（引脚 1~3）：地址线

SDA（引脚 5）：数据总线引脚。

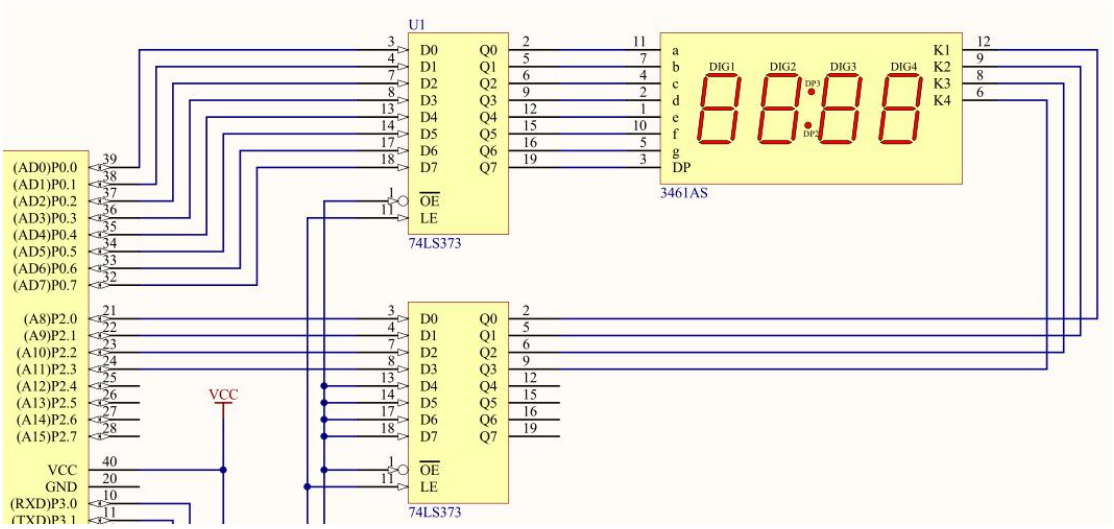
SCL（引脚 6）：时钟总线引脚。

WP（引脚 7）：测试引脚。

3） 其他器件

图中 R3、R4 是上拉电阻，其作用是减少 AT24C02 的静态功耗，由于 AT24C02 的数据线和地址线是复用的，采用串口的方式传送数据，所以只用两根线 SCL（移位脉冲）和 SDA（数据/地址）与单片机传送数据。

3.3 显示电路



1）LED 显示

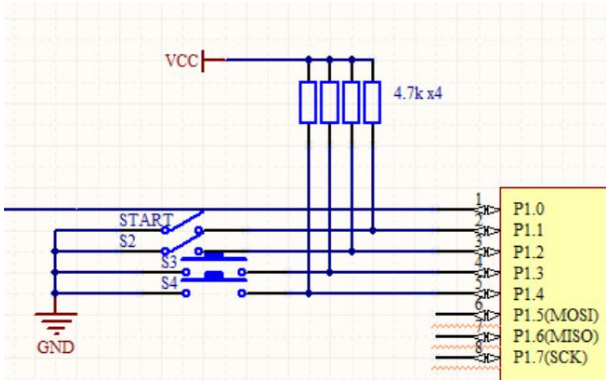
在显示模块需要对时间、单价、总价进行显示，采用的是 4 位 LED 数码管实现里程和价格的交替显示。LCD 虽然也能实现显示功能，但在距屏幕 1 米之外就无法看清数据，不能满足要求，而且在白天其对比度也不能满足要求，因此采用了 LED 数码管显示。

数码管的扫描选用了动态扫描的方法，由于 LED 显示器的余辉和人眼的视觉暂留现象，只要频率在 50Hz 以上，人眼就会觉得数字是一直亮着的。

2）驱动芯片

由于单片机输出电流太小，因此需要选用驱动芯片进行驱动。两片 73LS373 芯片分别实现数码管的段选和位选。OE 是三态门控制端，低电平有效，接地；LE 是锁存控制端，高电平时右边数据等于左边的输入数据，低电平时右边数据不随输入而改变。

3.4 按键电路

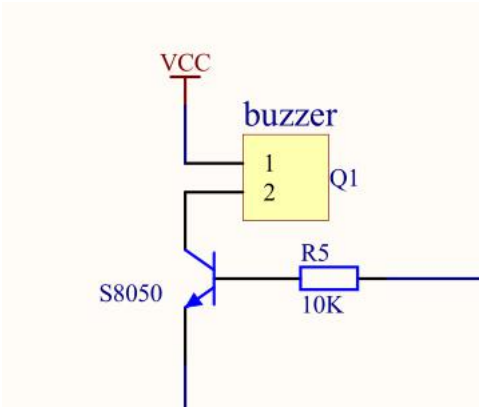


按键控制模块使用了 4 个开关，低电平有效，四个上拉电阻保证开关断开时输入非空，处于高电平状态。其中：

1)两个可锁定的按钮开关：START——计价开始和结束，模拟现实场景中出租车上的“空车计价器”，扳下来计价，推上去不计价；S2——时钟显示按钮，按下可以显示当前时间。

