

十天学会单片机实例 100

目录

目录.....	1
函数的使用和熟悉	3
实例 3: 用单片机控制第一个灯亮.....	3
实例 4: 用单片机控制一个灯闪烁: 认识单片机的工作频率.....	4
实例 5: 将 P1 口状态分别送入P0、P2、P3 口: 认识I/O口的引脚功能.....	4
实例 6: 使用P3 口流水点亮 8 位LED.....	5
实例 7: 通过对P3 口地址的操作流水点亮 8 位LED	6
实例 8: 用不同数据类型控制灯闪烁时间	7
实例 9: 用P0 口、P1 口分别显示加法和减法运算结果	8
实例 10: 用P0、P1 口显示乘法运算结果	8
实例 11: 用P1、P0 口显示除法运算结果	9
实例 12: 用自增运算控制P0 口 8 位LED流水花样.....	9
实例 13: 用P0 口显示逻辑"与"运算结果	10
实例 14: 用P0 口显示条件运算结果	10
实例 15: 用P0 口显示按位"异或"运算结果	10
实例 16: 用P0 显示左移运算结果.....	10
实例 17: "万能逻辑电路"实验	11
实例 18: 用右移运算流水点亮P1 口 8 位LED	11
实例 19: 用if语句控制P0 口 8 位LED的流水方向.....	12
实例 20: 用swtich语句的控制P0 口 8 位LED的点亮状态.....	12
实例 21: 用for语句控制蜂鸣器鸣笛次数	13
实例 22: 用while语句控制LED	15
实例 23: 用do-while语句控制P0 口 8 位LED流水点亮.....	15
实例 24: 用字符型数组控制P0 口 8 位LED流水点亮	16
实例 25: 用P0 口显示字符串常量.....	17
实例 26: 用P0 口显示指针运算结果	18
实例 27: 用指针数组控制P0 口 8 位LED流水点亮.....	18
实例 28: 用数组的指针控制P0 口 8 位LED流水点亮	19
实例 29: 用P0 、P1 口显示整型函数返回值	20
实例 30: 用有参函数控制P0 口 8 位LED流水速度.....	20
实例 31: 用数组作函数参数控制流水花样	21
实例 32: 用指针作函数参数控制P0 口 8 位LED流水点亮	22
实例 33: 用函数型指针控制P1 口灯花样	23
实例 34: 用指针数组作为函数的参数显示多个字符串	24
实例 35: 字符函数ctype.h应用举例	26
实例 36: 内部函数intrins.h应用举例	26
实例 37: 标准函数stdlib.h应用举例	26
实例 38: 字符串函数string.h应用举例	27
实例 39: 宏定义应用举例 2	28
实例 40: 宏定义应用举例 2	28

实例 41: 宏定义应用举例 3	28
中断、定时器	29
实例 42: 用定时器T0 查询方式P2 口 8 位控制LED闪烁	29
实例 43: 用定时器T1 查询方式控制单片机发出 1KHz音频	30
实例 44: 将计数器T0 计数的结果送P1 口 8 位LED显示	30
实例 45: 用定时器T0 的中断控制 1 位LED闪烁	31
实例 46: 用定时器T0 的中断实现长时间定时	32
实例 47: 用定时器T1 中断控制两个LED以不同周期闪烁	32
实例 48: 用计数器T1 的中断控制蜂鸣器发出 1KHz音频	33
实例 49: 用定时器T0 的中断实现"渴望"主题曲的播放	34
实例 50-1: 输出 50 个矩形脉冲	37
实例 50-2: 计数器T0 统计外部脉冲数	38
实例 51-2: 定时器T0 的模式 2 测量正脉冲宽度	38
实例 52: 用定时器T0 控制输出高低宽度不同的矩形波	39
实例 53: 用外中断 0 的中断方式进行数据采集	40
实例 54-1: 输出负脉宽为 200 微秒的方波	40
实例 54-2: 测量负脉冲宽度	41
实例 55: 方式 0 控制流水灯循环点亮	42
实例 56-1: 数据发送程序	43
实例 56-2: 数据接收程序	44
实例 57-1: 数据发送程序	45
实例 57-2: 数据接收程序	46
实例 58: 单片机向PC发送数据	47
实例 59: 单片机接收PC发出的数据	48
数码管显示	49
实例 60: 用LED数码显示数字 5	49
实例 61: 用LED数码显示器循环显示数字 0~9	49
实例 62: 用数码管慢速动态扫描显示数字"1234"	50
实例 63: 用LED数码显示器伪静态显示数字 1234	51
实例 64: 用数码管显示动态检测结果	51
实例 65: 数码秒表设计	53
实例 66: 数码时钟设计	55
实例 67: 用LED数码管显示计数器T0 的计数值	58
实例 68: 静态显示数字 "59"	59
键盘控制	60
实例 69: 无软件消抖的独立式键盘输入实验	60
实例 70: 软件消抖的独立式键盘输入实验	60
实例 71: CPU控制的独立式键盘扫描实验	61
实例 72: 定时器中断控制的独立式键盘扫描实验	64
实例 73: 独立式键盘控制的 4 级变速流水灯	67
实例 74: 独立式键盘的按键功能扩展: "以一当四"	69
实例 75: 独立式键盘调时的数码时钟实验	71
实例 76: 独立式键盘控制步进电机实验	75
实例 77: 矩阵式键盘按键值的数码管显示实验	78

实例 78: 矩阵式键盘按键音.....	81
实例 79: 简易电子琴	82
实例 80: 矩阵式键盘实现的电子密码锁.....	88
液晶显示LCD	91
实例 81: 用LCD显示字符'A'	91
实例 82: 用LCD循环右移显示"Welcome to China"	95
实例 83: 用LCD显示适时检测结果.....	98
实例 84: 液晶时钟设计.....	102
一些芯片的使用*****24c02 DS18B20 X5045 ADC0832 DAC0832	
DS1302 红外遥控 108	
实例 85: 将数据"0x0f"写入AT24C02 再读出送P1 口显示.....	108
实例 86: 将按键次数写入AT24C02, 再读出并用 1602LCD显示.....	112
实例 87: 对I2C总线上挂接多个AT24C02 的读写操作	119
实例 88: 基于AT24C02 的多机通信 读取程序	125
实例 88: 基于AT24C02 的多机通信 写入程序	129
实例 90: DS18B20 温度检测及其液晶显示	140
实例 91: 将数据"0xaa"写入X5045 再读出送P1 口显示.....	148
实例 92: 将流水灯控制码写入X5045 并读出送P1 口显示.....	152
实例 93: 对SPI总线上挂接多个X5045 的读写操作.....	156
实例 94: 基于ADC0832 的数字电压表	160
实例 95: 用DAC0832 产生锯齿波电压	166
实例 96: 用P1 口显示红外遥控器的按键值	166
实例 97: 用红外遥控器控制继电器	169
实例 98: 基于DS1302 的日历时钟	171
实例 99: 单片机数据发送程序.....	180
实例 100: 电机转速表设计	181
//模拟霍尔脉冲	186

函数的使用和熟悉

实例 3: 用单片机控制第一个灯亮

```
#include<reg51.h> //包含 51 单片机寄存器定义的头文件
void main(void)
{
    P1=0xfe; //P1=1111 1110B, 即 P1.0 输出低电平
}
```

实例 4：用单片机控制一个灯闪烁：认识单片机的工作频率

```
#include<reg51.h>    //包含单片机寄存器的头文件
/*****
函数功能：延时一段时间
*****/
void delay(void)      //两个 void 意思分别为无需返回值，没有参数传递
{
    unsigned int i;   //定义无符号整数，最大取值范围 65535
    for(i=0;i<20000;i++) //做 20000 次空循环
        ;             //什么也不做，等待一个机器周期
}
/*****
函数功能：主函数（C 语言规定必须有也只能有 1 个主函数）
*****/
void main(void)
{
    while(1)         //无限循环
    {
        P1=0xfe;    //P1=1111 1110B, P1.0 输出低电平
        delay();    //延时一段时间
        P1=0xff;    //P1=1111 1111B, P1.0 输出高电平
        delay();    //延时一段时间
    }
}
```

实例 5：将 P1 口状态分别送入 P0、P2、P3 口：认识 I/O 口的引脚功能

```
#include<reg51.h>    //包含单片机寄存器的头文件
/*****
函数功能：主函数（C 语言规定必须有也只能有 1 个主函数）
*****/
void main(void)
{
    while(1)         //无限循环
    {
        P1=0xff;    // P1=1111 1111B,熄灭 LED
        P0=P1;      // 将 P1 口状态送入 P0 口
        P2=P1;      // 将 P1 口状态送入 P2 口
        P3=P1;      // 将 P1 口状态送入 P3 口
    }
}
```

```

    }
}

