一、课题概述

1.1 系统基本功能

设计并制作智能农业大棚控制系统。开机后，屏幕（使用LCD12864屏）第一行显示“合肥工业大学，第二行与第三行显示“姓名+学号”，最后一行显示日期，三秒后进入工作界面。

（1）使用显示屏显示大棚温湿度（单位摄氏度）、光照（单位 lux）信息；显示补光灯、鼓风机（直流电机模拟）工作状态（开启或关闭）。其中大棚温度来自于温度传感器、光照来自于光敏电阻或其他传感器、补光灯为高亮 LED 输出。

（2）温度数据、光照数实时刷新；采用合理方式调整温度、光照信号变化，以产生可见动态效果；使用电位器模拟输入，改变温度（矩阵键盘切换）的阈值；使用试验箱上面相应的模块，实现设置的温度以及光照强度阈值信息掉电不丢失功能。

（3）当大棚温湿度高于阈值，蜂鸣器报警，同时鼓风机（电机模拟）工作；当光照低于阈值时，蜂鸣器报警，补光灯亮起，高于阈值时，补光灯灭，并在屏幕上显示报警原因。

1.2 系统附加功能

（1）当补光灯亮起后，具有自动调节亮度功能，分为三档“弱、中、强”，根据不同光照强度切换对应档位。（也可以考虑让led亮的数量不同）

（2）使用红外遥控装置实现（比如改变阈值、“打开/关闭鼓风机”、“打开/关闭补光灯”）打开和关闭直流电机、高亮LED灯。

（3）在矩阵键盘上设置加减按钮以及相应的功能按钮，可以通过按键来改变温度以及光照强度阈值。

（4）实现与PC主机间的通信，PC主机可以将数据传输到单片机上，实现相应功能（如更改阈值），模拟真实场景。

（5）将上述的所有功能用菜单图形化界面显示。

二、设计方案

2.1 设计方案硬件电路原理图

由于Proteus软件没有20pin的LCD12864，我们使用元件库更为齐全的软件立创EDA进行硬件电路原理图设计，具体硬件电路原理图如图2-1所示

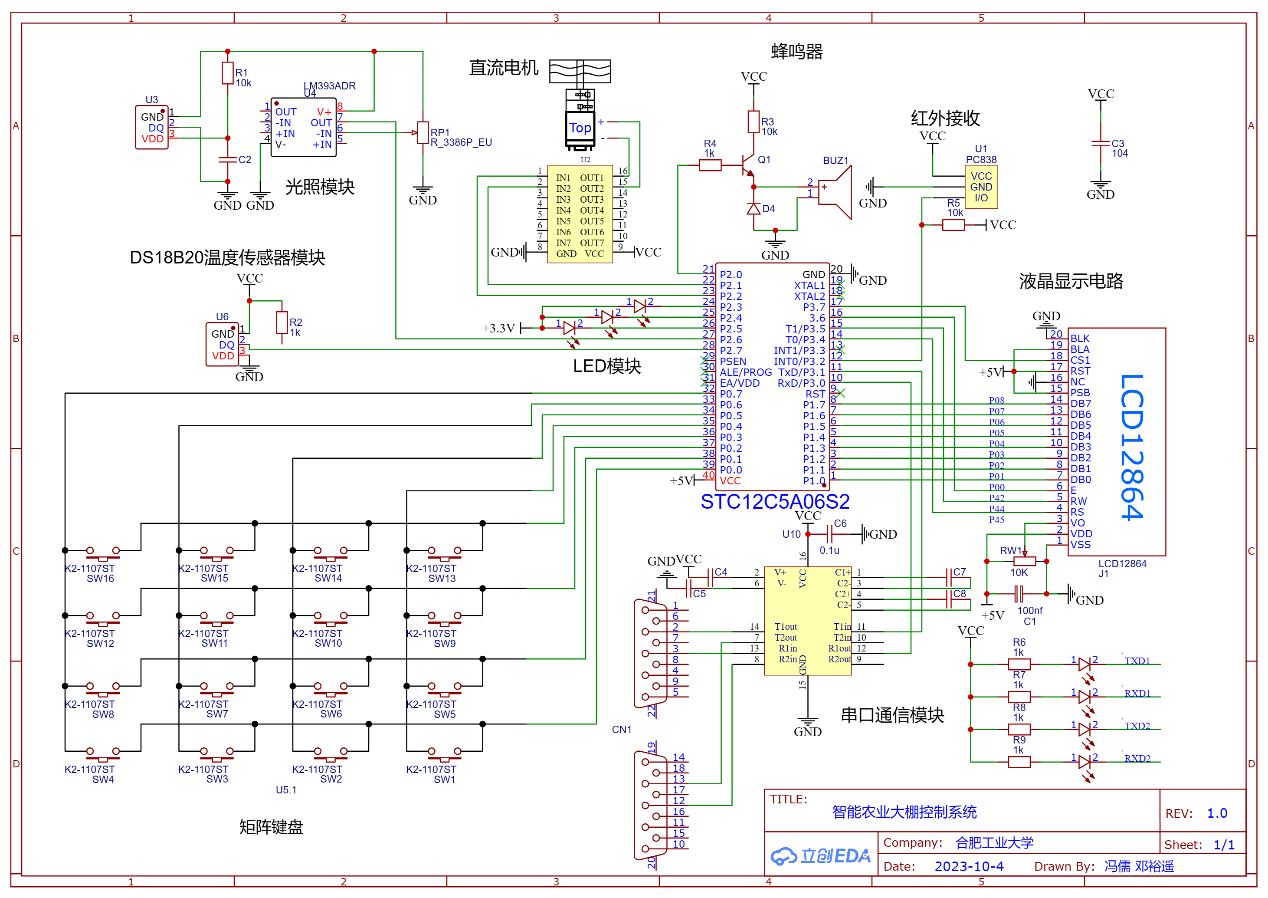


图2-1 硬件电路原理图

2.2 软件核心模块及功能说明

此次课程设计涉及的核心模块有：LCD12864、矩阵键盘、红外接收模块、光敏传感器、温度传感器DS18B20、直流电机、串口通信、LED模块、蜂鸣器

2.2.1 LCD12864

LCD12864是一种128×64像素的液晶显示屏，用于显示设备当前温度、当前光照强度、当前鼓风机状态、当前补光灯状态、当前温度阈值、当前光照阈值及报警原因。其硬件电路原理图如图2-2所示，D0-D7为数据端口，单片机P1口与底板JP41排针相连，分别为：P10-D0, P11-D1，P12-D2，P13-D3，P14-D4，P15-D5，P16-D6，P17-D7；单片机P34~P37口与底板JP36相连，分别为：P34-RS,P35-RW,P36-EN，P37-CS1，CS2悬空。