2)两个按键开关：S3、S4——单价的加、减。

3.5 提示音电路



采用蜂鸣器实现启动和停止计价的提示音。通过软件控制提示音时间，启动时短鸣，结束时长鸣。

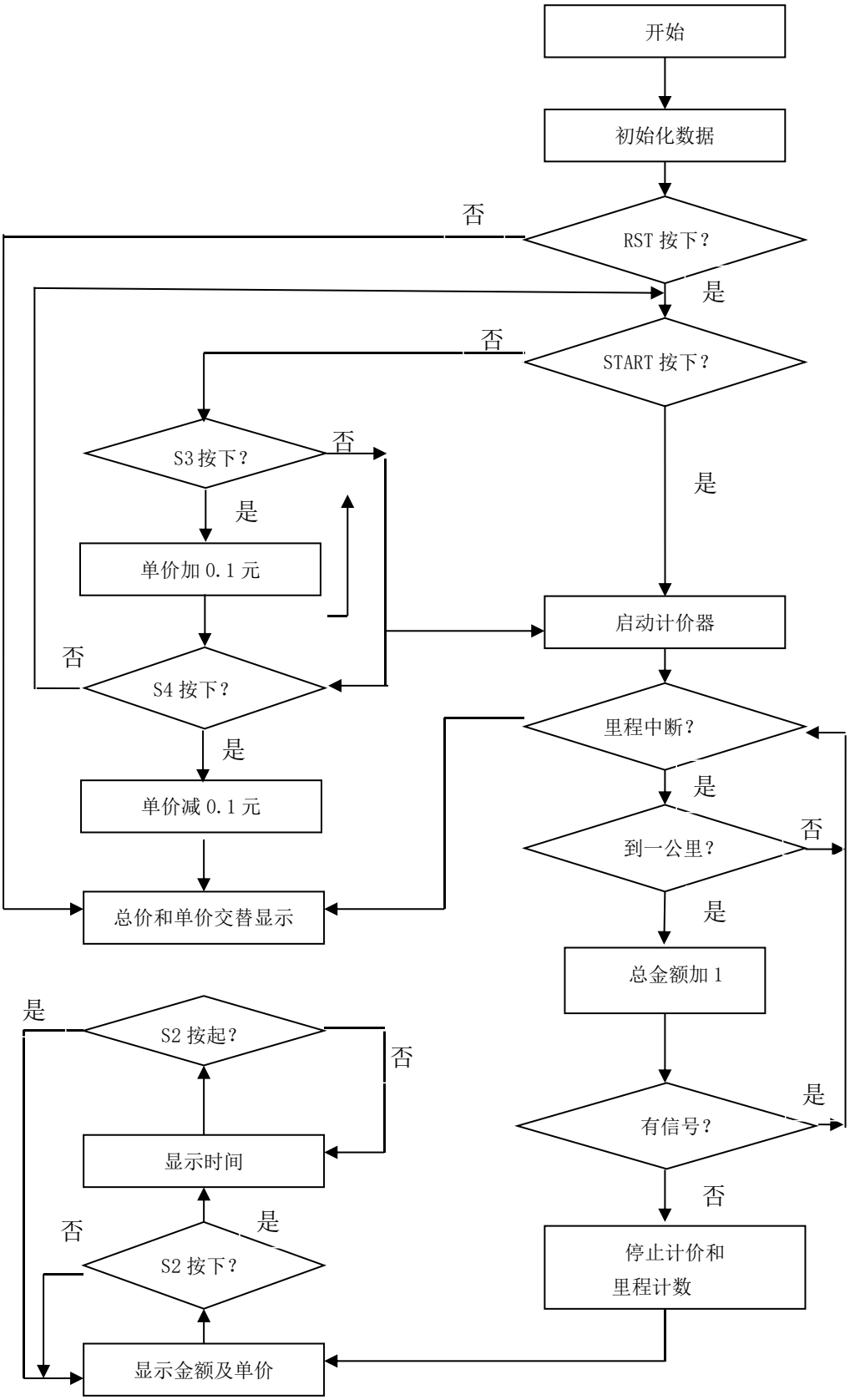
四、系统软件设计

本系统的软件设计主要可分为主程序模块、里程及计价换算模块、数据(时间、价格)存储模块、键盘处理模块、显示模块、提示音模块、定时计数中断模块、中途等待中断服务模块等几大模块。

