

# 单片机第五六八次实验

学号 21160806

姓名 邓 艳 红

# 实验五 重量测量（液晶显示）

## 一、原理总结

- （该实验涉及的基本原理及其在实验中的使用方法）
- 1、在液晶显示中,自定义图形和文字的字模对应的字节表需要使用专门的字模软件来生成。可以使用 PCtoLCD2002 字模软件提取。
- 2、字符点阵等数据,需要定义在 code 数据段中,具体原理参见示例程序设计部分。
- 3、向 LCM 输出一个命令或数据时,应当在选通信号为高时准备好数据,然后延迟若干指令周期,再将选通信号置为低。
- 4、与 A/D 转换相关的寄存器
- ADC\_POWER: ADC 电源控制位, 0 关 1 开。
- SPEED1,SPEED0: 模数转换器速度控制位, 控制 A/D 转换所需时间。
- ADC\_FLAG: 模数转换结束标志位, AD 转换完后, ADC\_FLAG=1, 一定要软件清 0。
- ADC\_START: 模数转换器 (ADC) 转换启动控制位, 1 开始转换, 转换结束后为 0。
- CHS2/CHS1/CHS0: 模拟输入通道选择, 选择使用 P1.0~P1.7 作为 A/D 输入。
- ADC\_RES、ADC\_RES1: A/D 转换结果寄存器, 是特殊功能寄存器, 用于保存 A/D 转换结果。
- IE: 中断允许寄存器 (可位寻址)
- EA: CPU 的中断开放标志, EA=1, CPU 开放中断, EA=0, CPU 屏蔽所有中断申请。
- EADC: A/D 转换中断允许位。1 允许 0 禁止。
- IPH: 中断优先级控制寄存器高 (不可位寻址)。
- IP: 中断优先级控制寄存器低 (可位寻址)。
- 5、重量传感器采用压敏电阻。利用压敏电阻采集应变,产生变化的阻值。利用放大电路将其转化为电压值,通过数模转换将电压值转化成 CPU 处理的数字信号。传感器根据编制的程序将数字信号转换为砝码重量显示输出。

### 液晶显示屏硬件说明:

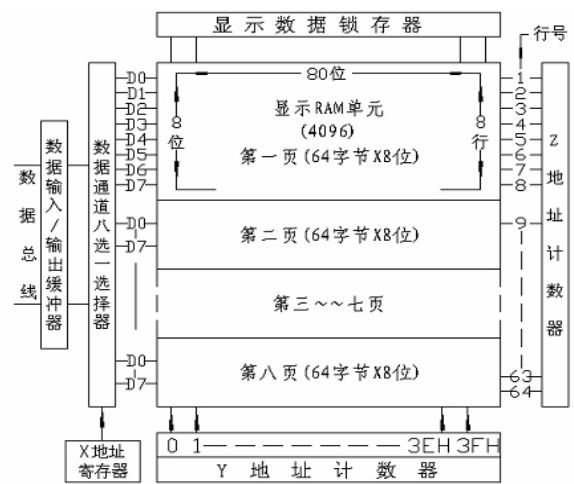
引脚特性如下表所示:

引脚名称	级别	引脚功能描述
CS1	H/L	片选信号, 当/CS1=L 时,液晶左半屏显示
CS2	H/L	片选信号, 当/CS2=L 时,液晶右半屏显示
VSS	0V	电源地
VDD	+5V	电源电压

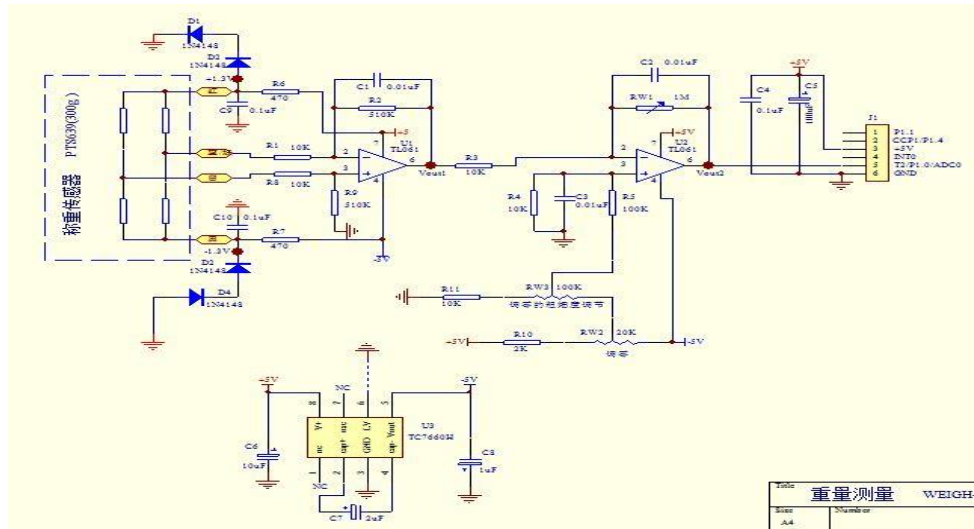
V0	0 至-10V	LCD 驱动负电压，要求 VDD-VLCD=10V
RS	H/L	寄存器选择信号
R/W	H/L	读/写操作选择信号
E	H/L	使能信号
DB0	H/L	八位三态并行数据总线
DB1		
DB2		
DB3		
DB4		
DB5		
DB6		
DB7		
RES	H/L	复位信号，低电平有效
VOUT	-10V	输出-10V 的负电压（单电源供电）
LED+(EL)	+5V	背光电源, Idd≤960mA
LED-(EL)	0V	