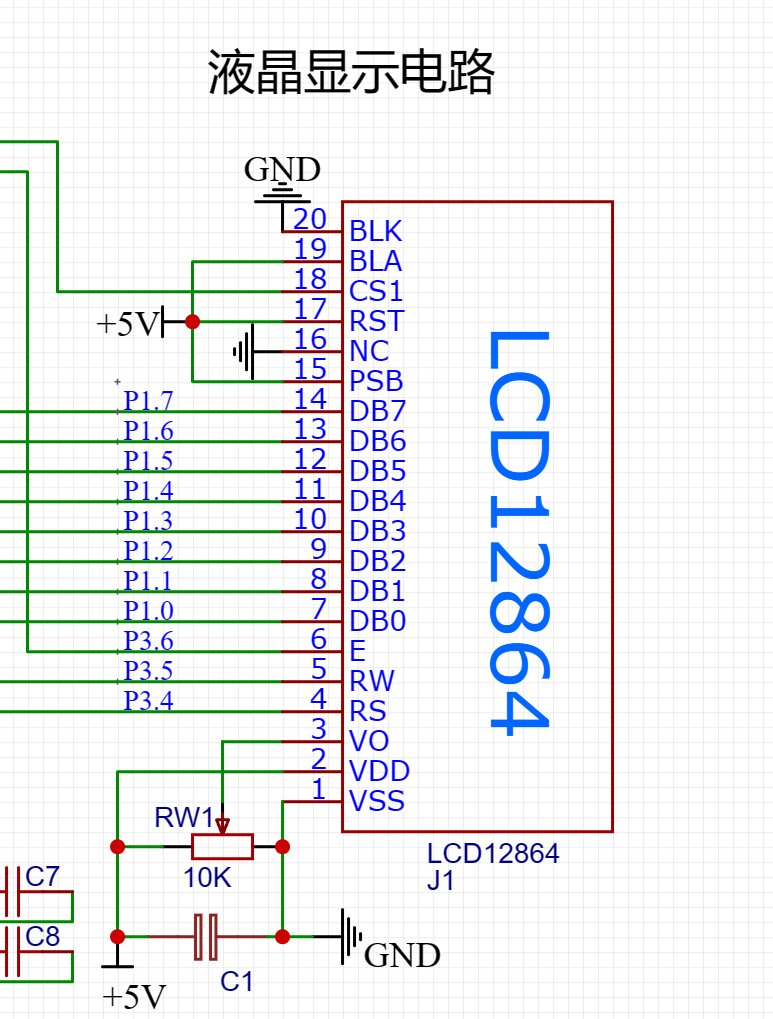


图2-2 LCD12864原理图

2.2.2 温度传感器DS18B20

用于获取温度的重要模块，将测量环境温度并将温度数据以数字形式提供给连接它的电子设备或微控制器。其硬件电路原理图如图2-3所示，P27为数据输入端口。

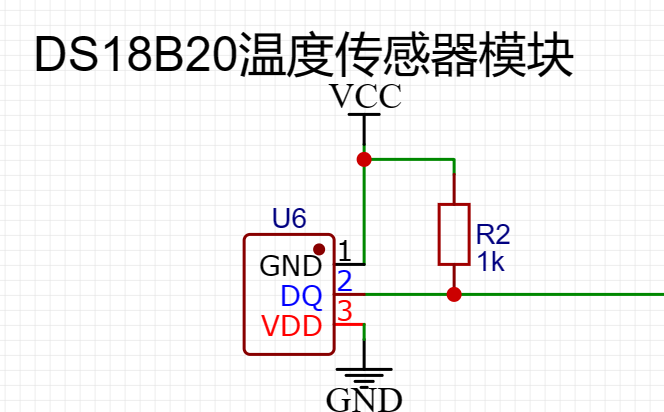


图2-3 DB18B20原理图

2.2.3 红外接收模块

此模块可以使用户能够通过红外遥控器向单片机发出指令，并执行相应功能，做到温度和光照阈值的更改。其硬件电路原理图如图2-4所示，单片机外部中断P32口接底板JP12。

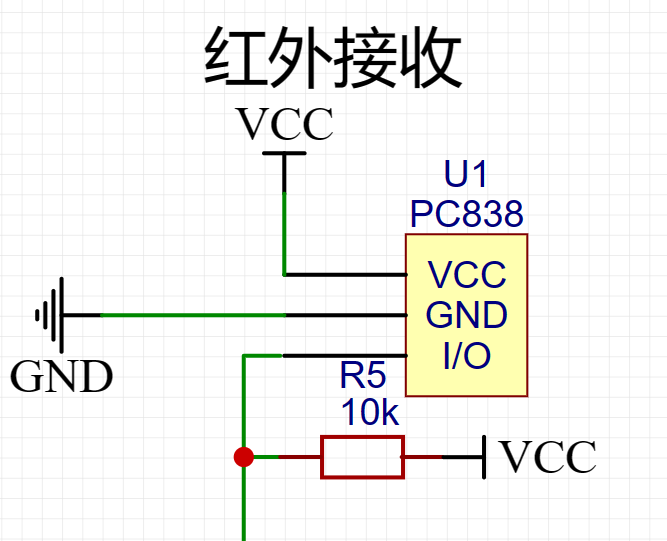


图2-4 红外接收原理图

2.2.4 串口通信

此模块可以使单片机与PC主机之间相互通信，实现PC主机远程发出指令让单片机执行，以及单片机向PC主机发出当前的阈值信息。其硬件电路原理图如图2-5所示，单片机P31接底板JP8的TX1，单片机P30接底板JP8的RX1。串口线接DB1串口母座。

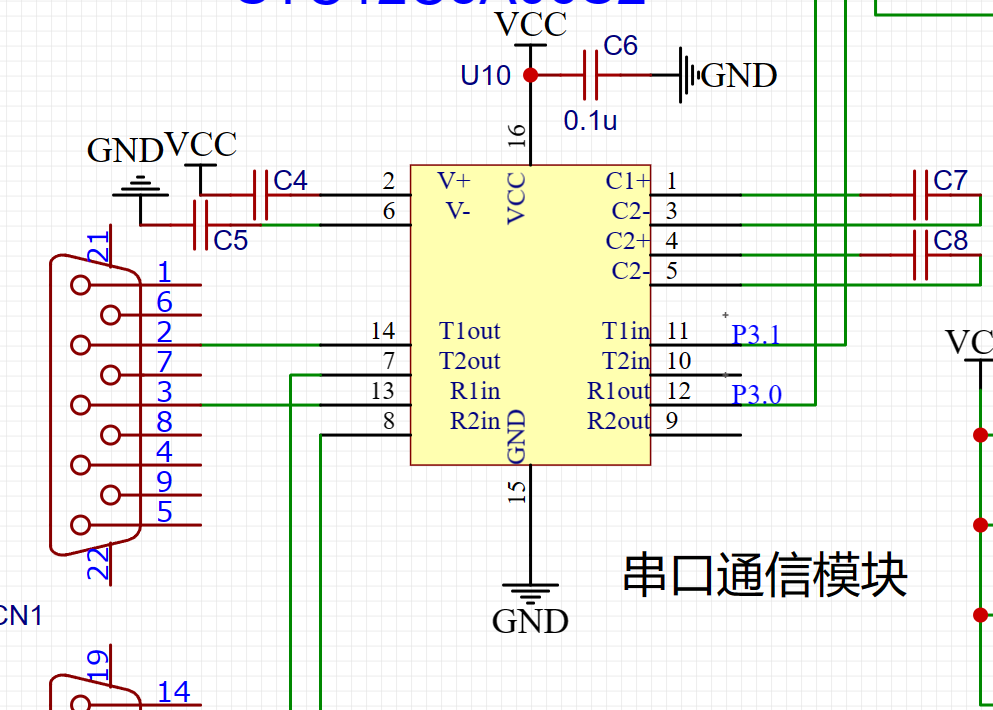


图2-5 串口通信原理图

2.2.5 矩阵键盘

此模块可以对单片机进行操控，当用户按下不同的键位时执行相应功能，用户按下按键1到按键4时继续温度和光照阈值的更改，用户按下按键5时，单片机向PC主机发送当前阈值信息。其硬件电路原理图如图2-6所示。

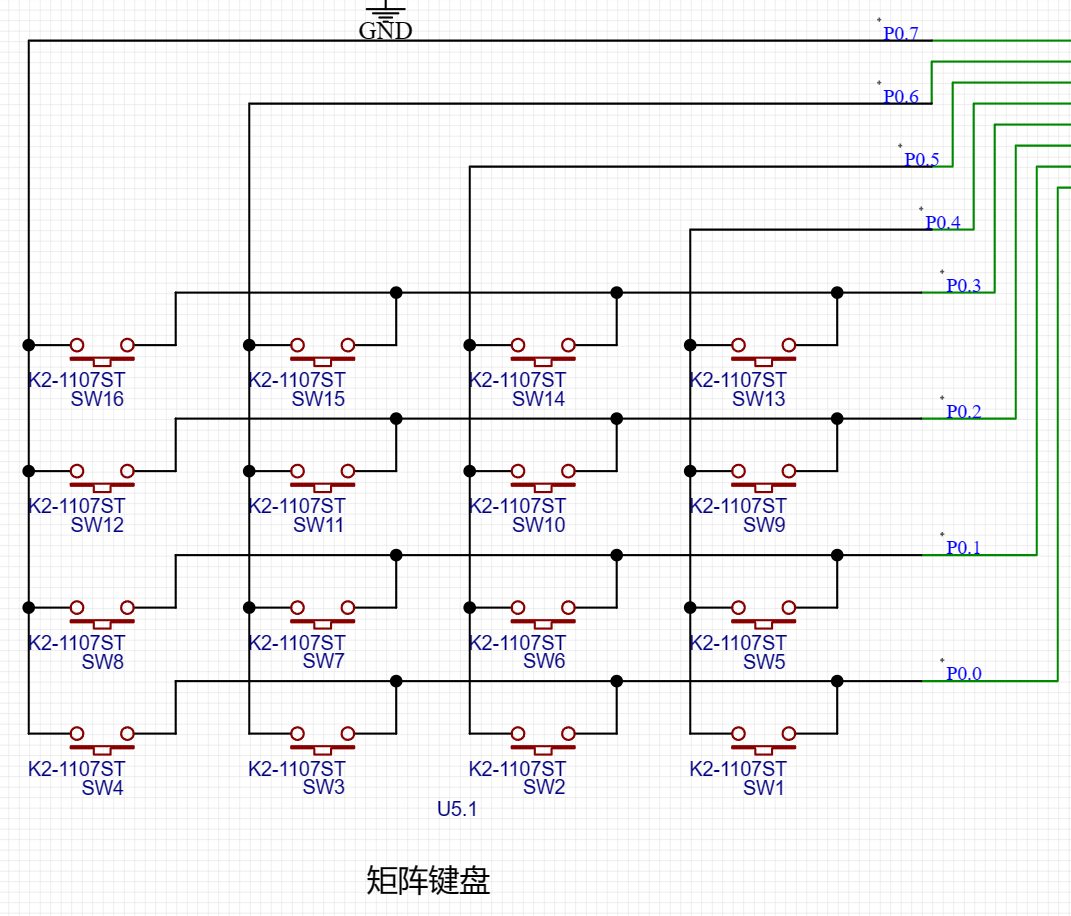


图2-6 矩阵键盘原理图

2.2.6 直流电机

此模块用于模拟鼓风机，电机转代表鼓风机开，当温度高于设定阈值时，鼓风机开始工作，对温度传感器进行降温，当温度低于阈值后，鼓风机关闭。其硬件电路原理图如图2-7所示。使用2P杜邦线连接核心板P00接JP32的I6，P01接JP32的I7，直流电机接JP38的两端。