```

实例 6：使用 P3 口流水点亮 8 位 LED

```
#include<reg51.h> //包含单片机寄存器的头文件
```

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

```
函数功能：延时一段时间
```

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

```
void delay(void)
```

```
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}
```

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

```
函数功能：主函数
```

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

```
void main(void)
```

```
{
    while(1)
    {
        P3=0xfe; //第一个灯亮
        delay(); //调用延时函数
        P3=0xfd; //第二个灯亮
        delay(); //调用延时函数
        P3=0xfb; //第三个灯亮
        delay(); //调用延时函数
        P3=0xf7; //第四个灯亮
        delay(); //调用延时函数
        P3=0xef; //第五个灯亮
        delay(); //调用延时函数
        P3=0xdf; //第六个灯亮
        delay(); //调用延时函数
        P3=0xbf; //第七个灯亮
        delay(); //调用延时函数
        P3=0x7f; //第八个灯亮
        delay(); //调用延时函数
    }
}
```

实例 7：通过对 P3 口地址的操作流水点亮 8 位 LED

```

#include<reg51.h> //包含单片机寄存器的头文件
sfr x=0xb0; //P3 口在存储器中的地址是 b0H， 通过 sfr 可定义 8051 内核单
片机
//的所有内部 8 位特殊功能寄存器,对地址 x 的操作也就是对 P1 口的
操作
/*****
函数功能：延时一段时间
*****/
void delay(void)
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ; //利用循环等待若干机器周期，从而延时一段时间
}
/*****
函数功能：主函数
*****/
void main(void)
{
    while(1)
    {
        x=0xfe; //第一个灯亮
        delay(); //调用延时函数
        x=0xfd; //第二个灯亮
        delay(); //调用延时函数
        x=0xfb; //第三个灯亮
        delay(); //调用延时函数
        x=0xf7; //第四个灯亮
        delay(); //调用延时函数
        x=0xef; //第五个灯亮
        delay(); //调用延时函数
        x=0xdf; //第六个灯亮
        delay(); //调用延时函数
        x=0xbf; //第七个灯亮
        delay(); //调用延时函数
        x=0x7f; //第八个灯亮
        delay(); //调用延时函数
    }
}

```

}

实例 8：用不同数据类型控制灯闪烁时间

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能：用整形数据延时一段时间
*****/
void int_delay(void) //延时一段较长的时间
{
    unsigned int m; //定义无符号整形变量，双字节数据，值域为 0~65535
    for(m=0;m<36000;m++)
        ; //空操作
}
/*****
函数功能：用字符型数据延时一段时间
*****/
void char_delay(void) //延时一段较短的时间
{
    unsigned char i,j; //定义无符号字符型变量，单字节数据，值域 0~255
    for(i=0;i<200;i++)
        for(j=0;j<180;j++)
            ; //空操作
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    while(1)
    {
        for(i=0;i<3;i++)
        {
            P1=0xfe; //P1.0 口的灯点亮
            int_delay(); //延时一段较长的时间
            P1=0xff; //熄灭
            int_delay(); //延时一段较长的时间
        }
        for(i=0;i<3;i++)
        {
            P1=0xef; //P1.4 口的灯点亮

```



```

        char_delay(); //延时一段较长的时间
        P1=0xff;      //熄灭
        char_delay(); //延时一段较长的时间
    }

}

}

```

实例 9：用 P0 口、P1 口分别显示加法和减法运算结果

```

#include<reg51.h>
void main(void)
{
    unsigned char m,n;
    m=43;      //即十进制数 2x16+11=43
    n=60;      //即十进制数 3x16+12=60
    P1=m+n;   //P1=103=0110 0111B,结果 P1.3、P1.4、P1.7 口的灯被点亮
    P0=n-m;   //P0=17=0001 0001B,结果 P0.0、P0.4 的灯被熄灭
}

```

实例 10：用 P0、P1 口显示乘法运算结果

```

#include<reg51.h> //包含单片机寄存器的头文件
void main(void)
{
    unsigned char m,n;
    unsigned int s;
    m=64;
    n=71;
    s=m*n;      //s=64*71=4544,需要 16 位二进制数表示,高 8 位送 P1 口,
                //低 8 位送 P0 口
                //由于
                4544=17*256+192=H3*16*16*16+H2*16*16+H1*16+H0
                //两边同除以 256,可得 17+192/256=H3*16+H2+
                (H1*16+H0)/256
                //因此,高 8 位 16 进制数 H3*16+H2 必然等于 17,即 4544
                除以 256 的商
                //低 8 位 16 进制数 H1*16+H0 必然等于 192,即 4544 除以
                256 的余数
}

```

```

    P1=s/256;    //高 8 位送 P1 口 , P1=17=11H=0001 0001B, P1.0 和 P1.4
    口灭, 其余亮
    P0=s%256;    //低 8 位送 P0 口 , P3=192=c0H=1100
    0000B,P3.1,P3.6,P3.7 口灭, 其余亮
}

```

实例 11: 用 P1、P0 口显示除法运算结果

```

#include<reg51.h>    //包含单片机寄存器的头文件
void main(void)
{
    P1=36/5;        //求整数
    P0=((36%5)*10)/5; //求小数
    while(1)
        ;          //无限循环防止程序“跑飞”
}

```

实例 12: 用自增运算控制 P0 口 8 位 LED 流水花样

```

#include<reg51.h>    //包含单片机寄存器的头文件
/*****
函数功能: 延时一段时间
*****/
void delay(void)
{
    unsigned int i;
    for(i=0;i<20000;i++)
        ;
}
/*****
函数功能 : 主函数
*****/
void main(void)
{
    unsigned char i;
    for(i=0;i<255;i++) //注意 i 的值不能超过 255
    {
        P0=i;    //将 i 的值送 P0 口
        delay(); //调用延时函数
    }
}

```

实例 13: 用 P0 口显示逻辑"与"运算结果

```
#include<reg51.h> //包含单片机寄存器的头文件
void main(void)
{
    P0=(4>0)&&(9>0xab);//将逻辑运算结果送 P0 口
    while(1)
        ; //设置无限循环, 防止程序“跑飞”
}
```

实例 14: 用 P0 口显示条件运算结果

```
#include<reg51.h> //包含单片机寄存器的头文件
void main(void)
{
    P0=(8>4)?8:4;//将条件运算结果送 P0 口, P0=8=0000 1000B
    while(1)
        ; //设置无限循环, 防止程序“跑飞”
}
```

实例 15: 用 P0 口显示按位"异或"运算结果

```
#include<reg51.h> //包含单片机寄存器的头文件
void main(void)
{
    P0=0xa2^0x3c;//将条件运算结果送 P0 口, P0=8=0000 1000B
    while(1)
        ; //设置无限循环, 防止程序“跑飞”
}
```

实例 16: 用 P0 显示左移运算结果

```
#include<reg51.h> //包含单片机寄存器的头文件
void main(void)
{
    P0=0x3b<<2;//将左移运算结果送 P0 口, P0=1110 1100B=0xec
    while(1)
        ; //无限循环, 防止程序“跑飞”
}
```

```
}

```

实例 17: "万能逻辑电路"实验

```
#include<reg51.h> //包含单片机寄存器的头文件
sbit F=P1^4;      //将 F 位定义为 P1.4
sbit X=P1^5;      //将 X 位定义为 P1.5
sbit Y=P1^6;      //将 Y 位定义为 P1.6
sbit Z=P1^7;      //将 Z 位定义为 P1.7
void main(void)
{
    while(1)
    {
        F=((~X)&Y)|Z; //将逻辑运算结果赋给 F
        ;
    }
}

```

实例 18: 用右移运算流水点亮 P1 口 8 位 LED

```
#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能: 延时一段时间
*****/
void delay(void)
{
    unsigned int n;
    for(n=0;n<30000;n++)
        ;
}
/*****
函数功能: 主函数
*****/
void main(void)
{
    unsigned char i;
    while(1)
    {
        P1=0xff;
        delay();
        for(i=0;i<8;i++)//设置循环次数为 8
        {

```

```

        P1=P1>>1; //每次循环 P1 的各二进制位右移 1 位，高位补 0
        delay(); //调用延时函数
    }
}
}

```

实例 19：用 if 语句控制 P0 口 8 位 LED 的流水方向

```

#include<reg51.h> //包含单片机寄存器的头文件
sbit S1=P1^4; //将 S1 位定义为 P1.4
sbit S2=P1^5; //将 S2 位定义为 P1.5
/*****
函数功能：主函数
*****/
void main(void)
{
    while(1)
    {
        if(S1==0) //如果按键 S1 按下
            P0=0x0f; //P0 口高四位 LED 点亮
        if(S2==0) //如果按键 S2 按下
            P0=0xf0; //P0 口低四位 LED 点亮
    }
}

```

实例 20：用 switch 语句的控制 P0 口 8 位 LED 的点亮状态