4.1 系统主程序设计

在主程序模块中，需要完成对各接口芯片的初始化、出租车起价和单价的初始化、中断向量的设计以及开中断、循环等待等工作。另外，在主程序模块中还需要设置启动/清除标志寄存器、里程寄存器和价格寄存器，并对它们进行初始化。

主程序将根据各标志寄存器的内容，判断行驶路程是否在两公里（起步价 8 元两公里）之内，若在两公里之内，则按照起步价计算总金额，若超过两公里，则按照起步价加上超出部分金额再加上等待时间金额进行计算，将时间，金额，单价等信息发送到显示电路。当乘客到达目的地，按下停止按钮，总金额，里程，时间等信息将显示在显示器上，在交易完成后，出租车司机按下清零键，主程序模块对各个模块重新初始化，为下一次启动做好准备。主程序的流程图如下图所示。


开始
初始化数据
RST 按下？
否
是
START 按下？
否
是
S3 按下？
否
是
单价加 0.1 元
启动计价器
S4 按下？
否
是
单价减 0.1 元
里程中断？
是
到一公里？
否
是
总价和单价交替显示
总金额加 1
S2 按起？
是
否
有信号？
是
否
显示时间
S2 按下？
否
是
停止计价和
里程计数
显示金额及单价

4.2 里程计数中断服务程序模块

每次信号源传来脉冲信号，在控制器允许的情况下，就会引起控制器中断，控制器转向处理中断子程序，中断子程序根据设定好的车轮周长计算出里程数，并将结果送达显示电路。如果未经控制器允许传感器就传送来脉冲，则视为作弊，不予处理。里程计数中断流程如上图 4.2 所示。

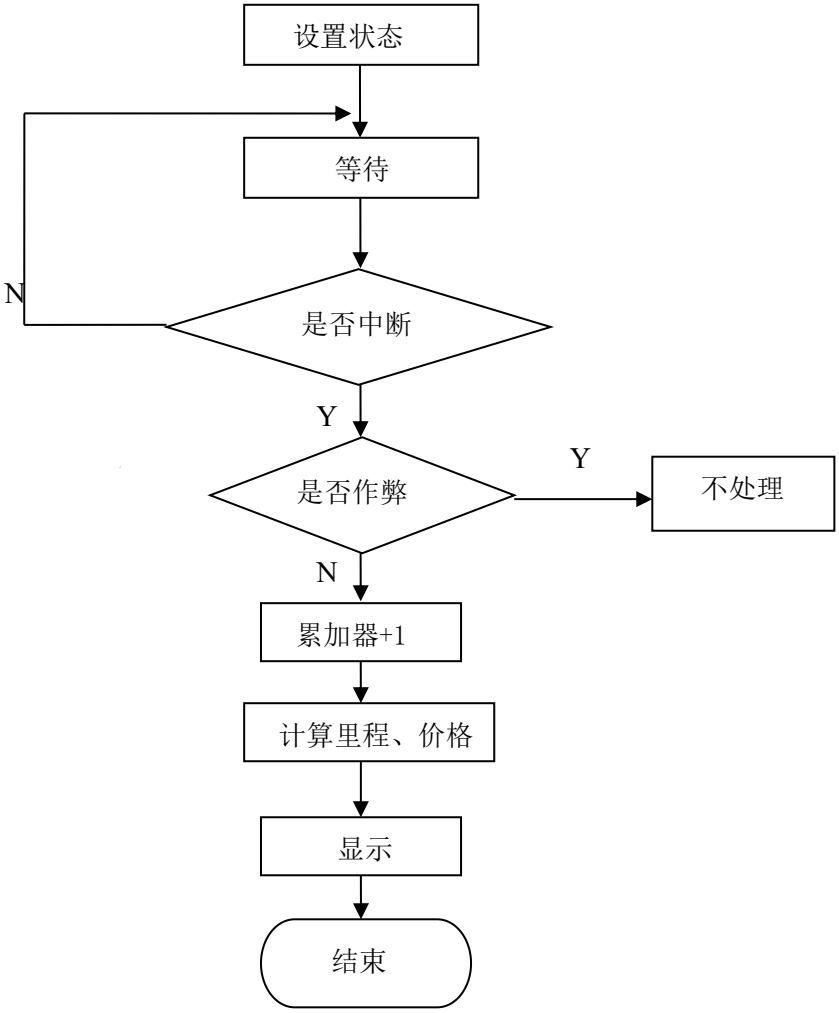


图 4.2 里程中断计数服务流程图

4.3 中途等待中断服务程序模块

当控制器在计价状态下时，控制器内的定时器也随之启动并将等待时间设为 1 秒。如果信号源在 1 秒之内没有传来脉冲信号，那么就进入等待金额计算公式（中途等待价为当前单价+0.3 元）。当信号源又重新有脉冲信号输入时，表明出租车开始行驶，控制器就转到里程计价模式，并且记录当前等待时间。中途等待终端服务流程图如图 4.3 所示。