YM12864C 的液晶分为左边和右边两个 64×64 的子屏，分别通过 CS1 和 CS2 选通，每个子屏相应的内部寄存器是相互独立的。在一个时刻只能选择一个子屏操作。

LCM 的内部寄存器：



本实验涉及到原理图：



## 二、程序分析

(程序设计的思路、程序代码+注释)

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit CS1=P1^7;///左半边
sbit CS2=P1^6;///右半边
sbit E=P3^3;///使能信号
sbit RW=P3^4;///读写操作选择
sbit RS=P3^5;///寄存器选择(数据/指令)
sbit RES=P1^5;///复位 低电平有效
sbit BUSY=P2^7;

/**Declare SFR associated with the ADC */
sfr ADC_CONTR = 0xBC; ///ADC control register
sfr ADC_RES = 0xBD; ///ADC high 8-bit result register
sfr ADC_LOW2 = 0xBE; ///ADC low 2-bit result register
sfr P1ASF = 0x9D; ///P1 secondary function control register
sfr AUXR1 = 0xA2; ///AUXR1 中的 ADJ 位用于转换结果寄存器的数据格式调整控制

/**Define ADC operation const for ADC_CONTR*/
#define ADC_POWER 0x80 ///ADC power control bit
#define ADC_FLAG 0x10 ///ADC complete flag
#define ADC_START 0x08 ///ADC start control bit
#define ADC_SPEEDLL 0x00 ///540 clocks
#define ADC_SPEEDL 0x20 ///360 clocks
#define ADC_SPEEDH 0x40 ///180 clocks
```

```

#define ADC_SPEEDHH 0x60  ///90 clocks
uchar ch = 0;  ///ADC channel NO.0
uchar code zima[20][32]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x00,///"0"*
0/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x00,///"1"*
1/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x00,///"2"*
2/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x00,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x00,///"3"*
3/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,///"4"*
4/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,///"5"*
5/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x00,///"6"*
6/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,///"7"*
7/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x00,///"8"
*8/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0x00,///"9"*
9/

```

```
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x00,//*"
重"*10/
```

```
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,//*" 量
"*11/
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,//*" 1
2/
```

```
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x00,//*" 克
"*13/
```

```
};
void send_byte(uchar dat ,uchar cs1,uchar cs2);
void send_all(uint page,uint lie,uint offset);
void delay(uint x);
void init_adc();
void init_yejing();
void calibrate();
int get_ad_result();
void clearscren();
int cweight;
int weight;
void main()
{
    init_yejing();
    init_adc();
    calibrate();//校准
    while(1)
    {
        weight=(get_ad_result()-cweight-70)/2;
        weight += weight/10;//真实重量
        clearscren();
        send_all(1,1,10);//重
        send_all(1,2,11);//量
        send_all(1,3,12);//:
        send_all(4,3,weight/100);//百
        send_all(4,4,(weight/10)%10);//十
        send_all(4,5,weight%10);//个
        send_all(4,6,13);//克
    }
}
```

```

        delay(50000);

    }

}

void init_yejing()
{
    send_byte(192,1,1);///设置起始行
    send_byte(63,1,1);///打开显示开关
}

void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;
    while(BUSY);

    ///送数据或控制字
    E=0;
    RS=!(cs1&&cs2),RW=0;
    P2=dat;
    E=1; delay(3); E=0;

    CS1=CS2=0;
}

void send_all(uint page,uint lie,uint offset)
{
    uint i,j,k=0;
    for(i=0;i<2;++i)
    {
        send_byte(184+i+page,1,1);///选择页面
        send_byte(64+lie*16-(lie>3)*64,1,1);///选择列号
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);///送数
    }
}

void init_adc()
{
    P1ASF = 1; ///Set P1.0 as analog input port
    AURX1 |= 0X04; ///AURX1 中的 ADJR 位用于转换结果寄存器的数据格式调整控制

```

