

MCS-51单片机应用实例

市电定时开关

市电定时开关功能描述

- 通过单片机控制220V市电的定时通断；
- 工作方式可分为定时开、定时关；
- 开关模式、定时时间可设置；
- 通过按键开关、数码管、LED等和用户接口。

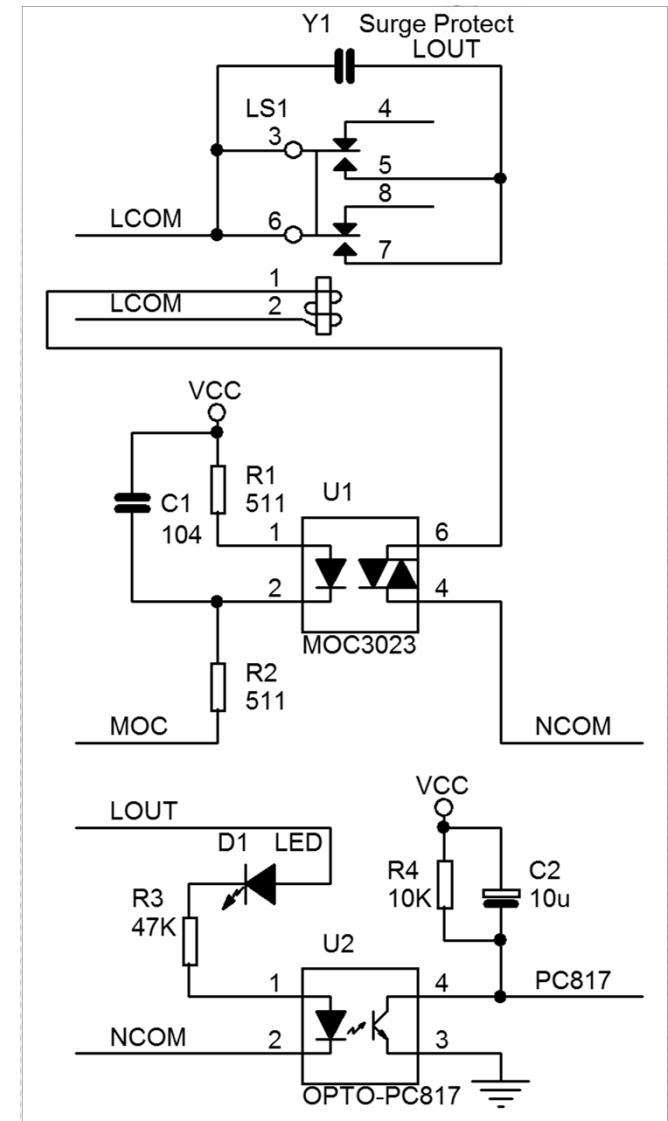
- 具体功能描述
 - 4只按键开关用于设置开关模式及计时值；
 - 2只数码管用于显示计时时间；
 - 4只LED用于指示工作状态；
 - 1只蜂鸣器用于声音提醒；
 - 1个220V交流继电器用于控制市电的通断；
 - 选用AT89C2051作为CPU。