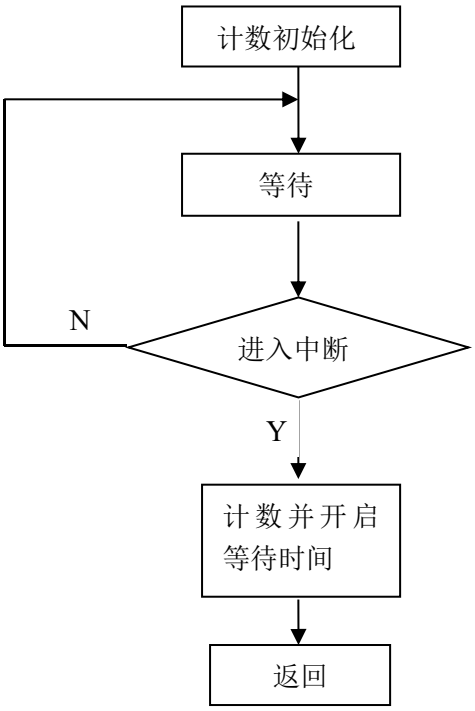


图 4.3 中途等待中断服务流程图

4.4 显示模块

显示单元利用了动态显示，将要显示的数据通过数码管显示。利用 delay()函数实现短暂延时，从而利用软件实现的数据的“锁存”。

在未按下“START”时，单价和起步价交替显示；

按下 START，显示当前行车金额；

按下 S2 可以显示当前时间，时间数据存储在 AT24C02 中；

按 S3、S4 对单价调高或者调低时，显示当前单价价格。

4.5 掉电保护模块

AT24C02 存储器主要用与计价器非正常停止状态，以及用于存储时钟数据。在拔掉电源后再重新接上后，里程、金额、单价数据不会丢失，重启后显示。时钟也因为有数据存储，所以是实际时间。