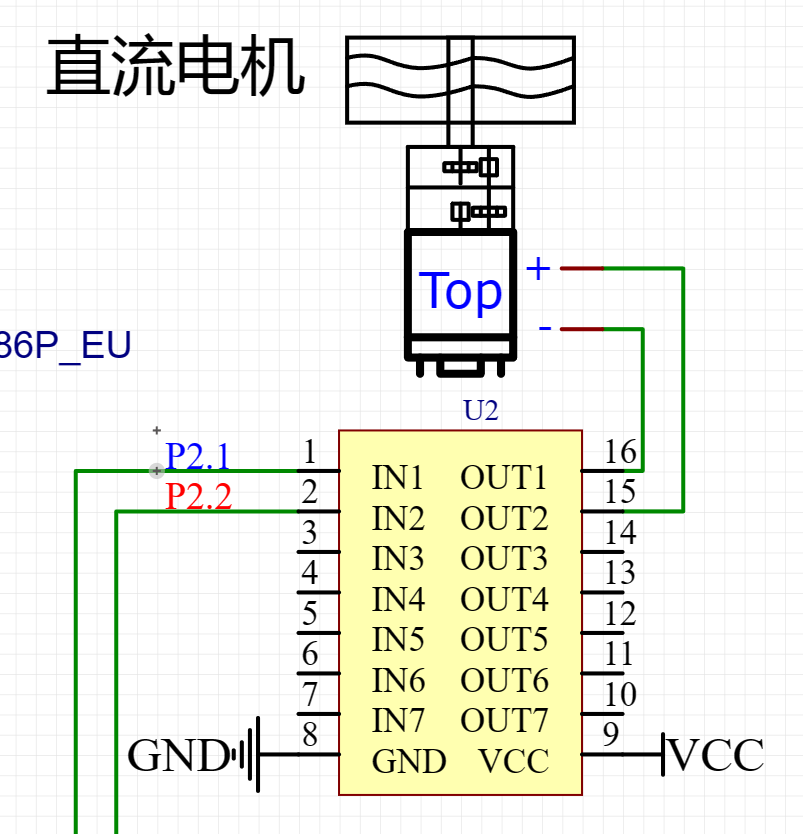


图2-7 直流电机原理图

2.2.7 蜂鸣器

此模块用于模拟报警装置，当出现温度高于阈值或者光照低于阈值后，蜂鸣器开始鸣叫。其硬件电路原理图如图2-8所示，引脚接P20

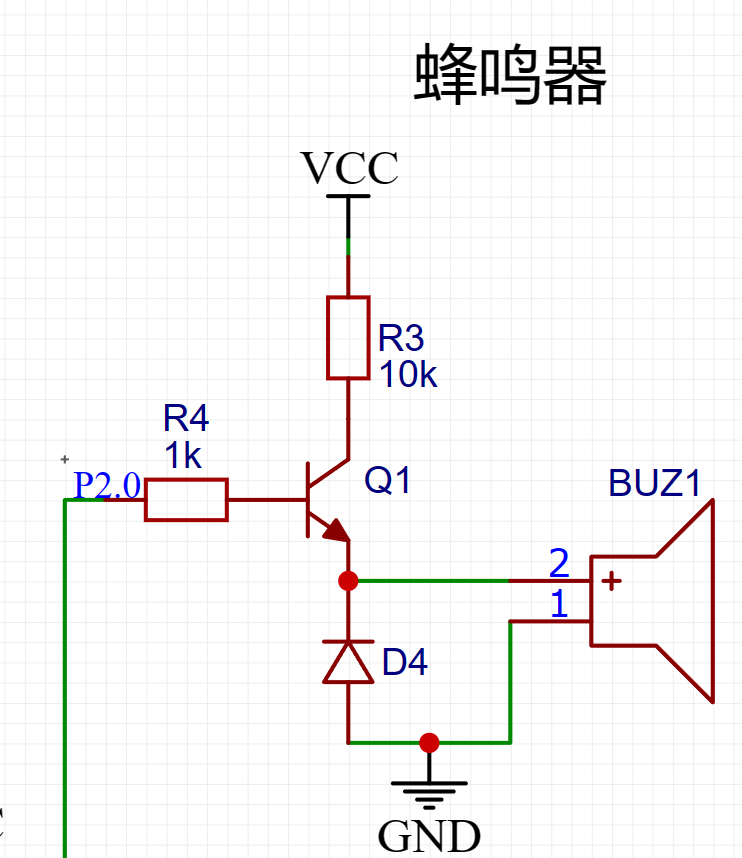


图2-8 蜂鸣器原理图

2.2.8 光敏传感器

此模块用于获取当前的光照值，由于试验箱提供的光敏传感模块不具有相关功能，因此在此次课设中设定光照强度为固定70lux。其硬件电路原理图如图2-9所示。

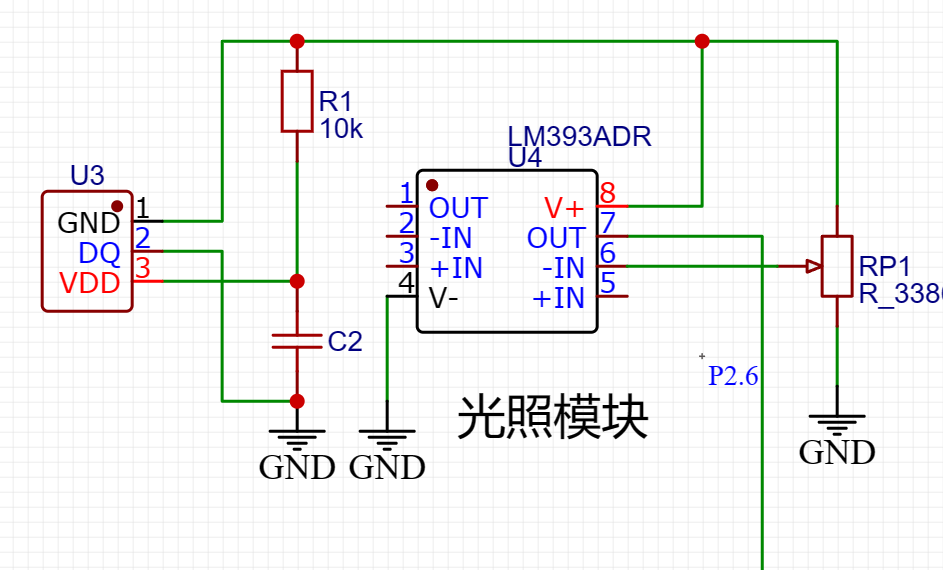


图2-9 光敏传感器原理图

2.2.9 LED模块

此模块用于模拟补光灯，当光照强度低于设定阈值后补光灯开启，补光灯有三档，低于光照阈值10时开启第二档（即亮两个LED灯），低于光照阈值15时开启第三档（即亮三个LED灯）。其原理图如图2-10所示。

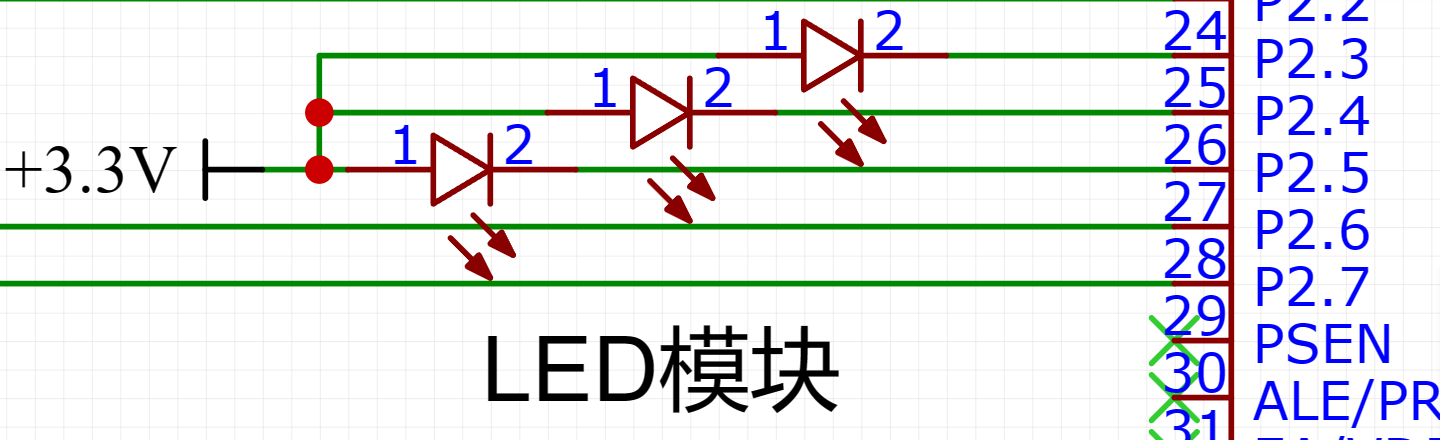


图2-10 LED模块原理图

2.3 软件核心模块流程图

按照系统设计的要求，确定系统由LCD12864、矩阵键盘、红外接收模块、光敏传感器、温度传感器DS18B20、直流电机、串口通信、LED模块、蜂鸣器共九个模块组成，系统构成框图如图2-11所示，图中箭头方向表示数据传输方向。

