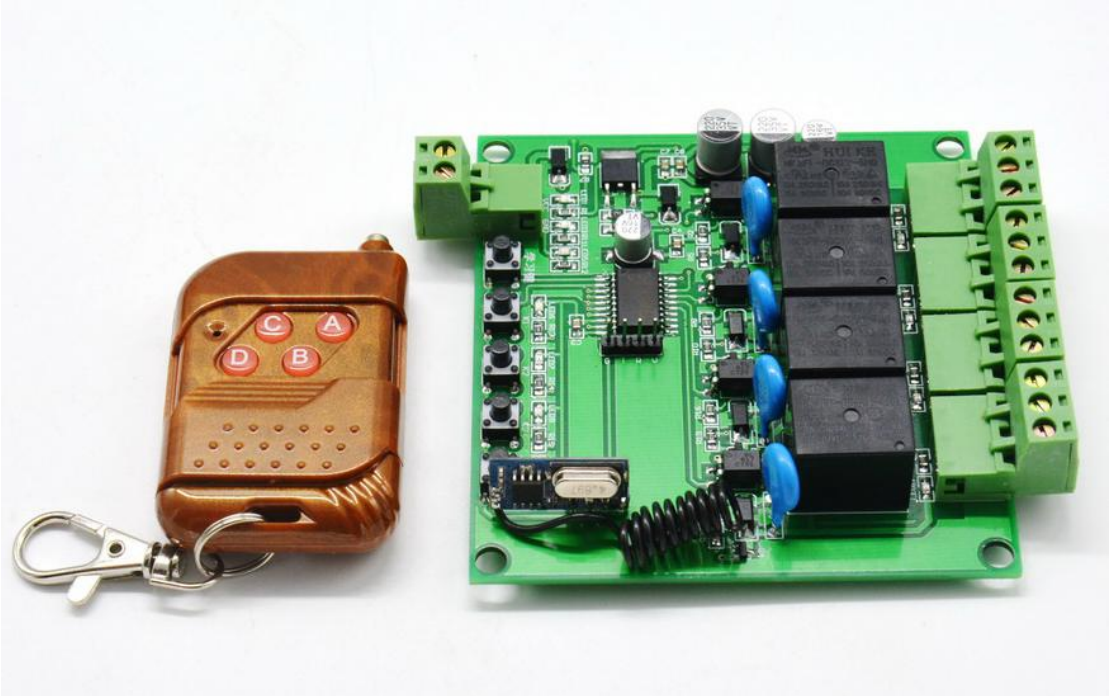
GYJ-0282_四路遥控可编程开发板产品使用手册



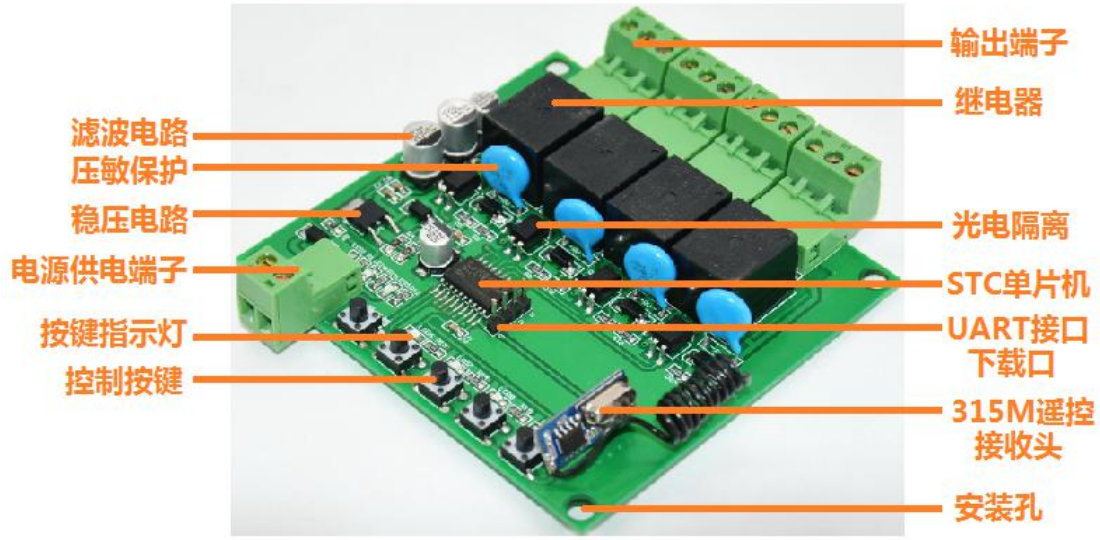
【简要说明】

尺寸：长 93mmX 宽 87mmX 高 18mm
二、主要芯片：STC15W408AS
三、工作电压：直流 6~36 伏
四、支持：UATR 接口下载程序及 USB 下载（需要安卓接口线）
五、特点：
1、具有电源指示功能；
2、所有输入输出均有 LED 灯做信号指示；
3、可以实现 315M 遥控器对继电器任意控制；
4、工作频率 315M
5、抗干扰能力强，穿墙能力强。
6、宽电压供电支持 5~36V 供电最大稳压输出电流 1.5A；
7、继电器控制最大负载 220V10A（300W）
8、采用螺旋端子压接，接线可靠方便扩展；
9、遥控距离，无障碍小于 100 米，有障碍小于 30 米；
10、供电具有防反接保护。电路工作稳定可靠；
11、工作环境：湿度小于 80% ，温度 -20 度至 70 度
六、提供相关软件、原理图 例程及相关资料；

