湖南大学电气与信息工程学院

本科生课程设计

题 目：简易灯光节能控制器

课 程：微机应用系统综合设计

专 业：电气工程及其自动化

班 级：电自1902

学 号：201907010226

姓 名：黄嘉祥

同组人姓名：胡家豪

指导 老师：张志文

设计 时间：

前言

现如今很多学校的教室在白天室内照度很高的情况下，仍然存在开灯现象；或者夜间许多教室，即使仅有几个学生在教室自习，但室内照明全部开启。这种看似有意无意的小事，实际上是对资源的一种浪费。

本文介绍了简易灯光节能控制器及其原理，提出了较为有效的节能控制方法。

## 目 录

一、任务及设计要求

二、设计实现功能

三、系统原理框图

3.1 液晶显示模块

3.2 移位寄存器 74HC595 驱动数码管

3.3 光照传感器 BH1750 / GY-30模块

3.4 PWM调节

四、硬件设计及说明

4.1 具体元器件清单

4.2 硬件电路模块图及功能说明

五、程序流程图设计及程序模块功能描述

六、系统操作说明

七、心得体会

八、参考文献

九、附录（代码、图片）

一、任务及设计要求

简易灯光节能控制器

（1）实时显示光照强度；

（2）测试环境光线变暗，则启动灯光照明，环境光线变亮，则关闭灯光照明，实现光照节能。照明灯光开启、关闭可用LED模拟；

（3）环境光照自动多级调节功能，可依据环境光线的强弱实现多盏灯的开启、关闭有 级控制，或可对单个灯的亮暗进行PWM无级调节；

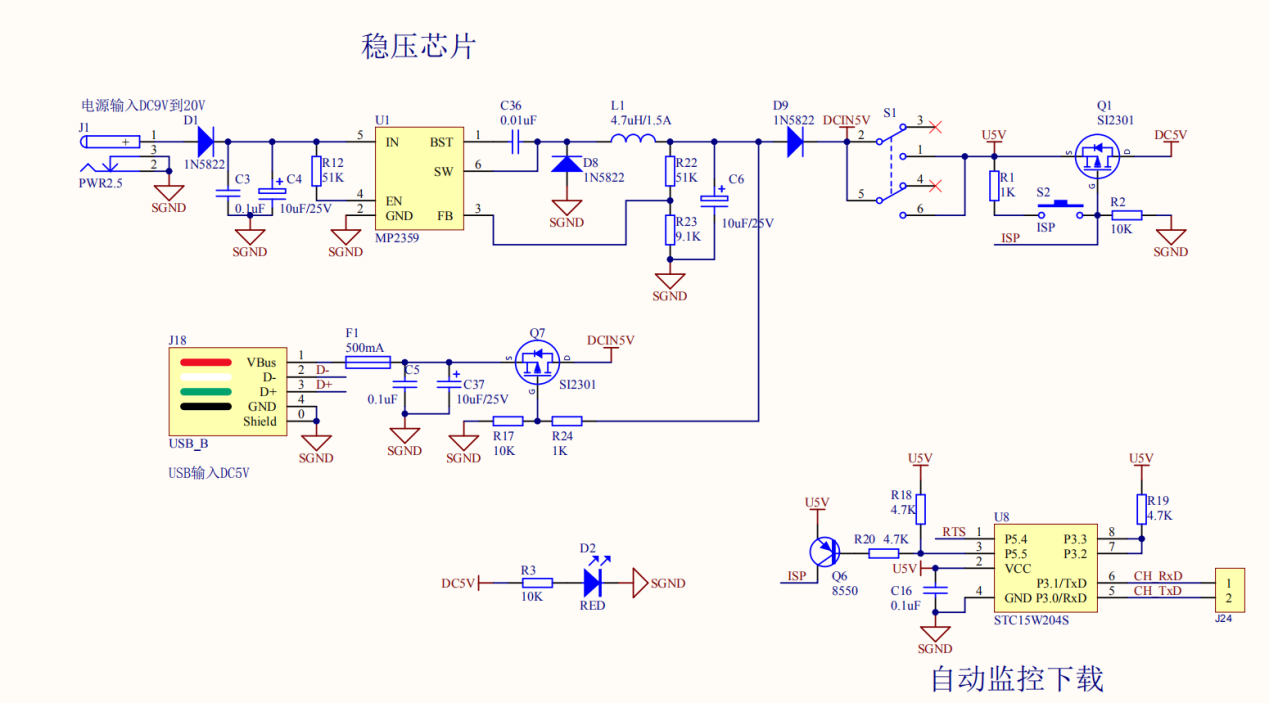
（4）光照度控制阀值可预设；

（5）具有简单电子表功能，可显示年、月、日、星期、时、分、秒等（根据实际显示方式的不同，可选择部分或全部功能）；

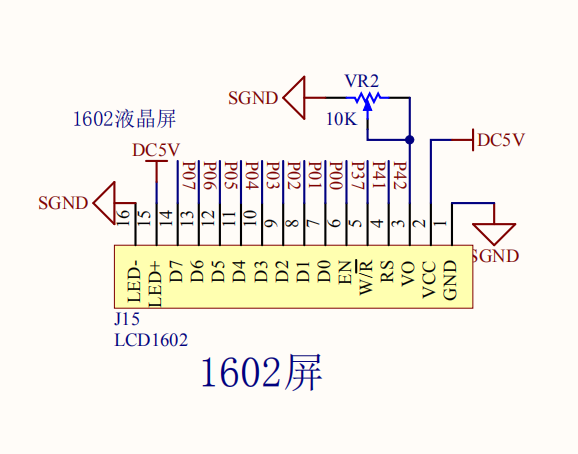
（6）具有电子表的校准功能；

二、设计实现功能

三、系统原理框图

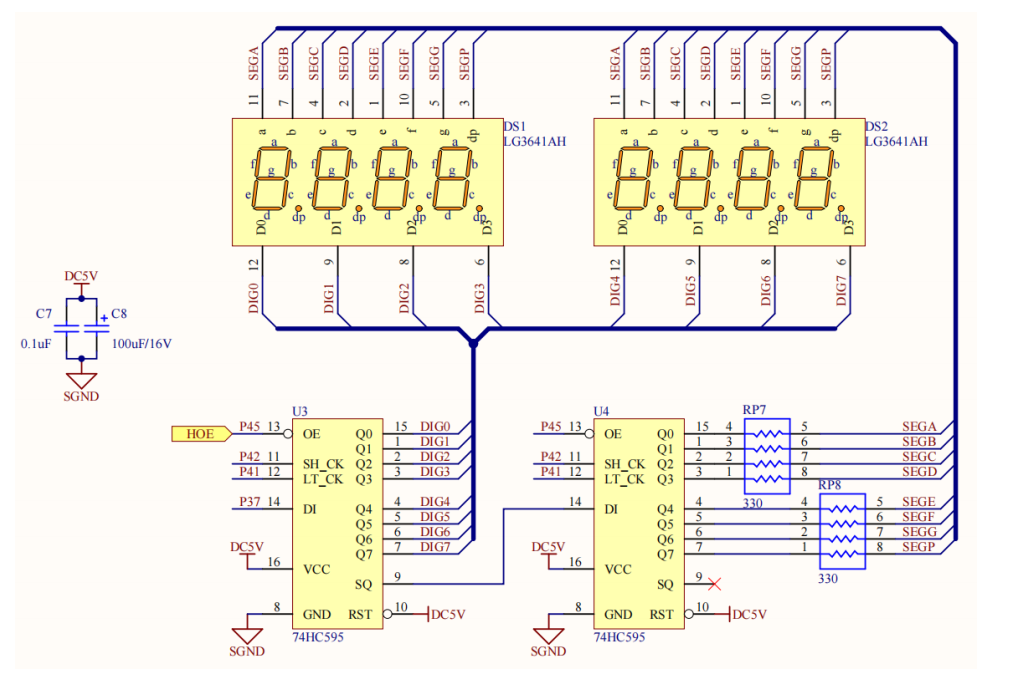


3.1液晶显示模块LCD1602

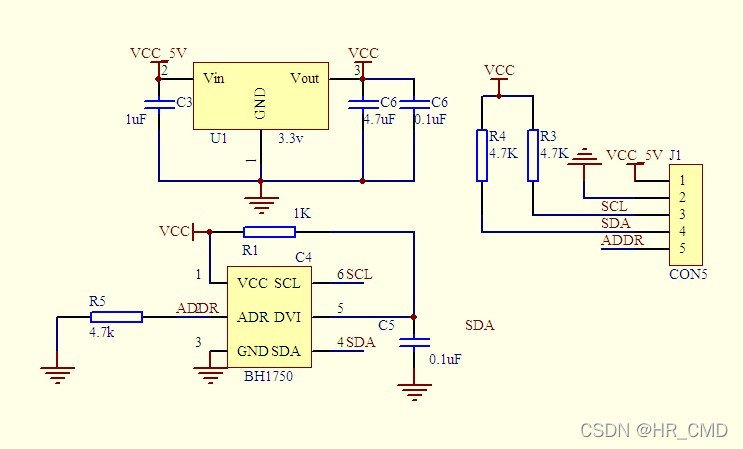


3.2移位寄存器 74HC595 驱动数码管

2 片 74HC595 级联即可驱动 2 个 4 位数码管