```

#include<reg51.h> //包含单片机寄存器的头文件
sbit S1=P1^4; //将 S1 位定义为 P1.4
/*****
函数功能：延时一段时间
*****/
void delay(void)
{
    unsigned int n;
    for(n=0;n<10000;n++)
        ;
}
/*****
函数功能：主函数
*****/

```

```

void main(void)
{
    unsigned char i;
    i=0;    //将 i 初始化为 0
    while(1)
    {
        if(S1==0)    //如果 S1 键按下
        {
            delay(); //延时一段时间
            if(S1==0) //如果再次检测到 S1 键按下
                i++;    //i 自增 1
            if(i==9) //如果 i=9, 重新将其置为 1
                i=1;
        }
        switch(i)    //使用多分支选择语句
        {
            case 1: P0=0xfe; //第一个 LED 亮
                    break;
            case 2: P0=0xfd; //第二个 LED 亮
                    break;
            case 3: P0=0xfb; //第三个 LED 亮
                    break;
            case 4: P0=0xf7; //第四个 LED 亮
                    break;
            case 5: P0=0xef; //第五个 LED 亮
                    break;
            case 6: P0=0xdf; //第六个 LED 亮
                    break;
            case 7: P0=0xbf; //第七个 LED 亮
                    break;
            case 8: P0=0x7f; //第八个 LED 亮
                    break;
            default: //缺省值, 关闭所有 LED
                    P0=0xff;
        }
    }
}

```

实例 21: 用 for 语句控制蜂鸣器鸣笛次数

```

#include<reg51.h> //包含单片机寄存器的头文件
sbit sound=P3^7; //将 sound 位定义为 P3.7

```

```
/******  
函数功能：延时形成 1600Hz 音频  
*****/  
void delay1600(void)  
{  
    unsigned char n;  
    for(n=0;n<100;n++)  
        ;  
}  
/******  
函数功能：延时形成 800Hz 音频  
*****/  
void delay800(void)  
{  
    unsigned char n;  
    for(n=0;n<200;n++)  
        ;  
}  
  
/******  
函数功能：主函数  
*****/  
void main(void)  
{  
    unsigned int i;  
    while(1)  
    {  
        for(i=0;i<830;i++)  
        {  
            sound=0; //P3.7 输出低电平  
            delay1600();  
            sound=1; //P3.7 输出高电平  
            delay1600();  
        }  
        for(i=0;i<200;i++)  
        {  
            sound=0; //P3.7 输出低电平  
            delay800();  
            sound=1; //P3.7 输出高电平  
            delay800();  
        }  
    }  
}
```

}

实例 22: 用 while 语句控制 LED

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能: 延时约 60ms (3*100*200=60000 μ s)
*****/
void delay60ms(void)
{
    unsigned char m,n;
    for(m=0;m<100;m++)
        for(n=0;n<200;n++)
            ;
}
/*****
函数功能: 主函数
*****/
void main(void)
{
    unsigned char i;
    while(1) //无限循环
    {
        i=0; //将 i 初始化为 0
        while(i<0xff) //当 i 小于 0xff (255)时执行循环体
        {
            P0=i; //将 i 送 P0 口显示
            delay60ms(); //延时
            i++; //i 自增 1
        }
    }
}

```

实例 23: 用 do-while 语句控制 P0 口 8 位 LED 流水点亮

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能: 延时约 60ms (3*100*200=60000 μ s)
*****/
void delay60ms(void)
{
    unsigned char m,n;
    for(m=0;m<100;m++)

```



```

        for(n=0;n<200;n++)
            ;
    }
    /*****
    函数功能：主函数
    *****/
    void main(void)
    {
        do
        {
            P0=0xfe;    //第一个 LED 亮
            delay60ms();
            P0=0xfd;    //第二个 LED 亮
            delay60ms();
            P0=0xfb;    //第三个 LED 亮
            delay60ms();
            P0=0xf7;    //第四个 LED 亮
            delay60ms();
            P0=0xef;    //第五个 LED 亮
            delay60ms();
            P0=0xdf;    //第六个 LED 亮
            delay60ms();
            delay60ms();
            P0=0xbf;    //第七个 LED 亮
            delay60ms();
            P0=0x7f;    //第八个 LED 亮
            delay60ms();
        }while(1);    //无限循环，使 8 位 LED 循环流水点亮
    }

```

实例 24：用字符型数组控制 P0 口 8 位 LED 流水点亮

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能：延时约 60ms (3*100*200=60000 μs)
*****/
void delay60ms(void)
{
    unsigned char m,n;
    for(m=0;m<100;m++)
        for(n=0;n<200;n++)
            ;
}

```

```

/*****
函数功能： 主函数
*****/
void main(void)
{
    unsigned char i;
    unsigned char code Tab[ ]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f}; //定义
无符号字符型数组
    while(1)
    {
        for(i=0;i<8;i++)
        {
            P0=Tab[i];//依次引用数组元素，并将其送 P0 口显示
            delay60ms();//调用延时函数
        }
    }
}

```

实例 25： 用 P0 口显示字符串常量

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能： 延时约 150ms (3*200*250=150 000 μ s=150ms
*****/
void delay150ms(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能： 主函数
*****/
void main(void)
{
    unsigned char str[]={ "Now, Temperature is :"}; //将字符串赋给字符型全部元
素赋值
    unsigned char i;
    while(1)
    {
        i=0; //将 i 初始化为 0，从第一个元素开始显示
        while(str[i]!='\0') //只要没有显示到结束标志'\0'

```

```

    {
        P0=str[i];    //将第 i 个字符送到 P0 口显示
        delay150ms(); //调用 150ms 延时函数
        i++;         //指向下一个待显字符
    }
}
}

```

实例 26：用 P0 口显示指针运算结果

```

#include<reg51.h>
void main(void)
{
    unsigned char *p1,*p2; //定义无符号字符型指针变量 p1,p2
    unsigned char i,j;     //定义无符号字符型数据
    i=25;                  //给 i 赋初值 25
    j=15;
    p1=&i;                 //使指针变量指向 i ，对指针初始化
    p2=&j;                 //使指针变量指向 j ，对指针初始化
    P0=*p1+*p2;           // *p1+*p2 相当于 i+j,所以 P0=25+15=40=0x28
                          //则 P0=0010 1000B，结果 P0.3、P0.5 引脚 LED 熄灭，其余点亮
    while(1)
        ;                 //无限循环，防止程序“跑飞”
}

```

实例 27：用指针数组控制 P0 口 8 位 LED 流水点亮

```

#include<reg51.h>
/*****
函数功能：延时约 150ms (3*200*250=150 000 μs=150ms)
*****/
void delay150ms(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能：主函数
*****/

```

```

void main(void)
{
    unsigned char code Tab[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
    unsigned char *p[ ]={&Tab[0],&Tab[1],&Tab[2],&Tab[3],&Tab[4],&Tab[5],
        &Tab[6],&Tab[7]};

    unsigned char i;    //定义无符号字符型数据
    while(1)
    {
        for(i=0;i<8;i++)
        {
            P0=*p[i];
            delay150ms();
        }
    }
}

```

实例 28：用数组的指针控制 P0 口 8 位 LED 流水点亮

```

#include<reg51.h>
/*****
函数功能：延时约 150ms (3*200*250=150 000 μs=150ms
*****/
void delay150ms(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    unsigned char Tab[ ]={0xFF,0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,
        0x7F,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,0xFE,
        0xFE,0xFC,0xFB,0xF0,0xE0,0xC0,0x80,0x00,
        0xE7,0xDB,0xBD,0x7E,0x3C,0x18,0x00,0x81,
        0xC3,0xE7,0x7E,0xBD,0xDB,0xE7,0xBD,0xDB};
        //流水灯控制码

```

```

unsigned char *p;    //定义无符号字符型指针
p=Tab;              //将数组首地址存入指针 p
while(1)
{
    for(i=0;i<32;i++) //共 32 个流水灯控制码
    {
        P0=*(p+i);    /* (p+i)的值等于 a[i]
        delay150ms(); //调用 150ms 延时函数
    }
}
}

```

实例 29：用 P0 、 P1 口显示整型函数返回值

```

#include<reg51.h>
/*****
函数功能：计算两个无符号整数的和
*****/
unsigned int sum(int a,int b)
{
    unsigned int s;
    s=a+b;
    return (s);
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned z;
    z=sum(2008,2009);
    P1=z/256;    //取得 z 的高 8 位
    P0=z%256;   //取得 z 的低 8 位
    while(1)
        ;
}

```

实例 30：用有参函数控制 P0 口 8 位 LED 流水速度