```

ADC_RES = ADC_LOW2 = 0; ///Clear previous result

ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch;    ///ch=0 ADC
channel NO.0
delay(4);    ///ADC power-on delay and Start A/D conversion
}

int get_ad_result()
{
    int ADC_result;
    ADC_RES = ADC_LOW2 = 0; ///Clear previous result
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START;
    _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); ///Must wait before inquiry
    while (!(ADC_CONTR & ADC_FLAG)); ///Wait complete flag
    ADC_result = (ADC_RES & 0x03) * 256 + ADC_LOW2; ///ADC_RES 中存高 2 位
    ADC_CONTR &= ~ADC_FLAG; ///Close ADC flag 位置 0
    return ADC_result;    ///Return ADC result
}

void calibrate()
{
    cweight=(get_ad_result()-0)/2;
}

void delay(uint x)
{
    while(x--);
}

void clearsreen()
{
    int i,j;
    for(i=0;i<8;++i)
    {
        send_byte(184+i,1,1);///页
        send_byte(64,1,1);///列
        for(j=0;j<64;++j)
        {
            send_byte(0x00,0,1);
            send_byte(0x00,1,0);
        }
    }
}

```



}

### 三、问题分析

（实验过程中遇到的问题及解决方法）

液晶屏上显示的重量与砝码实际的重量之间存在较大误差

通过一次次调整主函数中 `weight` 的值和调整单片机上的旋钮来使最终的显示结果与实际结果较为接近。

## 实验六 直流电机脉宽调制调速

### 一、原理总结

（该实验涉及的基本原理及其在实验中的使用方法）

1、对于直流电机来说，其转速由输入电压决定，因此具有平滑调速的效果；相比而言，交流电机的转速由交流电频率和电机结构决定，难以改变速度。当然，交流电机构造简单，没有换向器，所以容易制造高转速、高电压、大电流、大容量的电机；而直流电机一般用在负荷小，但要求转速连续可调的场合，如伺服电机。

2、脉宽调制（Pulse Width Modulation, PWM）是一种能够通过开关量输出达到模拟量输出效果的方法。使用 PWM 可以实现频率调制、电压调制等效果，并且需要的外围器件较少，特别适合于单片机控制领域。这里只关心通过 PWM 实现电压调制，从而控制直流电机转速的效果。也称作脉宽调制调速。

3、PWM 的基本原理是通过输出一个很高频率的 0/1 信号，其中 1 的比例为  $\delta$ （也叫做占空比），在外围积分元件的作用下，使得总的效果相当于输出  $\delta \times A$ （ $A$  为高电平电压）的电压。通过改变占空比就可以调整输出电压，从而达到模拟输出并控制电机转速的效果

使用单片机实现 PWM，就是根据预定的占空比  $\delta$  来输出 0 和 1，这里  $\delta$  就是控制变量。最简单的办法就是以某个时间单位（如 0.1ms，相当于 10kHz）为基准，在前  $N$  段输出 1，后  $M-N$  段输出 0，总体的占空比就是  $N/M$ 。这种方法由于 0 和 1 分布不均匀，所以要求基准频率要足够高，否则会出现颠簸现象。

要达到更稳定的效果，可以采用累加进位法如果将总的周期内的 0 和 1 均匀分散开。设置一个累加变量  $x$ ，每次加  $N$ ，若结果大于  $M$ ，则输出 1，并减去  $M$ ；否则输出 0。这样整体的占空比也是  $N/M$ 。在实验中取  $M=256$  可以使程序更加简单。

另外，由于本实验板的设计，输出 0 使电机工作。因此对于本实验，上面所说的 0 和 1 要翻转过来用。

4、在本实验板中，电机每转动一次，与之相连的偏心轮将遮挡光电对管一次，因此会产生一个脉冲，送到 INT0。要测量转速，既可以测量相邻两次中断之间的时间；也可以测量一秒种之内发生的中断次数。显然，后一种方法更加简单。

进行转速控制时，涉及到三个变量：预期转速，实际转速和控制变量。这里控制变量就是占空比。我们并不能够预先精确知道某个控制变量的值会导致多少的实际转速，因为这里有很多内部和外部因素起作用（如摩擦力，惯性等），但可以确定就是随着控制变量的增加，实际转速会增加。

5、反馈控制的基本原理就是根据实际结果与预期结果之间的差值，来调节控制变量的值。当实际转速高于预期转速时，我们需要减少控制变量，以降低速度；反之则需要调高控制变量

6、本实验的转速控制可以使用简单的比例控制算法，也就是当转速  $S$  大于预定值时，将输出 0 的个数减少；当转速小于预定值时，将输出 0 的个数增加。改变值正比于测量出的差值。也可自行使用其他更加复杂的算法。