MCS-51单片机应用实例

市电定时开关 - 电路设计

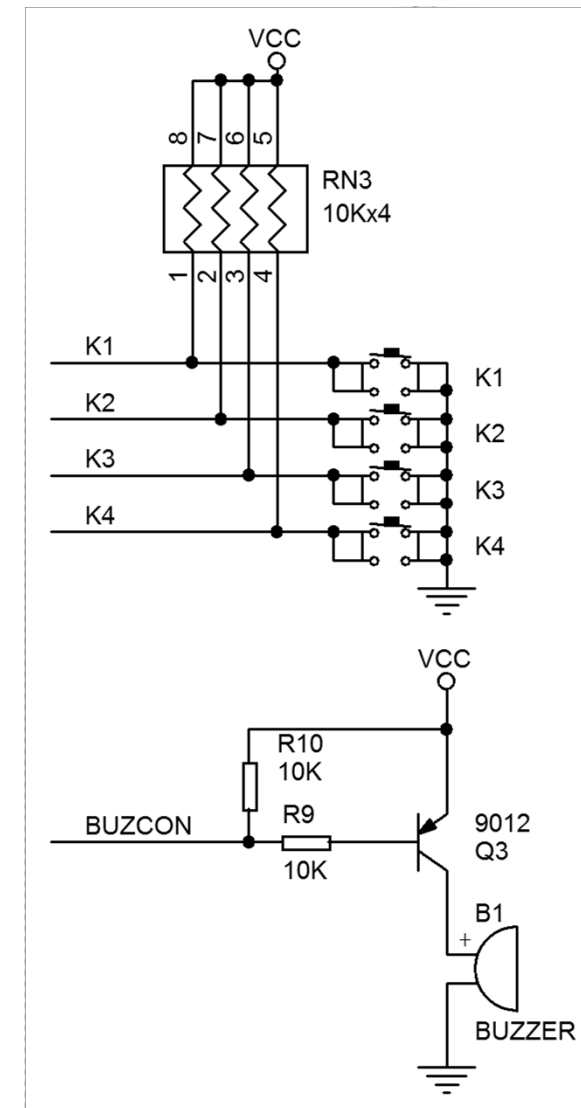
市电通断控制电路

- LS1为220V交流继电器，线圈一端接L线输入（ LCOM ），另一端接U1的受控输出，即N线（ NCOM ）；
- 单片机通过MOC端控制MOC3023（ U1 ）的通断；
- MOC端输出0，则U1导通，LS1吸合，LCOM端输入的市电经继电器的3/6触点输出到5/7触点（ LOUT ）；
- LOUT输出有效后，通过D1、R3到NCOM，使U2导通，U2的引脚4将变为低电平，供单片机判断市电输出是否正常。

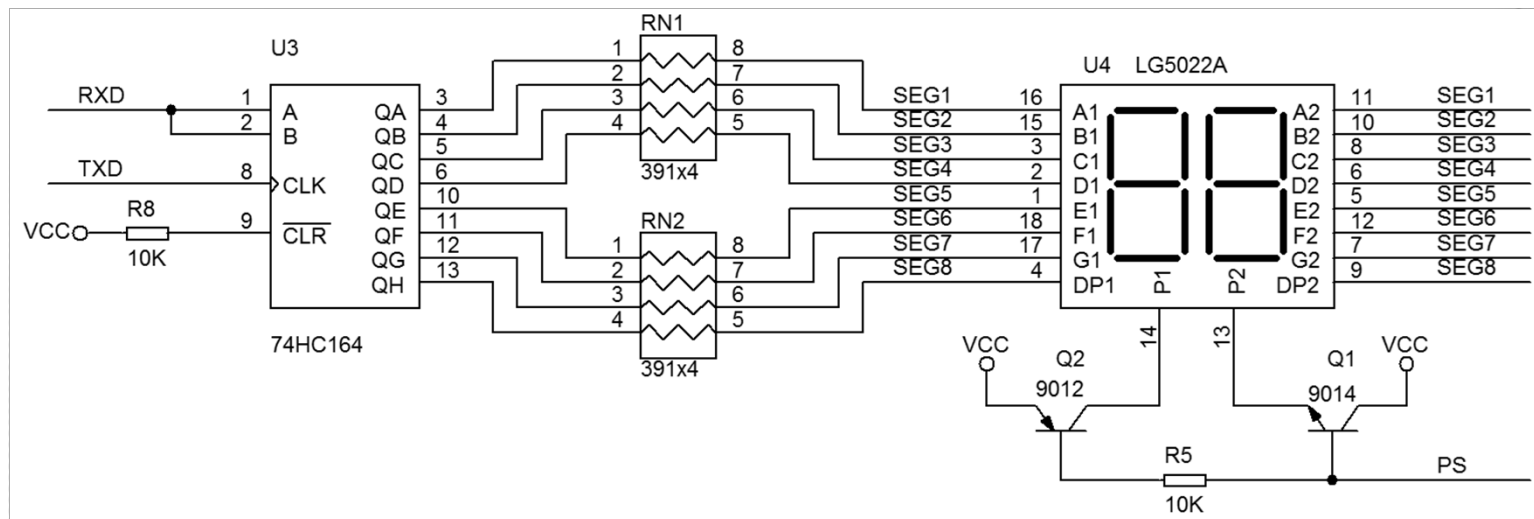


按键及蜂鸣器驱动

- 系统设置4个按键K1~K4，功能分别为+1、-1、功能切换、确认；
- 按键一端上拉，输入单片机端口，另一端接地。所以按键未按下，单片机读到高电平，按键按下时，单片机读到低电平；
- 单片机通过端口引脚驱动BUZCON，输出低电平则蜂鸣器鸣响。