3.3光照传感器 BH1750 / GY-30模块



3.4 PWM调节

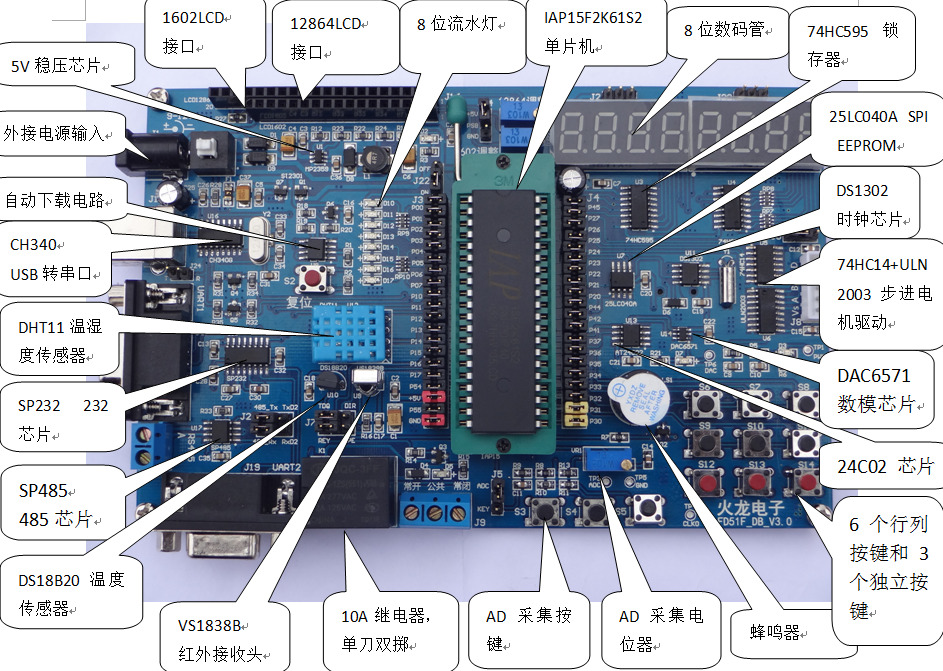
四、硬件设计及说明

1、具体元器件清单

开发板1、LCD1602显示屏、光照传感器模块

2、硬件电路模块及功能

（1）开发板1简介



1）板载一个USB转串口芯片CH340G，直接通过USB线将开发板连接到电脑，可直接对开发板供电和下载程序，方便使用。

2）全自动下载，板载STC15W204S八脚单片机，可监控下载软件发出的下载命令流指令，自动控制板子冷启动，实现全自动下载，该方式为全硬件方式，不占用IAP15F2K61S2单片机任何资源。

3）板载两片74HC595锁存器芯片，用于驱动8位数码管显示，可节约单片机管脚。

4）板载一个DHT11温湿度传感器，可测量温度和相对湿度，该传感器为8位温度和湿度，温度分辨率1℃，湿度分辨率±1%RH,该芯片通过单总线与单片机通信。

5）板载一路74HC14+ULN2003步进电机驱动，可驱动小型步进电机电机，非常方便地进行步进电机驱动控制。同时该接口也可以输出STC15单片机内部三路PWM波。