```

#include<reg51.h>
/*****
函数功能：延时一段时间

```

```

*****/
void delay(unsigned char x)
{
    unsigned char m,n;
    for(m=0;m<x;m++)
        for(n=0;n<200;n++)
            ;
}
/*****
函数功能： 主函数
*****/
void main(void)
{
    unsigned char i;
    unsigned char code
Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
//流水灯控制码

while(1)
{
    //快速流水点亮 LED
    for(i=0;i<8;i++) //共 8 个流水灯控制码
    {
        P0=Tab[i];
        delay(100); //延时约 60ms, (3*100*200=60 000 μ s)
    }
    //慢速流水点亮 LED
    for(i=0;i<8;i++) //共 8 个流水灯控制码
    {
        P0=Tab[i];
        delay(250); //延时约 150ms, (3*250*200=150 000 μ s)
    }
}
}

```

实例 31：用数组作函数参数控制流水花样

```

#include<reg51.h>
/*****
函数功能： 延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)

```

```

        for(n=0;n<250;n++)
            ;
    }
    /*****
    函数功能：流水点亮 P0 口 8 位 LED
    *****/
    void led_flow(unsigned char a[8])
    {
        unsigned char i;
        for(i=0;i<8;i++)
        {
            P0=a[i];
            delay();
        }
    }

    /*****
    函数功能：主函数
    *****/
    void main(void)
    {
        unsigned char code
        Tab[]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
                //流水灯控制码

        led_flow(Tab);
    }

```

实例 32：用指针作函数参数控制 P0 口 8 位 LED 流水点亮

```

#include<reg51.h>
/*****
函数功能：延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能：流水点亮 P0 口 8 位 LED

```

```

*****/
void led_flow(unsigned char *p) //形参为无符号字符型指针
{
    unsigned char i;
    while(1)
    {
        i=0; //将 i 置为 0，指向数组第一个元素
        while(*(p+i)!='\0') //只要没有指向数组的结束标志
        {
            P0=*(p+i); // 取的指针所指变量（数组元素）的值，送 P0 口
            delay(); //调用延时函数
            i++; //指向下一个数组元素
        }
    }
}

/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char code
    Tab[] = {0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F,
            0x7F,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,0xFE,
            0xFF,0xFE,0xFC,0xFB,0xF0,0xE0,0xC0,0x80,
            0x00,0xE7,0xDB,0xBD,0x7E,0xFF,0xFF,0x3C,
            0xFF,0x7E};
            //流水灯控制码

    unsigned char *pointer;
    pointer=Tab;
    led_flow(pointer);
}

```

实例 33：用函数型指针控制 P1 口灯花样

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
unsigned char code Tab[]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
//流水灯控制码，该数组被定义为全局变量

```



```

/*****
函数功能： 延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能： 流水灯左移
*****/
void led_flow(void)
{
    unsigned char i;
    for(i=0;i<8;i++) //8 位控制码
    {
        P0=Tab[i];
        delay();
    }
}
/*****
函数功能： 主函数
*****/
void main(void)
{
    void (*p)(void); //定义函数型指针，所指函数无参数，无返回值
    p=led_flow; //将函数的入口地址赋给函数型指针 p
    while(1)
        (*p)(); //通过函数的指针 p 调用函数 led_flow ()
}

```

实例 34：用指针数组作为函数的参数显示多个字符串

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
unsigned char code str1[ ]="Temperature is tested by DS18B20";//C 语言中，
字符串是作为字符数组来处理的
unsigned char code str2[ ]="Now temperature is:"; //所以，字符串的名字就
是字符串的首地址
unsigned char code str3[ ]="The System is designed by Zhang San";
unsigned char code str4[ ]="The date is 2008-9-30";

```

```
unsigned char *p[ ]={str1,str2,str3,str4}; //定义 p[4]为指向 4 个字符串的字符型
指针数组
```

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

```
函数功能： 延时约 150ms
```

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

```
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
```

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

```
函数功能： 流水点亮 P0 口 8 位 LED
```

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

```
void led_display(unsigned char *x[ ]) //形参必须为指针数组
```

```
{
    unsigned char i,j;
    for(i=0;i<4;i++) //有 4 个字符串要显示
    {
        j=0; //指向待显字符串的第 0 号元素
        while(*(x[i]+j)!='\0') //只要第 i 个字符串的第 j 号元素不是结束标志
        {
            P0=*(x[i]+j); //取得该元素值送到 P0 口显示
            delay(); //调用延时函数
            j++; //指向下一个元素
        }
    }
}
```

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

```
函数功能： 主函数
```

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

```
void main(void)
{
    unsigned char i;
    while(1)
    {
        for(i=0;i<4;i++)
            led_display(p); //将指针数组名作实际参数传递
    }
}
```

实例 35: 字符函数 ctype.h 应用举例

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#include<ctype.h>
void main(void)
{
    while(1)
    {
        P3=isalpha('_')?0xf0:0x0f;//条件运算, 若'_'是英文字母, P3=0xf0
    }
}

```

实例 36: 内部函数 intrins.h 应用举例

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#include<intrins.h> //包含函数 isalpha () 声明的头文件
/*****
函数功能: 延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能: 主函数
*****/
void main(void)
{
    P3=0xfe; //P3=1111 1110B
while(1)
    {
        P3=_crol_(P3,1);// 将 P3 的二进制位循环左移 1 位后再赋给 P3
        delay(); //调用延时函数
    }
}

```

实例 37: 标准函数 stdlib.h 应用举例

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#include<stdlib.h> //包含函数 isalpha ( ) 声明的头文件
/*****
函数功能：延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    while(1)
    {
        for(i=0;i<10;i++) //产生 10 个随机数
        {
            P3=rand()/160; //将产生的随机数缩小 160 倍后送 P3 显示
            delay();
        }
    }
}

```

实例 38：字符串函数 string.h 应用举例

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#include<string.h> //包含函数 isalpha ( ) 声明的头文件
void main(void)
{
    unsigned char str1[ ]="Now, The temperature is :";
    unsigned char str2[ ]="Now, The temperature is 36 Centgrade.";
    unsigned char i;
    i=strcmp(str1,str2); //比较两个字符串，并将结果存入 i
    if(i==0) //str1=str2
        P3=0x00;
    else
        if(i<0) //str1<str2
            P3=0xf0;

```

```

    else    //str1>str2
        P3=0x0f;
while(1)
    ;    //防止程序“跑飞”
}

```

实例 39：宏定义应用举例 2

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
#define F(a,b) (a)+(a)*(b)/256+(b)    //带参数的宏定义，a 和 b 为形参
void main(void)
{
    unsigned char i,j,k;
    i=40;
    j=30;
    k=20;
    P3=F(i,j+k);    //i 和 j+k 分别为实参，宏展开时，实参将替代宏定义中的形参
while(1)
    ;
}

```

实例 40：宏定义应用举例 2

```

#include<AT89X51.h>
#include<ctype.h>
void main(void)
{
    P3_0=0;    //将 P3.0 引脚置低电平，LED 点亮
    P3_1=0;    //将 P3.0 引脚置低电平，LED 点亮
    P3_2=0;    //将 P3.0 引脚置低电平，LED 点亮
    P3_3=0;    //将 P3.0 引脚置低电平，LED 点亮
    P3_4=1;    //将 P3.4 引脚置高电平，LED 熄灭
    P3_5=1;    //将 P3.5 引脚置高电平，LED 熄灭
    P3_6=1;    //将 P3.7 引脚置高电平，LED 熄灭
    P3_7=1;    //将 P3.7 引脚置高电平，LED 熄灭
while(1)
    ;
}

```

实例 41：宏定义应用举例 3

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#define MAX 100 //将 MAX 宏定义为字符串 100
void main(void)
{
    #if MAX>80 //如果字符串 100 大于 80
        P3=0xf0; //P3 口低四位 LED 点亮
    #else
        P3=0x0f; //否则, P3 口高四位 LED 点亮
    #endif //结束本次编译
}

```

中断、定时器

实例 42: 用定时器 T0 查询方式 P2 口 8 位控制 LED 闪烁

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
/*****
函数功能: 主函数
*****/
void main(void)
{
    // EA=1; //开总中断
    // ET0=1; //定时器 T0 中断允许
    TMOD=0x01; //使用定时器 T0 的模式 1
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-46083)%256; //定时器 T0 的高 8 位赋初值
    TR0=1; //启动定时器 T0
    TF0=0;
    P2=0xff;
    while(1)//无限循环等待查询
    {
        while(TF0==0)
            ;
        TF0=0;
        P2=~P2;
        TH0=(65536-46083)/256; //定时器 T0 的高 8 位赋初值
        TL0=(65536-46083)%256; //定时器 T0 的高 8 位赋初值
    }
}