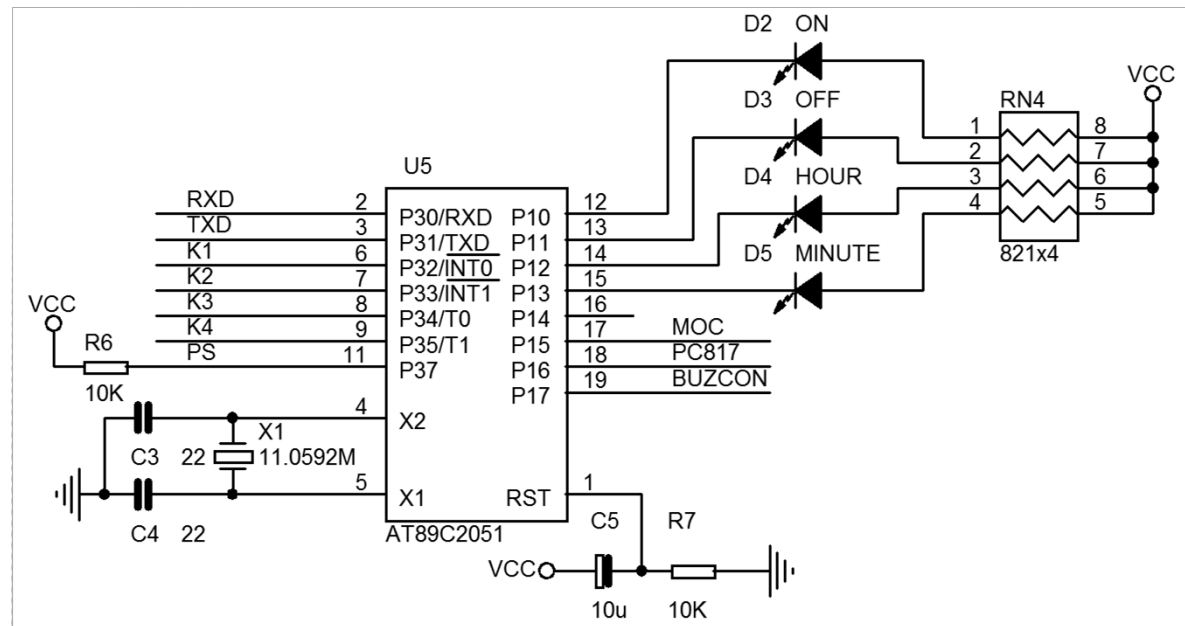


数码管驱动电路



- 由于单片机端口引脚只有15个，因此使用74HC164通过单片机串行口方式0扩展一个并行端口用于输出数码管的段码；
- 数码管选用二联共阳型模块LG5022A；
- 使用单片机的1个I/O端口引脚分时导通两个不同类型的三极管（PNP和NPN），完成两个数码管的位驱动。