6）板载一个10位DAC数模转换芯片DAC6571，可直接通过电压表测量DAC输出测试点了解输出的电压情况。该芯片为超小型封装，通过I2C接口与单片机通信。

7）板载一个12位高精度的数字温度传感器DS18B20，可直接读取温度值，并且在数码管上实时显示当前温度。

8）板载一个红外接收头芯片VS1838B，可配合随开发板配送的迷你遥控器进行红外遥控实验，并且在数码管上实时显示接收到的遥控码。

9）板载一个实时钟芯片DS1302，带后备电池，开发板断电也可保存时钟信息

10）板载一个I2C接口的EEPROM芯片AT24C02，可用于保存系统常用的数据，该EEPROM断电可依旧保存数据不丢失。

11）板载一个SPI接口的EEPROM芯片25LC040A，可用于保存系统常用的数据，用于测试STC15单片机内部的SPI硬件接口。

12）板载一路10位ADC模数转换接口，可直接通过调整电位器改变模拟输入的电压大小，可通过数码管显示测得的模拟电压，可通过测试点直接测量电压大小。

13）板载一个RS232转换芯片SP232，两路DB9串口输出，通过DB9串口线可连接到电脑的串口，实现与PC机的串口通信，使用灵活。

14）板载一个RS485转换芯片SP485，可进行多个开发板的RS485通信实验，该485使用的是单片机的串口2进行通信。

15）板载一路220V/10A继电器，单刀双掷，一组常开，一组常闭。

16）板载3个独立按键，可做独立键盘扫描实验，同时可作为其他实验的输入设备。

17）板载2x3行列按键，可做行列键盘扫描实验，同时可作为其他实验的输入设备。

18）板载三个AD采集的按键，该三个按键通过单片机的内部AD来采集确定按键值，只需一个端口可采集三个按键。

19）板载一路PWM控制产生DAC电路，通过PWM来驱动RC电路从而产生DAC模拟电压，用于对精度要求不高，低成本的场合。

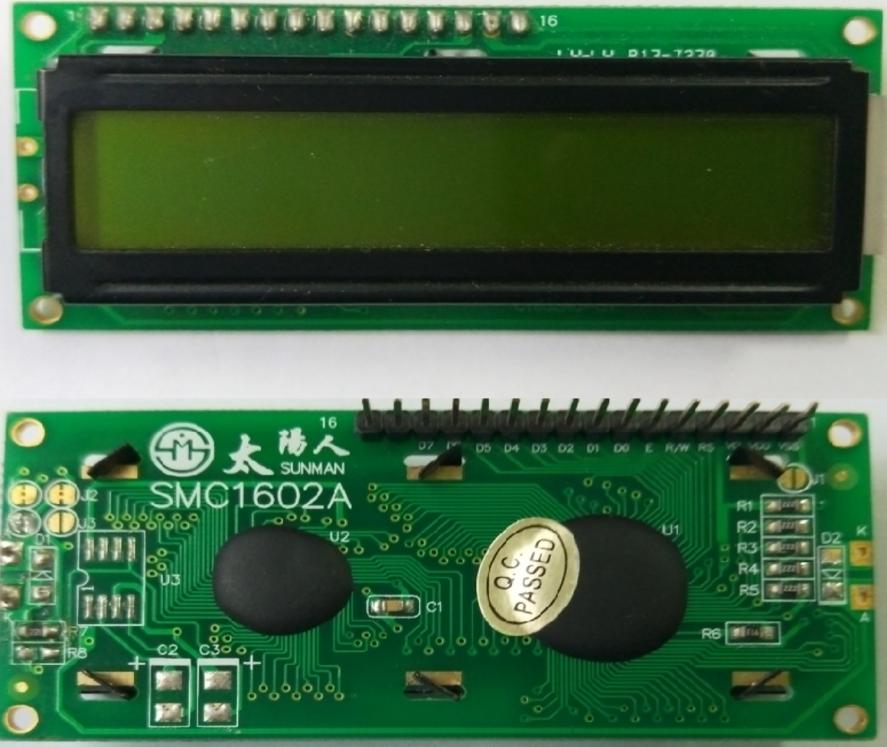