适用场合：单片机学习、电子竞赛、产品开发、毕业设计、项目开发、程序调试。。。

注意啦：本产品提供的所有程序都附带原理图以及说明！

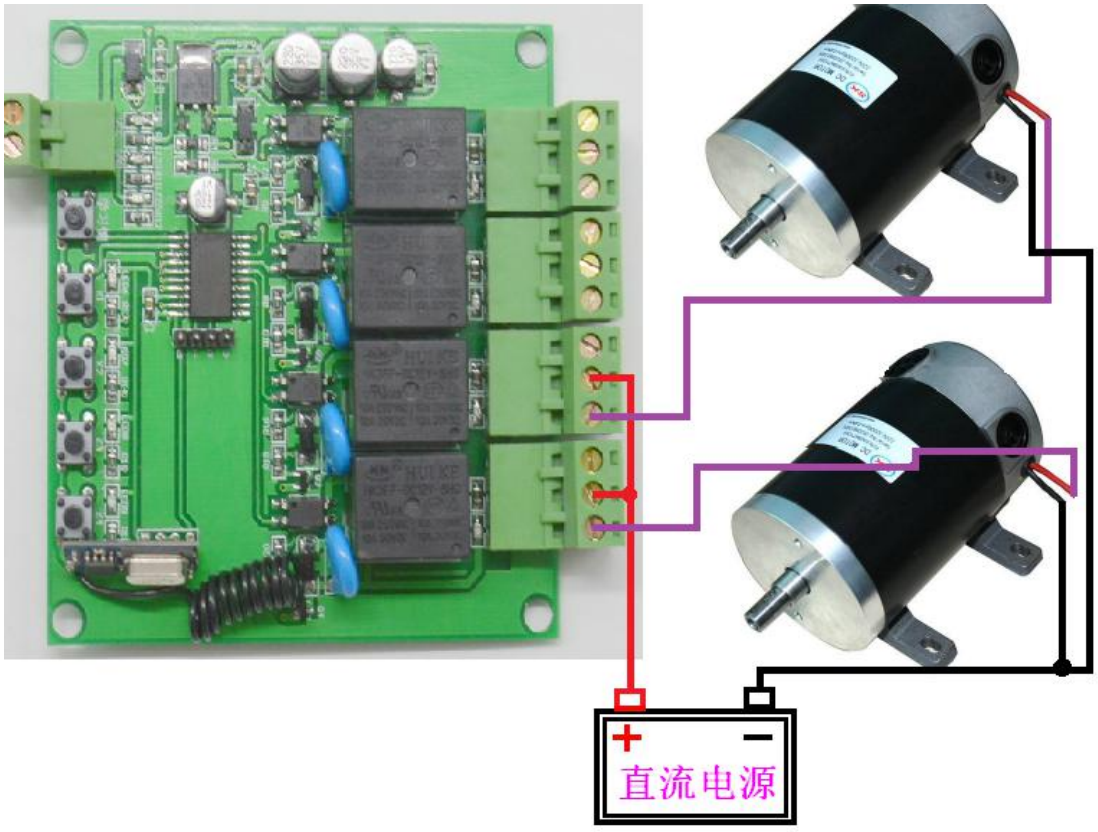
【产品标注】



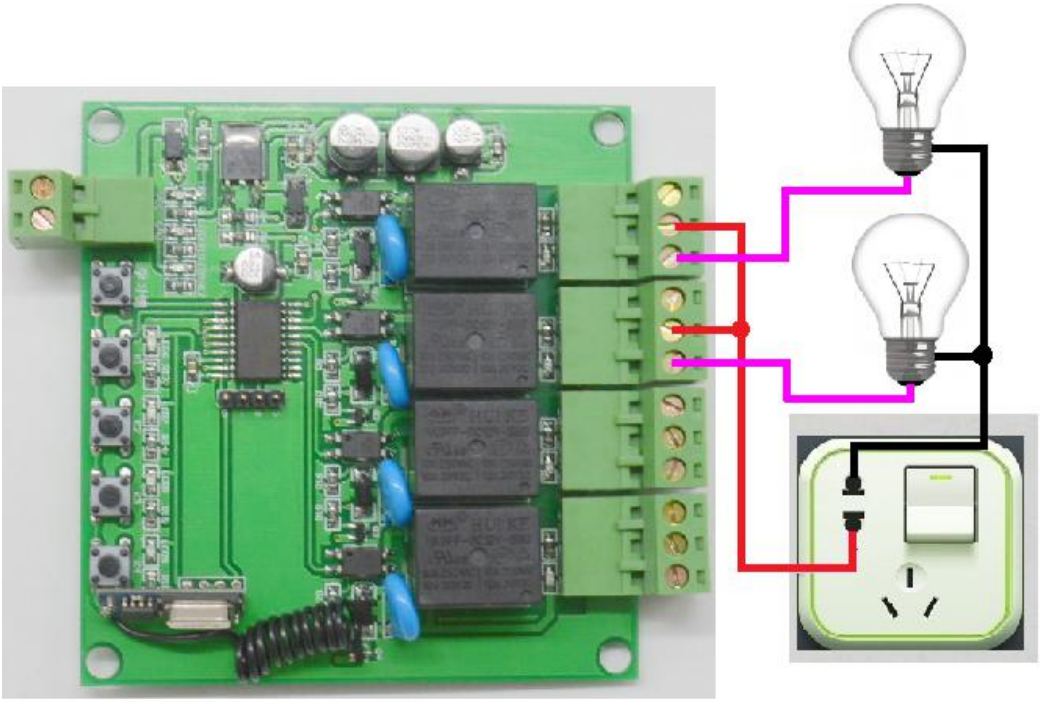
【输入输出接线图】

【应用举例】

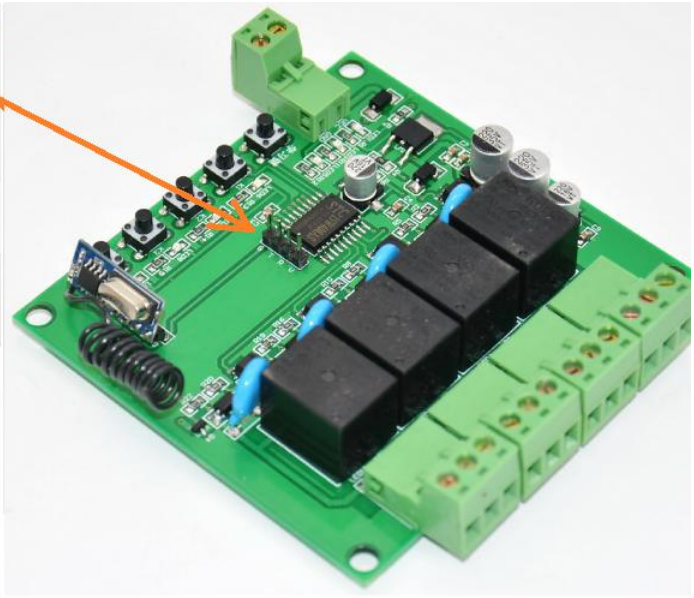
用第三路和第四路控制两台直流电机的接线，第一路和第二路同理。



用第 1 路和第 2 路控制两个 220V 灯泡，第 3 路和第 4 路同理。



【程序下载线接线图】提供编程软件及下载软件



连 接 图

STC下载线	工控板
GND----------	G (GND)
RXD----------	T (P3.1)
TXD----------	R (P3.0)
5V0----------	V (+5V)

