

单片机实验报告

班级：计科 8 班

姓名：闫英杰

学号：21160823

实验五 重量测量

一、实验目的和要求

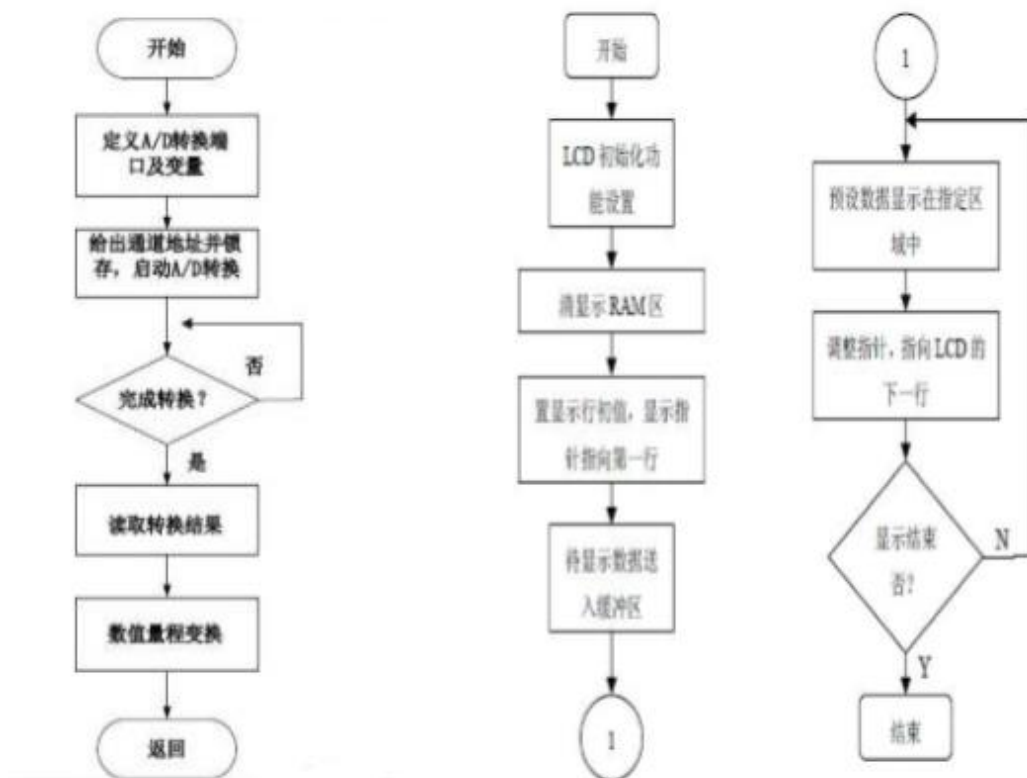
1. 掌握点阵式液晶显示屏的原理和控制方法，掌握点阵字符的显示方法。
2. 掌握 A/D 转换方式，进一步掌握使用 C51 语言编写程序的方法，使用 C51 语言编写实现重量测量的功能。

二、实验内容

参考辅助材料，学习 C51 语言使用

编写 C51 程序，使用测量实验板测量标准砝码的重量将结果(以 g 计)显示到液晶屏上。误差控制在允许的范围之间。

三、流程图



四、实验代码

```
#include <reg52.h>
#include <intrins.h>

#define uchar unsigned char
#define uint unsigned int

sbit CS1=P1^7;///左半边
```

单片机实验报告

```
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 AUX1 = 0xA2; ///AUX1 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/
```

单片机实验报告

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/

单片机实验报告

```
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, ///  
*": " *12/
```

```
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 clearsreen();
```

```
int cweight;
```

```
int weight;
```

```
int temp;
```

```
void main()
```

```
{
```

```
    init_yejing();
```

```
    init_adc();
```

```
    calibrate();///  
校准
```

```
    while(1)
```

```
    {
```

```
        weight=(get_ad_result()-cweight)/2.05;
```

```
        temp = weight;
```

```
        if(temp%10<5)
```

```
            weight = temp -temp%10;
```

```
        else
```

```
        {
```

```
            weight = temp - temp%10;
```

单片机实验报告

```
        weight += 10;
    }

    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;///输出，读取状态字
    while(BUSY) ;

    ///送数据或控制字
    E=0;

    RS=!(cs1&&cs2), RW=0;
    P2=dat;
    E=1; ///输入，写显示数据
    delay(3);

    E=0;
    CS1=CS2=0;
}
```

单片机实验报告

```
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 选择 P1.0 口
    AURX1 |= 0X04; ///AURX1 中的 ADRJ 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();
}
```