```

sbit sclk=P4^4;
sbit sdata=P4^5;
sbit swl1=P3^6;
sbit swl2=P3^7;
sbit motor=P1^1;

uchar tab[15]=
    {0xC0,0xF9,0xA4,0xB0,0x99,
     0x92,0x82,0x0F8,0x80,0x90};

uchar tspeed=0;///累加转数
uchar cspeed=0;///当前速度
uchar xspeed=100;///期望速度

uchar t1_cnt=0; ///1s 延时控制变量 50ms*20 次
int N=100;///占空比
int M=256;
int X=0;///起始变量

init();
void sendbyte(uchar ch);
void display(uchar n);
void delay1();
void delay2();
//void ex_int0() interrupt 0 ;
//void t1_int() interrupt 3 ;
//void t0_int() interrupt 1 ;

uchar flag1=0;
uchar flag2=0;

void main()
{
    init();
    motor=0;
    while(1)
    {
        display(cspeed);
        delay2();
        display(xspeed);
        delay1();
    }
}

```

```
    }  
}
```

```
init()  
{  
    P4SW=0x30;  
  
    IT0=1; ///设置 INT0 为边沿触发  
  
    EA=1;  
    ET1=1;  
    ET0=1;  
    EX0=1;  ///中断允许  
  
    TMOD=0x11; ///设置定时器 0 和定时器 1 的工作方式  
    TH1=0x3C;  
    TL1=0xB0;  ///50ms 计数值  
    TH0=0xFF;  
    TL0=0x9C;  ///0.1ms 计数值  
  
    TR0=1;  
    TR1=1;  ///启动定时器  
}
```

```
void ex_int0() interrupt 0  ///外部中断 INT0  
{  
    tspeed++;  
}
```

```
void t1_int() interrupt 3  ///50ms 定时器中断 T1  
{  
    if(++t1_cnt<20)  
    {  
        if(swh1==0)  
        {  
            flag1=1;  
        }  
        if(swh1==1 && flag1==1)  
        {  
            xspeed++;  
            flag1=0;  
        }  
    }  
}
```

```

    }

    if(swh2==0)
        flag2=1;
    if(swh2==1 && flag2==1)
    {
        xspeed--;
        flag2=0;
    }

    return;
}
t1_cnt=0;
cspeed=tspeed;
tspeed=0;
if(cspeed>xspeed) N++;
if(cspeed<xspeed) N--;
}

void t0_int() interrupt 1    ///0.1ms 定时器中断 T0
{
    X+=N;
    if(X>M)
    {
        motor=1; ///不转
        X-=M;
    }
    else
        motor=0; ///转
}

void sendbyte(uchar ch)
{
    uchar shape,c;
    shape=tab[ch];
    for(c=0;c<8;c++)
    {
        sclk=0;
        sdata=shape & 0x80;
        sclk=1;
        shape <<= 1;
    }
}

```

```

void display(uchar n)
{
    sendbyte(n%10);      ///  

    sendbyte((n/10)%10); ///  

    sendbyte(n/100);     ///  

}

```

```

void delay1()
{
    int i,j;
    for(i=0;i<1000;i++)
        for(j=0;j<500;j++);
}

```

