红外学习型万能解码遥控开关及调光调速器

一、概述

本产品能实现白炽灯及各种灯具的控制。手控、遥控均可实现相同功能。其最大特点是不需特定遥控器，对任何遥控器的任何按键都可以学习后进行操作，所以具有很大的灵活性及实用性。

二、使用说明

1、产品设置

1．10A 红外线遥控开关——CPU 第 19 脚接地，20 脚悬空。（可选择可控硅或继电器）。
2．800W 红外线调光开关——CPU 第 19、20 脚均悬空。
3．500VA 风机红外线调速——CPU 第 19 脚悬空，20 脚接地。

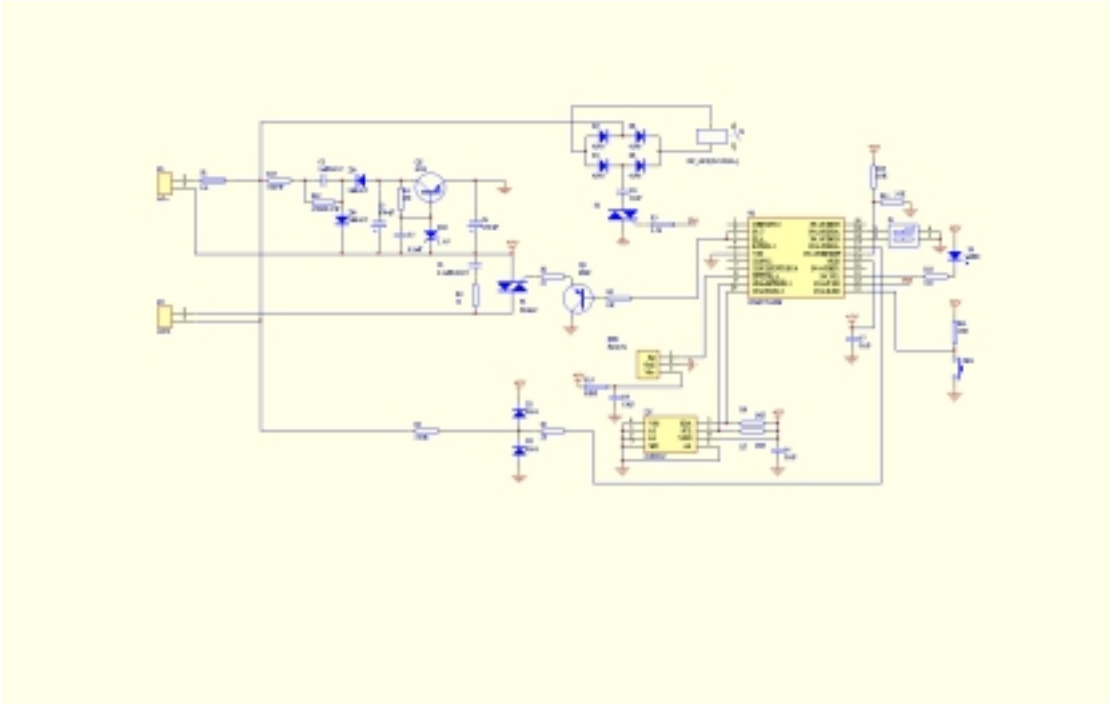
2、红外学习

1．10A 遥控开关学习方式为：持续按下手控按钮 4 秒钟进入学习状态，指示灯闪烁，按下遥控器上的某个指定键，指示灯闪烁两下，学习成功。不按遥控键，10 秒后开关进入正常状态。
2．其他两种产品的学习方式为：连续点击两下后持续按住进入学习状态，后同。

3、操作说明

1．10A 红外线遥控开关：按一次键，改变工作状态一次]，即开变关，或关变开。
2．800W 红外线调光开关
点动一次：关——〉渐亮到上一次调定的亮度。
亮——〉渐暗到最小亮度后关断 。
连续两次点动：
关——〉立即达到前一次亮度。
亮——〉立即熄灭。
持续按住：亮度逐渐增加，合适即可放开。
点动后再持续按住：亮度逐渐减小，合适即可放开。
3．500VA 风机红外线调速
点动一次：关——〉中速启动，5 秒后逐渐转变为上次调定的转速。
转动——〉断电。
持续按住：逐渐增加转速到合适放开按钮。
点动后再持续按住：逐渐减小转速到合适放开按钮。

三、电路原理图



四、流程图

为便于理解，这里就主要程序做出部分流程图(见附图)