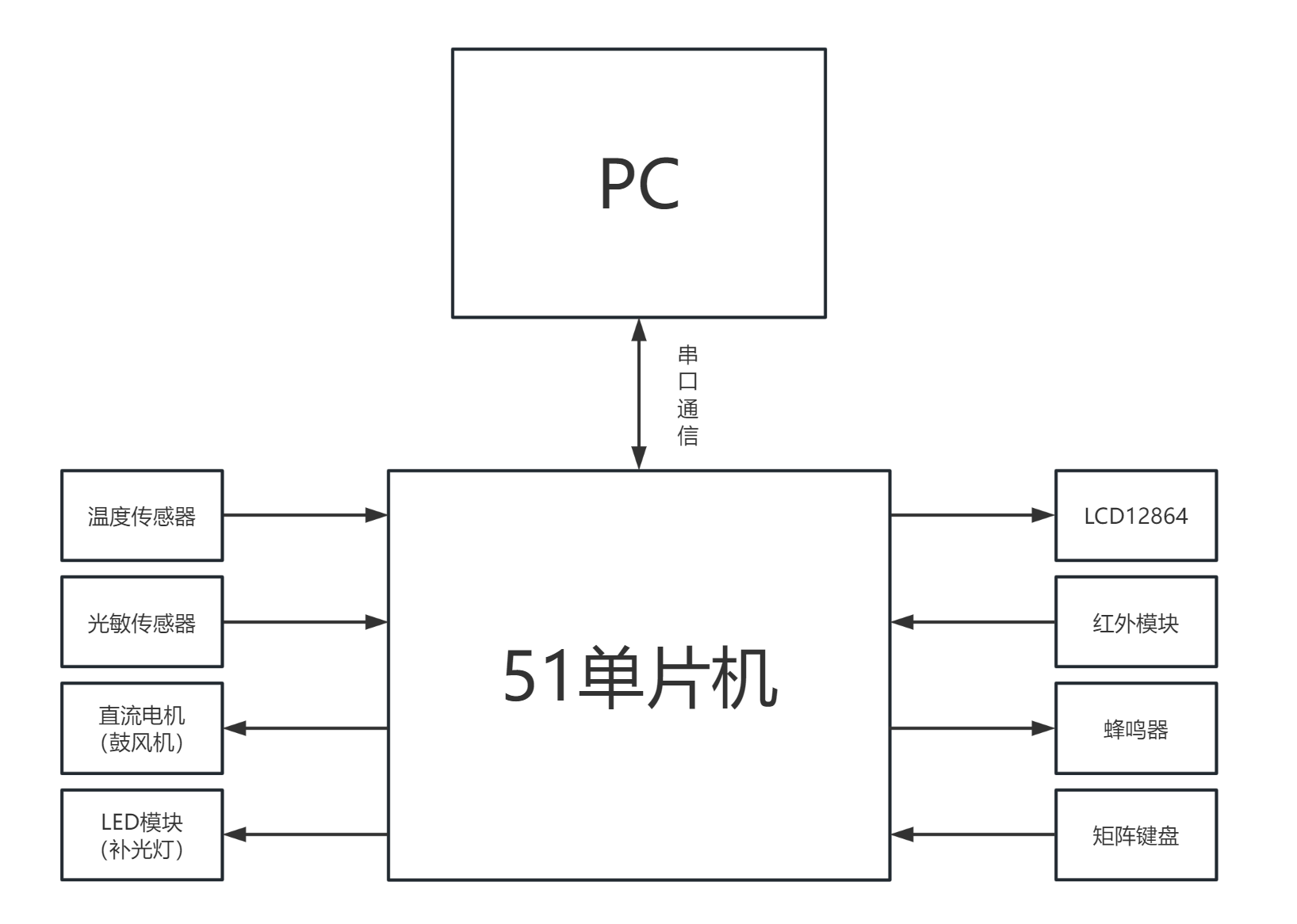


图2-11 系统构成框图

主程序流程图如图2-12所示。

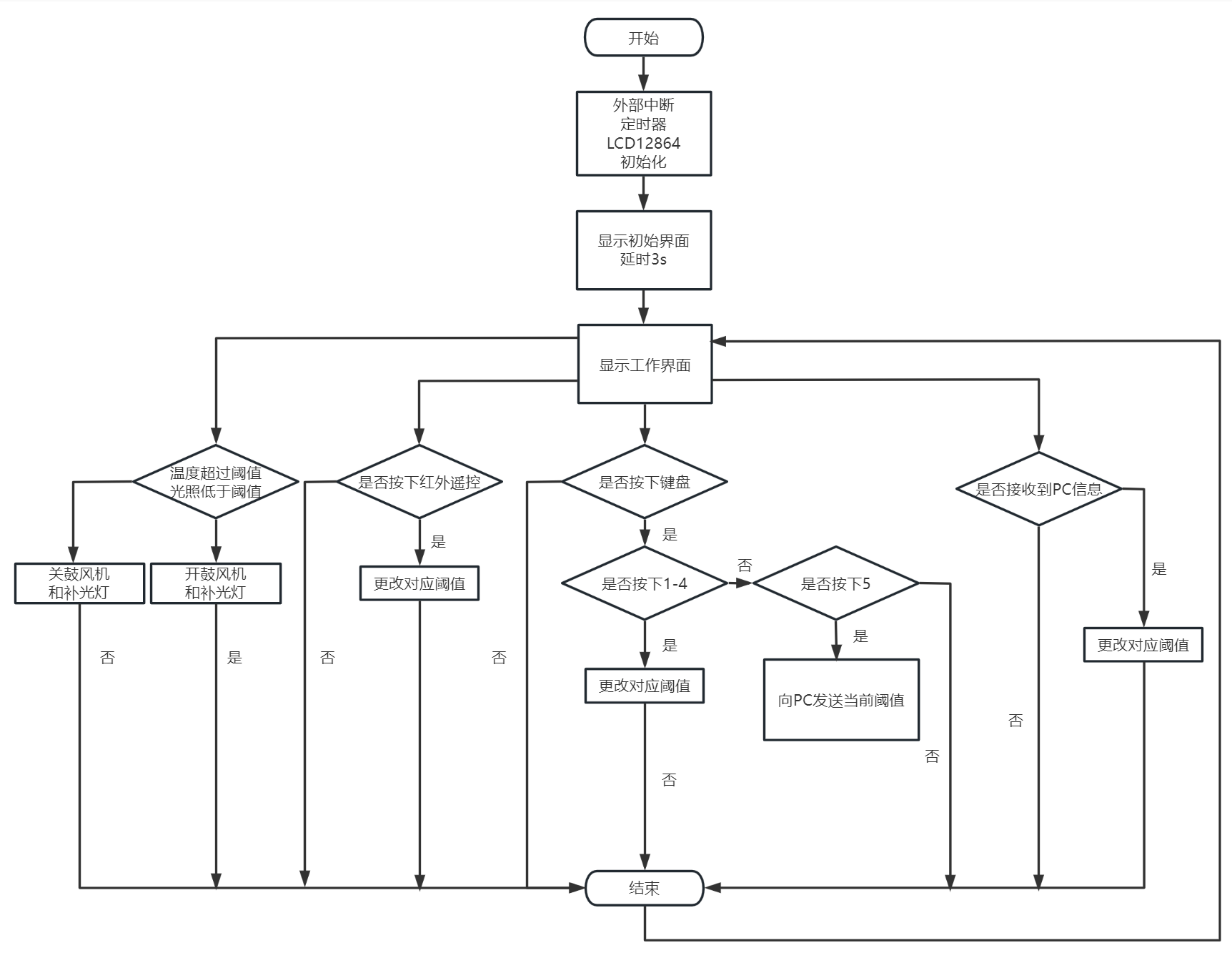


图2-12 主程序流程图

核心模块包括矩阵键盘、红外遥控、串口通信、温度读取、状态判定逻辑模块

矩阵键盘模块流程图如图2-13所示。

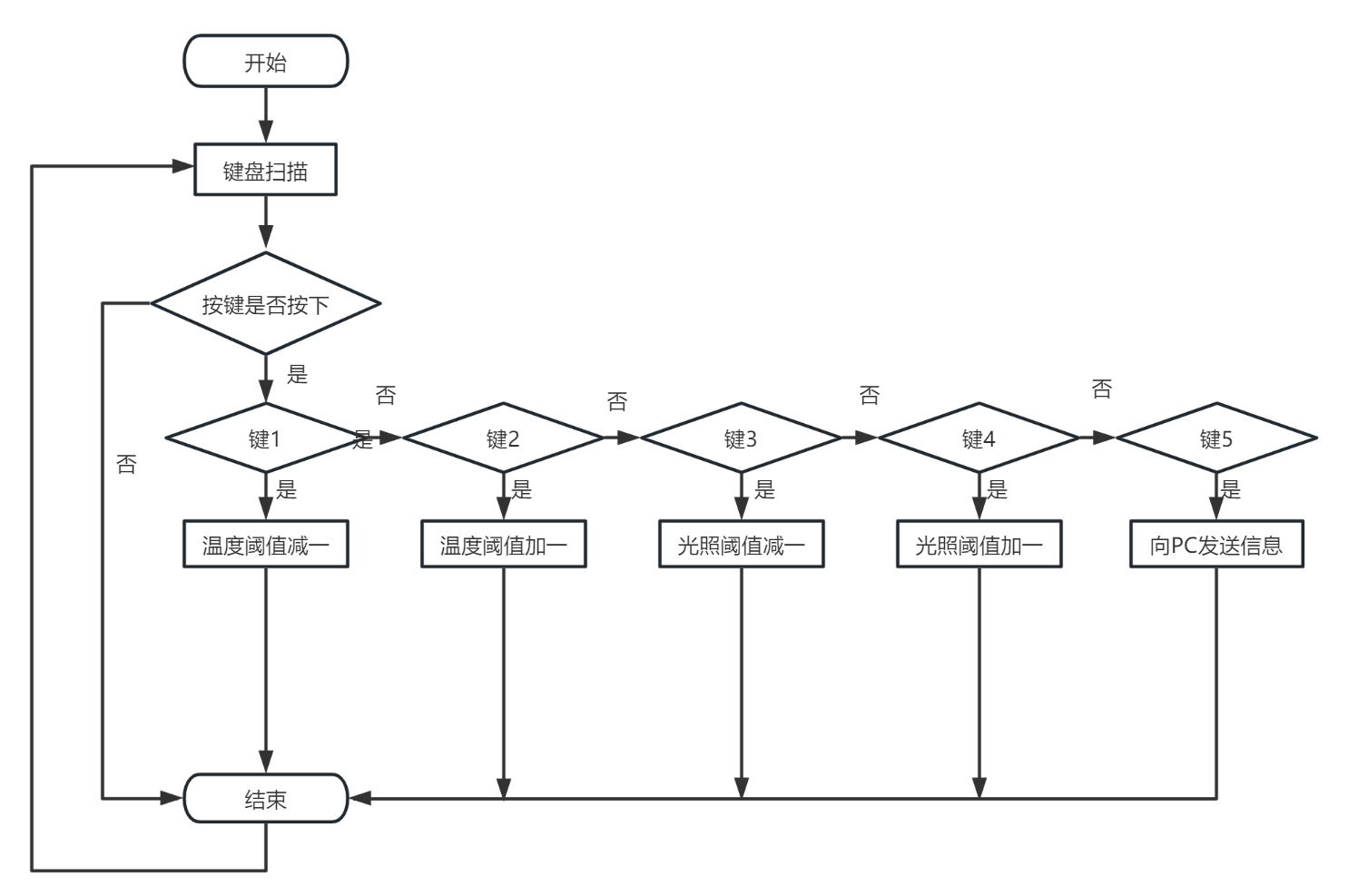


图2-13 矩阵键盘流程图

红外遥控流程图如图2-14所示。

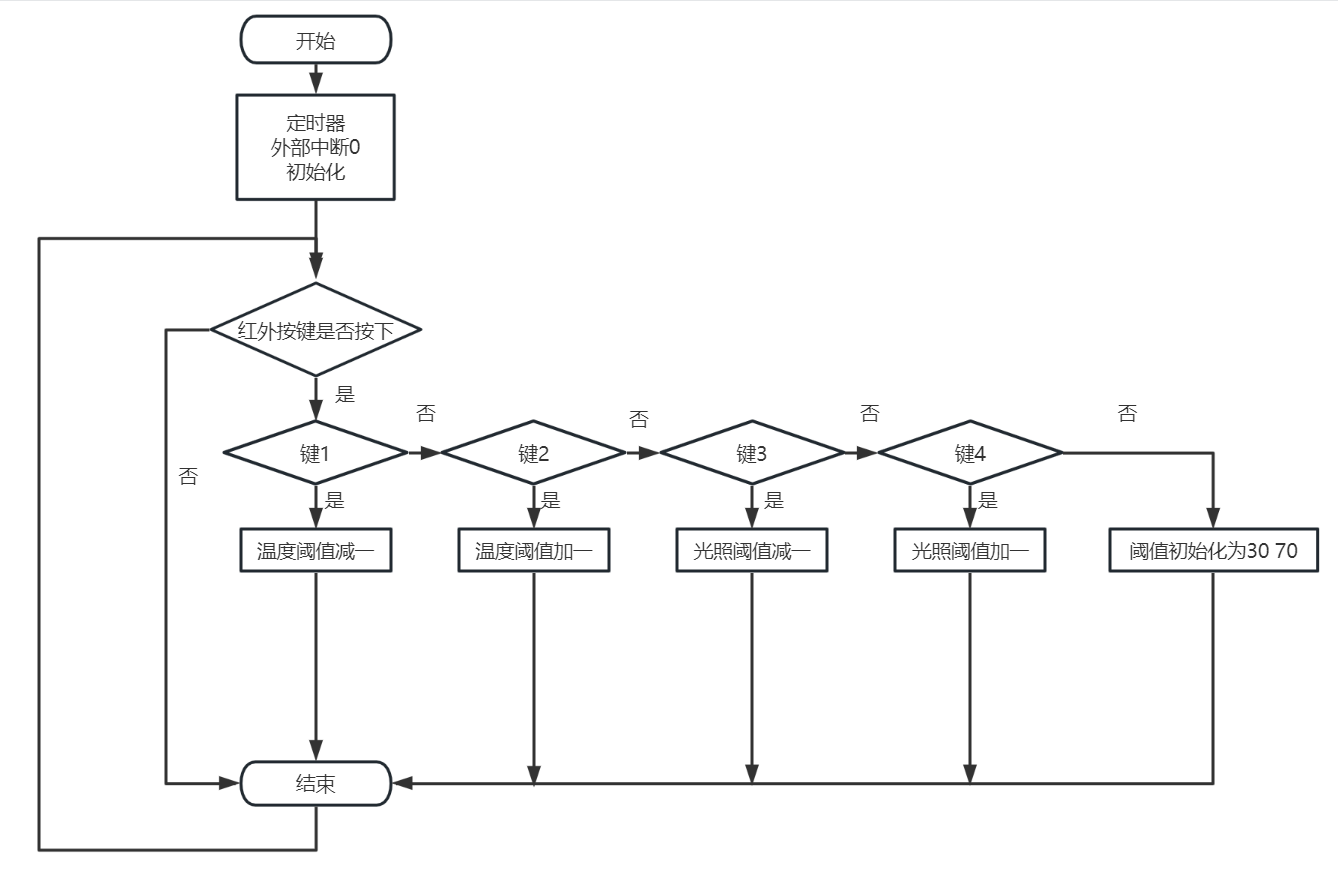


图2-14 红外遥控流程图

串口通信流程图如图2-15所示。

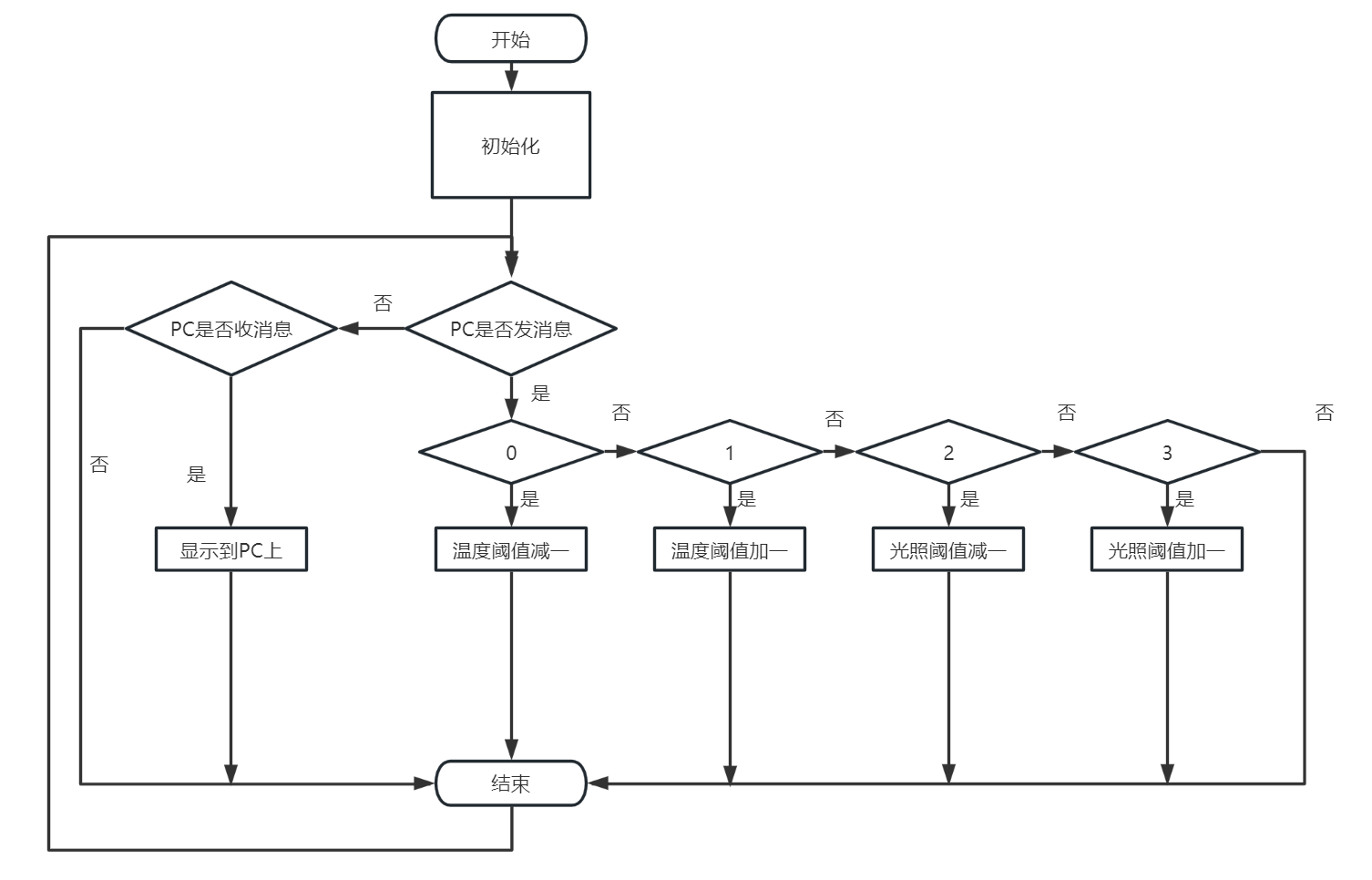


图2-15 串口通信流程图

温度读取流程图如图2-16所示。

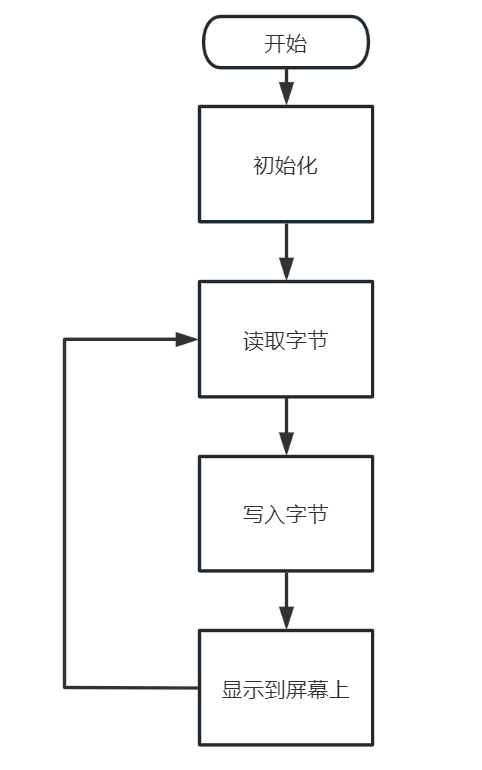


图2-16 温度读取流程图

状态判定逻辑模块流程图如图2-17所示，此流程图包括剩余模块。

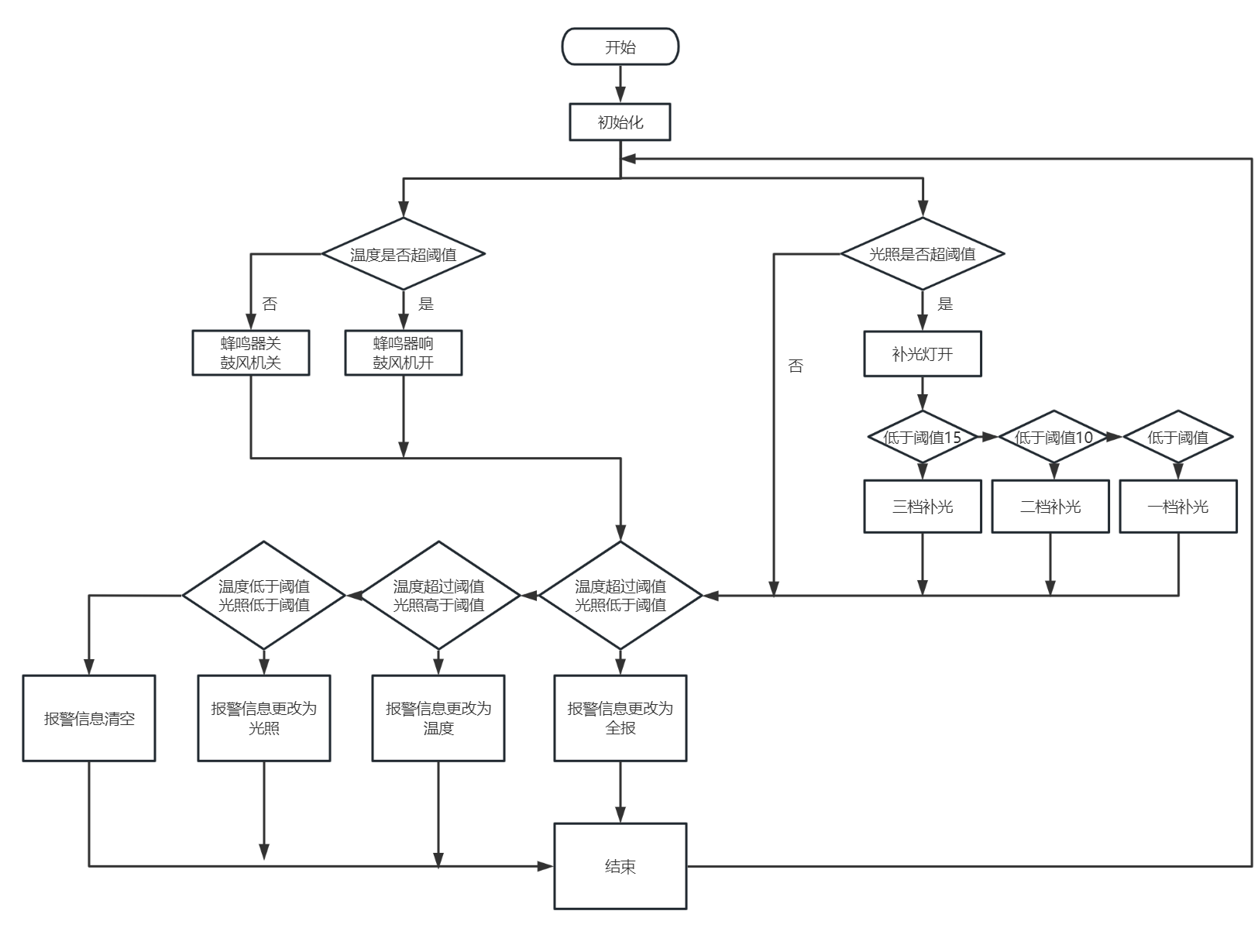


图2-17 状态判定逻辑模块流程图

三、设计实现与测试结果

3.1 核心模块代码及分析说明

3.1.1 LCD12864

此模块所需函数：

void LCD12864\_Init(void);         //LCD12864初始化函数

void LCD12864\_WriteInfomation(unsigned char ucData,bit bComOrData);