```

void delay2()
{
    int i,j;
    for(i=0;i<1000;i++)
        for(j=0;j<1000;j++);
}

```

### 三、问题分析

（实验过程中遇到的问题及解决方法）

实验中转速改变时，增加减少速度慢

解决方案：调整 N/M 的值，即占空比。或调整改变 N 值的函数部分，使 N 值不是依次加一，而是改变成其他值，但会遇到无法稳定在期望转速的问题。

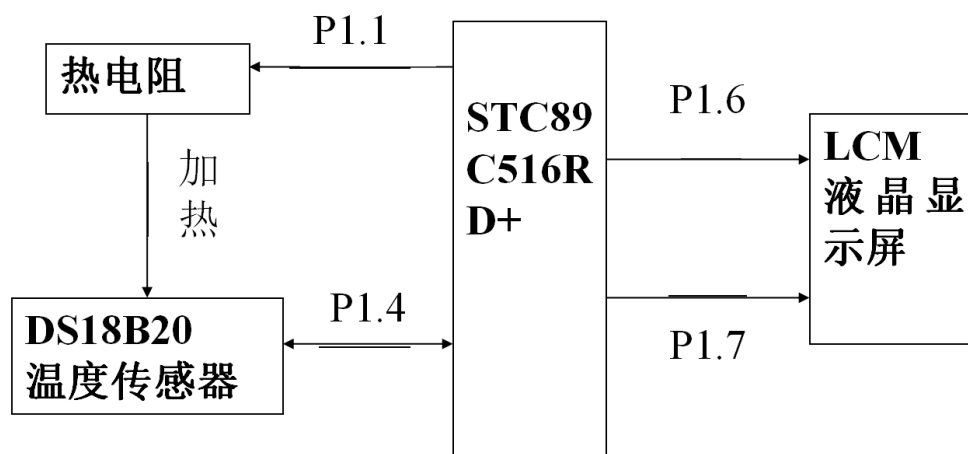
# 实验八温度测量与控制

## 一、原理总结

(该实验涉及的基本原理及其在实验中的使用方法)

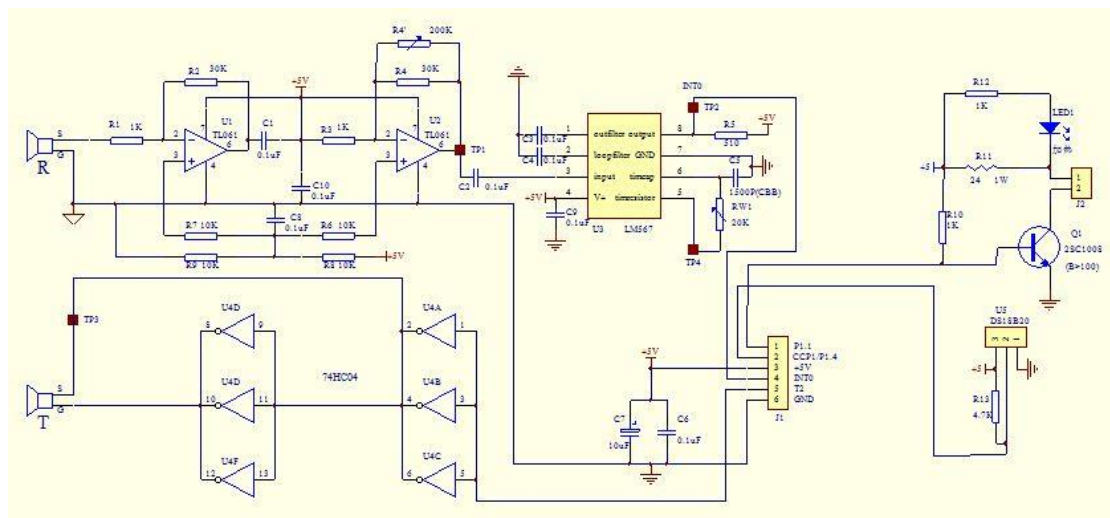
- 1、本实验使用的 DS18B20 是单总线数字温度计，测量范围从 $-55^{\circ}\text{C}$ 到 $+125^{\circ}\text{C}$ ，增量值为 $0.5^{\circ}\text{C}$ 。
- 2、用于贮存测得的温度值的两个 8 位存贮器 RAM 编号为 0 号和 1 号。  
1 号存贮器存放温度值的符号，如果温度为负 ( $^{\circ}\text{C}$ )，则 1 号存贮器 8 位全为 1，否则全为 0。  
0 号存贮器用于存放温度值的补码 LSB(最低位)的 1 表示  $0.5^{\circ}\text{C}$ 。
- 3、将存贮器中的二进制数求补再转换成十进制数并除以 2，就得到被测温度值。
- 4、温度检测与控制系统由加热灯泡，温度二极管，温度检测电路，控制电路和继电器组成。温度二极管和加热灯泡封闭在一个塑料保温盒内，温度二极管监测保温盒内的温度，用温控实验板内部的 A/D 转换器 ADC7109 检测二极管两端的电压，通过电压和温度的关系，计算出盒内空气的实际温度。
- 5、本实验使用 STC89C516RD+单片机实验板。单片机的 P1.4 与 DS18B20 的 DQ 引脚相连，进行数据和命令的传输。
- 6、单片机的 P1.1 连接热电阻。当 P1.1 为高电平时，加热热电阻。
- 7、温度控制的方法采用 PID 控制实现

本实验设备的原理图：





## 本实验示意电路原理图：



## DS18B20 工作过程如下：

初始化--ROM 操作命令--存储器操作命令--处理数据。其中初始化序列过程如下：将 DS18B20 的数据总线拉低 500 微秒以上然后释放，收到信号后等待 16-60 微秒左右，之后发出 60-240 微秒的存在低脉冲，单片机收到此信号表示初始化成功。

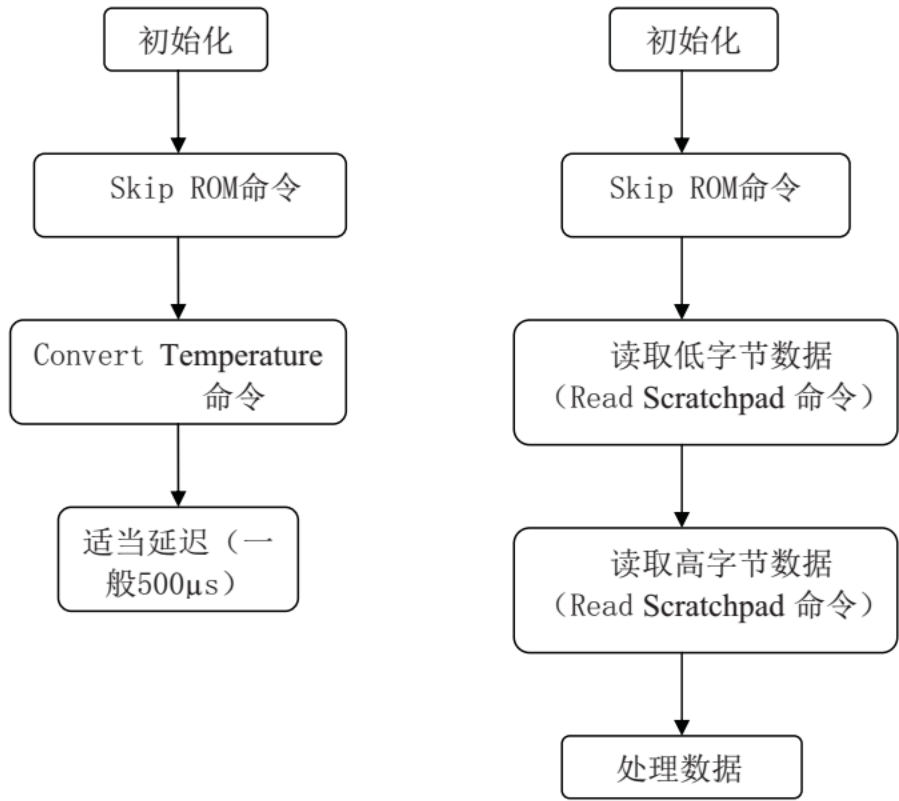