五、程序源代码

本程序用 Franklin C51 编写

```
/*****************************************/
/*遥控开关、调光调速器
/*广州周立功单片机发展有限公司
/*2000.6
/*****************************************/
#include "reg764.h"

#define AngleLimit 300   /*消除相角误差*/
#define SpeedLimit 700   /*控制最高和最低速度*/
                         /*  以控制 u、i 之间的相位差*/
sbit   SDA=P1^2;
sbit   SCL=P1^3;

sbit Led=P0^7;
sbit Key=P1^1;
sbit FIRE=P1^6;
sbit CtlLamp=P0^1;
sbit NO_R_SW=P0^2;
sbit SW=P1^0;
```

```
/*预留堆栈*/
unsigned char bdata SP1;
unsigned char bdata SP2;
unsigned char bdata SP3;
unsigned char bdata SP4;
unsigned char bdata SP5;
unsigned char bdata SP6;
unsigned char bdata SP7;
unsigned char bdata SP8;

unsigned char bdata FLAG;
sbit ON=FLAG^0;
sbit KeyDown=FLAG^1;      /*用于识别按键口线为 1 时,是否被按下过*/
sbit CONTI=FLAG^2;        /*标志键被持续按下*/
sbit Task_Ovr=FLAG^3;     /*任务处理标志*/
sbit FuncSet=FLAG^4;      /*功能处理标志*/
sbit KeyStart=FLAG^5;     /*键识别周期开始标志*/
sbit IsStudy=FLAG^6;      /*红外学习标志*/
sbit StudyOK=FLAG^7;      /*红外学习成功标志*/

unsigned char bdata FLAG1;
sbit RmtOK =FLAG1^0;      /*遥控键正确识别标志*/
sbit CONTI_R=FLAG1^1;     /*遥控键被持续按下*/
sbit RKeySt1=FLAG1^2;     /*遥控键状态 1*/
sbit RKeySt2=FLAG1^3;     /*遥控键状态 2*/
sbit RmtStart=FLAG1^4;

unsigned char data AGE;              /*按键年龄*/
unsigned char data SwStyTime;     /*开关状态时,按键按下时间*/
unsigned char data FuncCode;      /*功能代码*/
unsigned char data DownTimes;     /*按键连击次数*/

unsigned int data Fire_Angle;
unsigned int data MaxFireAngle;      /*交流电半周期值对应的定时器值*/
unsigned int Last_FireAngle;
unsigned char data IRCLK; /*遥控识别计时时间,在过零中断每 10ms 增 1*/
unsigned char data IKCLK;     /*键识别时间,在过零中断每 10ms 增 1*/

unsigned char data Buf[50];
unsigned char data Buf1[2];

void Delay(unsigned int);
int WriteEE(unsigned char SubAdr,unsigned char Num,unsigned char *Wbuf);
int ReadEE(unsigned char SubAdr,unsigned char Num,unsigned char *Rbuf);
```

```
unsigned char ReadByte();
int    SendByte(unsigned char);
void Start(void);
void Stop(void);
void SendACK(void);
void SendNoACK(void);

void WDFeed(void);
void Task_10ms(void);
void FuncHandle(void);
void Calc_MaxFireAngle(void);
void Read_Last_FireAngle(void);
void Write_FireAngle(void);
void Read_MaxFireAngle(void);
void Write_MaxFireAngle(void);

unsigned char Get_L_Width();
unsigned char Get_H_Width();

/*定时器 0 中断服务程序,用于可控硅点火*/
void Service_Timer0() interrupt 1
{
              FIRE=0;       /*产生点火脉冲*/
              FIRE=0;
             FIRE=0;
             FIRE=1;
              FIRE=1;
             FIRE=1;

             FIRE=0;       /*产生点火脉冲*/
              FIRE=0;
             FIRE=0;
             FIRE=1;
              FIRE=1;
             FIRE=1;

             FIRE=0;       /*产生点火脉冲*/
              FIRE=0;
             FIRE=0;
             FIRE=1;
              FIRE=1;
             FIRE=1;

             TR0=0;                /*关闭定时器,禁止中断*/
```

```
        ET0=0;
}

/*外部中断 1 服务程序,用于遥控操作*/
void Service_Int1() interrupt 2
{
    unsigned char data i,j,k;
    EX1=0;
    if(IsStudy==0){
        /*遥控识别*/
        if(StudyOK==1) goto Int1_reti;/*等待处理学习结果*/
        if(FuncSet==1) goto Int1_reti;/*等待功能处理*/
        if(CONTI_R==1) goto Int1_reti;/*持续按键,不再进行码识别*/

        while(IRCLK<100){