五、心得体会与总结

5.1 软件调试

5.1.1 单片机仿真软件—PROTEUS

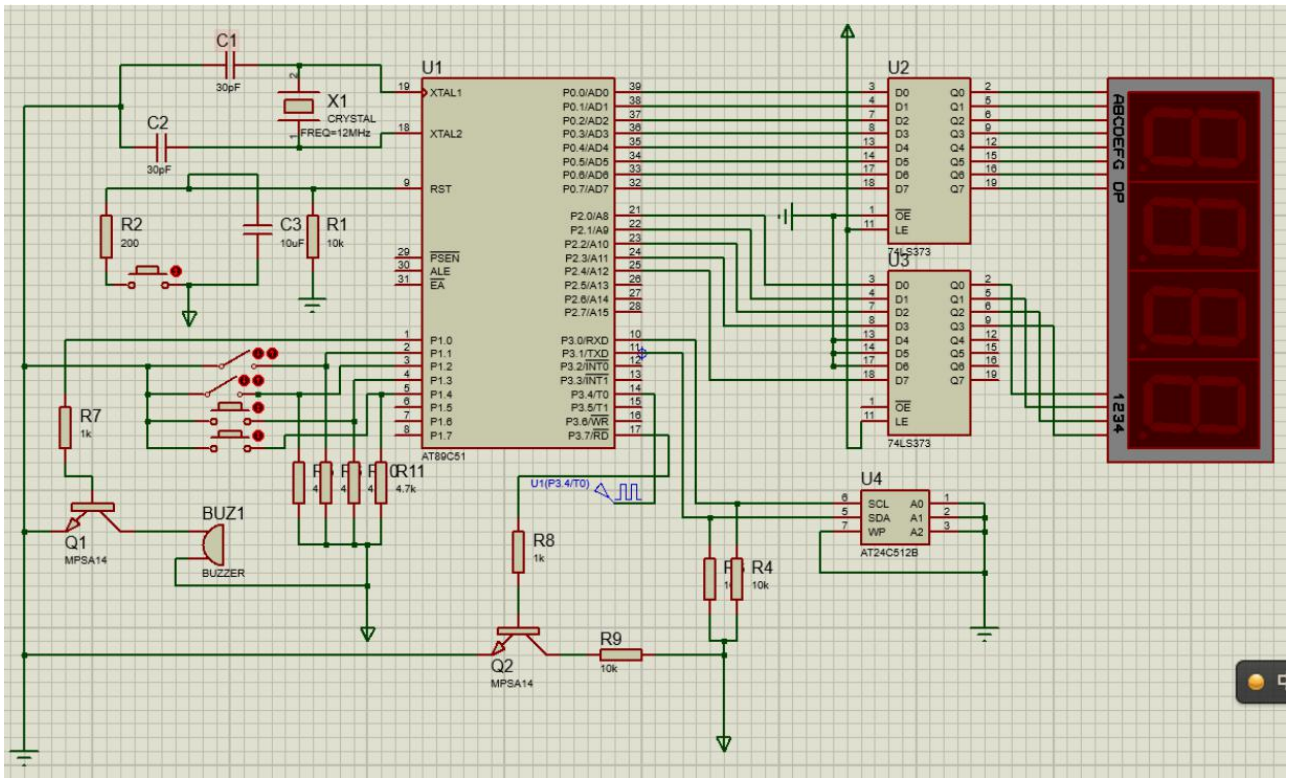
1.打开 Proteus 软件。

2.选择 file 菜单下的 open design 选项，找到所需的元器件，元器件上单击右键选中，再单击左键对其进行命名和赋值，接着在编辑器左边的一栏中，找出并绘制设计所要的各种元器件，按照电路图连接后并保存。

3.将用 keil 编译产生的 hex 文件下载到单片机中：双击 51 单片机，在对话框中把保存过的 hex 文件打开，再单击确定。

4.单击左下角运行按钮，进行软件仿真调试，直到出现正确的结果。

下图为软件仿真图：



5.1.2 软件调试

在软件编程时，应充分利用原理图，不然会在编程时导致硬件与软件不能对应起来，使程序无法

实现功能，还有在编程时子程序的名称应与其功能对应，否则会使别人在读程序的时候不能及时理解程序含义，而且在软件调试时逻辑一定要清晰，不然在找错时会浪费很多时间。

5.2 硬件检测

5.2.1 元器件检查

在焊接电路前，首先要进行元器件的检测。检测主要是测出各个元器件的型号，尤其是电阻，很容易焊接错误。识别电阻时可根据各环的数量级和色码表，判断电阻的阻值；更简便的方法是用万用表直接测量。对于集成芯片的检测，就是根据它的管脚图，来识别各个引脚，以方便焊接。

5.2.2 电路测试

焊好电路板，接通电源后，按对应开关，观察电路是否工作正常，是否符合逻辑。

1）按下复位键，是否数据复位；

2）分别在未接入信号源（等待状态）、接入信号源（行驶状态）进行以下操作：

按下各个功能键，与最初设计的功能对比，是否符合逻辑；

3）由于之前软件仿真无误，因此如果电路无法正常工作，需要进行硬件检查。检测步骤如下：

检测步骤：

1）晶振部分使用示波器查看波形。如果出现看不到 12MHZ 的正弦波形的现象，说明此部分电路不正常。

2）蜂鸣器使用万用表测量引脚两端电压，与额定工作电压对比；

3）各个芯片用示波器测量引脚波形，与理论值对比看是否符合实际；

5.2.3 错误及解决方法

因为选用了单片机来做出租车计费器，PCB 板比其他组小很多，元器件也较少，因此遇到的硬件上的错误较少。

错误 1：按下复位按钮时，电路没有反应。

解决办法：测量了复位电路按下开关时，开关串联的电阻的电压，认为没有问题，高电平，应该工作正常啊。后来三个人一起讨论之后才发现，把 RST 按键和上面几个按键的逻辑弄反了，其他功能键是低电平有效，RST 键连的引脚在单片机程序中，写的是低电平有效，然而实际电路中，按下开关是接高电平。

最终取下单片机，重新更改了代码，RST 键正常工作。

错误 2：启动和结束计价时蜂鸣器不响。

解决办法：我们先是测量了蜂鸣器引脚电压，4.7V，与工作电压相差不大；而后用 4.7V 电压通过杜邦线轻触蜂鸣器引脚，蜂鸣器响了，排除器件损坏问题。因为怀疑电流太小，为达到额定工作电

压。

经检查，发现基极电阻焊错，本来应该是 10k 电阻，结果焊接成了 1k 电阻，由于 1k 和 10k 电阻色环差别太小，难免在焊接时弄混。

因此拆下电阻重新焊上 10k 电阻，重新测试，蜂鸣器正常工作。

5.3 性能分析

排除了所有软硬件的故障后，电路能够正常工作。能够实现：①行车里程计费、等候时间计费和起步费 3 项计费功能；②起步、停车均有提示音；③具有数据复位功能；而且也实现了本组的扩展功能：能进行手动修改单价（加、减）和时钟显示功能。