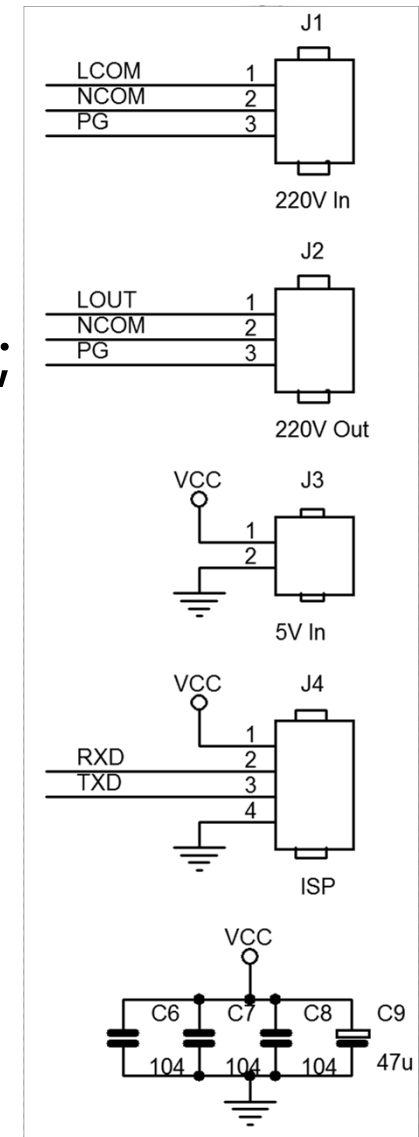
CPU及LED驱动电路



- 系统设计了4只LED，由P1.0~P1.3驱动，分别表示定时开、定时关、设置小时、设置分钟状态；
- P3.0、P3.1用于扩展并行I/O端口及ISP；
- P3.2~P3.5用于扫描按键，P3.7用于数码管位驱动；
- P1.5~P1.7分别为可控硅驱动、市电状态回读及蜂鸣器控制。

其他电路

- J1为220V输入接线端子；
- J2为220V输出三相插座；
- J4为ISP插座，和单片机的串行口相连，供外部TTL电平的串行口接入，方便在线更新程序；
- C6~C8为各芯片的去耦电容，C9为续流电容，确保系统工作时电压的纯净、稳定。