            WDRST=0X1E;
            WDRST=0XE1;   /*清除看门狗*/
            RmtOK=1;

          TH1=TL1=0;
            TR1=1;      /*启动定时器*/
            while(INT1!=1);
            TR1=0;
            /*宽度小于 0.25ms,是干扰脉冲*/
            if(TH1==0 && TL1<120) {
                if(RmtStart)
                    goto Ignore;
                else
                goto Int1_reti;/*干扰脉冲,且为首次,退出中断*/
            }
            j=(TH1*256+TL1 )/250;
            j/=2;
            if(j!=Buf[0]) break;
          k=Get_H_Width();
            if(k!=Buf[1]) break;

            for(i=1;i<25;i++){
                j=Get_L_Width();
                k=Get_H_Width();
                if(k==255) break;/*检测到 255,不再比较*/
                if(j!=Buf[2*i]) {RmtOK=0;break;}
                if(k!=Buf[2*i+1]) {RmtOK=0;break;}
            }
```

```
        if(RmtOK==0) break;

        /*是遥控开关,设定状态,退出*/
        if(!NO_R_SW){
            RKeySt1=1;
            RKeySt2=0;
            IRCLK=120;
            i=250;          /*延迟 250ms*/
        while(i--) for(j=0;j<125;j++);
            break;
        }

        If(RKeySt1==0 && RKeySt2==0)
           {RmtStart=1;IRCLK=0;RKeySt1=1;RKeySt2=0;}/*开始计时*/
        /*接收成功,更新状态*/
        else   {RKeySt1=0;RKeySt2=1;}

        /*检测连续按键*/
        k=0;
    while(k<8){

            TF1=0;
        TH1=TL1=0;
            TR1=1;      /*启动定时器*/
            while(INT1!=0){
                WDRST=0X1E;
                WDRST=0XE1;   /*清除看门狗*/

                if(TF1==1) break;
            }
            TR1=0;
            if(TF1==1) break;

        j=(TH1*256+TL1)/250;
            if(j<50) continue;/*间隔>25ms,表示为下一帧信号*/

            /*获取低电平宽度(抗干扰)*/
            TH1=TL1=0;
            TR1=1;
            while(INT1!=1);
            TR1=0;
            if(TH1>0 || TL1>120) k++; /*干扰脉冲不计数*/
        }
    if(k>7){
```

```
                    CONTI_R=1;
                    goto Int1_reti;
                }
            else
                    CONTI_R=0;
Ignore:
                while(INT1){        /*等待 INT1 变低*/
                    WDRST=0X1E;   /*清看门狗*/
                WDRST=0XE1;
                    if(IRCLK>120) goto Int1_reti;
                }
        }

    }

 else{
        /*红外学习*/
        if(StudyOK==1) goto Int1_reti;/*等待处理学习结果*/
        EA=0;
        for(i=0;i<25;i++){
            Buf[2*i]=Get_L_Width();
            Buf[2*i+1]=Get_H_Width();
        }
        for(i=0;i<6;i++) WriteEE(i*8,8,Buf+i*8);/*写学习结果*/
        WriteEE(48,2,Buf+48);
        EA=1;

        i=150;        /*延迟 150ms 滤除后续信号*/
    while(i--) for(j=0;j<125;j++);

        StudyOK=1;          /*置完成标志*/
    }

Int1_reti:
    EX1=1;
}

/*比较器中断,检测过零,作为相移控制的起始点(每 10ms 中断一次)*/
void Service_Cmp1() interrupt 12
{
    CMP1 &= 0xFE; /*清比较器 1 中断标志*/
    /*根据触发角设置时间常数*/
    TH0= -(Fire_Angle)/256;
    TL0=-(Fire_Angle)%256;
```

```
if(ON==1){
    TR0=1;              /*启动定时器 0,开定时器 0 中断*/
    ET0=1;
}
else{
    TR0=0;
    ET0=0;
}

Task_Ovr=1;     /*设置任务进程*/
if(RmtStart) IRCLK++;           /*遥控识别计时时间+*/

/*以下处理按键*/
if(KeyStart){
     IKCLK++;          /*计时*/

     if(Key==0){
          KeyDown=1;       /*键已按下*/
          AGE++;           /*按键年龄增长*/