5.4 收获

本款出租车计价器由于使用单片机制作，比起硬件电路，软件代码更容易实现修正与提升，可以根据市场需要实现功能扩展。多功能出租车计价器具有性能可靠、电路简单、成本低、实用性强等特点，加上经过优化的程序，使其有很高的智能化水平。

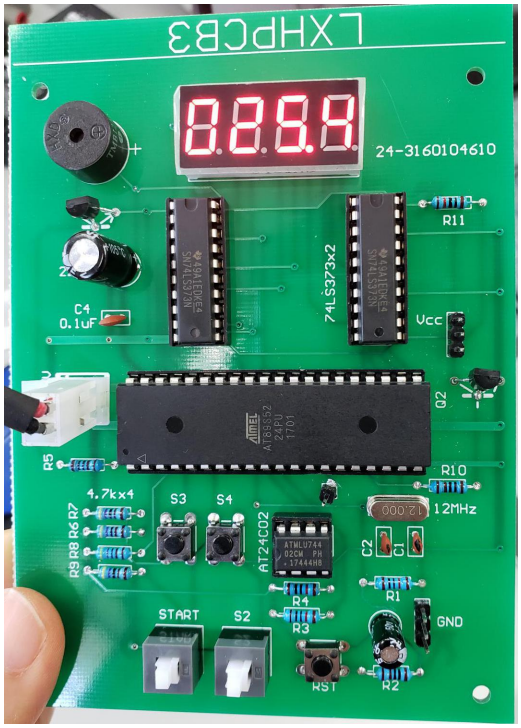
当初选择用单片机来做这个课程设计，就是希望通过此次实验，能够了解单片机，实现单片机入门，学到一些新东西，掌握一门很实用的技术。本次课程设计，确实让我学到了许多理论课中无法学到的知识,也深该体会到单片机技术应用领域的广泛，不仅使我对学过的一些关于 CPU 的知识有了很多的巩固，同时也对单片机这一门课程产生了更大的兴趣。

在显示电路模块中，我们已经写好代码，主要用 delay 函数，通过延时来达到想要的数据锁存的效果，而且在软件上仿真也没有错误。后来老师提议让我们用硬件实现数据的锁存，因为单片机的数据输出不能够持续在显示屏上显示，为了以防万一，小组内决定把硬件锁存当作备用方案，如果软件无法实现，就用 74LS373 的 LE（锁存使能端）和 AT89S51 的 W/R 端口相配合，实现硬件锁存，因此才有了原理图中“备用方案”那部分电路。通过跳线，方案一：LE 直接接高电平，右端数据一直等于左边，通过软件来实现数据持续显示。备用方案：跳帽连接 LE 和 W/R 反相端，有写数据信号（W）时，右端等于左端数据，没有写数据信号时，LE 低电平，数据锁存。

PCB 板发下后，经过实际电路检测，发现方案一能行，于是采用原本的方案。虽然没有采用老师的方案，但是任然教会了我们，在设计中需要周全的考虑，一个方案可能是行不通的，然而电路板不可能任意你更改，需要在 PCB 绘制中就考虑到可能遇到的问题，增加测试电路，多加几个测试接口，这样才能提高电路的正确性。

在设计过程中，我学会了熟练在网络上查找有关本设计的各硬件的资源，其中包括：AT89S51 单片机及其引脚说明、AT24C02 引脚图及其引脚功能等，为本次课程设计提供了一定的资料。由于平时很少进行课程设计，所以对于课程设计报告的格式也是近期才接触到，经过这次的设计，为我们以后毕业设计的制作也奠定了一定的基础。