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

void clearscren()
{
    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);
        }
    }
}
```

五、思考题

1. 调零的原理，软件调零和硬件调零的区别。

答：

硬件调零:使用外接电路或者改变压敏电阻的初始阻止等方式实现的调零。通过附加电路或者对压敏电阻调整；

软件调零:在不适用任何外接电路的情况下，对采集的数据进行数学处理从而实现调零的过程。

使用软件对 A/D 采集值进行调整来达到在未放置物体时候显示 0。

2. A/D 和 D/A 信号的转换原理。

答:A/D 逐次逼近法:由一个比较器、D/A 转换器、缓冲寄存器及控制逻辑电路组成。初始化时将逐次逼近寄存器各位清零；

转换开始时，先将逐次逼近寄存器最高位置 1, 送入 D/A 转换器，

经 D/A 转换后生成的模拟量送入比较器，称为 V_o , 与送入比较器的待转换的模拟量 V_i 进行比较，若 $V_o < V_i$, 该位 1 被保留，否则被清除。然后再置逐次逼近寄存器次高位为 1,

单片机实验报告

将寄存器中新的数字量送 D/A 转换器，输出的 V_o 再与 V_i 比较，若 $V_o < V_i$ ，该位 1 被保留，否则被清除。

重复此过程，直至逼近寄存器最低位。

转换结束后，将逐次逼近寄存器中的数字量送入缓冲寄存器，得到数字量的输出。D/A 转换：将二进制数的每位按权大小转换为相对应的模拟量，然后将代表的各位的模拟量相加，就得到对应数字量对应的模拟量。

3. I²C 总线在信号通讯过程中的应用。

答：主器件用于启动总线传送数据，并产生时钟以开放传送的器件，此时任何被寻址的器件均被认为是从器件。在总线上主和从、发和收的关系不是恒定的，而取决于此时数据传送方向。如果主机要发送数据给从器件，则主机首先寻址从器件，然后主动发送数据至从器件，最后由主机终止数据传送。

如果主机要接收从器件的数据，首先由主器件寻址从器件，然后主机接收从器件发送的数据，最后由主机终止接收过程，在这种情况下，主机负责产生定时时钟和终止数据传送。

六、遇到的问题及解决办法

显示的值与砝码实际重量不符，但结果程线性。通过软件调零，将 A/D 转换的结果进行数值调整，使得最终测量结果在误差范围内；

问题分析：在调试校准中出现问题（其中软件调零通过测量点对数据进行曲线拟合；

实验体会：通过本次实验，学会了 A/D 转换的原理，以及液晶显示屏的使用方法。

实验六 直流电机脉宽调制调速

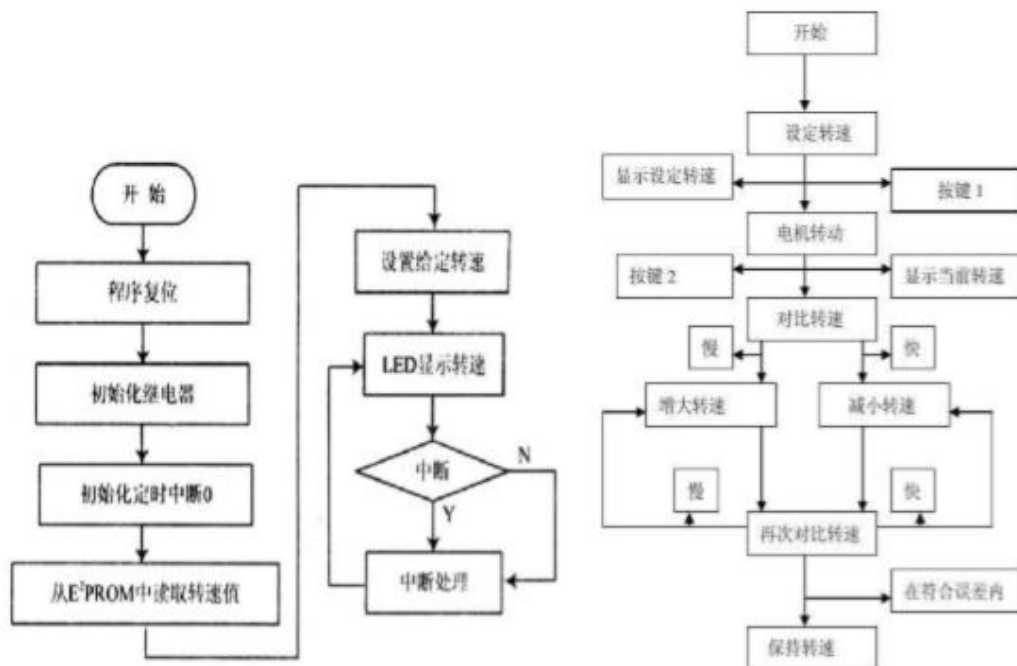
一、 实验目的和要求

掌握脉宽调制调速的原理与方法，学习频率/周期测量的方法，了解闭环控制的原理。

二、实验内容

1. 在液晶显示屏上显示出直流电机的:当前转速、低目标转速、高目标转速。
2. 固定向 PL. 1 输出 0, 然后测量每秒钟电机转动的转数，将其显示在数码管，每秒刷新一次即可。
- 3 使用脉宽调制的方法，动态调整向 P1. 1 输出的内容，使得电机转速能够稳定在一个预定值附近，同时实时显示当前转速。
4. 根据输入修改电机得目标转速值，设置两个转速目标值：低转速和高转速。
5. 每隔秒钟读取两个开关的状态，如果 s1 按下，动态调整输出，使得电机转速能够稳定到低转速目标值附近，如果 S2 按下，动态调整输出，使得电机转速能够稳定到高转速目标值附近。交替显示目标值和当前转速值。

三、流程图



四、实验代码