【配套程序】

```
/*********************************************************************

                    河南钰平电子科技有限公司

实现功能:应用程序

使用芯片：STC15W408AS

晶振：11.0592MHZ

波特率：9600

编译环境：Keil 4

作者：张新春

微信/手机：13603455408

QQ：115451619

产品有售淘宝    1 店：https://ourhc.taobao.com

产品有售淘宝    2 店：https://g88888.taobao.com

产品有售淘宝企业店：https://shop404420384.taobao.com

【声明】此程序仅用于学习与参考，引用请注明版权和作者信息！

*********************************************************************/

//-----------------兼容 2262 1527 的解码实验程序----------------//

//测试单片机：STC15W408AS

//晶振：内部 11.0592mHZ

//复位方式：内部复位

//串口通讯：波特率 9600/数据位 8/停止位 1/无校验

//调试环境：KEIL3

//程序功能：实现 2262 解码，学习、自适应多阻值,片内 EEPROM，存储 60 个遥控器数据

//              不依赖硬件，不占用硬件资源。移植更加方便
```

```
//            学习遥控器：按一下学习键，学习灯点亮，松开学习键，按动要学习的遥控器按键，学习灯熄灭，学习成功。重复上述操作可学习多个遥控器.

//            清除：按住学习键不放，直到学习灯自动熄灭，擦除成功.

/*************************************************************/


#include <STC15W408AS.h>

#include <intrins.h>

#define uchar unsigned char

#define uint  unsigned int

sbit RF         =   P1^1;     //信号输入

sbit LED        =   P3^3;     //学习指示灯

sbit set        =   P1^2;     //学习键

sbit D0         =   P3^4;     //1 号继电器解码输出

sbit D1         =   P3^5;     //2 号继电器

sbit D2         =   P3^6;     //3 号继电器

sbit D3         =   P3^7;     //4 号继电器

sbit deng1      =   P1^7;     //K1 指示灯

sbit deng2      =   P5^4;     //K2 指示灯

sbit deng3      =   P5^5;     //K3 指示灯

sbit deng4      =   P3^2;     //K4 指示灯

sbit VT         =   P1^0;     //接收指示灯

sbit aj1        =   P1^3;     //K1

sbit aj2        =   P1^4;     //K2

sbit aj3        =   P1^5;     //K3
```

```
sbit aj4        =   P1^6;     //K4

bit  decode_ok;       //解码成功

bit rf_ok;          //收到有效数据

bit study;          //学习标志

bit jmnx;  //编码类型 0 是 2262，1 是 1527

bit
m=0,ba=0,ca=0,da=0,ea=0,za=0,aa=0,hv=0,ff=0,ra=0,rb=0,rc=0,rd=0,g=0,hm=0,biao=0,
kt=0;

bit biao_1=0,g_1=0,kt_1=0,hm_1=0;

bit biao_2=0,g_2=0,kt_2=0,hm_2=0;

bit biao_3=0,g_3=0,kt_3=0,hm_3=0;

uchar da1527[2][3];  //解码过程中临时数组

uchar key_d;  //遥控器按键码

uchar short_k;      //窄脉冲宽度

uchar
ss=0,sn=0,h=0,yz=0,hh=0,yy=0,yk=0,yu=0;hk=0,sj=0,so=0,jk=0,hu=0,t1=0,t2=0,t3=0,
t4=0;

uint dt=0,dr=0,dq=0,rv=0,rv_1=0,rv_2=0,rv_3=0;

uchar trg=0,trg_1=0,trg_2=0,trg_3=0,cont=0,cont_1=0,cont_2=0,cont_3=0;

uchar ReadData=0,ReadData_1=0,ReadData_2=0,ReadData_3=0;

static uint sk=0;

uchar xdata key_number[181];          //遥控器编码数组，存放 60 个遥控器

void delay_1ms(uint x)     //1 毫秒延时

{

    uchar b,c;
```

```
    for(x;x>0;x--)
        {
            for(b=3;b>0;b--)
                {
                    for(c=150;c>0;c--);
                }
        }
}
void delay(uint ms)//
{
  while(ms--)
    {
     ms++;

     ms--;
    }
 }
//=====================================================
/////////片内 EEPROM 读写驱动程序//////////////////////////
//=====================================================
void IAP_Disable()      //关闭 IAP
{
    //关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
    //一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
```

```
    IAP_CONTR = 0;       //关闭 IAP 功能
    IAP_CMD   = 0;       //清命令寄存器,使命令寄存器无命令,此句可不用
    IAP_TRIG = 0;       //清命令触发寄存器,使命令触发寄存器无触发,此句可不用
    IAP_ADDRH = 0;
    IAP_ADDRL = 0;
}//
//读一字节，调用前需打开 IAP 功能，入口:DPTR = 字节地址，返回:A = 读出字节
uchar read_add(uint addr)    //读 EEPROM
{
    IAP_DATA = 0x00;
    IAP_CONTR = 0x84;          //打开 IAP 功能, 设置 Flash 操作等待时间
    IAP_CMD = 0x01;                  //IAP/ISP/EEPROM 字节读命令
    IAP_ADDRH = addr>>8;    //设置目标单元地址的高 8 位地址
    IAP_ADDRL = addr&0xff;    //设置目标单元地址的低 8 位地址
    EA = 0;
    IAP_TRIG = 0x5a;    //先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    IAP_TRIG = 0xa5;    //送完 B9h 后，ISP/IAP 命令立即被触发起动
    _nop_();
    EA = 1;
    IAP_Disable(); //关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
                   //一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都
关
    return (IAP_DATA);
}//-------------------------------------------------------------------
--
```

```
//字节编程，调用前需打开 IAP 功能，入口:DPTR = 字节地址, A= 须编程字节的数据
void write_add(uint addr,uchar ch)   //直接写 EEPROM
{
    IAP_CONTR = 0x84;          //打开 IAP 功能, 设置 Flash 操作等待时间
    IAP_CMD = 0x02;                  //IAP/ISP/EEPROM 字节编程命令
    IAP_ADDRH = addr>>8;     //设置目标单元地址的高 8 位地址
    IAP_ADDRL = addr&0xff;     //设置目标单元地址的低 8 位地址
    IAP_DATA = ch;                   //要编程的数据先送进 IAP_DATA 寄存器
    EA = 0;
    IAP_TRIG = 0x5a;    //先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    IAP_TRIG = 0xa5;    //送完 B9h 后，ISP/IAP 命令立即被触发起动
    _nop_();
    EA = 1;
    IAP_Disable(); //关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
                   //一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都
关
}//-------------------------------------------------------------------------
--
//擦除扇区, 入口:DPTR = 扇区地址
void Sector_Erase(uint addr)      //扇区擦除
{
     IAP_CONTR = 0x84;          //打开 IAP 功能, 设置 Flash 操作等待时间
    IAP_CMD = 0x03;                  //IAP/ISP/EEPROM 扇区擦除命令
    IAP_ADDRH =addr>>8;     //设置目标单元地址的高 8 位地址
    IAP_ADDRL =addr&0xff;     //设置目标单元地址的低 8 位地址
```

```
    EA = 0;
    IAP_TRIG = 0x5a;    //先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    IAP_TRIG = 0xa5;    //送完 B9h 后，ISP/IAP 命令立即被触发起动
    _nop_();
    EA = 1;
}//-----------------------------------------------------------------------------
//============================      接      收      解      码      部      分
=====================================//

void RF_decode()
{
      uchar ii=0,j=0,k=0,rep=0;
      uint head_k=0;            //短脉冲宽度
      uchar s;
//-----------------------------          数          据          接          收
---------------------------------------

      short_k=0;
      while(RF && j<250)      //检测头信号前一个高脉冲的宽度
         {
            delay(1);
            short_k++;
         }
      while(!RF)
         {
            delay(1);
            head_k++;
```

```
        } //检测头脉冲的宽度

      if(((short_k*24)<head_k) && (head_k<(short_k*38)))   //引导码宽度是窄脉冲
的 32 倍 24/38

        {

           for(rep=0;rep<2;rep++)

              {

                for(ii=0;ii<3;ii++)//3 字节

                    {

                      for(k=0;k<8;k++)//每个字节 8 位

                          {

                            j=0;

                            while(RF && j<245)

                                {

                                    delay(1);

                                    j++;

                                }//

if(j>(short_k-short_k/2-short_k/3)&&j<(short_k*1.96))

                                {

                                    da1527[rep][ii]&=~(1<<((7-k)));

                                }

                                else
if(j>(short_k*1.96)&&j<(short_k*5))da1527[rep][ii]|=(1<<(7-k));

                                    else {return;}          //乱码退出
```

```
                    j=0;
                    while(!RF && j<150){delay(2);j++;}       //跳过低电平

                }
            }//for(ii=0;ii<12;ii++)
            j=0;while(RF && (j<200)){delay(1);j++;}              //跳个最后一个高脉冲
            head_k=0;while(!RF) {delay(1);head_k++;} //检测下一个前导信号的宽度
            if((head_k<(short_k*26)) || (head_k>(short_k*38)))  {return;} //引导码宽度是窄脉冲的 32 倍  //乱码退出
        }
    //++++++++++++++++++++++++++2262  与  1527  数  据  分  离  处  理 ++++++++++++++++++++++++++++++++++++++++
     if((da1527[0][0]==da1527[1][0]) && (da1527[0][1]==da1527[1][1]) && (da1527[0][2]==da1527[1][2]))   //两次接收到的数据相同
        {
            uchar u,i,x;
            rf_ok=1;
            for(i=0;i<3;i++)  //判定 2262 与 1527
                {
                    for(u=0;u<4;u++) {if(((da1527[0][i]>>(u*2)) & 3)==2) {i=80;break;}}  //有 10 则为 1527
                    if(i==80) break;
                }
            if(i==80)  //1527
                {
```

```
                key_d=da1527[1][2] & 0x0f;          //分出 1527 的按键值
                da1527[0][2]=da1527[1][2]>>4; //分出 1527 的后 4 位地址
                jmnx=1;          //为 0 是 2262，1 是 1527
              }
            else      //2262
              {
                key_d=0;
                for(i=0;i<4;i++){if(((da1527[0][2]>>(i*2))&3)==3)
key_d|=1<<i;}   //计算出 2262 的按键数据
                da1527[0][2]=0x00; //2262 无后 4 位地址,全为 0
                jmnx=0;          //为 0 是 2262，1 是 1527
                jk++;//自锁用，作用：按下按键不松手继电器状态不变，松开再按下改变，一次只改变一次状态，因为按下按键后遥控会一直发码，所以让 jk 一直自加，但是只取 jk=1 的值的状态
              }

            if (!study)       //非学习状态
              {
                rf_ok=0;
                for(x=0;x<60;x++)
                    {

    if((da1527[0][0]==key_number[x*3+1])&&(da1527[0][1]==key_number[x*3+2])

    &&(da1527[0][2]==key_number[x*3+3]))//判断是否已学习过的编码
                                {
```

```
//                                          D0=!(key_d&0x08);       //取得按键
码

//                                          D1=!(key_d&0x04);

//                                          D2=!(key_d&0x02);

//                                          D3=!(key_d&0x01);

                                            if(m==1)  //互锁

                                            {

                                            if(key_d                        ==
0x01){D0=0;D1=1;D2=1;D3=1;}//D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}//else{
D0=1;}

                                            if(key_d                        ==
0x02){D0=1;D1=0;D2=1;D3=1;}//D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x03){D0=1;D1=1;D2=0;D3=1;D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}

                                            if(key_d                        ==
0x04){D0=1;D1=1;D2=0;D3=1;}//D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x05){D0=1;D1=1;D2=1;D3=1;D4=0;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x06){D0=1;D1=1;D2=1;D3=1;D4=1;D5=0;D6=1;D7=1;D8=1;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x07){D0=1;D1=1;D2=1;D3=1;D4=1;D5=1;D6=0;D7=1;D8=1;D9=1;D10=1;D11=1;}

                                            if(key_d                        ==
0x08){D0=1;D1=1;D2=1;D3=0;}//D4=1;D5=1;D6=1;D7=0;D8=1;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x09){D0=1;D1=1;D2=1;D3=1;D4=1;D5=1;D6=1;D7=1;D8=0;D9=1;D10=1;D11=1;}

//                                          if(key_d                        ==
0x0a){D0=1;D1=1;D2=1;D3=1;D4=1;D5=1;D6=1;D7=1;D8=1;D9=0;D10=1;D11=1;}
```

```
//                                          if(key_d                    ==
0x0b){D0=1;D1=1;D2=1;D3=1;D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=0;D11=1;}

//                                          if(key_d                    ==
0x0c){D0=1;D1=1;D2=1;D3=1;D4=1;D5=1;D6=1;D7=1;D8=1;D9=1;D10=1;D11=0;}

                                            }

                                            else

                                            {

                                            if(ba==0) //B 按键自锁

                                            {

                                             if(key_d == 0x04)

                                               {

                                                   if(jk==1)

                                                    {

                                                      D2=~D2;

                                                     }

                                                }

                                             } //B 自锁


                                             if(da==0) //A 按键自锁

                                            {

                                             if(key_d == 0x08)

                                               {

                                                  if(jk==1)

                                                    {

                                                      D3=~D3;
```

```
        }
    }
} //A 自锁
if(ca==0) //C 按键自锁
{
if(key_d == 0x02)
  {
     if(jk==1)
       {
         D1=~D1;
        }
    }
} //C 自锁
if(ea==0) //D 按键自锁
{
if(key_d == 0x01)
  {
     if(jk==1)
       {
         D0=~D0;
        }
    }
} //D 自锁
```

```
                if(ba==1)//B 按键点动,ba 为自锁转点
动标志位
                {
                 if(key_d == 0x04){D2=0;}
                 }
                if(da==1)//A 按键点动,da 为自锁转点
动标志位
                {
                 if(key_d == 0x08){D3=0;}
                 }
                 if(ca==1)//C 按键点动， da 为自锁转
点动标志位
                {
                 if(key_d == 0x02){D1=0;}
                 }
                 if(ea==1)//D 按键点动， da 为自锁转
点动标志位
                {
                 if(key_d == 0x01){D0=0;}
                 }
               }

               decode_ok=1;
               ss=1;
               sn=1;
               sj=1;
```

```
                                        so=1;
                                        VT=0;

                                        s=30;
                                        break;
                                    }
                                }
                                    }
                            }
                        }
                if(decode_ok)    //解码有效信号，类似 2272 PT 脚
                    {
                        s--;
                        if(!s)
                            {
                                jk=0; //按一下按键只做一次动作信号没有后清零
                            decode_ok=0;
                            if(h==1){ss=2;} //点动标志位
                            if(hh==1){sn=2;}
                            if(hk==1){sj=2;}//点动标志位
                            if(hu==1){so=2;}
                            VT=1;
                            }
}
```

```
void key_buffer()          //把遥控器码从 EEPROM 复制到 DATA
{
    uchar n;

    for(n=0;n<181;n++)
        {
           key_number[n]=read_add(0x0000+n);
        }
}

void KEY_study()          //遥控器学习
{
    uchar num_rf;
    uint d_num;
    if(study)
        {
            rf_ok=0;
            d_num=0;

            while(!rf_ok)
                {
                    RF_decode();
                    d_num++;
```

```
            if(d_num>50000) break;
        }
    d_num=0;
    if(rf_ok==1)
        {

            num_rf=key_number[0];                    //取已学习的遥控器
数量
            if(num_rf>60){num_rf=0;}          //如果遥控器数量超过 10 个，
覆盖最先学习的
            key_number[num_rf*3+1]=da1527[0][0];
            key_number[num_rf*3+2]=da1527[0][1];
            key_number[num_rf*3+3]=da1527[0][2];
            key_number[0]=num_rf+1;

            Sector_Erase(0x0000);
            for(num_rf=0;num_rf<181;num_rf++)
                {
                    write_add(0x0000+num_rf,key_number[num_rf]);

                }
            rf_ok=0;
            LED=1;          //学习成功
          }
        else
```

```
                    {
                        rf_ok=0;

                        for(num_rf=0;num_rf<8;num_rf++)          //操作超时
                            {
                                LED=!LED;
                                delay_1ms(100);
                            }
                         LED=1;

                    }
                d_num=0;
                study=0;
            }
}

void system_res()  //系统清零
{
    Sector_Erase(0x0000);
    write_add(0x0000,0x00);
    key_buffer();
}
void set_scan()   //判断学习键状态
{
```

```
uchar h=0;


while(!set)
  {
      if(h>5)
          {
              study=1;
              h=0;
              LED=0;
              while(!set)
                  {
                    delay_1ms(100);
                    h++;
                    if(h>80)
                      {
                          study=0;
                          h=0;
                          system_res();
                          LED=1;
                          while(!set);
                      }

                  }
          }
```

```
            delay_1ms(100);
            h++;
        }
    if(study)
          {
              KEY_study();
          }
 }


/*****************************************************************************
*                            按键读取
*****************************************************************************/
void KeyRead()//读取按键 IO 口函数
{
    ReadData = aj1^0xff;  // 读取按键状态取反后赋值给 ReadData
    trg = ReadData & (ReadData ^ cont);  //trg 短按，每按下按键 trg=1；抬手后为 trg=0，长按为 trg=0
    cont = ReadData;   //cont 长按，长按 cont=1，抬手后 cont=0
  /*****************************************************************************/
    ReadData_1 = aj2^0xff;  // 读取按键状态取反后赋值给 ReadData
    trg_1 = ReadData_1 & (ReadData_1 ^ cont_1);  //trg 短按，每按下按键 trg=1；抬手后为 trg=0，长按为 trg=0
    cont_1 = ReadData_1;   //cont 长按，长按 cont=1，抬手后 cont=0
  /*****************************************************************************/
```

```
    ReadData_2 = aj3^0xff;  // 读取按键状态取反后赋值给 ReadData

    trg_2 = ReadData_2 & (ReadData_2 ^ cont_2);  //trg 短按，每按下按键 trg=1；抬手后为 trg=0，长按为 trg=0

    cont_2 = ReadData_2;   //cont 长按，长按 cont=1，抬手后 cont=0

  /*****************************************************************************/

    ReadData_3 = aj4^0xff;  // 读取按键状态取反后赋值给 ReadData

    trg_3 = ReadData_3 & (ReadData_3 ^ cont_3);  //trg 短按，每按下按键 trg=1；抬手后为 trg=0，长按为 trg=0

    cont_3 = ReadData_3;   //cont 长按，长按 cont=1，抬手后 cont=0

}

/*****************************************************************************
*                          A 按键
*****************************************************************************/

void key_1()

{

  if(trg & 0x01) //短按

  {  rv=0;   //长按延时标志位防止短按到 200 次后长按启动

     biao=0; //这是长按屏蔽短按标志位

     g=0;  //这是长按只执行一次标志位

     kt=1; //这是短按标志位，kt=1 说明短按了

  }

  if((aj1!=0)&&(kt==1)&&(biao==0))//判断

    {

     D3=~D3;kt=0;

    }      // 短按
```

```
  if((cont & 0x01)&&(g==0))//长按
  {
   rv++;   //长按延时
   if(rv>200)
   {
    biao=1; //屏蔽短按
                hm=1;
                yz+=1;//选位标志位
                if(yz==1){h=1;da=1;za=0;hm=0;}
if(yz==2){h=2;m=1;yy=0;yk=0;yu=0;hh=0;hk=0;hu=0;ba=0;ca=0;ea=0;da=0;za=0;}
                if(yz==3){
                  yz=0;      //选位标志重个位开始重新选
                  h=0;
                  m=0;za=0;
                  }
    g=1;  //只让输出一次
    rv=0; //延时时间归零
   }
  }//长按
}

/*************************************************************************
*                          B 选位按键
```

```
**********************************************************************/
void key_2()
{
  if(trg_1 & 0x01) //短按
  {  rv_1=0;      //长按延时标志位防止短按到 200 次后长按启动
     biao_1=0; //这是长按屏蔽短按标志位
     g_1=0;  //这是长按只执行一次标志位
     kt_1=1; //这是短按标志位，kt=1 说明短按了
  }
  if((aj2!=0)&&(kt_1==1)&&(biao_1==0))//判断
    {
     D2=~D2;kt_1=0;
    }       // 短按
  if((cont_1 & 0x01)&&(g_1==0))//长按
  {
   rv_1++;    //长按延时
   if(rv_1>200)
   {
    biao_1=1;    //屏蔽短按
                hm_1=1;
                yy+=1;//选位标志位
                if(yy==1){hh=1;ba=1;aa=0;hm_1=0;}

if(yy==2){hh=2;m=1;hk=0;h=0;hu=0;yz=0;yk=0;yu=0;ba=0;ca=0;ea=0;da=0;aa=0;}
```

```
                if(yy==3){
                  yy=0;      //选位标志重个位开始重新选
                  hh=0;
                  m=0;aa=0;
                }
    g_1=1;  //只让输出一次
    rv_1=0; //延时时间归零
   }
  }//长按
}
/**************************************************************************
*                          C 选位按键
**************************************************************************/
void key_3()
{
  if(trg_2 & 0x01) //短按
  {  rv_2=0;      //长按延时标志位防止短按到 200 次后长按启动
     biao_2=0; //这是长按屏蔽短按标志位
     g_2=0;  //这是长按只执行一次标志位
     kt_2=1; //这是短按标志位，kt=1 说明短按了
  }
  if((aj3!=0)&&(kt_2==1)&&(biao_2==0))//判断
    {
     D1=~D1;kt_2=0;
```

```
    }       // 短按
  if((cont_2 & 0x01)&&(g_2==0))//长按
  {
   rv_2++;   //长按延时
   if(rv_2>200)
   {
    biao_2=1;   //屏蔽短按
                hm_2=1;
                yk+=1;//选位标志位
                if(yk==1){hk=1;ca=1;ff=0;hm_2=0;}

if(yk==2){hk=2;m=1;h=0;hh=0;hu=0;yz=0;yy=0;yu=0;ba=0;ca=0;ea=0;da=0;ff=0;}
                if(yk==3){
                  yk=0;      //选位标志重个位开始重新选
                  hk=0;
                  m=0;ff=0;
                }
    g_2=1;  //只让输出一次
    rv_2=0; //延时时间归零
   }
  }//长按
}

/*********************************************************************
```

```
*                          D 选位按键

**********************************************************************/

void key_4()

{

  if(trg_3 & 0x01) //短按

  {  rv_3=0;      //长按延时标志位防止短按到 200 次后长按启动

     biao_3=0; //这是长按屏蔽短按标志位

     g_3=0;  //这是长按只执行一次标志位

     kt_3=1; //这是短按标志位，kt=1 说明短按了

  }

  if((aj4!=0)&&(kt_3==1)&&(biao_3==0))//判断

    {

     D0=~D0;kt_3=0;

    }       // 短按

  if((cont_3 & 0x01)&&(g_3==0))//长按

  {

   rv_3++;    //长按延时

   if(rv_3>200)

   {

    biao_3=1;    //屏蔽短按

                hm_3=1;

                yu+=1;//选位标志位

                if(yu==1){hu=1;ea=1;hv=0;hm_3=0;}
```

```
if(yu==2){hu=2;m=1;h=0;hh=0;hk=0;yz=0;yy=0;yk=0;ba=0;ca=0;ea=0;da=0;hv=0;}
                if(yu==3){
                  yu=0;      //选位标志重个位开始重新选
                  hu=0;
                  m=0;hv=0;
                }
    g_3=1;  //只让输出一次
    rv_3=0; //延时时间归零
   }
  }//长按
}
/**************************************************************************
*                          定时器配置
**************************************************************************/
void ConfigTimer0(){
    TMOD=0x01;//将定时器 0,1 都设置为模式 1
    TH0=0XFC;//1ms
    TL0=0X66;
    TR0=1;//开启定时器 0
    ET0=1;//开定时器 0 的中断
    EA=1;//开总中断
}
/**************************************************************************
```

```
*                          t0 定时器

*************************************************************************/

void timer0() interrupt 1

{

    TH0=0XFC;//1ms

    TL0=0X66;

    sk++;

    if((t1==1)||(t2==1)||(t3==1)||(t4==1)) //闪烁一下

    {

     dq++;

     if(dq==1000)

     {

     if(ra==1){deng1=0;}

     if(rb==1){deng2=0;}

     if(rc==1){deng3=0;}

     if(rd==1){deng4=0;}

     }

    if(dq==2500)

     {

     if(ra==1){deng1=1;t1=0;ra=0;}

     if(rb==1){deng2=1;t2=0;rb=0;}

     if(rc==1){deng3=1;t3=0;rc=0;}

     if(rd==1){deng4=1;t4=0;rd=0;}

       dq=0;
```

```
 }
}
if((t1==2)||(t2==2)||(t3==2)||(t4==2)) //闪烁两下
{
 dt++;
 if(dt==1000)
 {
 if(ra==1){deng1=0;}
 if(rb==1){deng2=0;}
 if(rc==1){deng3=0;}
 if(rd==1){deng4=0;}
 }
 if(dt==2500)
 {
 if(ra==1){deng1=1;}
 if(rb==1){deng2=1;}
 if(rc==1){deng3=1;}
 if(rd==1){deng4=1;}
 }
 if(dt==4000)
 {
 if(ra==1){deng1=0;}
 if(rb==1){deng2=0;}
 if(rc==1){deng3=0;}
```

```
 if(rd==1){deng4=0;}
 }
 if(dt==5500)
 {
 if(ra==1){deng1=1;t1=0;ra=0;}
 if(rb==1){deng2=1;t2=0;rb=0;}
 if(rc==1){deng3=1;t3=0;rc=0;}
 if(rd==1){deng4=1;t4=0;rd=0;}

   dt=0;

 }
}

if((t1==3)||(t2==3)||(t3==3)||(t4==3)) //闪烁三下
{
     dr++;
 if(dr==1000)
 {
 if(ra==1){deng1=0;}
 if(rb==1){deng2=0;}
 if(rc==1){deng3=0;}
 if(rd==1){deng4=0;}
 }
```