附录 1：实际电路图



附录 2：代码

```
/* Main.c file generated by New Project wizard
 *
 * Created:   Fri Jul 13 2018
 * Processor: AT89C51
 * Compiler:  Keil for 8051
 */

#include <reg51.h>
#include <intrins.h>
#include "i2c.h"
```

```
#define uchar unsigned char
#define uint unsigned int

#define I2C_SCL P3^0
#define I2C_SDA P3^1
#define I2C_DELAY 10

uint totalPrice,distance,timerCount,innerCount,waitPrice;
uint perPrice=9,startPrice=80;
uint running=0;
uint key1DownCount,key1UpCount,key2DownCount,key2UpCount,key1State,key2State,keyLock;
uchar minute;

uchar LedChar[] = {
   0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
   0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71
};

uchar Position[]={0xfe,0xfd,0xfb,0xf7};

uchar show0,show1,show2,show3;

uchar perms,ms,second,minute;

sbit P00=P0^0;
sbit P01=P0^1;
sbit P02=P0^2;
sbit P03=P0^3;
sbit P04=P0^4;
sbit P05=P0^5;
sbit P06=P0^6;
sbit P07=P0^7;

sbit P20=P2^0;
sbit P21=P2^1;
sbit P22=P2^2;
sbit P23=P2^3;

sbit key0=P1^1;
sbit key1=P1^2;
sbit io_key_1=P1^3;
sbit io_key_2=P1^4;

sbit buzzer=P1^0;
```

```
sbit SCL=I2C_SCL;
sbit SDA=I2C_SDA;

static void i2c_delay(uint8 p_loop)
{
    while(p_loop--);
}

static void i2c_start()
{
    SDA=1;
    SCL=1;
    i2c_delay(I2C_DELAY);
    SDA=0;
    i2c_delay(I2C_DELAY);
    SCL=0;
}

static void i2c_stop()
{
    SDA=0;
    SCL=1;
    i2c_delay(I2C_DELAY);
    SDA=1;
    i2c_delay(I2C_DELAY);
}

static void i2c_send_ack(bool p_ack)
{
    SDA=p_ack;
    SCL=1;
    i2c_delay(I2C_DELAY);
    SCL=0;
    i2c_delay(I2C_DELAY);
}

static uint8 i2c_recv_ack()
{
    uint8 t_ack;
    SCL=1;
    i2c_delay(I2C_DELAY);
    t_ack=SDA;
    SCL=0;
```

```
    i2c_delay(I2C_DELAY);
    return t_ack;
}

static uint8 i2c_send_byte(uint8 p_data)
{
    uint8 t_i;
    for(t_i=0;t_i<8;t_i++)
    {
        SDA=p_data&0x80;
        p_data<<=1;
        SCL=1;
        i2c_delay(I2C_DELAY);
        SCL=0;
        i2c_delay(I2C_DELAY);
    }
    return i2c_recv_ack();
}

static uint8 i2c_recv_byte()
{
    uint8 t_i;
    uint8 t_data=0;
    SDA=1;
    for(t_i=0;t_i<8;t_i++)
    {
        t_data<<=1;
        SCL=1;
        i2c_delay(I2C_DELAY);
        t_data|=SDA;
        SCL=0;
        i2c_delay(I2C_DELAY);
    }
    return t_data;
}

void i2c_write(uint8 p_address,uint8 p_register,uint8 p_data)
{
    p_address<<=1;
    i2c_start();
    i2c_send_byte(p_address);
    i2c_send_byte(p_register);
    i2c_send_byte(p_data);
    i2c_stop();
```

```
}

uint8 i2c_read(uint8 p_address,uint8 p_register)
{
    uint8 t_data;
    i2c_multi_read(p_address,p_register,1,&t_data);
    return t_data;
}

void i2c_multi_write(uint8 p_address,uint8 p_register_start,uint8 p_count,uint8* p_buffer)
{
    uint8 t_i;
    for(t_i=0;t_i<p_count;t_i++)
        i2c_write(p_address,p_register_start+t_i,p_buffer[t_i]);
}

void i2c_multi_read(uint8 p_address,uint8 p_register_start,uint8 p_count,uint8* p_buffer)
{
    uint8 t_i;
    p_address<<=1;
    i2c_start();
    i2c_send_byte(p_address);
    i2c_send_byte(p_register_start);
    i2c_start();
    i2c_send_byte(p_address+1);
    for(t_i=0;t_i<p_count;t_i++)
    {
        p_buffer[t_i]=i2c_recv_byte();
        i2c_send_ack(t_i==p_count-1);
    }
    i2c_stop();
}

//////////////////////////////////////////
void delay(uint x)
{
   int i,j;
   for(i=x;i>0;i--)
     for(j=400;j>0;j--);
}
//////////////////////////////////////////////////
void delay_ms(uint16 p_ms)
{
    uint8 t_timer;
```

```
    while(p_ms--)
        for(t_timer=0;t_timer<250;t_timer++);
}

void realShow()
{

   P0=show0;
   P2=Position[0];
   delay(1);
   P0=0x00;

   P0=show1;
     if(key1==0&&key0==1)
        P07=1;
   P2=Position[1];
   delay(1);
   P0=0x00;

   P0=show2;
     if(key1==1)
   P07=1;
   P2=Position[2];
   delay(1);
   P0=0x00;

   P0=show3;
   P2=Position[3];
   delay(1);
   P0=0x00;
}

void timeDisplay()
{
   show0=LedChar[0];
   show1=LedChar[9];
   show2=LedChar[0];