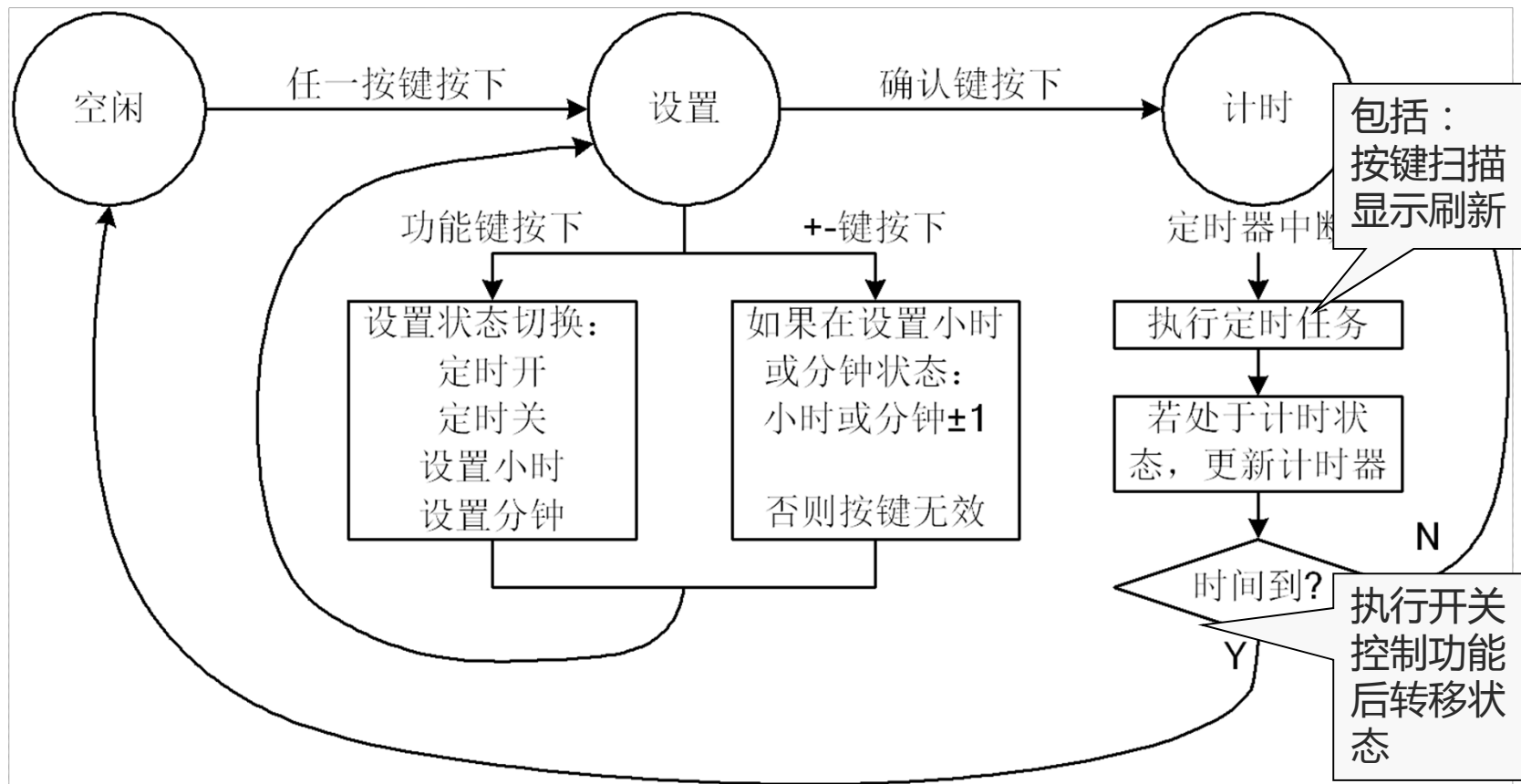


MCS-51单片机应用实例

市电定时开关 - 系统分析

系统状态转移图

- 系统状态分为三种：空闲、设置、计时；
- 外部事件（定时中断调度的按键扫描及状态处理）的发生驱动着系统在各状态之间的转换。



程序设计：系统参数定义

```
#include <reg51.h>
#define OSC 11059200 // 定义系统晶振频率
#define PSEC 100 // 系统时间片10ms(100Hz)

#define ON 0
#define OFF 1 // 定义开关控制的宏

#define RELAYCTRL P15 // 继电器控制引脚
#define RELAYSTAT P16 // 继电器状态检测
#define BUZCTRL P17 // 蜂鸣器控制

#define LEDHOUR P10
#define LEDMINUTE P11
#define LEDTIMERON P12
#define LEDTIMEROFF P13 // 定义各LED的驱动引脚

#define KEYUP 1
#define KEYDOWN 2
#define KEYFUNCTION 3
#define KEYENTER 4 // 定义各按键编号

#define POSCTRL P37 // 定义数码管驱动引脚
```

程序设计：系统状态及全局变量定义

// 定义系统状态编码

```
#define S_FREE 0x00
```

```
#define S_SETUP 0x01
```

```
#define S_TIMER 0x02
```

// 定义全局变量

```
unsigned char code CHTable[] = {0x03,0x9f,0x25,0x0d,0x99,0x49,0x41,0x1f,\n    0x01,0x09,0x05,0xc1,0xe5,0x85,0x61,0x71,0xc5,0xd5}; // 定义数码管字形码, 0~F,o,n
```

```
unsigned char code KeyMap[] = {0x01,0x02,0x04,0x08}; // 定义按键扫描掩码
```

```
unsigned char Hour,Minute; // 倒计时小时、分钟
```

```
unsigned long SysTick; // 系统秒计时
```

```
unsigned char Tick; // 10ms计时
```

```
unsigned long Seconds; // 总倒计时秒数
```

```
unsigned char DispBuf[2]; // 数码管显示缓冲区
```

```
unsigned char SysState; // 系统状态
```

```
unsigned char LEDNum; // 当前点亮的LED号
```

```
unsigned char KeyNum; // 当前按下的键号(刚释放)
```

```
unsigned char PrevKey,CurrKey; // 上次和本次按键扫描值
```

```
bit TimeOut; // "定时时间到"标志
```

程序设计：系统初始化

```
void SysInit(void)
{
    SCON = 0x00;                // 串口模式0，8位移位寄存器
    DispBuf[0] = 0xff;
    DispBuf[1] = 0xff;
    SBUF = 0xff;
    while(!TI);
    TI = 0;                      // 送0xFF并等待发完，熄灭LED

    TMOD = 0x01;                // 定时器0,方式1:16位定时器
    TH0 = (65536 - OSC/12/PSEC)/256; // 初始化10ms定时常数
    TL0 = (65536 - OSC/12/PSEC)%256;
    TR0 = 1;                    // 启动定时器0
    ET0 = 1;                    // 允许T0中断
    EA = 1;                     // 打开总中断允许

    PrevKey = 0x0f;
    CurrKey = 0x0f;
    KeyNum = 0;
    LEDNum = 0;
    TimeOut = 0;
    SysState = S_FREE;
}
```