20）板载8位LED流水灯，可做流水灯实验，定时闪烁实验等，使用端口直接驱动。

21）板载1个有源蜂鸣器，可做报警实验等。

22）板载1个开关稳压芯片MP2359，该芯片为开关型稳压芯片，效率高，不发烫，支持宽输入电压9V~20V，推荐操作的输入电压范围9~12V。

23）板上还预留了1602和12864液晶屏接口，可进行液晶屏显示字符和图片实验。

（2）LCD1602显示屏



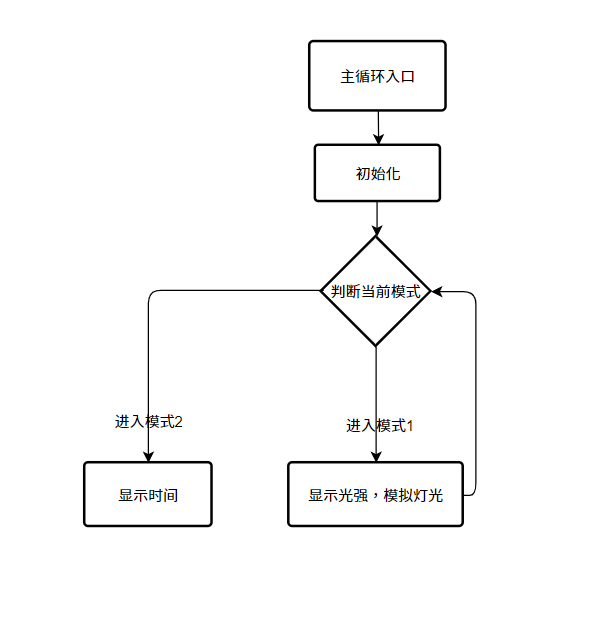
（3）光照传感器模块



利用光电转换，将光照强度值转化为电压值，再经调理电路将此电压值转换为0~2V或4~20V

五、程序流程图设计及程序模块功能描述

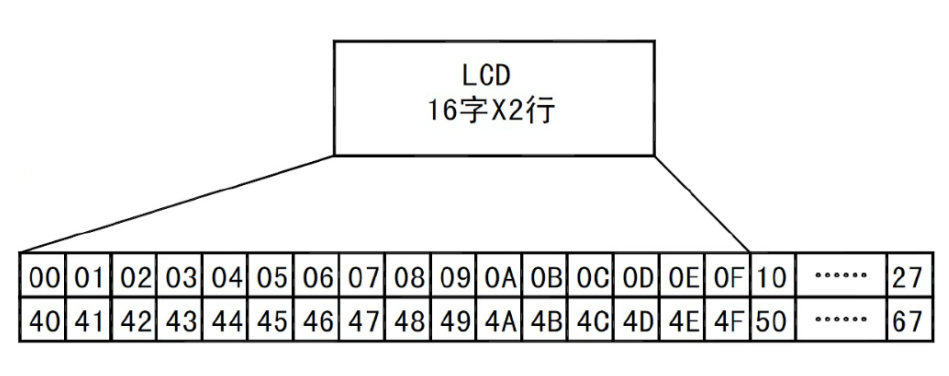
程序流程图



**1.液晶显示模块LCD1602**

液晶屏幕总共 2 行，每行可以显示 16 个字符，总共 32 个字符

根据 1 个字节确定位置 2^8 > 32



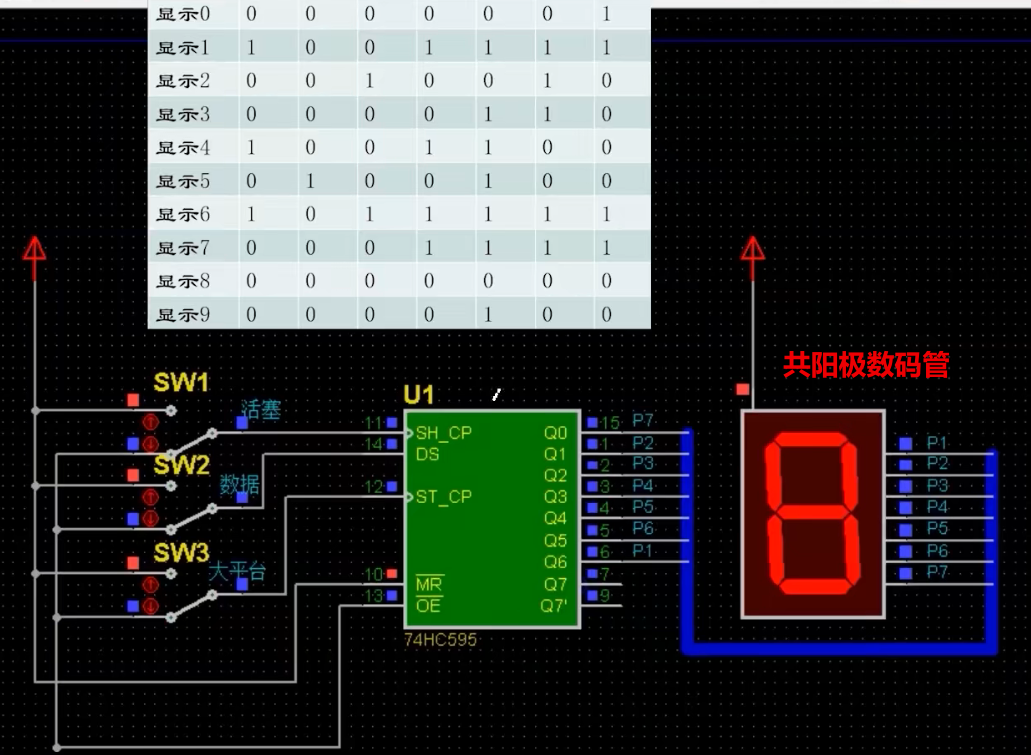
根据另 1 个字节确定字符内容

根据 RS 引脚的电平状态，确定所接收字节的类型

IMG_256

**2.光照传感器 BH1750 / GY-30模块**

**3.移位寄存器 74HC595 驱动数码管**



主要用于拓展 IO 口，用 3 个引脚控制 8 个引脚

两片 74HC595 级联即可驱动 2 个 4 位数码管

**4.**PWM **调节，思路简单，不再赘述，详见代码**

1. 系统操作说明

|  |
| --- |
| 设备共分为两个模式  上电后默认即进入模式1，此模式下可以实时显示环境光强，  并且根据当前光强实时调整灯（数码管模拟灯）的亮度  从而达到省电的效果。  按下s14即切换为模式2，在此模式下可以实时显示当前时间。  再次按下s14 回到模式1，如此反复。  在不同模式切换时，时钟总是保持转动。 |