```

```
}

```

实例 43：用定时器 T1 查询方式控制单片机发出 1KHz 音频

```
#include<reg51.h>          // 包含 51 单片机寄存器定义的头文件
sbit sound=P3^7;          //将 sound 位定义为 P3.7 引脚
/*****
函数功能：主函数
*****/
void main(void)
{
    // EA=1;                //开总中断
    // ET0=1;              //定时器 T0 中断允许
    TMOD=0x10;             //使用定时器 T1 的模式 1
    TH1=(65536-921)/256;   //定时器 T1 的高 8 位赋初值
    TL1=(65536-921)%256;   //定时器 T1 的高 8 位赋初值
    TR1=1;                 //启动定时器 T1
    TF1=0;
    while(1)//无限循环等待查询
    {
        while(TF1==0)
            ;
        TF1=0;
        sound=~sound;      //将 P3.7 引脚输出电平取反
        TH1=(65536-921)/256; //定时器 T0 的高 8 位赋初值
        TL1=(65536-921)%256; //定时器 T0 的高 8 位赋初值
    }
}

```

实例 44：将计数器 T0 计数的结果送 P1 口 8 位 LED 显示

```
#include<reg51.h>          // 包含 51 单片机寄存器定义的头文件
sbit S=P3^4;              //将 S 位定义为 P3.4 引脚
/*****
函数功能：主函数
*****/
void main(void)
{
    // EA=1;                //开总中断
    // ET0=1;              //定时器 T0 中断允许
    TMOD=0x02;            //使用定时器 T0 的模式 2
    TH0=256-156;          //定时器 T0 的高 8 位赋初值

```

```

TL0=256-156; //定时器 T0 的高 8 位赋初值
TR0=1;           //启动定时器 T0
while(1)//无限循环等待查询
{
    while(TF0==0) //如果未计满就等待
    {
        if(S==0) //按键 S 按下接地, 电平为 0
            P1=TL0; //计数器 TL0 加 1 后送 P1 口显示
    }
    TF0=0; //计数器溢出后, 将 TF0 清 0
}
}

```

实例 45: 用定时器 T0 的中断控制 1 位 LED 闪烁

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
sbit D1=P2^0; //将 D1 位定义为 P2.0 引脚
/*****
函数功能: 主函数
*****/
void main(void)
{
    EA=1;           //开总中断
    ET0=1;         //定时器 T0 中断允许
    TMOD=0x01;     //使用定时器 T0 的模式 2
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-46083)%256; //定时器 T0 的高 8 位赋初值
    TR0=1;         //启动定时器 T0
    while(1)//无限循环等待中断
        ;
}
/*****
函数功能: 定时器 T0 的中断服务程序
*****/
void Time0(void) interrupt 1 using 0 // “interrupt” 声明函数为中断服务函数
//其后的 1 为定时器 T0 的中断编号; 0 表示使用第 0 组工作
寄存器
{
    D1=~D1; //按位取反操作, 将 P2.0 引脚输出电平取反
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位重新赋初值
    TL0=(65536-46083)%256; //定时器 T0 的高 8 位重新赋初值
}

```


实例 46：用定时器 T0 的中断实现长时间定时

```
#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
sbit D1=P2^0; //将 D1 位定义为 P2.0 引脚
unsigned char Counter; //设置全局变量，储存定时器 T0 中断次数
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1; //开总中断
    ET0=1; //定时器 T0 中断允许
    TMOD=0x01; //使用定时器 T0 的模式 2
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-46083)%256; //定时器 T0 的高 8 位赋初值
    TR0=1; //启动定时器 T0
    Countor=0; //从 0 开始累计中断次数
    while(1)//无限循环等待中断
        ;
}
/*****
函数功能：定时器 T0 的中断服务程序
*****/
void Time0(void) interrupt 1 using 0 //“interrupt”声明函数为中断服务函数
//其后的 1 为定时器 T0 的中断编号；0 表示使用第 0 组工作
寄存器
{
    Counter++; //中断次数自加 1
    if(Counter==20) //若累计满 20 次，即计时满 1s
    {
        D1=~D1; //按位取反操作，将 P2.0 引脚输出电平取反
        Countor=0; //将 Countor 清 0，重新从 0 开始计数
    }
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位重新赋初值
    TL0=(65536-46083)%256; //定时器 T0 的高 8 位重新赋初值
}
```

实例 47：用定时器 T1 中断控制两个 LED 以不同周期闪烁

```
#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
sbit D1=P2^0; //将 D1 位定义为 P2.0 引脚
```

```

sbit D2=P2^1; //将 D2 位定义为 P2.1 引脚
unsigned char Countor1; //设置全局变量, 储存定时器 T1 中断次数
unsigned char Countor2; //设置全局变量, 储存定时器 T1 中断次数
/*****
函数功能: 主函数
*****/
void main(void)
{
    EA=1;           //开总中断
    ET1=1;         //定时器 T1 中断允许
    TMOD=0x10;     //使用定时器 T1 的模式 1
    TH1=(65536-46083)/256; //定时器 T1 的高 8 位赋初值
    TL1=(65536-46083)%256; //定时器 T1 的高 8 位赋初值
    TR1=1;         //启动定时器 T1
    Countor1=0;    //从 0 开始累计中断次数
    Countor2=0;    //从 0 开始累计中断次数
    while(1)//无限循环等待中断
        ;
}
/*****
函数功能: 定时器 T1 的中断服务程序
*****/
void Time1(void) interrupt 3 using 0 //“interrupt”声明函数为中断服务函数
//其后的 3 为定时器 T1 的中断编号; 0 表示使用第 0 组工作
寄存器
{
    Countor1++; //Countor1 自加 1
    Countor2++; //Countor2 自加 1
    if(Countor1==2) //若累计满 2 次, 即计时满 100ms
    {
        D1=~D1; //按位取反操作, 将 P2.0 引脚输出电平取反
        Countor1=0; //将 Countor1 清 0, 重新从 0 开始计数
    }
    if(Countor2==8) //若累计满 8 次, 即计时满 400ms
    {
        D2=~D2; //按位取反操作, 将 P2.1 引脚输出电平取反
        Countor2=0; //将 Countor1 清 0, 重新从 0 开始计数
    }
    TH1=(65536-46083)/256; //定时器 T1 的高 8 位重新赋初值
    TL1=(65536-46083)%256; //定时器 T1 的高 8 位重新赋初值
}

```

实例 48: 用计数器 T1 的中断控制蜂鸣器发出 1KHz 音频

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
sbit sound=P3^7; //将 sound 位定义为 P3.7 引脚
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1;           //开总中断
    ET1=1;         //定时器 T1 中断允许
    TMOD=0x10;     //TMOD=0001 000B，使用定时器 T1 的模式 1
    TH1=(65536-921)/256; //定时器 T1 的高 8 位赋初值
    TL1=(65536-921)%256; //定时器 T1 的高 8 位赋初值
    TR1=1;        //启动定时器 T1
    while(1)//无限循环等待中断
        ;
}
/*****
函数功能：定时器 T1 的中断服务程序
*****/
void Time1(void) interrupt 3 using 0 // “interrupt” 声明函数为中断服务函数
{
    sound=~sound;
    TH1=(65536-921)/256; //定时器 T1 的高 8 位重新赋初值
    TL1=(65536-921)%256; //定时器 T1 的高 8 位重新赋初值
}

```

实例 49：用定时器 T0 的中断实现"渴望"主题曲的播放

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit sound=P3^7; //将 sound 位定义为 P3.7
unsigned int C; //储存定时器的定时常数
//以下是 C 调低音的音频宏定义
#define l_dao 262 //将“l_dao”宏定义为低音“1”的频率 262Hz
#define l_re 286 //将“l_re”宏定义为低音“2”的频率 286Hz
#define l_mi 311 //将“l_mi”宏定义为低音“3”的频率 311Hz
#define l_fa 349 //将“l_fa”宏定义为低音“4”的频率 349Hz
#define l_sao 392 //将“l_sao”宏定义为低音“5”的频率 392Hz
#define l_la 440 //将“l_a”宏定义为低音“6”的频率 440Hz
#define l_xi 494 //将“l_xi”宏定义为低音“7”的频率 494Hz
//以下是 C 调中音的音频宏定义
#define dao 523 //将“dao”宏定义为中音“1”的频率 523Hz
#define re 587 //将“re”宏定义为中音“2”的频率 587Hz
#define mi 659 //将“mi”宏定义为中音“3”的频率 659Hz

```