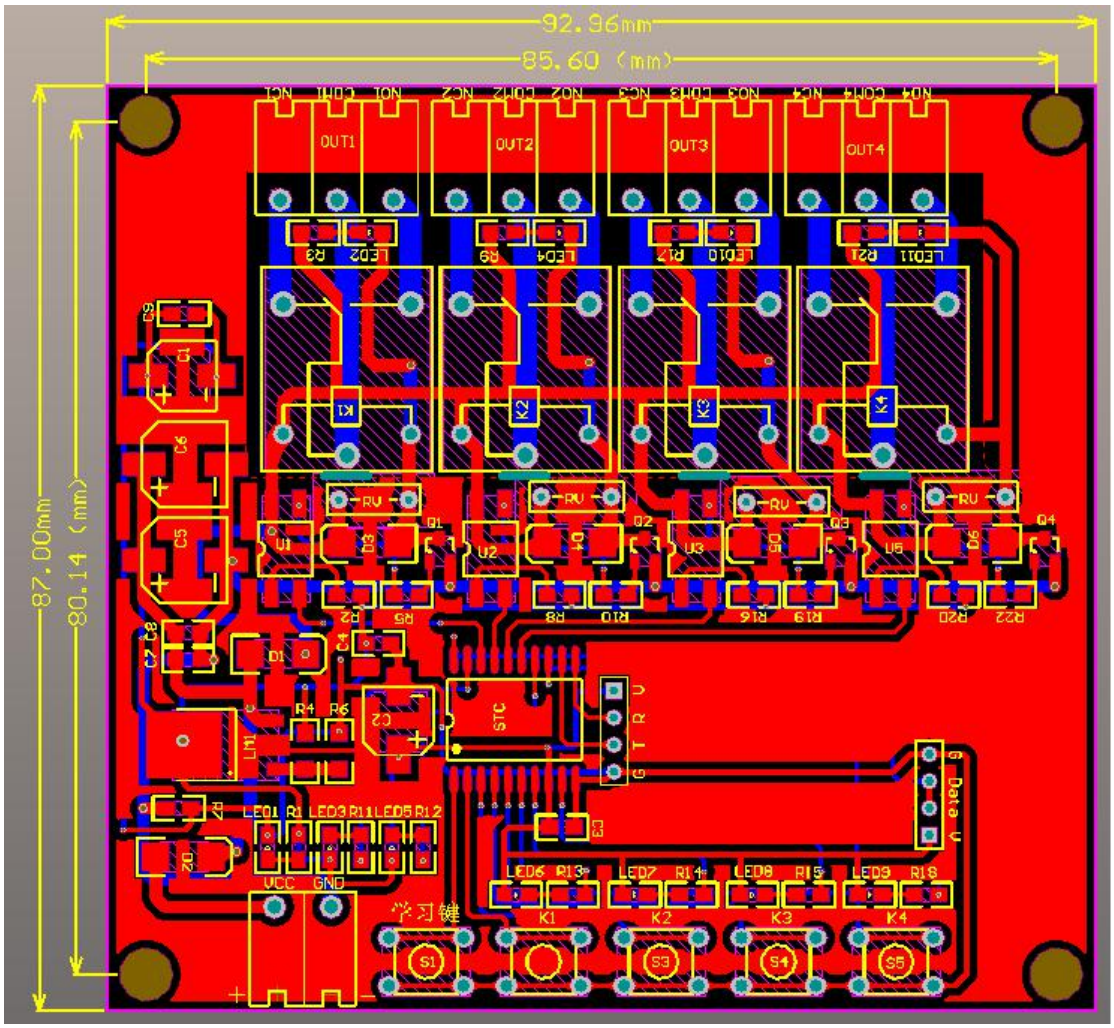
```
if(dr==2500)
{
if(ra==1){deng1=1;}
if(rb==1){deng2=1;}
if(rc==1){deng3=1;}
if(rd==1){deng4=1;}
}
if(dr==4000)
{
if(ra==1){deng1=0;}
if(rb==1){deng2=0;}
if(rc==1){deng3=0;}
if(rd==1){deng4=0;}
}
if(dr==5500)
{
if(ra==1){deng1=1;}
if(rb==1){deng2=1;}
if(rc==1){deng3=1;}
if(rd==1){deng4=1;}
}
if(dr==7000)
{
if(ra==1){deng1=0;}
```