七、心得体会

在本次课设过程中，我遇到过许多的问题，既有硬件问题，也有代码上的问题。在老师与网络的帮助下，我成功地将它们解决。现摘录如下：

**问题1**：GY-30(bh1750)模块不同引脚对于电平的要求不同，需要对某些引脚额外增加上拉电阻。见附图1

**问题2：**不同单片机执行速度不相同，本次课设所使用的开发板为1T，速度比常规12T单片机执行速度快10倍。当进行iic通信以及1602显示时，尤其需要考虑，这一点。建议使用 ISP 生成 1T MCU 的软件延时程序。

**问题3：**IIC 通讯时发送应答时间要确定（详见代码）

**问题4：** 此次课设要求使用led灯进行模拟，由于显示模块1602已经占用P0 口，发生冲突，于是采用数码管进行模拟。

### **问题5：**无论是使用 1602 还是 12864，P42 引脚都会与控制数码管的 595 发生。于是我采用分时复用，并且记录引脚状态，从而解决冲突。但这也带来一个问题，造成了性能的浪费。

八、参考文献

<https://www.bilibili.com/video/BV1dt411D7nC>

<https://www.bilibili.com/video/BV1hb4y1m7k3>

<https://www.bilibili.com/video/BV12t4y1X7xk>

<https://baike.baidu.com/item/74HC595/9886491>

<https://blog.csdn.net/qq_51272949/article/details/120022849>

<https://zhuanlan.zhihu.com/p/503219395>

<https://blog.csdn.net/yueniaoshi/article/details/52854339>

[1] 丁向荣，陈崇辉 单片机原理与应用 清华大学出版社 2015

[2] 何立民 单片机高级教程——应用与设计 北京航空航天大学出版社 2000

[3] 李嗣福 计算机控制基础 中国科技大学出版社 2001

1. 附录：

**代码部分**

主程序 lamp.c

|  |
| --- |
| #include <stc15f2k60s2.h>  #include <intrins.h> // 调用nop  #include <delay.h> // 软件延时  #include <lcd1602.h> // 液晶显示屏  #include <bh1750.h> // 光传感器  #include <hc595.h> // 数码管  #include <key.h> // 独立按键、外部中断、扫描  unsigned int lux; // 当前光照强度（单位勒克斯）  unsigned int Status = 0; // 暂存 P4 状态，io 分时复用  unsigned int ratio = 1; // PWM 占空比  long int time = 2021400; // 初始化时间（23 日 9点 30 分 0 秒）  // 当前累计时间（单位秒）  // 长整型（-2147483648~ 2147483647）  // 全局变量可以在模式切换时保存时间  unsigned char second;  unsigned char minute;  unsigned char hour;  unsigned char day;  unsigned char d=1; // 方向（前进1、后退-1、暂停0）  unsigned char s=1; // 步长（1、60、3600）  void AdjustLamp()  {  if (lux > 4000)return;  Seg\_Single\_Display(0,0x80);  if (lux > 2200)  {  ratio = 1;  }  else if (lux > 1200)  {  ratio = 3;  }  else if (lux > 500)  {  ratio = 4;  }  else if (lux > 300)  {  ratio = 6;  }  else if (lux > 150)  {  ratio = 8;  }  else  {  ratio = 10;  }  Delay\_us(500 \* ratio);  }  void Timer0\_Init(void) // 每隔 20ms 中断 1 次  {  AUXR &= 0x7F; // 定时器时钟12T模式  TMOD &= 0xF0; // 设置定时器模式  TL0 = 0x00; // 设置初值  TH0 = 0xB8;  TF0 = 0; //清除TF0标记  ET0=1; //使能T0中断  EA=1; //使能总中断  TR0=1; //运行T0  }  // 中断服务程序 interrupt service routine  void Timer0\_ISR(void) interrupt 1  {  static unsigned char cnt=0; // 静态数据类型，用作计数器  if(flag == 1) // 模式1：光强检测、模拟调光  {  HC595\_OE = 0; // 打开 595  Status = P4; // 存入状态  AdjustLamp();  P4 = Status; // 取出状态  HC595\_OE = 1; // 关闭 595  }  // 时间总是在流逝，不论当前是什么模式  if(cnt++ == 50) //定时 20ms\*50 = 1s  {  cnt=0;  time = time + d\*s;// 暂未实现调节时间... 保留在这  }  }  void conversion(unsigned int t) // 获取各位数据（个十百千万）  {  // 1+0x30 = 0x31 = 49，在 ASCII 表中对应中字符 '1'  ARR[4]=t/10000+0x30;  t=t%10000;ARR[3]=t/1000+0x30;  t=t%1000;ARR[2]=t/100+0x30;  t=t%100;ARR[1]=t/10+0x30;  t=t%10;ARR[0]=t+0x30;  }  void processTime()// 处理时间数据，例如100000秒 = 1天3小时46分40秒  {  second = time%60; // 得秒  minute = time/60%60; // 得分  hour = time/3600%24; // 得时  day = time/3600/24%30; // 得天  }  void main()  {  while(1) // 大循环  {  unsigned char i;  unsigned int num;  unsigned char str[] = "light:";    Delay\_ms(100);  Init\_BH1750();  LCD\_Init();  HC595\_Init();  Timer0\_Init(); // 初始化定时器  ExterInt0\_Init(); // 初始化外部中断  while(flag == 1) // 模式1：光强检测、模拟调光  {  Single\_Write\_BH1750(0x01); // 上电命令 power on  Single\_Write\_BH1750(0x10); // 连续高分辨测量 H- resolution mode，测量时间一般为120ms    Delay\_ms(180); // 等待测量事件结束，适当地延时180ms，避免浪费时间    Multiple\_Read\_BH1750(); // 连续读出数据，存储在 BUF 中    num=BUF[0];  num=(num<<8) + BUF[1]; // 拼接数据    lux = (float)num/1.2; // 按照公式换算  conversion(lux); // 计算数据和显示    for(i=0;i<=5;i++) // 显示文本  {  LCD\_Display(i,0,str[i]);  }  for(i=6;i<=10;i++)  {  LCD\_Display(i,0,ARR[10-i]); // 显示数据  }  LCD\_Display(i+1,0,'L'); // 显示单位  LCD\_Display(i+2,0,'X');  }  HC595\_Init(); // 重新初始化数码管  LCD1602\_Write\_Cmd(0x01,1); // 显示清屏  while(flag != 1) // 模式2：时钟  {  processTime(); // 更新秒、分、时、天数据  Seg\_Multiple\_Display(day\*100 + hour,minute\*100 + second);  }  }  } |

延时函数头文件 delay.h

|  |
| --- |
| // 延时微秒 适用于 1T 的 15f2k60  void Delay\_us(unsigned int us)  {  while(us --)  {  \_nop\_();  \_nop\_();  \_nop\_();  }  }  // 延时毫秒 适用于 1T 的 15f2k60  void Delay\_ms(unsigned int ms)  {  unsigned char i, j;  while(ms--)  {  \_nop\_();  \_nop\_();  \_nop\_();  i = 11;  j = 190;  do  {  while (--j);  } while (--i);  }  } |