   show3=LedChar[minute%10];

   realShow();
}

void totalPriceDisplay()
{
```

```
   show0=LedChar[totalPrice/1000];
   show1=LedChar[totalPrice%1000/100];
   show2=LedChar[totalPrice%100/10];
   show3=LedChar[totalPrice%10];

  realShow();
}

void startPriceDisplay()
{
   show0=LedChar[startPrice/1000];
   show1=LedChar[startPrice%1000/100];
   show2=LedChar[startPrice%100/10];
   show3=LedChar[startPrice%10];

   realShow();
}

void perPriceDisplay()
{
   show0=LedChar[perPrice/1000];
   show1=LedChar[perPrice%1000/100];
   show2=LedChar[perPrice%100/10];
   show3=LedChar[perPrice%10];

   realShow();
}

void priceCount()
{
   if(key0==1)
   {
      totalPrice=0;
   }
   else
   {
      if(distance<=4)
     totalPrice=startPrice;
      else
     totalPrice=startPrice+perPrice*(distance-4);
   }
   if(waitPrice>15)
   {
      totalPrice+=(waitPrice-15);
```

```
    }
    totalPrice%=10000;
}

void keyControl()
{
    if(io_key_1==0)
    {
       if(io_key_1==key1State)
      key1DownCount++;
       else
      key1DownCount=0;
    }
    if(key1DownCount>=10)
    {
       if(keyLock==0)
       {
      keyLock=1;
      perPrice++;
                innerCount=201;
       }
       key1DownCount=0;
    }


    if(io_key_1==1)
    {
       if(io_key_1==key1State&&keyLock==1)
      key1UpCount++;
       else
      key1UpCount=0;
    }
    if(key1UpCount>=20)
    {
       if(keyLock==1)
       {
      keyLock=0;
       }
       key1UpCount=0;
    }

    if(io_key_2==0)
    {
       if(io_key_2==key2State)
```

```
      key2DownCount++;
       else
      key2DownCount=0;
   }
   if(key2DownCount>=10)
   {
      if(keyLock==0)
      {
      keyLock=1;
      if(perPrice>=2)
         perPrice--;
      innerCount=201;
       }
       key2DownCount=0;
   }


   if(io_key_2==1)
   {
      if(io_key_2==key2State&&keyLock==1)
      key2UpCount++;
       else
      key2UpCount=0;
   }
   if(key2UpCount>=20)
   {
      if(keyLock==1)
      {
      keyLock=0;
       }
       key2UpCount=0;
   }
}

void init()
{
   distance=0;
   totalPrice=0;
   timerCount=0;
   waitPrice=0;
   innerCount=0;
   key1UpCount=0;
   key1DownCount=0;
   key2UpCount=0;
```

```
   key2DownCount=0;
   key1State=1;
   key2State=1;
   keyLock=0;
}


void main()
{
     buzzer=0;
    perms=0;
   ms=0;
   second=0;
   minute=0;
//////////////////////////////////////////////
     minute=i2c_read(0x50,0);
//////////////////////////////////////////////
   init();
   perPrice=9;
   startPrice=80;

   TL0=(65536-5)%256;
   TH0=(65536-5)/256;
   TMOD=0x06;
   EA=1;
   ET0=1;
   TR0=0;
   TH1=0xB8;
   TL1=0x00;
   TR1=1;

   while(1)
   {perms++;
      if(perms>=122)
      {
     ms++;
     perms=0;
      }
      if(ms>=14)
      {
     second++;
     ms=0;
      }
      if(second>=2)
```

```
 {
minute++;
second=0;
 }


 if(key0==1)
 {
if(running==1)
{
   init();
   buzzer=1;
}

TR0=0;
running=0;
 }
 else
 {
if(running==0)
{
   init();
   buzzer=1;
}

TR0=1;
running=1;
 }

 if(running==0)
 {
if(innerCount>=250)
   buzzer=0;

if(innerCount<=200)
{
    if(key1==0)
        timeDisplay();
    else
startPriceDisplay();

}

else if(innerCount<=400)
```

```
        {
        if(key1==0)
            timeDisplay();
        else
    perPriceDisplay();

    }
    else
       innerCount=0;

    keyControl();
     }
     else
     {
    totalPriceDisplay();
    if(innerCount>=100)
       buzzer=0;
    if(innerCount>=300)
    {
       timerCount=0;
       waitPrice+=3;
       innerCount=0;
    }
     }

     if(TF1==1)
     {
    TF1=0;
    TH1=0xB8;
    TL1=0x00;
    innerCount++;
     }

     key1State=io_key_1;
     key2State=io_key_2;
     priceCount();
       ////////////////////////////////////
     i2c_write(0x50,0,minute);
       }
}

void Timer0() interrupt 1
{
   timerCount++;
```

```
    if(innerCount>=290)
        innerCount=0;

    if(timerCount>=5)
    {
        timerCount=0;
        distance++;
    }
}
```