```

#define fa 698 //将“fa”宏定义为中音“4”的频率 698Hz
#define sao 784 //将“sao”宏定义为中音“5”的频率 784Hz
#define la 880 //将“la”宏定义为中音“6”的频率 880Hz
#define xi 987 //将“xi”宏定义为中音“7”的频率 523H
//以下是 C 调高音的音频宏定义
#define h_dao 1046 //将“h_dao”宏定义为高音“1”的频率 1046Hz
#define h_re 1174 //将“h_re”宏定义为高音“2”的频率 1174Hz
#define h_mi 1318 //将“h_mi”宏定义为高音“3”的频率 1318Hz
#define h_fa 1396 //将“h_fa”宏定义为高音“4”的频率 1396Hz
#define h_sao 1567 //将“h_sao”宏定义为高音“5”的频率 1567Hz
#define h_la 1760 //将“h_la”宏定义为高音“6”的频率 1760Hz
#define h_xi 1975 //将“h_xi”宏定义为高音“7”的频率 1975Hz
/*****
函数功能：1 个延时单位，延时 200ms
*****/
void delay()
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i,j;
//以下是《渴望》片头曲的一段简谱
    unsigned int code f[]={re,mi,re,dao,l_la,dao,l_la, //每行对应一小节音符
        l_sao,l_mi,l_sao,l_la,dao,
        l_la,dao,sao,la,mi,sao,
        re,
        mi,re,mi,sao,mi,
        l_sao,l_mi,l_sao,l_la,dao,
        l_la,l_la,dao,l_la,l_sao,l_re,l_mi,
        l_sao,
        re,re,sao,la,sao,
        fa,mi,sao,mi,
        la,sao,mi,re,mi,l_la,dao,
        re,
        mi,re,mi,sao,mi,
        l_sao,l_mi,l_sao,l_la,dao,
        l_la,dao,re,l_la,dao,re,mi,

```

```

re,
l_la,dao,re,l_la,dao,re,mi,
re,
0xff}; //以 0xff 作为音符的结束标志

//以下是简谱中每个音符的节拍
//“4”对应 4 个延时单位, “2”对应 2 个延时单位, “1”对应 1 个延时单位
unsigned char code JP[ ]={4,1,1,4,1,1,2,
                        2,2,2,2,8,
                        4,2,3,1,2,2,
                        10,
                        4,2,2,4,4,
                        2,2,2,2,4,
                        2,2,2,2,2,2,2,
                        10,
                        4,4,4,2,2,
                        4,2,4,4,
                        4,2,2,2,2,2,2,
                        10,
                        4,2,2,4,4,
                        2,2,2,2,6,
                        4,2,2,4,1,1,4,
                        10,
                        4,2,2,4,1,1,4,
                        10
                        };
EA=1;          //开总中断
ET0=1;        //定时器 T0 中断允许
TMOD=0x00;    // 使用定时器 T0 的模式 1 (13 位计数器)
while(1)      //无限循环
{
    i=0;      //从第 1 个音符 f[0]开始播放
    while(f[i]!=0xff) //只要没有读到结束标志就继续播放
    {
        C=460830/f[i];
        TH0=(8192-C)/32; //可证明这是 13 位计数器 TH0 高 8 位的
        TL0=(8192-C)%32; //可证明这是 13 位计数器 TL0 低 5 位的
        TR0=1;          //启动定时器 T0
        for(j=0;j<JP[i];j++) //控制节拍数
            delay();        //延时 1 个节拍单位
        TR0=0;          //关闭定时器 T0
        i++;            //播放下一个音符
    }
}

```

```

    }
}
}
/*****
函数功能：定时器 T0 的中断服务子程序，使 P3.7 引脚输出音频的方波
*****/
void Time0(void ) interrupt 1 using 1
{
    sound=!sound;    //将 P3.7 引脚输出电平取反，形成方波
    TH0=(8192-C)/32; //可证明这是 13 位计数器 TH0 高 8 位的赋初值方法
    TL0=(8192-C)%32; //可证明这是 13 位计数器 TL0 低 5 位的赋初值方法
}

```

实例 50-1：输出 50 个矩形脉冲

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit u=P1^4;    //将 u 位定义为 P1.4
/*****
函数功能：延时约 30ms (3*100*100=30 000 μs =30m
*****/
void delay30ms(void)
{
    unsigned char m,n;
    for(m=0;m<100;m++)
        for(n=0;n<100;n++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    u=1;    //初始化输出高电平
    for(i=0;i<50;i++) //输出 50 个矩形脉冲
    {
        u=1;
        delay30ms();
        u=0;
        delay30ms();
    }
    while(1)
}

```

```

        ;//无限循环，防止程序“跑飞”
    }

```

实例 50-2: 计数器 T0 统计外部脉冲数

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
/*****
函数功能：主函数
*****/
void main(void)
{
    TMOD=0x06;      // TMOD=0000 0110B,使用计数器 T0 的模式 2
    EA=1;           //开总中断
    ET0=0;          //不使用定时器 T0 的中断
    TR0=1;          //启动 T0
    TH0=0;          //计数器 T0 高 8 位赋初值
    TL0=0;          //计数器 T0 低 8 位赋初值
    while(1)        //无限循环，不停地将 TL0 计数结果送 P1 口
        P1=TL0;
}

```

实例 51-2: 定时器 T0 的模式 2 测量正脉冲宽度

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
sbit ui=P3^2;       //将 ui 位定义为 P3.0 (INT0) 引脚，表示输入电压
/*****
函数功能：主函数
*****/
void main(void)
{
    TMOD=0x0a;      // TMOD=0000 1010B,使用定时器 T0 的模式 2, GATE 置
1
    EA=1;           //开总中断
    ET0=0;          //不使用定时器 T0 的中断
    TR0=1;          //启动 T0
    TH0=0;          //计数器 T0 高 8 位赋初值
    TL0=0;          //计数器 T0 低 8 位赋初值
    while(1)        //无限循环，不停地将 TL0 计数结果送 P1 口
    {
        while(ui==0) //INT0 为低电平，T0 不能启动
            ;
        TL0=0;       //INT0 为高电平，启动 T0 计时，所以将 TL0 清 0
    }
}

```

```

        while(ui==1) //在 INTO 高电平期间，等待，计时
            ;
        P1=TL0;    //将计时结果送 P1 口显示
    }
}

```

实例 52：用定时器 T0 控制输出高低宽度不同的矩形波

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
sbit u=P3^0;        //将 u 位定义为 P3.0，从该引脚输出矩形脉冲
unsigned char Countor; //设置全局变量，储存负跳变累计数
/*****
函数功能：延时约 30ms (3*100*100=30 000 μs =30ms)
*****/
void delay30ms(void)
{
    unsigned char m,n;
    for(m=0;m<100;m++)
        for(n=0;n<100;n++)
            ;
}

/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    EA=1;    //开放总中断
    EX0=1;  //允许使用外中断
    IT0=1;  //选择负跳变来触发外中断
    Countor=0;
    for(i=0;i<100;i++) //输出 100 个负跳变
    {
        u=1;
        delay30ms();
        u=0;
        delay30ms();
    }
    while(1)
        ; //无限循环，防止程序跑飞
}

```



```

/*****
函数功能：外中断 T0 的中断服务程序
*****/
void int0(void) interrupt 0 using 0 //外中断 0 的中断编号为 0
{
    Countor++;
    P1=Countor;
}

```

实例 53：用外中断 0 的中断方式进行数据采集

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit S=P3^2; //将 S 位定义为 P3.2,
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1; //开放总中断
    EX0=1; //允许使用外中断
    IT0=1; //选择负跳变来触发外中断
    P1=0xff;
    while(1)
        ;//无限循环，防止程序跑飞
}
/*****
函数功能：外中断 T0 的中断服务程序
*****/
void int0(void) interrupt 0 using 0 //外中断 0 的中断编号为 0
{
    P1=~P1; //每产生一次中断请求，P1 取反一次。
}

```

实例 54-1：输出负脉宽为 200 微秒的方波

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit u=P1^4; //将 u 位定义为 P1.4
/*****
函数功能：主函数
*****/

```

```

void main(void)
{
    TMOD=0x02;           //TMOD=0000 0010B, 使用定时器 T0 的模式 2
    EA=1;                //开总中断
    ET0=1;               //定时器 T0 中断允许
    TH0=256-200;        //定时器 T0 的高 8 位赋初值
    TL0=256-200;        //定时器 T0 的高 8 位赋初值
    TR0=1;               //启动定时器 T0
    while(1)             //无限循环, 等待中断
        ;
}
/*****
函数功能: 定时器 T0 的中断服务程序
*****/
void Time0(void) interrupt 1 using 0 //“interrupt”声明函数为中断服务函数
{
    u=~u; //将 P1.4 引脚输出电平取反, 产生方波
}

```

实例 54-2: 测量负脉冲宽度

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit u=P3^2; //将 u 位定义为 P3.2
/*****
函数功能: 主函数
*****/
void main(void)
{
    TMOD=0x02; //TMOD=0000 0010B,使用定时器 T0 的模式 2
    EA=1; //开放总中断
    EX0=1; //允许使用外中断
    IT0=1; //选择负跳变来触发外中断
    ET0=1; //允许定时器 T0 中断
    TH0=0; //定时器 T0 赋初值 0
    TL0=0; //定时器 T0 赋初值 0
    TR0=0; //先关闭 T0
    while(1)
        ;//无限循环, 不停检测输入负脉冲宽度
}
/*****
函数功能: 外中断 0 的中断服务程序
*****/