```
#include <reg52.h>
#include <intrins.h>

#define uchar unsigned char
```

单片机实验报告

```
#define uint unsigned int
sfr P4=0xC0;
sfr P4SW=0xBB;
sbit sclk=P4^4;
sbit sdata=P4^5;

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;

///sfr IE=0xA8;
sbit swh1=P3^6;//S1
sbit swh2=P3^7;//S2
sbit motor=P1^1;

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,
```

单片机实验报告

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/

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/

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, ///*”:”*12/

0x00, 0x04, 0x04, 0xE4, 0x24, 0x24, 0x24, 0x3F, 0x24, 0x24, 0x24, 0xE4, 0x04,

单片机实验报告

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

};

uchar tab[15]=
    {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
     0x92, 0x82, 0x0F8, 0x80, 0x90} ;//0-9

uchar tspeed=0;//累加转数
uchar cspeed=0;//当前速度
uchar xspeed=120;//期望速度
uchar speedUp = 140;//转速上限
uchar speedLow =100;//转速下限

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

void send_byte(uchar dat ,uchar cs1,uchar cs2);
void send_all(uint page,uint lie,uint offset);
void init();
void clearscreen();
void init_yejing();
void sendbyte(uchar ch);
void display(uchar n);
void delay1();
void delay2();
void delay(uint x)
{
    while(x--);
}

void main()
{
    init();
    init_yejing();
    motor=0;
    while(1)
    {
        clearscreen();
        send_all(1,3,speedLow/100);
```

单片机实验报告

```
        send_all(1, 4, (speedLow/10)%10);
        send_all(1, 5, speedLow%10);
    send_all(3, 3, cspeed/100);
        send_all(3, 4, (cspeed/10)%10);
        send_all(3, 5, cspeed%10);
    send_all(5, 3, speedUp/100);
        send_all(5, 4, (speedUp/10)%10);
        send_all(5, 5, speedUp%10);

    delay1();
        display(cspeed);
    delay(50000);
}

void init()
{
    P4SW=0x30;

    IT0=1; ///设置 INTO 为边沿触发

    EA=1; //系统中断允许
    ET1=1;///定时器中断允许
    ET0=1;
    EX0=1; //外部中断允许
    TMOD=0x11; ///设置定时器 0 和 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 INTO
{
    tspeed++;
}
```

单片机实验报告

```
void t1_int() interrupt 3    ///50ms 定时器中断 T1
{
    if(++t1_cnt<20)
    {
        TH1=0x3C;
        TL1=0xB0;
        if(swh1==0)
        {

            xspeed = speedLow;

        }

        if(swh2==0) {

            xspeed = speedUp;

        }
        if(swh1==1 && swh2==1 ) {
            xspeed = 120;
        }

        return;
    }

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

//累加进位法

```
void t0_int() interrupt 1    ///0.1ms 定时器中断 T0
{
    TH0=0xFF;
    TL0=0x9C;
    X+=N;
    if(X>M)
```

单片机实验报告

```
{
    motor=0;
    X-=M;
}
else
    motor=1;
}
void init_yejing()
{
    send_byte(192,1,1);
    send_byte(63,1,1); }
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;
    while(BUSY) ;