程序设计：延时及设置初始化

```
/******  
函数: void Delay1ms(int ms)  
功能: 循环延时ms*1ms  
*****/  
void Delay1ms(int ms) {  
    int i;  
    while(--ms) {  
        i = 128;  
        while(--i);  
    }  
}  
/******  
函数: void InitSetup(void)  
功能: 系统进入设置状态前的初始化过程  
*****/  
void InitSetup(void){  
    Hour = 0;  
    Minute = 0;                // 初始化定时的小时、分钟为0  
    LEDNum = 3;                // LED指示状态为设置" 小时 "  
    DispBuf[0] = CHTable[0];  
    DispBuf[1] = CHTable[0];   // 数码管显示 "00"  
}
```

程序设计：定时器0 ISR（系统核心程序）

```
/*  
函数: void T0ISR(void) interrupt 1  
功能: 定时器T0中断服务程序  
倪晓军 2007-06-09 10:34  
*/  
void T0ISR(void) interrupt 1  
{  
    // 首先完成系统计时工作  
    TH0 = (65536 - OSC/12/PSEC)/256;    // 重新设置10ms定时常数  
    TL0 = (65536 - OSC/12/PSEC)%256;  
  
    Tick ++;    // 更新系统Tick  
    if(Tick >= 100)    // 表示1秒定时时间到  
    {  
        Tick = 0;    // 清毫秒计时变量  
        SysTick ++;    // 系统秒计时变量+1  
    }  
}
```

程序设计：定时器0 ISR（系统核心程序）

```
if(SysState == S_TIMER)           // 如果处于计时状态则减1秒,每秒更新一次
{
    if(Seconds > 0) Seconds --;      // 总的计时时间减1秒
    if(Seconds >= 3600)              // 更新时、分计时变量
    {
        Hour = (unsigned char)(Seconds/3600);
        Minute = (unsigned char)((Seconds%3600)/60);
    }
    else
    {
        Hour = 0;
        Minute = (unsigned char)(Seconds);
    }

    if(LEDNum==4)                  // 定时关状态，此时继电器应吸合
    {
        if(RELAYSTAT != ON) BUZCTRL = ON;
        else BUZCTRL = OFF;
    }
}
}
```


程序设计：定时器0 ISR（系统核心程序）

// 显示处理。效果：1分钟以内显示秒倒计时，1分钟以上轮流显示时、分

```
if(SysState == S_TIMER)                // 如果处于计时状态
{
    if(Seconds < 60)                    // 剩余1分钟内直接显示秒倒计时
    {
        if(Tick<=50)                  // 前500ms显示秒数
        {
            DispBuf[0] = CHTable[Minute/10];
            DispBuf[1] = CHTable[Minute%10];
        }
        else                            // 后500ms显示秒+小数点
        {
            DispBuf[0] = CHTable[Minute/10] & 0xfe;
            DispBuf[1] = CHTable[Minute%10] & 0xfe;
        }
    }
    else                                // 剩余1分钟以上交替显示时、分
    {
```

程序设计：定时器0 ISR（系统核心程序）

```
if((SysTick % 2) == 0)                // 偶数秒，显示小时
{
    LEDHOUR = ON;
    LEDMINUTE = OFF;                  // 小时LED点亮，分钟LED熄灭

    if(Tick<=50)                      // 前500ms显示倒计时小时数
    {
        DispBuf[0] = CHTable[Hour/10];
        DispBuf[1] = CHTable[Hour%10];
    }
    else                              // 后500ms显示小时数+小数点
    {
        DispBuf[0] = CHTable[Hour/10] & 0xfe;
        DispBuf[1] = CHTable[Hour%10] & 0xfe;
    }
}
else                                  // 奇数秒，显示分钟
{
```

程序设计：定时器0 ISR（系统核心程序）

```
LEDHOUR = OFF;
LEDMINUTE = ON;           // 分钟LED点亮，小时LED熄灭

if(Tick<=50)              // 前500ms显示倒计时分钟数
{
    DispBuf[0] = CHTable[Minute/10];
    DispBuf[1] = CHTable[Minute%10];
}
else                       // 后500ms显示分钟数+小数点
{
    DispBuf[0] = CHTable[Minute/10] & 0xfe;
    DispBuf[1] = CHTable[Minute%10] & 0xfe;
}
}
```