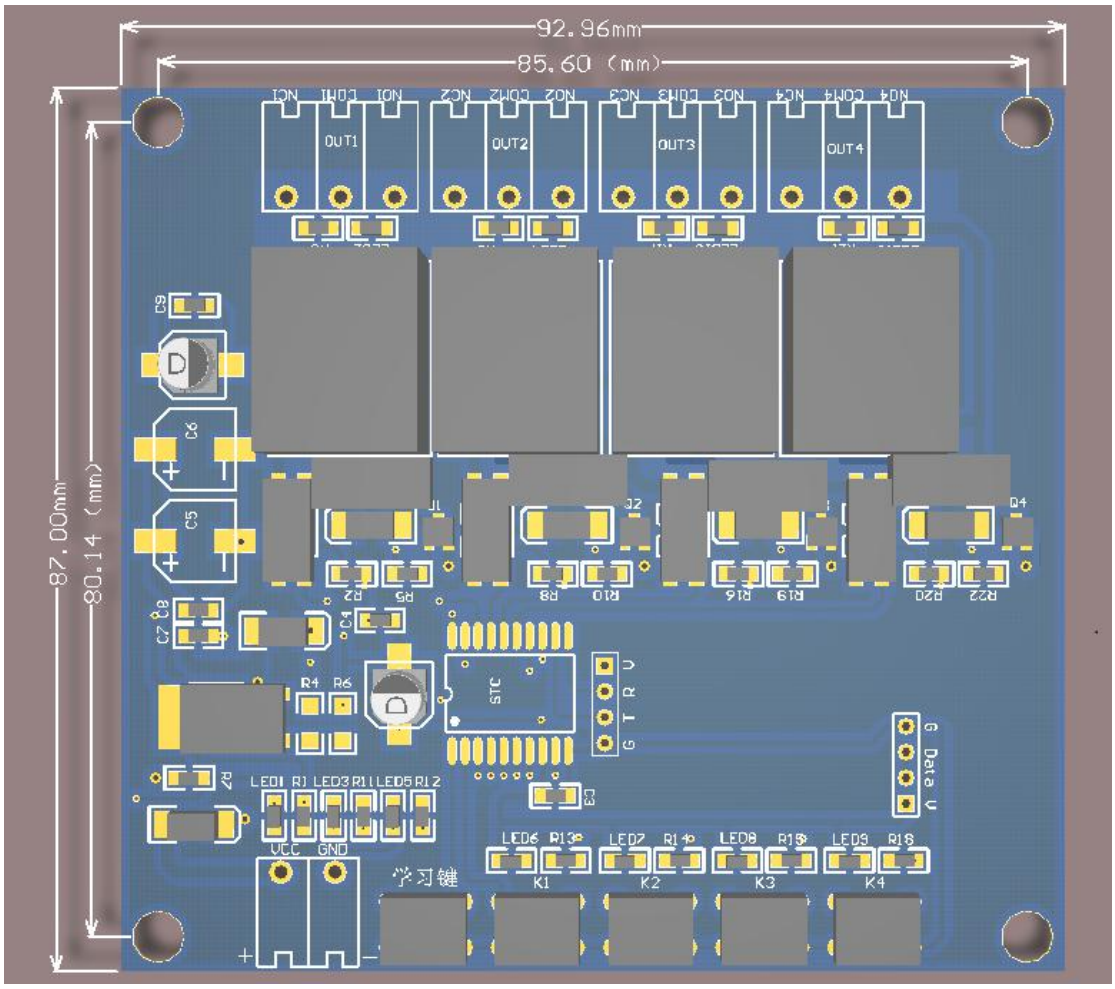
```
        if(rb==1){deng2=0;}
        if(rc==1){deng3=0;}
        if(rd==1){deng4=0;}
        }
        if(dr==8500)
        {
        if(ra==1){deng1=1;t1=0;ra=0;}
        if(rb==1){deng2=1;t2=0;rb=0;}
        if(rc==1){deng3=1;t3=0;rc=0;}
        if(rd==1){deng4=1;t4=0;rd=0;}
          dr=0;
        }
      }
}

void system_start()   //上电初始化
{
    AUXR=0xb5;
    P0=0xfe;
    P1=0xff;
    P3=0xff;
    key_buffer();
}
```

```
void main()
{
  system_start();  //上电初始化
  ConfigTimer0();  //t0 中断配置
  deng1=1;
  deng2=1;
  deng3=1;
  deng4=1;
  while(1)
    {
        KeyRead();  //按键读取函数
        key_1();
        key_2();
        key_3();
        key_4();
        RF_decode();  //读取信号波形分析
        set_scan();   //学习按键
          if((h==1)&&(ss==2)){D3=1;} //A 按键点动,h 为 k1 按键按下后 h=1，ss 为接收指示灯熄灭后 ss=2；控制对应输出断开，形成点动
          if((hh==1)&&(sn==2)){D2=1;}//B 按键点动，hh 为 k2 按键按下后 hh=1，sn 为接收指示灯熄灭后 sn=2；控制对应输出断开，形成点动
          if((hk==1)&&(sj==2)){D1=1;} //C 按键点动,hk 为 k3 按键按下后 hk=1，sj 为接收指示灯熄灭后 sj=2；控制对应输出断开，形成点动
          if((hu==1)&&(so==2)){D0=1;} //D 按键点动,hu 为 k4 按键按下后 hu=1，so 为接收指示灯熄灭后 so=2；控制对应输出断开，形成点动
```