液晶显示头文件 1602.h

|  |
| --- |
| #define DataPort P0 // LCD1602 数据端口  sbit RS=P4^2;  // 注意 P42 与 hc595 发生冲突  sbit RW=P4^1;  sbit EN=P3^7;  unsigned char BUF[8]; // 定义一个数组，作为缓存区  unsigned char ARR[5]; // 存放 ASCII 编码  void WaitForEnable(void)  {  DataPort=0xff;  RS=0;  RW=1;  \_nop\_();  EN=1;  \_nop\_();  \_nop\_();  while(DataPort&0x80);  EN=0;  }  void LCD1602\_Write\_Cmd(unsigned char cmd,unsigned char Attribc)  {  if(Attribc)WaitForEnable(); // 这一步的目的是什么............    RS=0; // 寄存器选择，低电平说明正在写指令  RW=0; // 始终保持低电平  \_nop\_(); // 微秒级延迟  DataPort=cmd; // 写入一个字节  \_nop\_();  EN=1; // 使能信号  \_nop\_();  \_nop\_();  EN=0;  }    void LCD1602\_Write\_Dat(unsigned char dat)  {  WaitForEnable();  RS=1; // 高电平说明正在写数据，后续步骤完全相同  RW=0;  \_nop\_();  DataPort=dat;  \_nop\_();  EN=1;  \_nop\_();  \_nop\_();  EN=0;  }  void Lcd\_Init() // 初始化1602  {  LCD1602\_Write\_Cmd(0x38,1); // 设置显示模式  LCD1602\_Write\_Cmd(0x08,1); // 显示关闭  LCD1602\_Write\_Cmd(0x01,1); // 显示清屏  LCD1602\_Write\_Cmd(0x06,1); // 显示光标移动  LCD1602\_Write\_Cmd(0x0c,1); // 显示开及光标设置  }  void LCD\_Display(unsigned char X,unsigned char Y,unsigned char DData)  {  Y&=1;  X&=15;  if(Y)  X|=0x40;  X|=0x80;  LCD1602\_Write\_Cmd(X,0);  LCD1602\_Write\_Dat(DData);  } |

光传感器头文件 bh1750.h

|  |
| --- |
| sbit SCL=P1^5; // IIC 时钟引脚  sbit SDA=P1^6; // IIC 数据引脚  #define SlaveAddress 0x46  /\*  定义器件在IIC总线中的从地址，ADD 引脚接地时地址为 0x46，接电源时地址为 0xB8  从设备地址用来区分总线上不同的从设备，因为因为有时候同一根时钟线和数据线可能会连接多个从设备  一般发送从设备地址的时候会在最低位加上读/写信号  比如设备地址为 0x50，则读数据就会发送 0x50，写数据就会发送 0x51  \*/  // 起始信号  void BH1750\_Start()  {  SDA = 1; //拉高数据线  SCL = 1; //拉高时钟线  Delay\_us(5);  SDA = 0; //数据线由高到低，产生下降沿  Delay\_us(5);  SCL = 0; //拉低时钟线，开始传输数据  }  // 停止信号  void BH1750\_Stop()  {  SDA = 0;  SCL = 1;  Delay\_us(5);  SDA = 1; //数据线由低到高，产生上升沿  Delay\_us(5);  }  // 想要接收数据，需要向发送方发送应答  void BH1750\_SendACK(bit ack)  {  // ack = 0: ACK  // ack = 1: NAK  SDA = ack; // 写应答信号  SCL = 1;  Delay\_us(5);  SCL = 0;  Delay\_us(5);  }  // 想要发送数据，需要等待接收方的应答  void BH1750\_RecvACK()  {  SCL = 1;  Delay\_us(5);  // CY = SDA; // 读应答信号 ，可有可无  // CY 溢出位，在此处作为存储应答信号的全局变量  SCL = 0;  Delay\_us(5);  }  // 向 IIC 总线发送一个字节数据  void BH1750\_SendByte(unsigned char dat)  {  unsigned char i;  for (i=0; i<8; i++)  {  dat <<= 1; // 移出数据的最高位，会影响溢出位 CY  SDA = CY; // 把 CY（即 dat 最高位）送数据口  SCL = 1; // 拉高时钟线，此时数据线状态必须保持稳定  Delay\_us(5);  SCL = 0; // 只有在时钟线为低电平期间，数据线状态才允许变化  Delay\_us(5);  }  BH1750\_RecvACK(); // 等待接收方的应答  }  // 从 IIC 总线接收一个字节数据  unsigned char BH1750\_RecvByte()  {  unsigned char i;  unsigned char dat = 0;  SDA = 1; // 必须上拉，为什么.................  for (i=0; i<8; i++)  {  dat <<= 1;  SCL = 1; // 拉高时钟线，此时数据线状态必须保持稳定  Delay\_us(5);  dat |= SDA; // 读数据  SCL = 0; // 只有在时钟线为低电平期间，数据线状态才允许变化  Delay\_us(5);  }  return dat;  }  void Single\_Write\_BH1750(unsigned char REG\_Address)  {  BH1750\_Start(); // 发送起始信号  BH1750\_SendByte(SlaveAddress); // 主设备发送设备地址 + 写信号 0  BH1750\_SendByte(REG\_Address); // 内部寄存器地址... 难道不是指令吗？ 比如上电 0x01  BH1750\_Stop(); // 发送停止信号  }  // 连续读出 BH1750 的内部数据（ 2 个字节）  void Multiple\_read\_BH1750(void)  {  unsigned char i;  BH1750\_Start(); // 发送起始信号  BH1750\_SendByte(SlaveAddress + 1); // 发送设备地址 + 读信号 1    for (i=0; i<2; i++) // 连续读取 2 个地址数据，存储在 BUF 中  {  BUF[i] = BH1750\_RecvByte(); // BUF[0]存储 0x32 地址中的数据，为什么是 0x32...... 不是很懂  if (i == 1)  {  BH1750\_SendACK(1); // 最后一个数据需要回 NOACK，接收的数据只有2个字节，结束通讯  }  else  {  BH1750\_SendACK(0); // 回应一个 ACK  }  }  BH1750\_Stop(); // 发送停止信号  Delay\_ms(5);  }  //初始化BH1750  void Init\_BH1750()  {  Single\_Write\_BH1750(0x01); // 上电命令  } |