          if(NO_R_SW){    /*不是遥控开关,才能设置为持续方式*/
              if(AGE>=30){
                CONTI=1;
                    goto retu;
              }/*按下按键超过 0.3s,为持续方式*/
          }
          else{/*开关模式下,按下 2.5s 进行红外学习*/
               if(AGE==50) SwStyTime++;
               if(SwStyTime==6){
          SwStyTime=0;
                    CONTI=1;
               DownTimes=2;
               IKCLK=120;
                    goto retu;
               }
          }
     }
     else{
          if(KeyDown==1 && AGE>=3 && AGE<=30){
               DownTimes++;
               if(!NO_R_SW) IKCLK=120; /*是遥控开关,立即执行功能*/
          }
          KeyDown=0;     /*键已释放*/
          AGE=0;         /*按键弹起时,清除年龄*/
          SwStyTime=0;
```

```
                CONTI=0;        /*清除持续标志*/
            }
        }

        if(Key==1) CONTI=0;   /*键已释放,清除持续标志*/

retu:;
}

/*主程序*/
main()
{
    WDFeed();           /*清看门狗*/
    WDCON=0X16;         /*设置看门狗*/

    /*初始化比较器,用于过零检测,Vref=5V*/
    CMP1 &= 0xFE; /*清比较器中断标志*/
    PT0AD=0X30;     /*  禁止 CN1A,CMPREF 作为数字输入口  */
    P0M2 &= 0XCF;       /*  禁止 CN1A,CMPREF 作为数字输出口*/
    P0M1 |= 0X30;     /*开放比较器功能*/
    CMP1=0X24; /*CHN1A 作为比较器正输入端,CMPREF 为参考电压,比较结果送 CMP1*/
    Delay(1);   /*延时 1ms*/
    CMP1 &= 0xFE; /*清比较器 1 中断标志*/
    EC1=1;/*使能比较器一中断*/

    /*设置定时器*/
    TMOD=0X11;       /*定时器 0,1 方式 1*/
    IT1=1;              /*外部中断 1 为边沿触发方式*/
    EX1=1;
    Led=0;               /*LED ON*/

    Calc_MaxFireAngle();   /*计算半周期对应的定时器值*/
    Write_MaxFireAngle();

    IP1H=0X20;              /*比较器 1 中断最高优先级*/
    IP1L=0X20;
    IP0L=0X02;
    ReadEE(0,50,Buf);

    Read_Last_FireAngle(); /*从存储器中读出上一次的速度(亮度)值*/
    Fire_Angle=Last_FireAngle;

    EA =1;                  /*开中断*/
    do{
```

```
        WDFeed();          /*清看门狗*/

        ReadEE(0,50,Buf);            /*循环读出,加强抗干扰能力*/
        Read_MaxFireAngle();

        if(Task_Ovr==1) Task_10ms();
        if(FuncSet==1) FuncHandle();
    }while(1);
}

/*任务处理,由电压过零激活*/
void Task_10ms()
{
    unsigned char data i,ret;

    if(KeyStart==0 && Key==0){/*检测键处理周期中的第一次按键*/
        AGE=0;                  /*清除按键年龄*/
        IKCLK=0;        /*清除识别计时器*/
        DownTimes=0;        /*清除击键次数*/
        KeyStart=1;          /*开始进行按键处理*/
    }

    if(IKCLK<100) goto Remote;/*未完,等待下一次任务*/
    if(CONTI==0)        /*识别时间完,设置功能代码*/
        FuncCode=DownTimes;
    else
        FuncCode=DownTimes+4;

    FuncSet=1;           /*允许进行功能处理*/

    IKCLK=0;
    DownTimes=0;
    KeyStart=0;      /*关闭按键处理*/
    goto Task_Ret;

Remote:
    if(IRCLK<100) goto Task_Ret;

    IRCLK=0;
    if(CONTI_R){
        if(RKeySt1)          FuncCode=4;
        else if(RKeySt2) FuncCode=5;
        else                    FuncCode=0;
    }
```

```
    else{
        if(RKeySt1)          FuncCode=1;
        else if(RKeySt2) FuncCode=2;
        else                      FuncCode=0;
    }
    RKeySt1=0;          /*清除状态*/
    RKeySt2=0;
    RmtStart=0;
    FuncSet=1;          /*允许进行功能处理*/

Task_Ret:
    Task_Ovr=0;    /*清任务进程标志*/
}


void FuncHandle()
{
    unsigned char data i;
    switch(FuncCode){
    case 1:          /*点击一次*/
        if(!NO_R_SW){ /*遥控开关*/