ROM 操作命令有：

指令	代码	解释
Read ROM (读ROM)	[33H]	此命令允许总线主机读DS18B20的8位产品系列编码，唯一的48位序列号，以及8位的CRC。
Match ROM (匹配ROM)	[55H]	此命令后继以64位的ROM数据序列，允许总线主机对多点总线上特定的DS18B20寻址。
Skip ROM (跳过ROM)	[CCH]	在单点总线系统中，此命令通过允许总线主机不提供64位ROM编码而访问存储器操作来节省时间。
Search ROM (搜索ROM)	[F0H]	此命令允许总线控制器用排除法识别总线上的所有从机的64位编码。
Alarm search (告警搜索)	[ECH]	此命令的流程与搜索ROM命令相同。但是，仅在最近一次温度测量出现告警的情况下，DS18B20才对此命令作出响应。

存储器操作命令如下：

命令	代码	解释
Write Scratchpad (写暂存存储器)	[4EH]	此命令向DS18B20的暂存器中写入数据，开始位置在地址2。接下来写入的两个字节将被存到暂存器中的地址位置2和3。
Read Scratchpad (读暂存存储器)	[BEH]	此命令读取暂存器的内容。读取将从字节0开始，一直进行下去，直到第9（字节8，CRC）字节读完。
Copy Scratchpad (复制暂存存储器)	[48H]	此命令把暂存器的内容拷贝到DS18B20的EPROM存储器里，即把温度报警触发字节存入非易失性存储器里。
Convert Temperature (温度变换)	[44H]	这条命令启动一次温度转换而无需其他数据。
Recall EPROM (重新调出)	[B8H]	此命令把贮存在EPROM中温度触发器的值重新调至暂存存储器。
Read Power Supply (读电源)	[B4H]	对于在此命令发送至DS18B20之后所发出的第一读数据的时间片，器件都会给出其电源方式的信号：“0”=寄生电源供电，“1”=外部电源供电。

单片机控制完成温度转换和读取数据的流程如下：



写操作(包括写指令)具体过程如下：

通过单总线采取移位的方式来向 DS18B20 写入数据，按照从低位到高位的方式每次一位的方式写进去，需要满足写时间间隙的要求。在写数据时间间隙的前 15us 数据总线需要是被单片机拉置低电平，而后则将是芯片对总线数据的采样时间，采样时间在 15 -60us，采样时间内如果单片机将总线拉高则表示写“1”，如果单片机将总线拉低则表示写“0”。每一位的发送都应该有一个至少 15us 的低电平起始位，随后的数据“0”或“1”应该在 45u 内完成。整个位的发送时间应该保持在 60- 120uS，否则不能保证通信的正常。