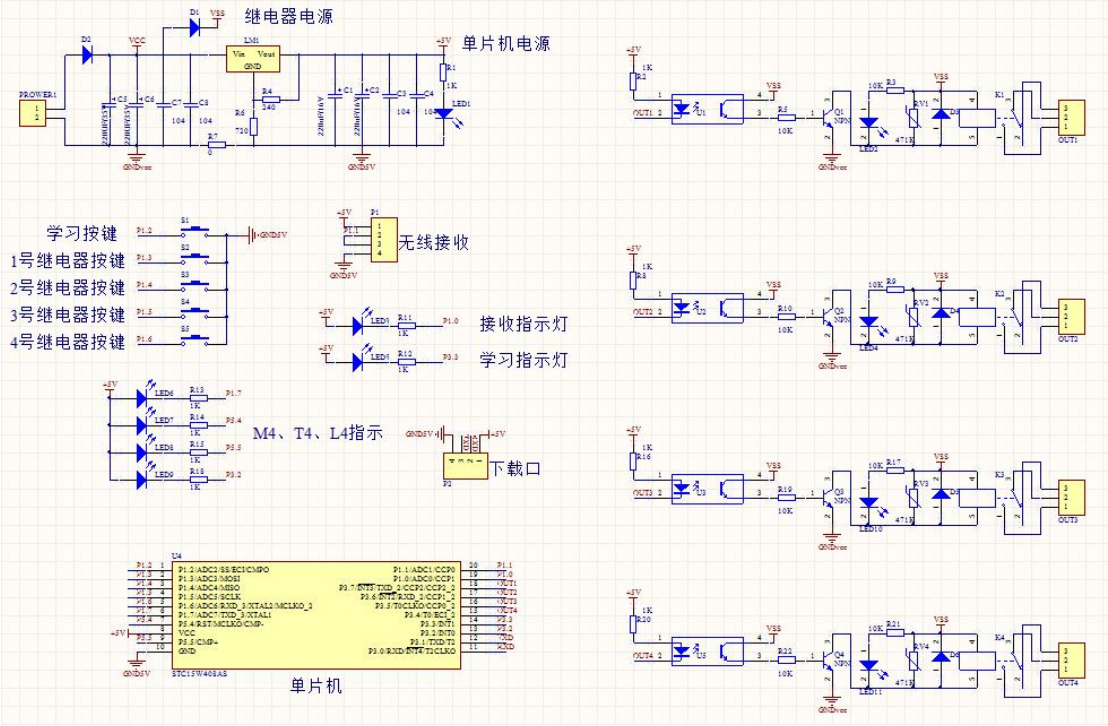
```
            if((yz==0)&&(za==0)&&(hm==1)){t1=1;za=1;ra=1;}//A 按键:主要控制对应灯闪烁，za 为只执行一次闪烁标志位，t1 为闪烁标志位 t1=1,闪一下，t1=2 闪两下,t1=3 闪三下

            if((yz==1)&&(za==0)){t1=2;za=1;ra=1;}      //ra 为对应灯输出标志位，ra=1 则输出，ra=0 则停止

            if((yz==2)&&(za==0)){t1=3;za=1;ra=1;}

            if((yy==0)&&(aa==0)&&(hm_1==1)){t2=1;aa=1;rb=1;}//B 按键:主要控制对应灯闪烁，aa 为只执行一次闪烁标志位，t2 为闪烁标志位 t2=1,闪一下，t2=2 闪两下,t2=3 闪三下

            if((yy==1)&&(aa==0)){t2=2;aa=1;rb=1;}     //rb 为对应灯输出标志位，rb=1 则输出，rb=0 则停止

            if((yy==2)&&(aa==0)){t2=3;aa=1;rb=1;}

            if((yk==0)&&(ff==0)&&(hm_2==1)){t3=1;ff=1;rc=1;}//C 按键:主要控制对应灯闪烁，ff 为只执行一次闪烁标志位，t3 为闪烁标志位 t3=1,闪一下，t3=2 闪两下,t3=3 闪三下

            if((yk==1)&&(ff==0)){t3=2;ff=1;rc=1;}    //rc 为对应灯输出标志位，rc=1 则输出，rc=0 则停止

            if((yk==2)&&(ff==0)){t3=3;ff=1;rc=1;}

            if((yu==0)&&(hv==0)&&(hm_3==1)){t4=1;hv=1;rd=1;}//D 按键:主要控制对应灯闪烁，hv 为只执行一次闪烁标志位，t4 为闪烁标志位 t4=1,闪一下，t4=2 闪两下,t4=3 闪三下

            if((yu==1)&&(hv==0)){t4=2;hv=1;rd=1;}    //rd 为对应灯输出标志位，rd=1 则输出，rd=0 则停止

            if((yu==2)&&(hv==0)){t4=3;hv=1;rd=1;}

    }


}
```