//向LCD12864写入数据，bComOrData为1时写入的是数据，0时写入的是命令

void LCD12864\_CheckBusy(void);      //忙检测函数

void LCD12864\_DisplayOneLine(unsigned char ucPos,unsigned char \*ucStr); //向LCD12864写入一行文字

void LCD12864\_value(unsigned char position,unsigned char \*p);

void Delay(unsigned int uiCount);

void DisplayOne(unsigned char Data);

重要函数代码分析：

void LCD12864\_DisplayOneLine(unsigned char position,unsigned char \*p)

{

     unsigned char i;

     LCD12864\_WriteInfomation(position,0);//写入要显示文字的行的首地址

     Delay(150);

     for(i = 0;i<16;i++)        //依次执行写入操作

     {

         LCD12864\_WriteInfomation(\*p,1);

         p++;

     }

}

此函数可以在LCD12864上显示一行信息，positon为地址，p为信息。通过循环将p中的信息依次写入LCD12864，写入调用LCD12864\_WriteInfomation(\*p,1) 函数。

void LCD12864\_value(unsigned char position,unsigned char \*p)

{

     unsigned char i;

     LCD12864\_WriteInfomation(position,0);//写入要显示文字的行的首地址

     Delay(150);

     for(i = 0;i<2;i++)     //依次执行写入操作

     {

         LCD12864\_WriteInfomation(\*p,1);

         p++;

     }

}

此函数用于在LCD12864上显示一个字节的信息，positon为地址，p为信息。

在编写代码的时候，我们要知晓地址和信息，其中地址的位置如图3-1所示，信息为字符串即可。

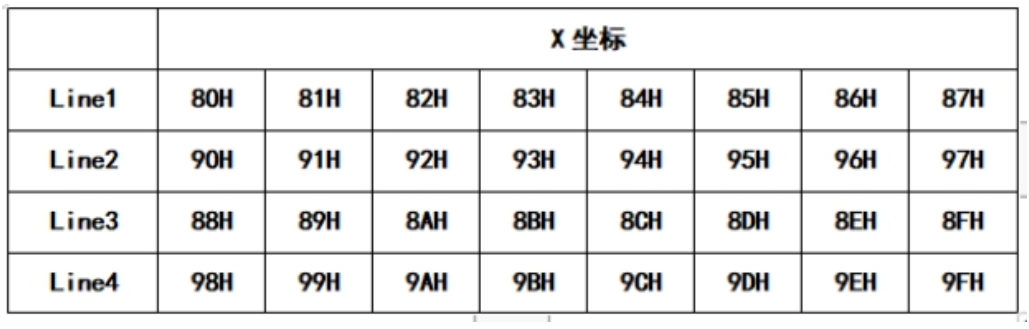


图3-1 LCD12864地址

3.1.2 温度传感器DS18B20

此模块所需函数：

void DS18B20\_Init(void);   //DS18B20初始化

unsigned char DS18B20\_ReadOneChar(void);   //读取一个数据

void DS18B20\_WriteOneChar(unsigned char dat);     //写入一个数据

unsigned int DS18B20\_ReadTemperature(void);       //读取温度

重要函数代码分析：

void DS18B20\_Init(void) //初始化

{

    unsigned char x=0;