读操作（包括读数据）具体过程如下：

也是通过移位的方法从 DS18B20 中读取数据，按照从低位到高位的方式每次一位的方式读入，需要满足读时间间隙的要求。读时间间隙时控制时的采样时间应该更加的精确才行，读时间间隙时也是必须先由单片机产生至少 1us 的低电平，表示读时间的起始。随后在总线被释放后的 15us 中 DS18B20 会发送内部数据位，这时单片机如果发现总线为高电平表示读出“1”，如果总线为低电平则表示读出数据“0”。

注意，必须在读间隙开始的 15us 内读取数据位才可以保证通信的正确。

### 3. LCM 的使用

YM12864C 是一种图形点阵液晶显示器。它主要采用动态驱动原理由行驱动控制器和列驱动器两部分组成了 128(列) X64(行)的全点阵液晶显示。此显示器采用了 COB 的软封装方式，通过导电橡胶和压框连接 LCD, 使其寿命长，连接可靠。

YM12864C 是全屏幕点阵，点阵数为 128(列) X64(行)，可显示 8(每行)x4(行)个(16X16 点阵)汉字，也可完成图形，字符的显示。与 CPU 接口采用 5 条位控制总线和 8 位并行数据总

线输入输出，适配 M6800 系列时序。内部有显示数据锁存器，自带上电复位电路。

一. 硬件说明

引脚特性如下表所示：

引脚号	引脚名称	级别	引脚功能描述
1	CS1	H/L	片选信号，当/CS1=L时，液晶左半屏显示
2	CS2	H/L	片选信号，当/CS2=L时，液晶右半屏显示
3	VSS	0V	电源地
4	VDD	+5V	电源电压
5	V0	0至-10V	LCD驱动负电压，要求VDD-VLCD=10V
6	RS	H/L	寄存器选择信号
7	R/W	H/L	读/写操作选择信号
8	E	H/L	使能信号
9	DB0	H/L	八位三态并行数据总线
10	DB1		
11	DB2		
12	DB3		
13	DB4		
14	DB5		
15	DB6		
16	DB7		
17	RES	H/L	复位信号，低电平有效
18	VOUT	-10V	输出-10V的负电压（单电源供电）
19	LED+ (EL)	+5V	背光电源, Idd≤960mA
20	LED- (EL)	0V	

二、程序分析

（程序设计的思路、程序代码+注释）

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code zima[20][32]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x00,/*"0"*
0/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x00,/*"1"*
```