【尺寸图】提供 PDF 版本详细图

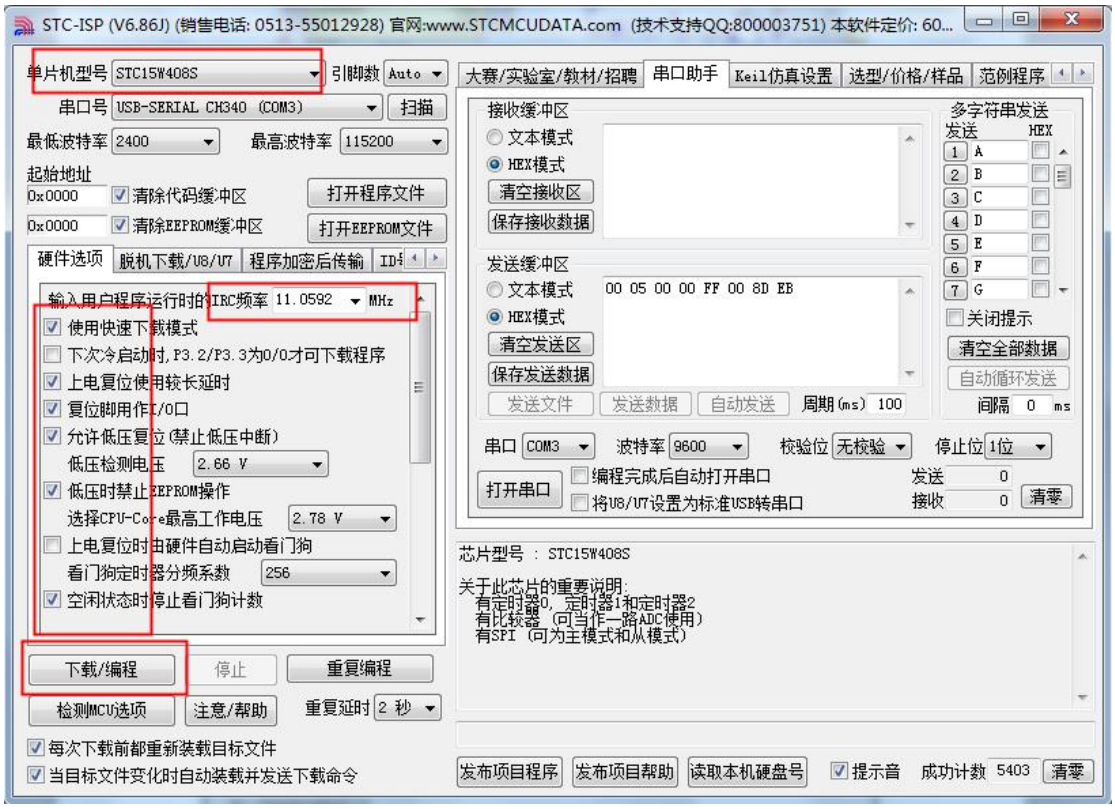




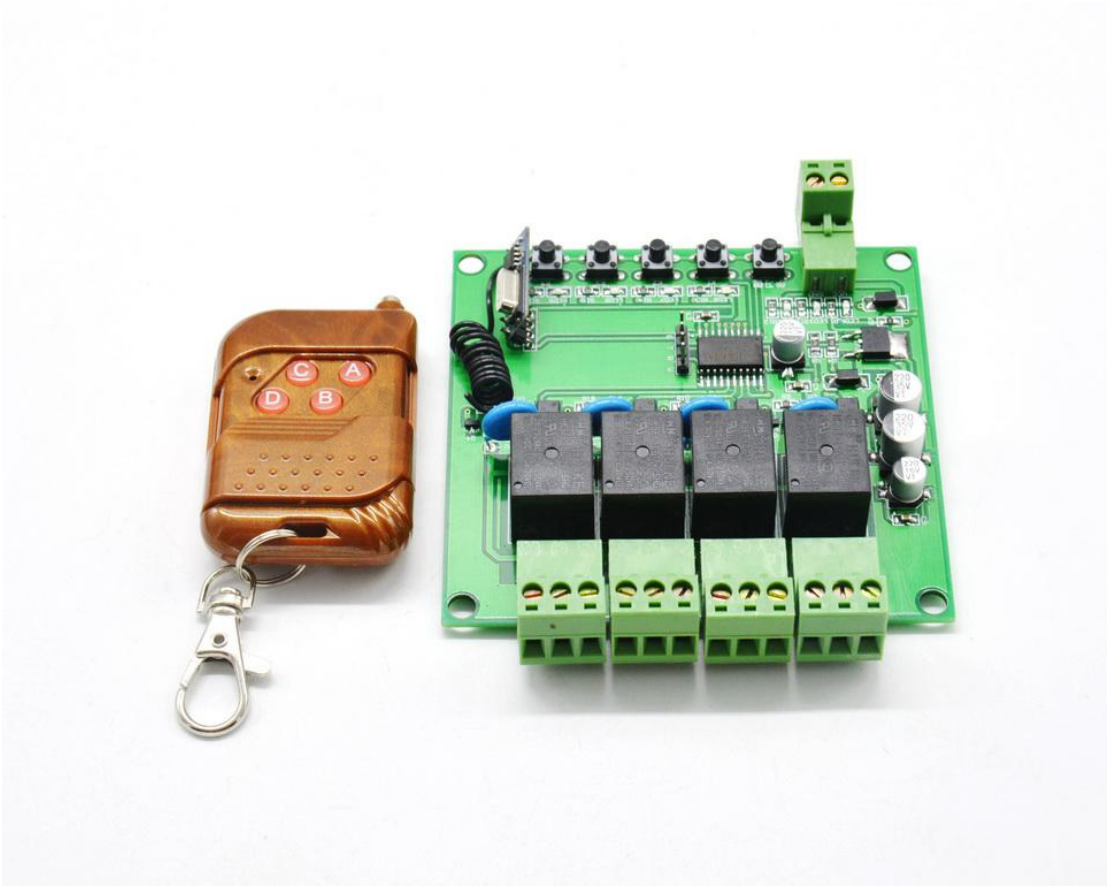
【原理图】提供 PDF 版本详细图

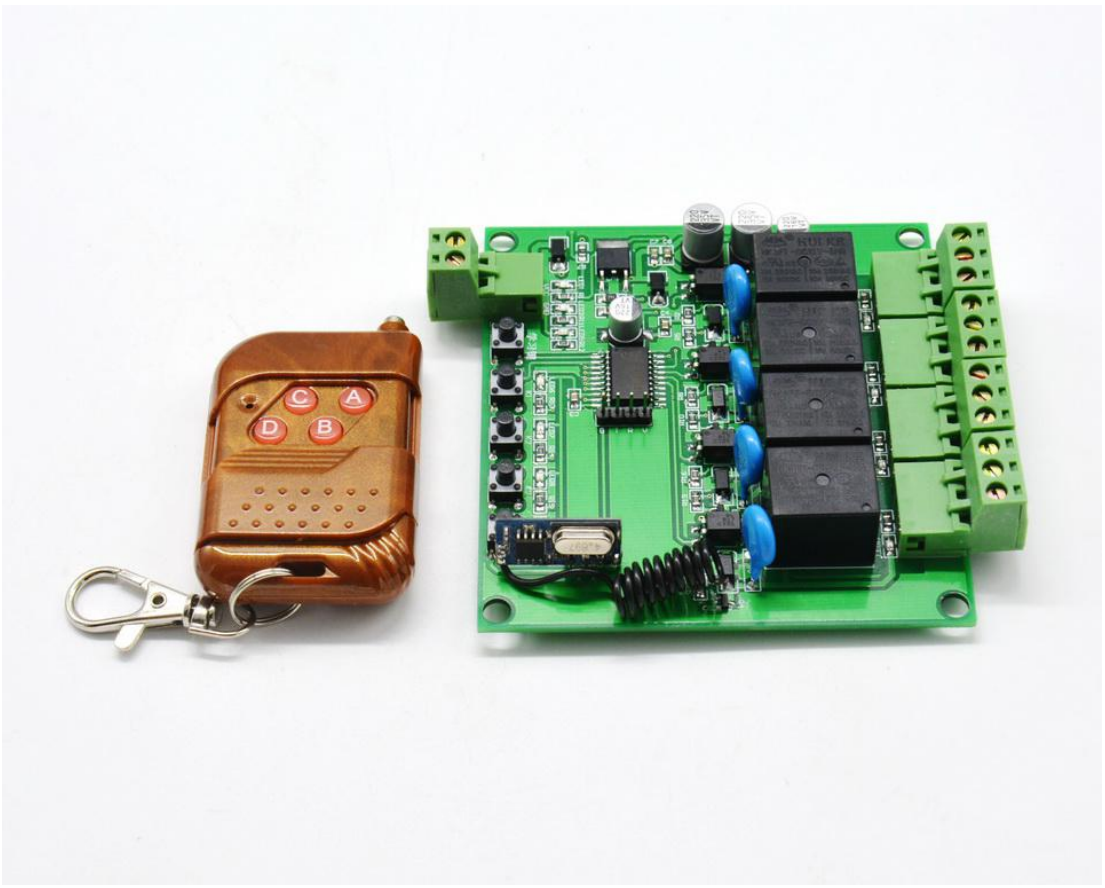

继电器电源
单片机电源
学习按键
1号继电器按键
2号继电器按键
3号继电器按键
4号继电器按键
无线接收
接收指示灯
学习指示灯
M4、T4、L4指示
下载口
单片机