            if(ON){
                ON=0;
                Led=0;
                SW=1;
            }
            else{
                Fire_Angle=AngleLimit;/*可控硅全导通*/
                ON=1;
                Led=1;
                SW=0;
            }
            break;
        }


        Read_Last_FireAngle();
        if(ON==0){
        /*渐亮到上一次调定的亮度*/
            ON=1; /*允许可控硅触发*/
            Led=1;      /*指示灯灭*/
            if(CtlLamp)
                for(Fire_Angle=MaxFireAngleAngleLimit;Fire_Angle>Last_FireAngle;Fire
_Angle--) Delay(1);
```

```
            else{
                Fire_Angle=MaxFireAngle/2;   /*中速运转*/
                Delay(3000);                 /*延迟 5s*/
                /*转变为上次调定的转速*/
                if(Fire_Angle>Last_FireAngle){
                    while(Fire_Angle>Last_FireAngle){
                        Fire_Angle--;
                        Delay(1);
                    }
                }
                else{
                    while(Fire_Angle<Last_FireAngle){
                        Fire_Angle++;
                        Delay(1);
                    }
                }
            }
        }
        else{
            Led=0;      /*指示灯点亮*/
            if(CtlLamp)
                /*渐暗到最小亮度后关闭*/
                for(Fire_Angle=Last_FireAngle;Fire_Angle<MaxFireAngle-
AngleLimit;Fire_Angle++) Delay(1);
            ON=0;       /*不允许可控硅触发*/
        }
        break;
    case 2:         /*点击二次*/
        if(!CtlLamp) break;

        if(ON==0){
            Read_Last_FireAngle();
            Fire_Angle=Last_FireAngle;
            ON=1;               /*可控硅打开为上次亮度*/
            Led=1;              /*指示灯关闭*/
        }
        else{
        /*可控硅关闭*/
            ON=0;
            Led=0;
        }
        break;
    case 3:         /*点击三次*/
        break;
```

```
case 4:          /*持续按住*/
    EX1=0;
    i=0;
    while(CONTI || CONTI_R){/*亮度逐渐增大,直到放开*/
        WDRST=0X1E;  /*清看门狗*/
        WDRST=0XE1;
        if(ON==0){   /*原来关闭,亮度从 0 开始,是电机,不予理会*/
            if(CtlLamp){
                ON=1;
                Led=1;
                Fire_Angle=MaxFireAngle-AngleLimit;
            }
        }

        if(ON){
            if(CtlLamp){/*灯*/
                if(Fire_Angle>AngleLimit){
                    Fire_Angle-=1;
                Delay(1);
                }
            else
                    Fire_Angle=AngleLimit;
            }
            else{/*电机*/
                if(Fire_Angle>AngleLimit+SpeedLimit){
                    Fire_Angle-=1;
                Delay(1);
                }
            else
                    Fire_Angle=AngleLimit+SpeedLimit;
            }
        }
        if(INT1){    /*等待遥控键释放*/
           if(i++>100) CONTI_R=0;
        }
        else
            i=0;
    }
    Write_FireAngle();
    EX1=1;
    break;
case 5:          /*点击一次,持续*/
EX1=0;
    i=0;
```

```
        while(CONTI || CONTI_R){/*亮度逐渐减小,直到放开*/
            WDRST=0X1E;  /*清看门狗*/
            WDRST=0XE1;
            if(ON){
                if(CtlLamp){
                    if(Fire_Angle<MaxFireAngle-AngleLimit) {
                        Fire_Angle+=1;
                        Delay(1);