数码管及移位寄存器头文件 hc595.h

|  |
| --- |
| sbit HC595\_DAT = P3^7; // 串行数据线 - 14  sbit HC595\_SCK = P4^2; // 移位时钟线 - 12，相当于活塞  // 注意 P42 与 LCD1602 发生冲突  sbit HC595\_RCK = P4^1; // 锁存时钟线 - 11，相当于平台  sbit HC595\_OE = P4^5; // 输出允许线 - 13  // 共阴数码管段码表，存入映射关系  unsigned char code DispCode[]={  0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,  // 0 1 2 3 4 5 6 7 8 9  0x77,0x7C,0x39,0x5E,0x79,0x71,0x3D,0x76,0x74,0x30,  // A b C d E F G H h I  0x10,0x1E,0x38,0x54,0x5C,0x73,0x67,0x50,0x31,0x78,  // i J L n o P q r R t  0x3E,0x1C,0x40,0x48,0x08,0x00};  // U V 一 二 \_ 灭  unsigned char DispBuf[8]; // 显示缓冲区  void HC595\_SendWord(unsigned char dat,unsigned char dig) // dat 传入段码，dig 传入位码  {  unsigned char n;  unsigned int temp=0;  temp=dat;  temp=temp<<8 | dig; // 8bit + 8bit = 16bit  for(n=0;n<16;n++)  {  HC595\_SCK = 0; // 置低时钟线，相当于活塞复位  if((temp&0x8000)==0x8000) // 从高到低串行发送  {  HC595\_DAT = 1; // 置高信号线，准备一个 1  }  else  {  HC595\_DAT = 0; // 置低信号线，准备一个 0  }  HC595\_SCK = 1; // 置高时钟线，相当于活塞前推  temp<<=1;  }    HC595\_RCK = 0;  HC595\_RCK = 1;  // 锁存信号上升沿锁存数据，数据输出到端口  // 将平台上的数据一次性推出  }  void Seg\_Single\_Display(unsigned char Dat,unsigned char Dig )  {  HC595\_SendWord(DispCode[Dat],~Dig);  // 若 Dig = 0x80 = 1000 0000，则 ~Dig = 0111 1111，故 D7 亮  }  void Seg\_Multiple\_Display(unsigned int High\_Dat,unsigned int Low\_Dat) // 选择 unsigned int 而不是 unsigned char（-128~127）  {  unsigned char i;  DispBuf[0]=High\_Dat/1000; //High\_Dat的千位  DispBuf[1]=High\_Dat%1000/100; //High\_Dat的百位  DispBuf[2]=High\_Dat%100/10; //High\_Dat的十位  DispBuf[3]=High\_Dat%10; //High\_Dat的个位  DispBuf[4]=Low\_Dat/1000; //Low\_Dat的千位  DispBuf[5]=Low\_Dat%1000/100; //Low\_Dat的百位  DispBuf[6]=Low\_Dat%100/10; //Low\_Dat的十位  DispBuf[7]=Low\_Dat%10; //Low\_Dat的个位    for(i=0;i<8;i++)  {  HC595\_SendWord(DispCode[DispBuf[i]],(~(0x01<<i)));  HC595\_OE = 0; // 打开显示  Delay\_ms(1); // 适当延时，利用视觉残留  HC595\_OE = 1; // 关闭显示  }  }  void HC595\_Clear(void)  {  unsigned char i;    for(i=0;i<8;i++) // 清除显示缓冲区内容  {  DispBuf[i]=35; // DispCode[35] = 0x00，即熄灭  }  HC595\_SendWord(0x00,0xff);  // 位码端低电平，对应的数码管才能点亮  }  void HC595\_Init(void)  {  //等待HC595上电稳定  Delay\_ms(10);    //初始化P41,P42,P45口为准双向口  P4M1 &=~( (1<<1) | (1<<2) | (1<<5));  P4M0 &=~( (1<<1) | (1<<2) | (1<<5));    //初始化P37口为准双向口  P3M1 &=~(1<<7);  P3M0 &=~(1<<7);    //将端口置低，赋予端口固定电平  HC595\_DAT = 0;  HC595\_SCK = 0;  HC595\_RCK = 0;  //打开显示  HC595\_OE = 0;  //清除显示内容  HC595\_Clear();  } |

按键及外部中断头文件 key.h

|  |
| --- |
| // 标记状态，用于切换模式  bit flag = 1;  void ExterInt0\_Init(void)  {  P3M1 &=~(1<<2); // 双向口为什么要初始化？怎么初始化呢？  P3M0 &=~(1<<2); // P32 端口初始化  IT0=1; // 设置为下降沿触发  EX0=1; // 开外部中断  EA=1; // 开总中断  }  // 中断服务程序 interrupt service routine  void ExterInt0\_ISR(void) interrupt 0  {  if(P3^2==0) //检测中断端口，判断是否确实产生中断  {  Delay\_ms(2); //适当延时  if(P3^2==0) //再次判断是否确实产生中断  {  flag = ~flag; // 切换状态  }  }  } |

**附图**