程序设计：定时器0 ISR（系统核心程序）

```
if(Seconds == 0)                                // 定时时间到
{
    if(LEDNum == 3)                              // 定时开，此时应吸合继电器
    {
        BUZCTRL = ON;                          // 蜂鸣器鸣响
        RELAYCTRL = ON;                        // 吸合继电器
    }

    if(LEDNum == 4)                              // 定时关，此时应释放继电器
    {
        BUZCTRL = ON;                          // 蜂鸣器鸣响
        RELAYCTRL = OFF;                       // 释放继电器
    }
    TimeOut = 1;                                // 设置计时时间到标志，通知
                                                // 主程序做相应处理
}

KeyNum = KeyPressed();                          // 扫描按键(10ms执行一次)
RefreshDisplay();                              // 刷新数码管显示(10ms执行一次)
}
```

程序设计：按键扫描

```
unsigned char KeyPressed(void)
{
    unsigned char ci;

    PrevKey = CurrKey;                                     // 将本次扫描结果保存到上次扫描结果中

    CurrKey = (P3>>2)&0x0f;                               // 再读取本次按键扫描结果
    if((PrevKey ^ CurrKey) != 0)                           // 如果两次扫描值不等
    {
        if(CurrKey != 0x0f) {                             // 表明有键按下
            BUZCTRL = ON;                                 // 蜂鸣器响
            return 0;                                     // 按键还未释放，不返回有效键值
        }
        else {                                             // 刚才按下的按键已释放
            BUZCTRL = OFF;                                // 蜂鸣器停
            for(ci=0;ci<4;ci++)                            // 判断到某个按键按下即返回其键值
                if((PrevKey & KeyMap[ci]) == 0) return (ci+1);
        }
    }
    return 0;
}
```

程序设计：显示信息刷新

```
/******  
函数: void RefreshDisplay(void)  
功能: 刷新数码管显示  
倪晓军 2007-06-09 11:00  
*****/  
void RefreshDisplay(void)  
{  
    // 轮流点亮左右两个数码管，10ms切换一次，显示频率为50Hz  
    if((Tick % 2) == 0)  
    {  
        SBUF = DispBuf[0];                // 显示数据送串行口，通过164并行输出  
        while(!TI);                        // 约10us完成  
        TI = 0;  
        POSCTRL = 0;                       // 第一个数码管显示  
    }  
    else  
    {  
        SBUF = DispBuf[1];                // 显示数据送串行口，通过164并行输出  
        while(!TI);                        // 约10us完成  
        TI = 0;  
        POSCTRL = 1;                       // 第二个数码管显示  
    }  
}
```

程序设计：显示信息刷新

```
// 根据当前状态更新LED显示
```

```
if(SysState == S_SETUP)
```

```
{
```

```
    switch(LEDNum)
```

```
{
```

```
    case 1:
```

```
        LEDHOUR = ON;
```

```
        LEDMINUTE = OFF;
```

```
        LEDTIMERON = OFF;
```

```
        LEDTIMEROFF = OFF;
```

```
        break;
```

```
    case 2:
```

```
        LEDHOUR = OFF;
```

```
        LEDMINUTE = ON;
```

```
        LEDTIMERON = OFF;
```

```
        LEDTIMEROFF = OFF;
```

```
        break;
```

```
// 如果系统处于设置状态
```

```
// LEDNum指示着当前的状态
```

```
// 设置小时状态
```

```
// 点亮小时指示LED
```

```
// 设置分钟状态
```

```
// 点亮分钟指示LED
```

程序设计：显示信息刷新

```
case 3:                                // 定时开状态
    LEDHOUR = OFF;
    LEDMINUTE = OFF;
    LEDTIMERON = ON;
    LEDTIMEROFF = OFF;
    break;
case 4:                                // 定时关状态
    LEDHOUR = OFF;
    LEDMINUTE = OFF;
    LEDTIMERON = OFF;
    LEDTIMEROFF = ON;
    break;
default:                               // 状态错误，熄灭所有LED
    LEDHOUR = OFF;
    LEDMINUTE = OFF;
    LEDTIMERON = OFF;
    LEDTIMEROFF = OFF;
    break;
    }
    }
}
```


程序设计：按键处理

```
/******  
函数: void KeyProcess(unsigned char State,unsigned char Key)  
功能: 根据系统所处状态和按键信息进行处理  
倪晓军 2007-06-09 12:21  
*****/  
void KeyProcess(unsigned char State,unsigned char Key)  
{  
    if(State == S_FREE)                // 如果当前状态为空闲  
    {  
        InitSetup();                  // 进行设置初始化  
        SysState = S_SETUP;           // 转移到设置状态  
        return;                       // 返回，后续情况无需处理  
    }  
}
```