                    }
                    else{
                        Fire_Angle=MaxFireAngle-AngleLimit;
                        ON=0;
                        Led=0;
                    }
                }
                else{
                    if(Fire_Angle<MaxFireAngle-AngleLimit-SpeedLimit) {
                        Fire_Angle+=1;
                        Delay(1);
                    }
                    else{
                        Fire_Angle=MaxFireAngle-AngleLimit-SpeedLimit;
                        ON=0;
                        Led=0;
                    }
                }
            }
            if(INT1){    /*等待遥控键释放*/
                if(i++>100) CONTI_R=0;
            }
            else
                i=0;
        }
        Write_FireAngle();
        EX1=1;
        break;
    case 6:          /*点击二次,持续*/
        ON=0;        /*禁止可控硅*/
        IsStudy=1;   /*设置红外学习标志*/
        StudyOK=0;
        i=20;        /*设置 10s 学习时间*/
        while(!StudyOK){/*等待学习完成,Led 闪烁*/
            Led=~Led;
            Delay(500);
```

```
                if(i--==0) goto NoSt;
            }
            /*学习成功*/
            Led=1;
            Delay(100);
            Led=0;
            Delay(100);
            Led=1;
            Delay(100);
NoSt:
            Led=0;
            IsStudy=0;/*进入接收状态*/
            StudyOK=0;
            break;      /*红外学习*/
        default:
            break;
        }
        FuncCode=0;
        FuncSet=0;    /*清除功能设置标志*/
}

void Write_FireAngle()
{
        Buf1[0]=Fire_Angle%256;
        Buf1[1]=Fire_Angle/256;
        WriteEE(128,2,Buf1);
}

void Read_Last_FireAngle(void)
{
        ReadEE(128,2,Buf1);
        Last_FireAngle=Buf1[0]+Buf1[1]*256;
}

void Write_MaxFireAngle()
{
        Buf1[0]=MaxFireAngle%256;
        Buf1[1]=MaxFireAngle/256;
        WriteEE(136,2,Buf1);
}

void Read_MaxFireAngle(void)
{
        ReadEE(136,2,Buf1);
```

```
    MaxFireAngle=Buf1[0]+Buf1[1]*256;
}

void WDFeed()   /*清看门狗*/
{
    WDRST=0X1E;
    WDRST=0XE1;
}

/*三次过零的时间间隔除于 2 即为半周期时间值*/
void Calc_MaxFireAngle()
{
   while((CMP1 & 1)==0); /*等待过零*/
   CMP1 &= 0xFE; /*清比较器中断标志*/
   TR0=1;            /*启动定时器*/
   while((CMP1 & 1)==0); /*等待过零*/
   CMP1 &= 0xFE; /*清比较器中断标志*/
   while((CMP1 & 1)==0); /*等待过零*/
   CMP1 &= 0xFE; /*清比较器中断标志*/
   TR0=0;            /*关闭定时器*/
   MaxFireAngle=(TH0*256+TL0)/2;
}

/*测量低电平宽度*/
unsigned char Get_L_Width()
{
    TH1=TL1=0;
    TR1=1;      /*启动定时器*/
    while(INT1!=1);
    TR1=0;
    TL1=(TH1*256+TL1 )/250;
    return(TL1/2);
}

/*测量高电平宽度*/
unsigned char Get_H_Width()
{
    TF1=0;
    TH1=TL1=0;
    TR1=1;      /*启动定时器*/
    while(INT1!=0){
        if(TH1>0X20) return(255);/*脉宽约为 16ms*/
        /*if(TF1==1) return(255);*/
    }
```

```
    TR1=0;
    TL1=(TH1*256+TL1 )/250;
    return(TL1/2);
}

/*页写存储器,Num<=8*/
int WriteEE(unsigned char SubAdr,unsigned char Num,unsigned char *Wbuf)
{
     unsigned char data i;
     Start();                    /*发送 I2C 总线起始条件*/
     if(SendByte(0xa0)!=0) return -1;   /*  发送被控器总线地址*/
     if(SendByte(SubAdr)!=0) return -1;   /*发送写的地址*/
     for(i=0;i<Num;i++)          /*重复操作直到发送完最后一个数据*/
        if(SendByte(*(Wbuf+i))!=0) return -1;