    DQ = 1;    //DQ复位

    Delay\_us(10);

    //Delay(8);  //稍做延时,10us

    DQ = 0;    //单片机将DQ拉低

    Delay\_us(500);

    //Delay(80); //精确延时 大于 480us ,498us

    DQ = 1;    //拉高总线

    Delay\_us(154);

    //Delay(14);    //154us

    x=DQ;      //稍做延时后 如果x=0则初始化成功 x=1则初始化失败

    Delay\_us(212);

    //Delay(20); //212us

}

此函数用于DS18B20的初始化，DS18B20的初始化较为复杂，DQ必须满足如图3-2所示的时序曲线，只有满足后才可以正确的进行温度读取。具体步骤如下：

1、数据线先拉到高电平，稍作延时即可（刚开始是高电平还是低电平芯片手册上其实不关心这一部分）

2、数据线拉到低电平

3、延时（480us~960us）

4、将数据线拉高电平

5、延时等待（大于60us)

6、判断有没有初始化成功；理论上电平在第4步置高后，DS18B20如果存在就会将数据线拉低如果不存在就还是高电平

7、延时（cpu读到18b20回应的低电平后，还要做延时；其时间是从发出高电平（第4步）时间算起，至少要480us

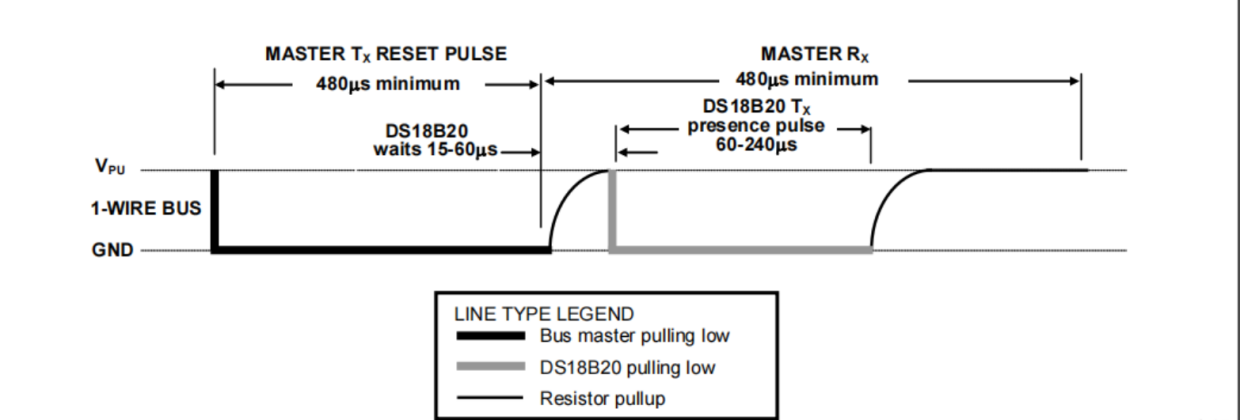


图3-2 初始化时序

读写字节同样需要满足一定的时序，具体如图3-3和图3-4所示。

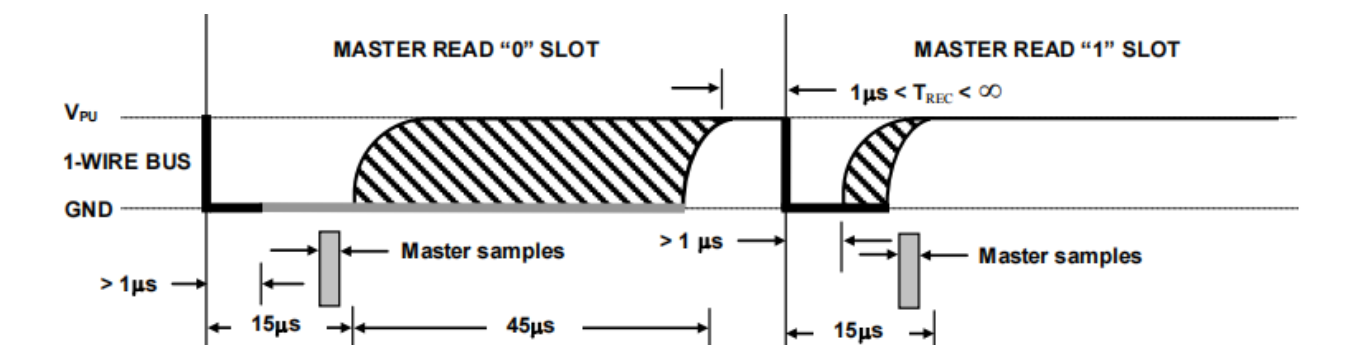


图3-3 读取字节时序

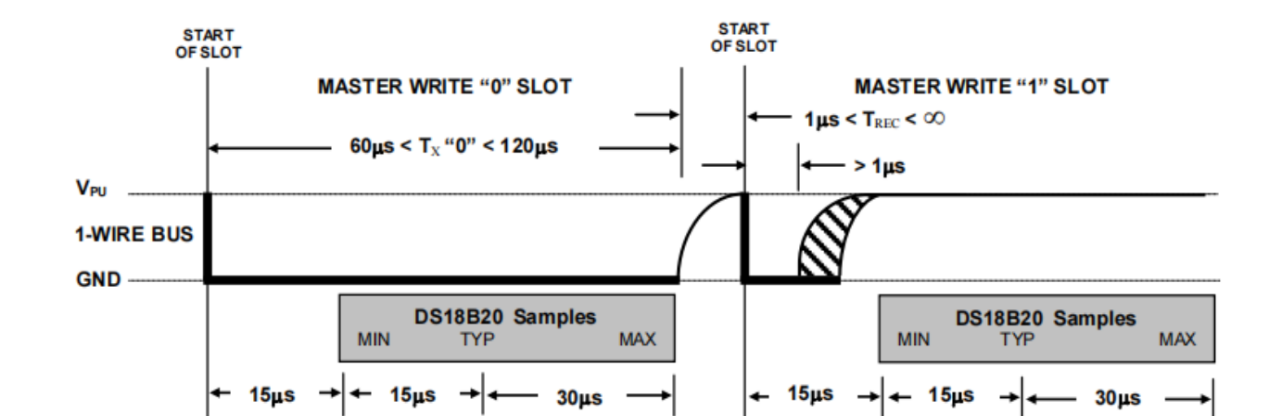


图3-4 写入字节时序

unsigned int DS18B20\_ReadTemperature(void)

{

    unsigned char a=0;

    unsigned char b=0;

    unsigned int t=0;

    DS18B20\_Init();

    DS18B20\_WriteOneChar(0xCC); // 跳过读序号列号的操作

    DS18B20\_WriteOneChar(0x44); // 启动温度转换

    DS18B20\_Init();

    DS18B20\_WriteOneChar(0xCC); //跳过读序号列号的操作

    DS18B20\_WriteOneChar(0xBE); //读取温度寄存器等（共可读9个寄存器） 前两个就是温度

    a=DS18B20\_ReadOneChar();

    b=DS18B20\_ReadOneChar();

    t=b;

    t<<=8;

    t=t|a;

    //t= t/2; //放大10倍输出并四舍五入---此行没用

    t = t\*0.0625;

    return(t);

}

此函数用与获取温度，DS18B20获取的是16位二进制数据，前八位和后八位用a和b表示，温度整数数值的位置在高位后三位和低位前四位，小数数值为低位后四位，格式如图3-5所示，因此需要对获取的数据进行转换，温度表示图如图3-6所示。