    ///???????
    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);
        send_byte(64+lie*16-(lie>3)*64,1,1);
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);
    }
}

void clearscren()
{
    int i,j;
    for(i=0;i<8;++i)
    {
```


单片机实验报告

```
        send_byte(184+i, 1, 1); ///初始化行
        send_byte(64, 1, 1); ///初始化页
        for(j=0; j<64; ++j)
        {
            send_byte(0x00, 0, 1);
            send_byte(0x00, 1, 0);
        }
    }
}
```

```
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;
    }
}
//LED??
void display(uchar n)
{
    sendbyte(n%10);        ///百
    sendbyte((n/10)%10);   ///十
    sendbyte(n/100);       ///个
}
```

```
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++);
}
```

单片机实验报告

五、思考题

1. 讨论脉宽调速和电压调速的区别、优缺点和应用范围。

答:脉宽调速:是一种能够通过开关量输出达到模拟量输出效果的方法。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 δ (也叫做占空比),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。并且需要的外围器件较少,特别适合于单片机控制领域。

电压调速:直接改变电压模拟量从而改变电机转速的方法。电压便于平滑性调节,可以实现无级调速,损耗小,调速经济性好。

2. 说明程序原理中累加进位法的正确性。

答: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 。每次循环中都有 M/N 次输出, 而其中只有一次输出为 1, 即总共输出了 N 个 1, 而且总输出次数是 M 次, 1 的比例就是 N/M 。

3. 计算转速测量的最大可能误差, 讨论减少误差的办法。

答:减少误差的方法:

外部因素:减少摩擦力

内部因素:让电机的转速保持在 200~40 转/s 之间的速度, 并保持一个比较低的速度(速度不要过快)。

六、遇到的问题及解决办法

实验中要求每一秒钟判断一次开关是否断开, 可用定时器中断设置 50ms 一次, 循环 20 次后进行判断, 更加准确。

通过本次实验, 对直流电机的脉冲调制有了基础的了解, 同时巩固了中断方面的知识, 收获很多。

实验八 温度测量与控制

一、实验目的和要求

1. 学习 DS18B20 温度传感器的编程结构。
2. 了解温度测量的原理。
3. 掌握 PID 控制原理及实现方法。
4. 加深 C51 编程语言的理解和学习。

二、实验设备

单片机测控实验系统

温控实验模块

Keil 开发环境

STC-ISP 程序下载工具

三、实验内容

掌握使用传感器测量与控制温度的原理与方法，使用 C51 语言编写实现温度控制的功能，使用超声波/温度实验板测量温度，将温度测量的结果（单位为摄氏度）显示到液晶屏上。

编程实现测量当前教室的温度，显示在 LCM 液晶显示屏上。

通过 S1 设定一个高于当前室温的目标温度值。

编程实现温度的控制，将当前温度值控制到目标温度值并稳定的显示。

四、实验步骤

1. 预习，参考附录三，预习 DS18B20 的编程结构，编程时注意 DS18B20 的时间要求，必须准确满足。根据实验原理附录中的流程图进行编程。
2. 将编译后的程序下载到 51 单片机，观察温度的测量结果。
3. 程序调试

五、实验原理

本实验使用的 DS18B20 是单总线数字温度计，测量范围从 -55°C 到 $+125^{\circ}\text{C}$ ，增量值为 0.5°C 。

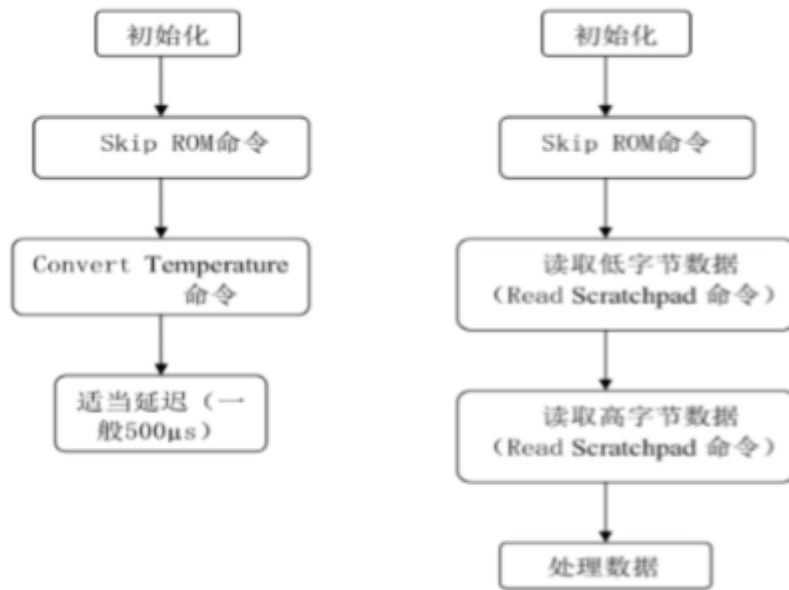
用于贮存测得的温度值的两个 8 位存贮器 RAM 编号为 0 号和 1 号。

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°C 。

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。

六、流程图



七、实验代码