     Stop();                     /*发送 I2C  总线停止条件*/
     Delay(10);                  /*延时 10ms,等待数据写完*/
    return 0;
}
/*读多个字节数据  */
int ReadEE(unsigned char SubAdr,unsigned char Num,unsigned char *Rbuf)
{
      unsigned char data i;
      Start();                    /*  发送 I2C 总线起始条件*/
      if(SendByte(0xa0)!=0) return -1;/*发送被控器总线写地址*/
      if(SendByte(SubAdr)!=0) return -1;    /*发送读首地址*/
      Start();                    /*发送 I2C 总线重复起始条件*/
      if(SendByte(0xa1)!=0) return -1; /*发送被控器总线读地址*/
      if(Num!=1){
            for(i=0;i<Num-1;i++)

                  *(Rbuf+i)=ReadByte();
            SendACK();              /*发送应答信号*/
            }
            *(Rbuf+Num-1)=ReadByte();
      }
      else
            *Rbuf=ReadByte();
      SendNoACK();                  /*最后一个字节,发送非应答信号*/
      Stop();                     /*发送 I2C  总线停止条件*/
      return 0;
}

/*字节数据传送子程序：发送一个字节数据或地址给被控器*/
int SendByte(unsigned char a)
```

```
{
    unsigned char data i,j;
    unsigned char bdata Sin;
    Sin=a;
    for(i=0;i<8;i++)
    {
        if((Sin & 0x80)==0)
            SDA=0;
        else
            SDA=1;
        for(j=0;j<2;j++);
        SCL=1;              /*置时钟线为高，通知被控器开始*/
        for(j=0;j<4;j++);/*保证时钟高周期大于 4μs*/
        SCL=0;
        Sin=Sin<<1;
    }
    for(j=0;j<2;j++);
    SDA=1;
    for(j=0;j<2;j++);
    SCL=1;

    for(j=0;j<2;j++);
    if (SDA==0)  {SCL=0;return 0;}   /*成功,返回 0*/
    for(j=0;j<1;j++);
    SCL=0;
    return -1;                      /*未收到应答,返回-1*/
}

/*数据接收子程序：从被控器接收一个字节数据*/
unsigned char ReadByte()
{
    unsigned char data i,j;
    unsigned char bdata Sin;
    SDA=1;
    for (i=0;i<7;i++)
    {
        SCL=0;
        for(j=0;j<2;j++);
        SCL=1;
        if(SDA==1)
            Sin|=0x01;
        else
            Sin&=0xfe;
        for(j=0;j<2;j++);
```

```
        Sin<<=1;
    }
    SCL=0;
    for(j=0;j<2;j++);
    SCL=1;
    if(SDA==1)
        Sin|=0x01;
    else
        Sin&=0xfe;
    for(j=0;j<2;j++);

    return(Sin);
}

/*发送应答位*/
void SendACK()
{
      unsigned char data j;
      SCL=0;
      for(j=0;j<2;j++);
      SDA=0;
      for(j=0;j<2;j++);
      SCL=1;
      for(j=0;j<4;j++);
      SCL=0;
}

/*发送非应答位*/
void SendNoACK()
{
      unsigned char data j;
      SCL=0;
      for(j=0;j<2;j++);
      SDA=1;
      for(j=0;j<2;j++);
      SCL=1;
      for(j=0;j<4;j++);
      SCL=0;
}

/*START 启动 I2C 总线子程序：发送 I2C 起始条件*/
void Start(void)
{
      unsigned char data i;
```

```
    SDA=1;                      /* 发送起始条件的数据信号*/
    for (i=0;i<2;i++);
    SCL=1;                     /*发送起始条件的时钟信号*/
    for(i=0;i<4;i++) ;         /*起始条件建立时间大于 4.7μs*/
    SDA=0;                      /*发送起始信号*/
    for(i=0;i<4;i++);     /*起始条件锁定时间大于 4μs*/
    SCL=0;                          /*钳住 I2C 总线，准备发送或接收数据*/
}
/*STOP 停止 I2C 总线子程序：发送 I2C 总线停止条件*/
void Stop(void)
{
    unsigned char data i;
    SDA=0;
    for(i=0;i<2;i++) ;
    SCL=1;
    for(i=0;i<4;i++)
    SDA=1;
    for(i=0;i<4;i++);
}

/*一次延时 1ms*/
void Delay(unsigned int x)
{
    unsigned char j;
    while(x--){
        WDRST=0X1E;  /*清看门狗*/
        WDRST=0XE1;
        for(j=0;j<120;j++);
    }
}
```