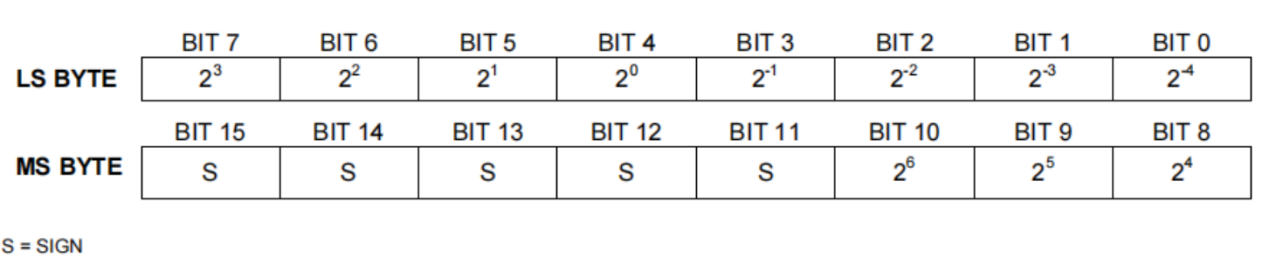


图3-5 16位二进制格式

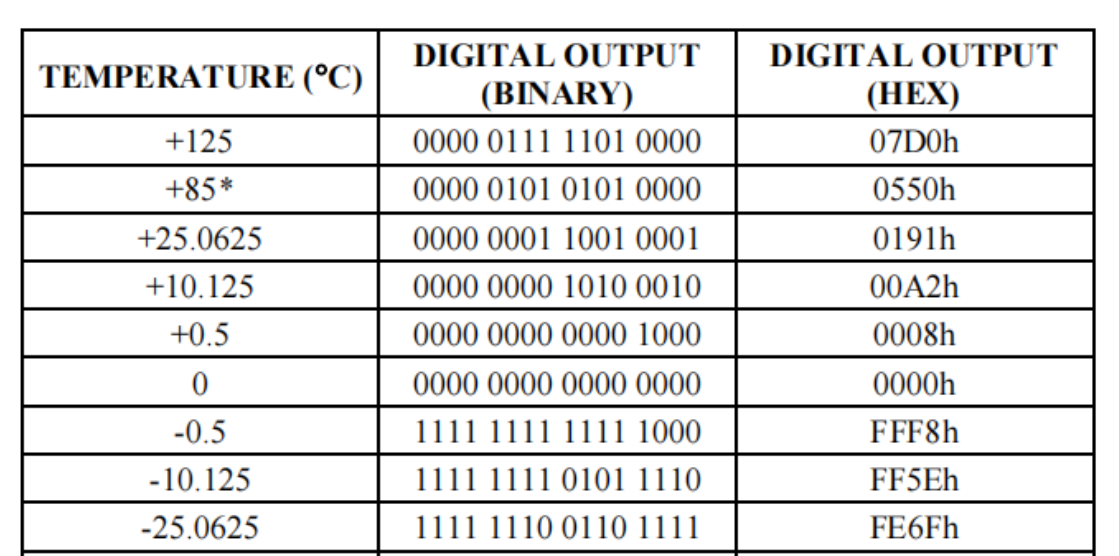


图3-6 温度表示图

不过我们无需进行复杂的计算，只需要将获得的值乘以0.0625即可得到所需温度值，此时得到的数值为温度的整数值。

3.1.3 红外接收模块

此模块所需函数：

void Ir\_work(void); //遥控

void Ircordpro(void);//遥控

void ShowString (unsigned char line,char \*ptr); //遥控

void ex0\_isr (void) interrupt 0 using 0//外部中断0服务函数

void tim0\_isr (void) interrupt 1 using 1//定时器0中断服务函数

重要函数代码分析：

void Ircordpro(void)//红外码值处理函数

{

  unsigned char i, j, k;

  unsigned char cord,value;

  k=1;

  for(i=0;i<4;i++)      //处理4个字节

     {

      for(j=1;j<=8;j++) //处理1个字节8位

         {

          cord=irdata[k];

          if(cord>7)//大于某值为1，这个和晶振有绝对关系，这里使用12M计算，此值可以有一定误差

            {

             value=value|0x80;

            }

          else

            {

             value=value;

            }

          if(j<8)

            {

             value=value>>1;

            }

           k++;

         }

     IRcord[i]=value;

     value=0;

     } irpro\_ok=1;//处理完毕标志位置1

}

此函数用于解码，将解码后的数值传到IRcord这个数组中。

void Ir\_work(void)        //红外键值散转程序

  {

         char TimeNum[]="  ";

       TimeNum[0] = Tab[IRcord[2]/16];  //处理数据码并显示

       TimeNum[1] = Tab[IRcord[2]%16];

//       LCD12864\_value(0x87,TimeNum);

     if(strcmp(TimeNum,"0C")==0)

                tmp\_--;

         else if(strcmp(TimeNum,"18")==0)

              tmp\_++;

         else if(strcmp(TimeNum,"5E")==0)

              light\_--;

         else if(strcmp(TimeNum,"08")==0)

              light\_++;

         else tmp\_++;

       irpro\_ok=0;           //处理完成后清楚标志位

  }

此函数用于判断用户按下的是哪个键位，执行对应的操作，遥控器对应键位数据码如图3-7所示，判断字符串是否相等采用的是strcmp函数。



图3-7 数据码

3.1.4 串口通信

此模块所需函数：

void UART\_Init(void);

void UART\_Send\_Byte(unsigned char ucData);

void UART\_Send\_Str(char \*pStr);

void UART\_Send\_Enter(void);

void UART\_Init(void);

void UartInit();

void Uartsend(unsigned char byte);

重要函数代码分析：

void UartInit()     //9600bps@11.0592MHz

{

    PCON &= 0x7F;       //波特率不倍速

    SCON = 0x50;        //8位数据,可变波特率

    TMOD &= 0x0F;       //清除定时器1模式位

    TMOD |= 0x20;       //设定定时器1为8位自动重装方式

    TL1 = 0xFD;         //设定定时初值

    TH1 = 0xFD;         //设定定时器重装值

    TR1 = 1;            //启动定时器1

    ET1 = 0;            //禁止定时器1中断

    EA=1;               //开放总中断

    ES=1;               //开放串口中断

}

此函数用于对串口通信进行初始化，在串口通信的过程中，对晶振频率，波特率等参数需要进行严格的设定，否则通信接收的会是乱码，计算参数可用stc-isp软件自带的计算功能进行设定，如图3-8所示。

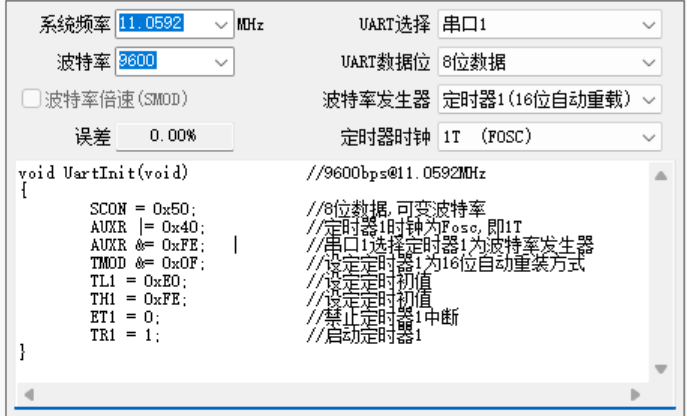


图3-8 参数计算设定

void UART\_ISR() interrupt 4//串口中断函数

{

    if(RI==1)//接收中断

    {

        a = SBUF;

        switch(a)

        {

            case '0':

                tmp\_--;break;

            case '1':

                tmp\_++;break;

            case '2':

                light\_--;break;

            case '3':

                light\_++;break;

            default:

                tmp\_=30;

                light\_=80;

                break;

        }

        Uartsend(SBUF);//把接收的数据发送到计算机

        RI=0;//软件清零

    }

}

此函数用于PC主机远程更改单片机温度和光照阈值，同时将更改后的数据返回到PC主机上进行验证，此函数重点是SBUF寄存器的使用。

3.1.5 矩阵键盘

此模块所需函数：

unsigned char Key\_Scan(void);

void DisplayOne(unsigned char Data); //

重要函数代码分析：

void DisplayOne(unsigned char Data)

{       char buf1[2];

        char buf2[2];

        sprintf(buf1,"%d",tmp\_);

        sprintf(buf2,"%d",light\_);

        switch(Data)

            {

                case 0:

                tmp\_--;break;

                case 1:

                tmp\_++;break;

                case 2:

                light\_--;break;

                case 3:

                light\_++;break;

                case 4:

                    UART\_Init();

                    UART\_Send\_Byte(11);

                    UART\_Send\_Str(buf1);

                    UART\_Send\_Str(buf2);

                    UART\_Send\_Enter();break;

                default:

                    break;

            }

}

此函数用于根据用户按下的键位进行相应操作，按键的判断代码过长，在此不进行展示，矩阵键盘判断的原理是读取第1行(列)→读取第2行(列) →读取第3行(列) → ……，然后快速循环这个过程，最终实现所有按键同时检测的效果。再按下按键1-按键4时，进行阈值的更改，在按下按键5时，将当前的阈值信息传给PC主机。

3.1.6 直流电机与蜂鸣器

重要代码分析：

if(t > tmp\_ )

{

    DCOUT1=0;

    DCOUT2=1;

    Delay\_ms(1);

    SPK=!SPK;

}

else

{

    DCOUT1=0;

    DCOUT2=0;

    }

DCOUT1和DCOUT2为直流电机的两个口，当当前温度（t）大于阈值（tmp\_）时，电机两侧产生电压差，电机开始转动，同时蜂鸣器（SPK）开始鸣叫。

3.1.7 光敏传感器与LED模块

重要代码分析：

if(light\_-light>15)

{

    LED1=0;

    LED2=0;

    LED3=0;

}

else if(light\_-light>10 && light\_-light<=15)

{

    LED1=0;

    LED2=0;

    LED3=1;

}

else if(light\_-light>0 && light\_-light<=10)

{

    LED1=0;

    LED2=1;

    LED3=1;

}

else{

    LED1=1;

    LED2=1;

    LED3=1;

}

LED1,LED2,LED3为补光灯的三个灯，在当前光照（light）小于阈值（light\_）时，LED开始亮起，低于10开启两个LED，低于15开启三个LED。