```

```

*****/
void int0(void) interrupt 0 using 0 //外中断 0 的中断编号为 0
{
    TR0=1; //外中断一到来，即启动 T0 计时
    TL0=0; //从 0 开始计时
    while(u==0) //低电平时，等待 T0 计时
        ;
    P1=TL0; //将结果送 P1 口显示
    TR0=0; //关闭 T0
}

```

实例 55：方式 0 控制流水灯循环点亮

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
#include<intrins.h> //包含函数_nop_() 定义的头文件
unsigned char code Tab[]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};//
流水灯控制码，该数组被定义为全局变量
sbit P17=P1^7;
/*****
函数功能：延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能：发送一个字节的数
*****/
void Send(unsigned char dat)
{
    P17=0; //P1.7 引脚输出清 0 信号，对 74LS164 清 0
    _nop_(); //延时一个机器周期
    _nop_(); //延时一个机器周期，保证清 0 完成
    P17=1; //结束对 74LS164 的清 0
    SBUF=dat; //将数据写入发送缓冲器，启动发送
    while(TI==0) //若没有发送完毕，等待
        ;
    TI=0; //发送完毕，TI 被置“1”，需将其清 0
}
/*****

```

函数功能：主函数

```

*****/
void main(void)
{
    unsigned char i;
    SCON=0x00; //SCON=0000 0000B, 使串行口工作于方式 0
    while(1)
    {
        for(i=0;i<8;i++)
        {
            Send(Tab[i]); //发送数据
            delay();      //延时
        }
    }
}

```

实例 56-1：数据发送程序

```

#include<reg51.h> //包含单片机寄存器的头文件
unsigned char code Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
//流水灯控制码，该数组被定义为全局变量

```

```

/*****
函数功能：向 PC 发送一个字节数据
*****/
void Send(unsigned char dat)
{
    SBUF=dat;
    while(TI==0)
        ;
    TI=0;
}
/*****
函数功能：延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****

```

函数功能：主函数

```

*****/
void main(void)
{
    unsigned char i;
    TMOD=0x20; //TMOD=0010 0000B, 定时器 T1 工作于方式 2
    SCON=0x40; //SCON=0100 0000B, 串口工作方式 1
    PCON=0x00; //PCON=0000 0000B, 波特率 9600
    TH1=0xfd; //根据规定给定时器 T1 赋初值
    TL1=0xfd; //根据规定给定时器 T1 赋初值
    TR1=1; //启动定时器 T1
    while(1)
    {
        for(i=0;i<8;i++) //模拟检测数据
        {
            Send(Tab[i]); //发送数据 i
            delay(); //50ms 发送一次检测数据
        }
    }
}

```

实例 56-2：数据接收程序

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能：接收一个字节数据
*****/
unsigned char Receive(void)
{
    unsigned char dat;
    while(RI==0) //只要接收中断标志位 RI 没有被置“1”
        ; //等待，直至接收完毕（RI=1）
    RI=0; //为了接收下一帧数据，需将 RI 清 0
    dat=SBUF; //将接收缓冲器中的数据存于 dat
    return dat;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    TMOD=0x20; //定时器 T1 工作于方式 2
    SCON=0x50; //SCON=0101 0000B, 串口工作方式 1,允许接收（REN=1）

```

```

PCON=0x00; //PCON=0000 0000B, 波特率 9600
TH1=0xfd; //根据规定给定时器 T1 赋初值
TL1=0xfd; //根据规定给定时器 T1 赋初值
TR1=1;    //启动定时器 T1
REN=1;    //允许接收
while(1)
{
    P1=Receive(); //将接收到的数据送 P1 口显示
}
}

```

实例 57-1: 数据发送程序

```

#include<reg51.h>          //包含单片机寄存器的头文件
sbit p=PSW^0;

unsigned char code Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
//流水灯控制码, 该数组被定义为全局变量
/*****
函数功能: 向 PC 发送一个字节数据
*****/
void Send(unsigned char dat)
{
    ACC=dat;
    TB8=p;
    SBUF=dat;
    while(TI==0)
        ;
    TI=0;
}
/*****
函数功能: 延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能: 主函数
*****/

```

```

void main(void)
{
    unsigned char i;
    TMOD=0x20; //TMOD=0010 0000B, 定时器 T1 工作于方式 2
    SCON=0xc0; //SCON=1100 0000B, 串口工作方式 3,
                //SM2 置 0, 不使用多机通信, TB8 置 0
    PCON=0x00; //PCON=0000 0000B, 波特率 9600
    TH1=0xfd; //根据规定给定时器 T1 赋初值
    TL1=0xfd; //根据规定给定时器 T1 赋初值
    TR1=1;    //启动定时器 T1
    while(1)
    {
        for(i=0;i<8;i++) //模拟检测数据
        {
            Send(Tab[i]); //发送数据 i
            delay(); //50ms 发送一次检测数据
        }
    }
}

```

实例 57-2: 数据接收程序

```

#include<reg51.h> //包含单片机寄存器的头文件
sbit p=PSW^0;
/*****
函数功能: 接收一个字节数据
*****/
unsigned char Receive(void)
{
    unsigned char dat;
    while(RI==0) //只要接收中断标志位 RI 没有被置"1"
        ; //等待, 直至接收完毕 (RI=1)
    RI=0; //为了接收下一帧数据, 需将 RI 清 0
    ACC=SBUF; //将接收缓冲器中的数据存于 dat
    if(RB8==p)
    {
        dat=ACC;
        return dat;
    }
}
/*****
函数功能: 主函数
*****/

```

```

void main(void)
{
    TMOD=0x20; //定时器 T1 工作于方式 2
    SCON=0xd0; //SCON=1101 0000B, 串口工作方式 1,允许接收 (REN=1)
    PCON=0x00; //PCON=0000 0000B, 波特率 9600
    TH1=0xfd; //根据规定给定时器 T1 赋初值
    TL1=0xfd; //根据规定给定时器 T1 赋初值
    TR1=1; //启动定时器 T1
    REN=1; //允许接收
    while(1)
    {
        P1=Receive(); //将接收到的数据送 P1 口显示
    }
}

```

实例 58：单片机向 PC 发送数据

```

#include<reg51.h> //包含单片机寄存器的头文件
unsigned char code Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
//流水灯控制码, 该数组被定义为全局变量
/*****
函数功能: 向 PC 发送一个字节数据
*****/
void Send(unsigned char dat)
{
    SBUF=dat;
    while(TI==0)
        ;
    TI=0;
}
/*****
函数功能: 延时约 150ms
*****/
void delay(void)
{
    unsigned char m,n;
    for(m=0;m<200;m++)
        for(n=0;n<250;n++)
            ;
}
/*****
函数功能: 主函数
*****/

```



```

void main(void)
{
    unsigned char i;
    TMOD=0x20; //TMOD=0010 0000B, 定时器 T1 工作于方式 2
    SCON=0x40; //SCON=0100 0000B, 串口工作方式 1
    PCON=0x00; //PCON=0000 0000B, 波特率 9600
    TH1=0xfd; //根据规定给定时器 T1 赋初值
    TL1=0xfd; //根据规定给定时器 T1 赋初值
    TR1=1; //启动定时器 T1
    while(1)
    {
        for(i=0;i<8;i++) //模拟检测数据
        {
            Send(Tab[i]); //发送数据 i
            delay(); //150ms 发送一次数据
        }
    }
}

```

实例 59：单片机接收 PC 发出的数据

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能：接收一个字节数据
*****/
unsigned char Receive(void)
{
    unsigned char dat;
    while(RI==0) //只要接收中断标志位 RI 没有被置“1”
        ; //等待，直至接收完毕（RI=1）
    RI=0; //为了接收下一帧数据，需将 RI 清 0
    dat=SBUF; //将接收缓冲器中的数据存于 dat
    return dat;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    TMOD=0x20; //定时器 T1 工作于方式 2
    SCON=0x50; //SCON=0101 0000B, 串口工作方式 1,允许接收（REN=1）
    PCON=0x00; //PCON=0000 0000B, 波特率 9600
    TH1=0xfd; //根据规定给定时器 T1 赋初值

```

```

    TL1=0xfd; //根据规定给定时器 T1 赋初值
    TR1=1;    //启动定时器 T1
    REN=1;    //允许接收
while(1)
{
    P1=Receive(); //将接收到的数据送 P1 口显示
}
}

```

数码管显示

实例 60：用 LED 数码显示数字 5