附:reg764.h 头文件

```
/*   Register Declarations for 87LPC764 Processor */

/*   BYTE Register   */
sfr P0     = 0x80;
sfr P1     = 0x90;
sfr P2     = 0xA0;

sfr PSW    = 0xD0;
sfr ACC    = 0xE0;
/*sfr IEN1 = 0xE8;*/
sfr B      = 0xF0;
```

```
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

/*  BIT Register  */
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;

/*  IP   */
```

```
sbit PS     = 0xBC;
sbit PT1    = 0xBB;
sbit PX1    = 0xBA;
sbit PT0    = 0xB9;
sbit PX0    = 0xB8;

/*管脚*/
sbit T1     = 0x87;
sbit T0     = 0x92;
sbit INT1 = 0x94;
sbit INT0 = 0x93;
sbit TXD    = 0x90;
sbit RXD    = 0x91;

/*   SCON   */
sbit SM0    = 0x9F;
sbit SM1    = 0x9E;
sbit SM2    = 0x9D;
sbit REN    = 0x9C;
sbit TB8    = 0x9B;
sbit RB8    = 0x9A;
sbit TI     = 0x99;
sbit RI     = 0x98;

/*特殊功能寄存器 SFR 和位地址定义*/
sfr AUXR1     = 0xA2;              /*辅助功能寄存器，不可位寻址*/
sfr CMP1      = 0xAC;              /*比较起 1 控制寄存器，不可位寻址*/
sfr CMP2 = 0xAD;                 /*比较起 2 控制寄存器，不可位寻址*/
sfr DIVM = 0x95;                 /*MCU 时钟除数控制寄存器，不可位寻址*/

/*I2C  总线 SFR*/
sfr I2CFG      = 0xC8;           /*I2C 总线配置寄存器，不可位寻址*/
sbit CT0  = 0xC8;                /*I2CFG.0  优化系统控制位*/
sbit CT1  = 0xC9;                /*I2CFG.1  优化系统控制位*/
sbit TIRUN     = 0xCC;           /*I2CFG.4  定时器 I 停止控制位*/
sbit CLRTI     = 0xCD;           /*I2CFG.5  定时器 I 中断标志控制位*/
sbit MASTARTQ=0xCE;              /*I2CFG.6  主控器设置位*/
sbit SLAVEN  = 0xCF;             /*I2CFG.7  被控器设置位*/

sfr I2CON      = 0xD8;           /*I2C 总线控制寄存器，不可位寻址*/
sbit XSTP      = 0xD8;           /*I2CON.0  产生停止条件*/
sbit MASTER = 0xD9;              /*I2CON.1  主控器标志*/
sbit XSTR      = 0xD9;           /*I2CON.1  产生重复起始条件*/
sbit STP   = 0xDA;               /*I2CON.2  停止标志位*/
```

```
sbit CSTP      = 0xDA;          /*I2CON.2 清除 STOP 标志*/
sbit STR  = 0xDB;          /*I2CON.3 起始标志位*/
sbit CSTR      = 0xDB;          /*I2CON.3 清除 START 标志*/
sbit ARL  = 0xDC;          /*I2CON.4 仲裁失败标志*/
sbit CARL      = 0xDC;          /*I2CON.4 清除 ARL 标志*/
sbit DRDY      = 0xDD;          /*I2CON.5 数据准备好标志*/
sbit CDR = 0xDD;           /*I2CON.5 清除 DRDY 标志*/
sbit ATN  = 0xDE;          /*I2CON.6 DRDY、ARL、STOP、START 任一为 1 时，ATN=1*/
sbit IDLE      = 0xDE;          /*I2CON.6 使被控器进入空闲模式*/
sbit RDAT      = 0xDF;          /*I2CON.7 最后接受到的数据*/
sbit CXA = 0xDF;           /*I2CON.7 清除数据传送状态*/

sfr I2DAT      = 0xD9;          /*I2C 数据读寄存器，接收数据位在 I2DAT.7(RDAT)
                                  读取数据位在 I2DAT.7（XDAT）*/
/*中断控制寄存器*/
sfr  IEN0     =    0xA8;        /*中断控制寄存器 0，可位寻址*/
sbit EB0  =    0xAD;        /*IEN0.5 掉电检测中断*/
sbit EWD=    0xAE;        /*IEN0.6 看门狗定时器中断*/

sfr  IEN1     =    0xE8;        /*中断控制寄存器 1，可位寻址*/
sbit EI2  =    0xE8;        /*IEN1.0 I2C 中断*/
sbit EKB =    0xE9;        /*IEN1.1 键盘中断*/
sbit EC2  =    0xEA;        /*IEN1. 比较 EC2 中断*/
sbit EC1      =    0xED;        /*IEN1.5 比较器中断*/

sfr IP0L  =    0xB8;        /*中断优先级 0 寄存器，低字节*/
sfr IP0H  =    0xB7;        /*中断优先级 0 寄存器，高字节*/
sfr IP1L  =    0xF8;        /*中断优先级 1 寄存器，低字节*/
sfr IP1H  =    0xF7;        /*中断优先级 1 寄存器，高字节*/

/*端口输出类型选择寄存器*/
sfr P0M1 =    0x84;         /*0 口输出模式选择 1*/
sfr P0M2 =    0x85;         /*0 口输出模式选择 2*/
sfr P1M1 =    0x91;         /*1 口输出模式选择 1*/
sfr P1M2 =    0x92;         /*1 口输出模式选择 2*/
sfr P2M1 =    0xA4;         /*2 口输出模式选择 1*/
sfr P2M2 =    0xA5;         /*2 口输出模式选择 2*/
sfr PT0AD     =    0xF6;         /*0 口数字输出禁止 */
sfr SADDR     =    0xA9;         /*串口地址寄存器  */
sfr SADEN     =    0xB9;         /*串口地址使能寄存器*/

/*看门狗寄存器*/
sfr WDCON     = 0xA7;      /*看门狗控制寄存器*/
sfr WDRST     = 0xA6;      /*看门狗复位寄存器*/
```