1/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x00,  
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x00,///  
"2"

\*2/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x00,  
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x00,///  
"3"

\*3/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x00,  
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,///  
"4"

\*4/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,  
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,///  
"5"

\*5/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00,  
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x00,///  
"6"

\*6/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x00,  
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,///  
"7"

\*7/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x00,  
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x00,///  
"8"

\*8/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,0x00,  
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0x00,///  
"9"

\*9/

0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x00,  
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x00,///  
"10"

重"10/

0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x00,  
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,///  
"11"

\*11/

0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,  
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,///  
":1"

\*1

2/

```
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x00,  
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x00,///  
" 克  
"*13/
```

```
0x10,0x21,0x86,0x70,0x00,0x7E,0x4A,0x4A,0x4A,0x4A,0x7E,0x00,0x00,0x00,0x00,  
0x02,0xFE,0x01,0x40,0x7F,0x41,0x41,0x7F,0x41,0x41,0x7F,0x41,0x41,0x7F,0x40,0x00,///  
" 温  
",14*/
```

```
0x00,0x00,0xFC,0x04,0x24,0x24,0xFC,0xA5,0xA6,0xA4,0xFC,0x24,0x24,0x24,0x04,0x00,  
0x80,0x60,0x1F,0x80,0x80,0x42,0x46,0x2A,0x12,0x12,0x2A,0x26,0x42,0xC0,0x40,0x00,///  
" 度  
",15*/
```

```
};
```

```
sbit CS1=P1^7;///  
sbit CS2=P1^6;///  
sbit E=P3^3;///  
sbit RW=P3^4;///  
sbit RS=P3^5;///  
sbit RES=P1^5;///  
sbit BUSY=P2^7;
```

```
sbit De=P1^1; ///  
sbit DQ=P1^4; ///  
uchar TPH,TPL; ///  
unsigned int t; ///  
unsigned int t1=30; ///  
//目标温度值
```

```
sbit swh1=P3^6;  
sbit swh2=P3^7;  
uchar flag1=0;  
uchar flag2=0;
```

```
void send_byte(uchar dat,uchar cs1,uchar cs2);  
void send_all(uint page,uint lie,uint offset);  
void delay(uint x);  
void init_yejing();  
void clearsreen();
```

```
void DelayXus(uchar n); ///  
void ow_rest(); ///  
void write_byte(char dat);///  
//写操作
```

```
unsigned char read_bit(void);///读操作
```

```
void main(void)
```

```
{
```

```
    init_yejing();
```

```
    t=0;
```

```
    while(1)
```

```
    {
```

```
        if(swh1==0)
```

```
            flag1=1;
```

```
        if(swh1==1 && flag1==1)
```

```
        {
```

```
            t1++;
```

```
            flag1=0;
```

```
        }
```

```
        if(swh2==0)
```

```
            flag2=1;
```

```
        if(swh2==1 && flag2==1)
```

```
        {
```

```
            t1--;
```

```
            flag2=0;
```

```
        }
```

```
        if(t<t1)
```

```
            De=1;
```

```
        else
```

```
            De=0;
```

```
        ow_rest(); ///设备复位
```

```
        write_byte(0xCC); ///跳过 ROM 命令
```

```
        write_byte(0x44); ///开始转换命令
```

```
        delayXus(100);
```

```
        while (!DQ); ///等待转换完成
```

```
        ow_rest(); ///设备复位
```

```
        write_byte(0xCC); ///跳过 ROM 命令
```

```
        write_byte(0xBE); ///读暂存存储器命令
```

```
        TPL = read_bit(); ///读温度低字节
```

```
        TPH = read_bit(); ///读温度高字节
```

```
        t=TPH; ///取温度高位
```

```
        t<<=8; ///高位 8 位
```

```
        t|=TPL; ///加上温度低位
```

```
        t*=0.625; ///实际温度 可直接显示
```

```

        t=t/10;
        send_all(1,1,14);///温
        send_all(1,2,15);///度
        send_all(1,3,12);///:
        send_all(4,2,t1/10);///十
        send_all(4,3,t1%10);///个

        send_all(4,5,t/10);///十
        send_all(4,6,t%10);///个

        delay(50000);
        clearscren();
    }
}

void DelayXus(uchar n)
{
    while (n--)
    {
        _nop_();
        _nop_();
    }
}

unsigned char read_bit(void)///读位
{
    uchar i;
    uchar dat = 0;
    for (i=0; i<8; i++) ///8 位计数器
    {
        dat >>= 1;
        DQ = 0; ///开始时间片
        DelayXus(1); ///延时等待
        DQ = 1; ///准备接收
        DelayXus(10); ///接收延时
        if (DQ) dat |= 0x80; ///读取数据
        DelayXus(60); ///等待时间片结束
    }
    return dat;
}

void ow_rest()///复位
{
    CY = 1;

```



```

while (CY)
{
    DQ = 0; ///送出低电平复位信号
    DelayXus(240); ///延时至少 480us
    DelayXus(240);
    DQ = 1; ///释放数据线
    DelayXus(60); ///等待 60us
    CY = DQ; ///检测存在脉冲,DQ 为 0 转换完成
    DelayXus(240); ///等待设备释放数据线
    DelayXus(180);
}
}

```

```

void write_byte(char dat)///写字节
{
    uchar i;
    for (i=0; i<8; i++) ///8 位计数器
    {
        DQ = 0; ///开始时间片
        DelayXus(1); ///延时等待
        dat >>= 1; ///送出数据
        DQ = CY;
        DelayXus(60); ///等待时间片结束
        DQ = 1; ///恢复数据线
        DelayXus(1); ///恢复延时
    }
}

```

```

void init_yejing()
{
    send_byte(192,1,1);///设置起始行
    send_byte(63,1,1);///打开显示开关
}

```

```

void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;
    while(BUSY) ;
    ///送数据或控制字
    E=0;
    RS=!(cs1&&cs2),RW=0;
    P2=dat;
}

```

```

    E=1; delay(3); E=0;
    CS1=CS2=0;
}

void send_all(uint page,uint lie,uint offset)
{
    uint i,j,k=0;
    for(i=0;i<2;++i)
    {
        send_byte(184+i+page,1,1);///选择页面
        send_byte(64+lie*16-(lie>3)*64,1,1);///选择列号
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);///送数
    }
}

void delay(uint x)
{
    while(x--);
}

void clearscreen()
{
    int i,j;
    for(i=0;i<8;++i)
    {
        send_byte(184+i,1,1);///页
        send_byte(64,1,1);///列
        for(j=0;j<64;++j)
        {
            send_byte(0x00,0,1);
            send_byte(0x00,1,0);
        }
    }
}

```

### 三、问题分析

（实验过程中遇到的问题及解决方法）

实验中需要多次调试使用 PID 时所需的  $K_p$ 、 $K_i$ 、 $K_d$  的数值，需在多次测试后，观察到其值能够成比例的拟合期望的数值。