```

#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, 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,

```

单片机实验报告

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,

单片机实验报告

```
0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, ///  
*": " *12/
```

```
    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, 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;///  
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;
```

```
sbit De=P1^1; ///  
加热
sbit DQ=P1^4; ///  
DS18B20 单数据总线
uchar TPH, TPL; ///  
温度值高位 低位
unsigned int t; ///  
温度值
unsigned int t1=30; ///  
目标温度值
```

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

单片机实验报告

```
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 clearscreen();
```

```
void DelayXus(uchar n); ///微秒级延时
void ow_rest(); ///复位
void write_byte(char dat);
unsigned char read_bit(void);
```

```
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
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);

clearscreen();
```



```
    }  
}
```

```
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(1); ///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  
    }  
}
```

单片机实验报告

```
        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 clearscren()
{
    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);
        }
    }
}
```

八、思考题

1. 进行精确的延时程序有几种方法?各有什么优缺点?

答:1. 循环方法:让单片机使用 while 等循环语句进行循环延时程序,实现简单但是浪费 CPU 资源,而且精确度较低。

2. 定时器方法:通过定时器来进行延时,通常可以规定 2-3 个计时器同时进行延时,好处是复用性好,效率和精确度比较高,缺点是计时时间有上限而且会浪费一个计时器。

2. 参考其他资料,了解 DS18B20 的其他命令的用法。

答:

单片机实验报告

1、Read ROM[33H]

2、Match ROM[55H]

这个是匹配 ROM 命令，后跟 64 位 ROM 序列，让总线控制器在多点总线上定位一只特定的 DS18B20。

3、Skip ROM[0CCH]

这条命令允许总线控制器不用提供 64 位 ROM 编码就使用存储器操作命令，在单点总线情况下，可以节省时间。

4、Search ROM[0F0H]

当一个系统初次启动时，总线控制器可能并不知道单线总线上有多个器件或它们的 64 位编码，搜索 ROM 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。

5、Alarm Search [0ECH]

这条命令的流程和 Search ROM 相同。然而，只有在最近一次测温后遇到符合报警条件的情况，DS18B20 才会响应这条命令。

6、Write Scratchpad [4EH]

这个命令向 DS18B20 的暂存器 TH 和 TL 中写入数据。可以在任何时刻发出复位命令来中止写入。

7、Read Scratchpad[0BEH]

这个命令读取暂存器的内容。读取将从第 1 个字节开始，一直进行下去，直到第 9(CRC)字节读完。如果不想读完所有字节，控制器可以在任何时间发出复位命令来中止读取。

8、Copy Scratchpad[48H]

这个命令把暂存器的内容拷贝到 DS18B20 的 E2ROM 存储器里，即把温度报警触发字节存入非易失性存储器里。

9、Convert T[44H]

这条命令启动一次温度转换而无需其他数据。

10、Recall E2

这条命令把报警触发器里的值拷贝回暂存器。这种拷贝操作在 DS18B20 上电时自动执行，这样器件一上电，暂存器里马上就存在有效的数据了。若在这条命令发出之后发出读数据除，器件会输出温度转换忙的标识：0 为忙，1 为完成。

11、Read Power Supply [0B4H]

若把这条命令发给 DS18B20 后发出读时间隙，器件会返回它的电源模式：。为寄生电源，1 为外部电源。

九、遇到的问题及解决办法

通过此次实验，我初步了解了 DS18B20 温度控制的使用方法，它的工作过程可分为初始化-ROM 操作命令-存储器操作命令-处理数据。它会返回 10 数据，取其高 2 位和低 8 位形成 8 位数据送入液晶屏。其次进一步理解了 ADC 模数转换器的使用，ADC 产生电压，通过电压与温度的关系，可以计算出真实的温度。而 PID 温度控制就是通过比例，积分，微分的方法来消除误差，比例的作用就是当误差产生

单片机实验报告

时，增大大比比比例例系数就可以减小小误差，积分的作用用就是消除稳态误差，每当误差产生生时就可以随时消除，而而微分的作用用就是让温度值更更加的接近真实值。

本次实验涉及了 PID 的使用，需自行查找资料做了解后方可灵活使用，明白了其中比例系数，微分及积分的作用，对这次实验有了更加深刻的体会。