程序设计：按键处理

```
if(State == S_SETUP)                // 如果系统处于设置状态
{
    switch(Key)                      // 根据按键值进行分支判断
    {
        case KEYUP:                 // 第一种情况：加1键按下
            if(LEDNum == 1)          // 如果处于设置小时状态
            {
                Hour ++;              // 定时小时数+1，并刷新显示
                if(Hour > 99) Hour = 99;
                DispBuf[0] = CHTable[Hour/10];
                DispBuf[1] = CHTable[Hour%10];
            }
            if(LEDNum == 2)          // 如果处于设置分钟状态
            {
                Minute ++;            // 定时分钟数+1，并刷新显示
                if(Minute > 59) Minute = 59;
                DispBuf[0] = CHTable[Minute/10];
                DispBuf[1] = CHTable[Minute%10];
            }
            break;
    }
}
```

程序设计：按键处理

```
case KEYDOWN:                                // 第二种情况，减1键按下
    if(LEDNum == 1)                          // 如果系统处于设置小时状态
    {
        if(Hour > 0) Hour --;                // 定时小时数-1，并刷新显示
        DispBuf[0] = CHTable[Hour/10];
        DispBuf[1] = CHTable[Hour%10];
    }
    if(LEDNum == 2)                          // 如果系统处于设置分钟状态
    {                                        // 定时分钟数-1，并刷新显示
        if(Minute > 0) Minute --;
        DispBuf[0] = CHTable[Minute/10];
        DispBuf[1] = CHTable[Minute%10];
    }
    break;
```

程序设计：按键处理

```
case KEYFUNCTION:                                // 第三种情况：功能(切换)键
    LEDNum++;
    if(LEDNum > 4) LEDNum = 1;                    // 切换LED显示1次，并循环

    if(LEDNum == 1)                               // 如果本次切换到小时显示
    {                                              // 更改显示缓冲区内容为小时
        DispBuf[0] = CHTable[Hour/10];
        DispBuf[1] = CHTable[Hour%10];
    }

    if(LEDNum == 2)                               // 如果本次切换到分钟显示
    {                                              // 更改显示缓冲区内容为分钟
        DispBuf[0] = CHTable[Minute/10];
        DispBuf[1] = CHTable[Minute%10];
    }

    break;
```

程序设计：按键处理

```
case KEYENTER:                                // 第四种情况：选择键
    if((LEDNum==3) || (LEDNum==4))
    {      // 只有处于定时开或定时关状态时此按键才有效
        SysState = S_TIMER;
        Seconds = Hour*3600 + Minute*60;
        if(LEDNum == 4) RELAYCTRL = ON;
        else RELAYCTRL = OFF;
        BUZCTRL = OFF; Delay1ms(200); BUZCTRL = ON; Delay1ms(500);
        BUZCTRL = OFF;
    }
    else
    {
        BUZCTRL = OFF; Delay1ms(200); BUZCTRL = ON; Delay1ms(200);
        BUZCTRL = OFF; Delay1ms(200); BUZCTRL = ON; Delay1ms(200);
        BUZCTRL = OFF;
    }
    break;
default:    break;
}
return;
}
```

程序设计：主函数

```
void main(void)
{
    unsigned char Key;

    SysInit();
    while(1)
    {
        if(KeyNum != 0) {
            Key = KeyNum;
            KeyNum = 0;
            KeyProcess(SysState,Key);
        }
        if(TimeOut) {
            TimeOut = 0;
            SysState = S_FREE;
            Delay1ms(500);
            if(LEDNum==3) {
                if(RELAYSTAT != ON) BUZCTRL = ON;
                else BUZCTRL = OFF;
            }
            else BUZCTRL = OFF;
        }
    }
}
```

```
// 如果有按键事件
// 暂存按键码
// 清按键码，以备记录下次按键事件
// 调用按键处理函数

// 如果定时时间到
// 首先清定时时间到标志
// 系统转移到空闲状态
// 延时，供继电器动作
// 如果是定时开状态，此时继电器应吸合
// 如继电器状态不对，则通过蜂鸣器报警

// 定时关，直接关闭蜂鸣器
```