【产品下载界面】

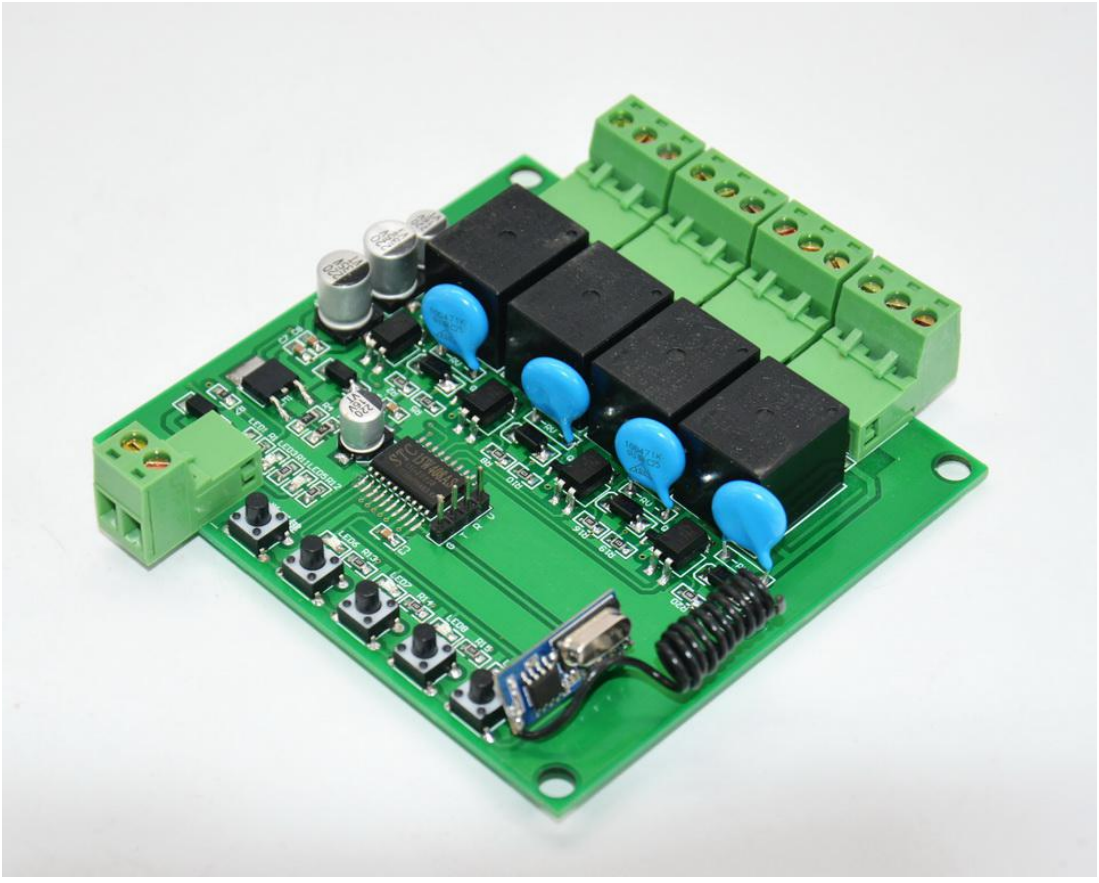

STC-ISP (V6.86J) (销售电话: 0513-55012928) 官网:www.STCMCUDATA.com (技术支持QQ:800003751) 本软件定价: 60...
单片机型号 STC15W408S
引脚数 Auto
串口号 USB-SERIAL CH340 (COM3)
扫描
最低波特率 2400
最高波特率 115200
起始地址
0x0000
清除代码缓冲区
打开程序文件
0x0000
清除EEPROM缓冲区
打开EEPROM文件
硬件选项
脱机下载/U8/U7
程序加密后传输
输入用户程序运行时的IRC频率 11.0592 MHz
使用快速下载模式
下次冷启动时, P3.2/P3.3为0/0才可下载程序
上电复位使用较长延时
复位脚用作I/O口
允许低压复位 (禁止低压中断)
低压检测电压 2.66 V
低压时禁止EEPROM操作
选择CPU-Core最高工作电压 2.78 V
上电复位时由硬件自动启动看门狗
看门狗定时器分频系数 256
空闲状态时停止看门狗计数
下载/编程
停止
重复编程
检测MCU选项
注意/帮助
重复延时 2 秒
每次下载前都重新装载目标文件
当目标文件变化时自动装载并发送下载命令
大赛/实验室/教材/招聘
串口助手
Keil仿真设置
选型/价格/样品
范例程序
接收缓冲区
文本模式
HEX模式
清空接收区
保存接收数据
发送缓冲区
00 05 00 00 FF 00 8D EB
清空发送区
保存发送数据
发送文件
发送数据
自动发送
周期(ms) 100
多字符串发送
发送
HEX
关闭提示
清空全部数据
自动循环发送
间隔 0 ms
串口 COM3
波特率 9600
校验位 无校验
停止位 1位
打开串口
编程完成后自动打开串口
将U8/U7设置为标准USB转串口
发送 0
接收 0
清零
芯片型号 : STC15W408S
关于此芯片的重要说明:
有定时器0, 定时器1和定时器2
有比较器 (可当作一路ADC使用)
有SPI (可为主模式和从模式)
发布项目程序
发布项目帮助
读取本机硬盘号
提示音
成功计数 5403
清零

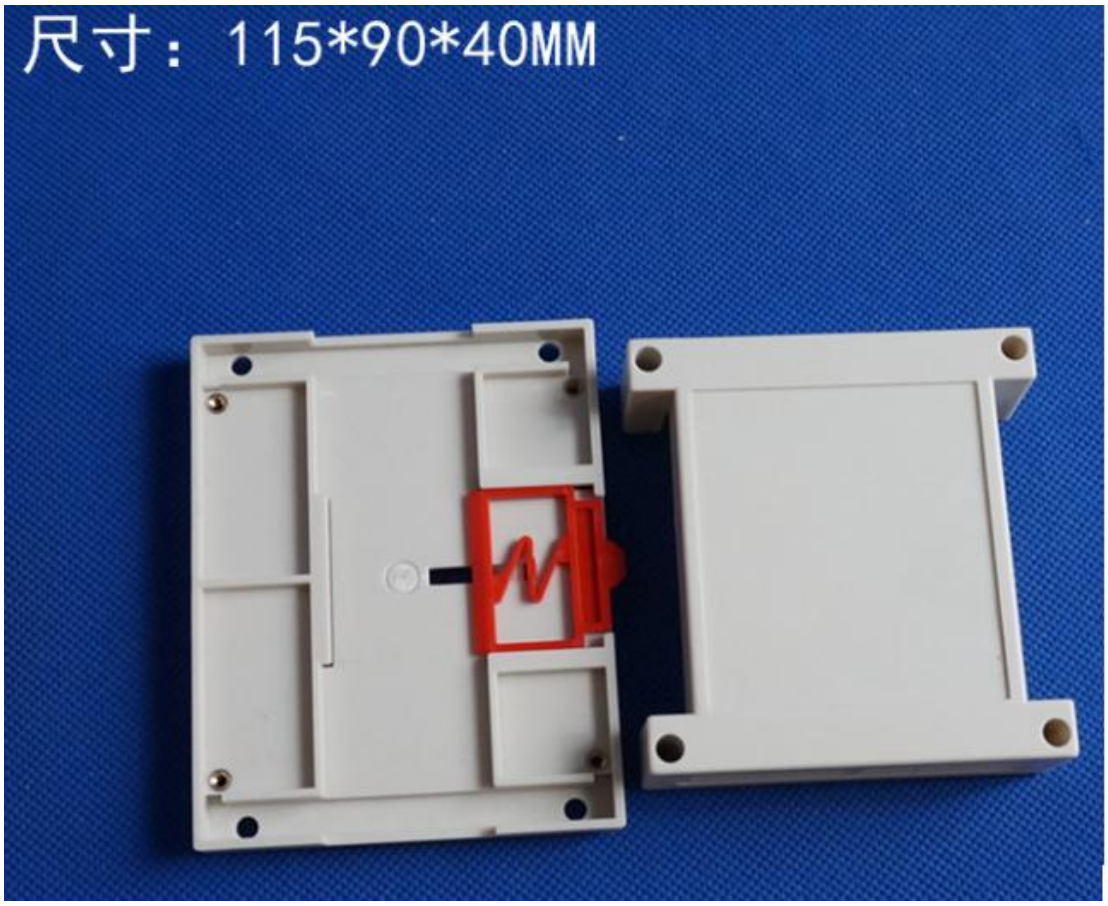
【产品图片】

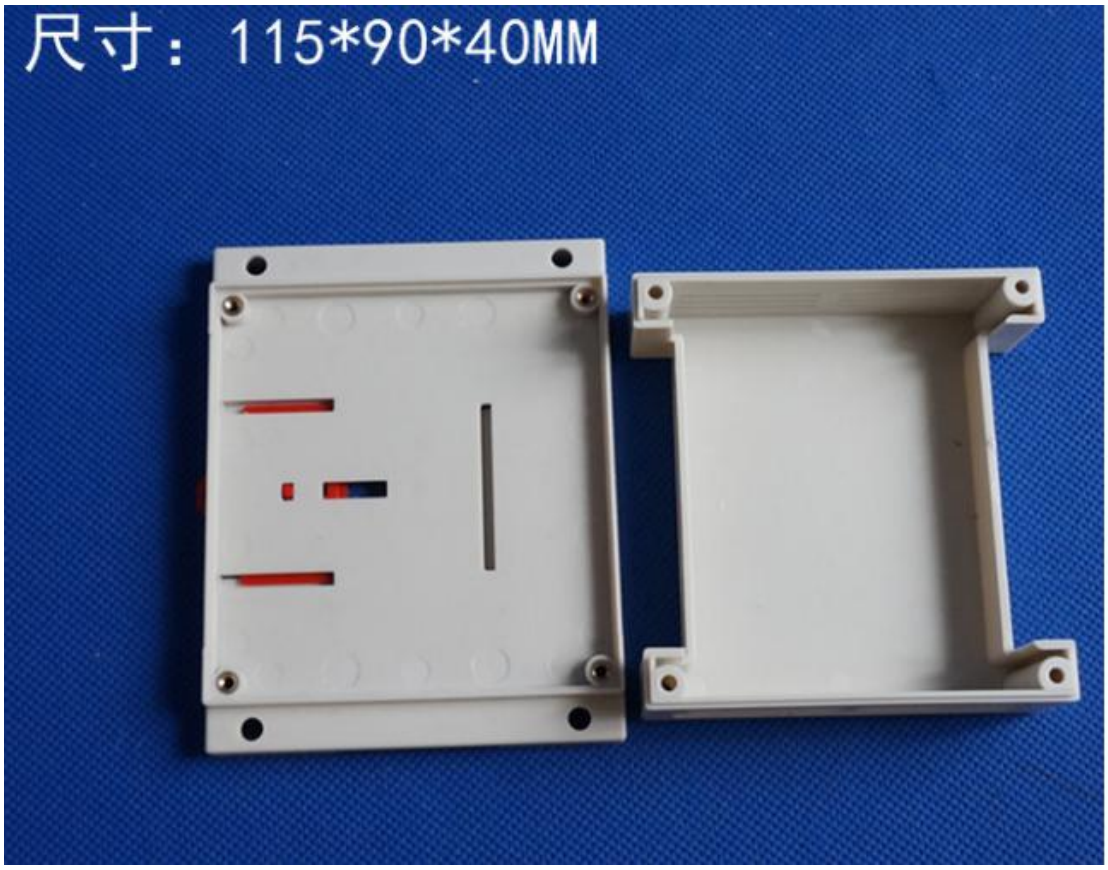

C
A
D
B


C
A
D
B


C
A
D
B




尺寸：115*90*40MM


尺寸：115*90*40MM