```

#include<reg51.h>           // 包含 51 单片机寄存器定义的头文件
void main(void)
{
    P2=0xfe; //P2.0 引脚输出低电平，数码显示器接通电源准备点亮
    P0=0x92; //让 P0 口输出数字"5"的段码 92H
}

```

实例 61：用 LED 数码显示器循环显示数字 0~9

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
/*****
函数功能：延时函数，延时一段时间
*****/
void delay(void)
{
    unsigned char i,j;
    for(i=0;i<255;i++)
        for(j=0;j<255;j++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;

```

```

unsigned char code
Tab[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//数码管显示 0~9 的段码表，程序运行中当数组值不发生变化
时，
//前面加关键字 code ，可以大大节约单片机的存储空间
P2=0xfe; //P2.0 引脚输出低电平，数码显示器 DS0 接通电源工作
while(1) //无限循环
{
    for(i=0;i<10;i++)
    {
        P0=Tab[i]; //让 P0 口输出数字的段码 92H
        delay(); //调用延时函数
    }
}
}

```

实例 62：用数码管慢速动态扫描显示数字"1234"

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
void delay(void) //延时函数，延时一段时间
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}
void main(void)
{
    while(1) //无限循环
    {
        P2=0xfe; //P2.0 引脚输出低电平，DS0 点亮
        P0=0xf9; //数字 1 的段码
        delay();
        P2=0xfd; //P2.1 引脚输出低电平，DS1 点亮
        P0=0xa4; //数字 2 的段码
        delay();
        P2=0xfb; //P2.2 引脚输出低电平，DS2 点亮
        P0=0xb0; //数字 3 的段码
        delay();
        P2=0xf7; //P2.3 引脚输出低电平，DS3 点亮
        P0=0x99; //数字 4 的段码
        delay();
        P2=0xff;
    }
}

```

```

}
}

```

实例 63：用 LED 数码显示器伪静态显示数字 1234

```

#include<reg51.h>           // 包含 51 单片机寄存器定义的头文件
void delay(void)           //延时函数，延时约 0.6 毫秒
{
    unsigned char i;
    for(i=0;i<200;i++)
        ;
}

void main(void)
{
    while(1)    //无限循环
    {
        P2=0xfe;    //P2.0 引脚输出低电平，DS0 点亮
        P0=0xf9;    //数字 1 的段码
        delay();
        P2=0xfd;    //P2.1 引脚输出低电平，DS1 点亮
        P0=0xa4;    //数字 2 的段码
        delay();
        P2=0xfb;    //P2.2 引脚输出低电平，DS2 点亮
        P0=0xb0;    //数字 3 的段码
        delay();
        P2=0xf7;    //P2.3 引脚输出低电平，DS3 点亮
        P0=0x99;    //数字 4 的段码
        delay();
        P2=0xff;
    }
}

```

实例 64：用数码管显示动态检测结果

```

#include<reg51.h>    // 包含 51 单片机寄存器定义的头文件
#include<stdlib.h>   //包含随机函数 rand()的定义文件
unsigned char i;     //记录中断次数
unsigned int x;      //随机检测的数据
unsigned char code
Tab[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90}; //数码管显
示 0~9 的段码表

```

```

/*****
函数功能：快速动态扫描延时，延时约 0.9 毫秒
*****/
void delay(void)
{
    unsigned int i;
    for(i=0;i<300;i++)
        ;
}
/*****
函数功能：4 位数的数码显示器显示
入口参数：k
出口参数：无
*****/
void display(unsigned int k)
{
    P2=0xfe; //即 P2=1111 1110B, P2.0 引脚输出低电平，数码显示器 DS0
    接通电源
    P0=Tab[k/1000]; //显示千位
    delay();
    P2=0xfd; //即 P2=1111 1101B, P2.1 引脚输出低电平，数码显示器 DS1
    接通电源
    P0=Tab[(k%1000)/100]; //显示百位
    delay();
    P2=0xfb; //即 P2=1111 1011B, P2.2 引脚输出低电平，数码显示器 DS2
    接通电源
    P0=Tab[(k%100)/10]; //显示十位
    delay();
    P2=0xf7; //即 P2=1111 0111B, P2.3 引脚输出低电平，数码显示器 DS3
    接通电源
    P0=Tab[k%10]; //显示个位
    delay();
    P2=0xff; //关闭所有显示器
}

void main(void) //主函数
{
    TMOD=0x01; //使用定时器 T0
    TH0=(65536-46083)/256; //将定时器计时时间设定为 46083×
    1.085 微秒=50000 微秒=50 毫秒
    TL0=(65536-46083)%256;
    EA=1; //开启总中断
}

```

```

        ET0=1;           //定时器 T0 中断允许
        TR0=1;           //启动定时器 T0 开始运行

    while(1)
    {
        display(x);      //调用检测结果的显示程序
    }

}

/*****
    函数功能：定时器 T0 的中断服务程序
*****/

void Time0(void) interrupt 1 using 1
{
    TR0=0;      //关闭定时器 T0
    i++;        //每来一次中断，i 自加 1
    if(i==20)   //够 20 次中断，即 1 秒钟进行一次检测结果采样
    {
        x=rand()/10;    //随机产生一个从 0 到 32767 的整数，再将其除以
        10，获得一个随机 4 位数，模拟检测结果
        i=0;          //将 i 清 0，重新统计中断次数
    }
    TH0=(65536-46083)/256; //重新给计数器 T0 赋初值
    TL0=(65536-46083)%256;
    TR0=1;        //启动定时器 T0
}

```

实例 65：数码秒表设计

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
unsigned char code
Tab[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//数码管显示 0~9 的段码表
unsigned char int_time;    //记录中断次数
unsigned char second;     //储存秒
/*****
    函数功能：快速动态扫描延时,延时约 0.6 毫秒
*****/

void delay(void)
{
    unsigned char i;

```

```

    for(i=0;i<200;i++)
        ;
}
/*****
函数功能：显示秒
入口参数：k
出口参数：无
*****/
void DisplaySecond(unsigned char k)
{
    P2=0xfb;          //P2.6 引脚输出低电平， DS6 点亮
    P0=Tab[k/10];     //显示十位
    delay();

    P2=0xf7;          //P2.7 引脚输出低电平， DS7 点亮
    P0=Tab[k%10];     //显示个位
    delay();
    P2=0xff;          //关闭所有数码管

}

void main(void)      //主函数
{
    TMOD=0x01;        //使用定时器 T0
    TH0=(65536-46083)/256; //将定时器计时时间设定为 46083×1.085
    //微秒
    //=50000 微秒=50 毫秒
    TL0=(65536-46083)%256;
    EA=1;              //开启总中断
    ET0=1;             //定时器 T0 中断允许
    TR0=1;             //启动定时器 T0 开始运行
    int_time=0;        //中断次数初始化
    second=0;          //秒初始化
    while(1)
    {
        DisplaySecond(second); //调用秒的显示子程序
    }
}
/*****
//函数功能：定时器 T0 的中断服务程序
*****/
void interserve(void ) interrupt 1 using 1

```

```

{
    TR0=0;    //关闭定时器 T0
    int_time++;    //每来一次中断,中断次数 int_time 自加 1
    if(int_time==20)    //够 20 次中断,即 1 秒钟进行一次检测结果采样
    {
        int_time=0;    //中断次数清 0
        second++;    //秒加 1
        if(second==60)
            second =0; //秒等于 60 就返回 0
    }
    TH0=(65536-46083)/256;    //重新给计数器 T0 赋初值
    TL0=(65536-46083)%256;
    TR0=1;    //启动定时器 T0
}

```

实例 66：数码时钟设计

```

#include<reg51.h>    // 包含 51 单片机寄存器定义的头文件
unsigned char Tab[ ]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//control shape
unsigned char port[8]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
unsigned char int_time ; //中断次数计数变量
unsigned char second;    //秒计数变量
unsigned char minute;    //分钟计数变量
unsigned char hour;    //小时计数变量

```

```

////////////////////////////////////

```

```

void delay(void)    //延时函数，延时约 0.6ms
{
    unsigned char j;
    for(j=0;j<200;j++)
        ;
}

```

```

/*****

```

函数功能：显示秒的子程序

入口参数：s

```

*****/

```

```

void DisplaySecond(unsigned char s)
{
    P2=0xbf;    //P2.6 引脚输出低电平， DS6 点亮
    P0=Tab[s/10];    //显示十位
}

```



```

delay();

P2=0x7f;      //P2.7 引脚输出低电平, DS7 点亮
P0=Tab[s%10]; //显示个位
delay();
P2=0xff;     //关闭所有数码管

}

/*****
函数功能: 显示分钟的子程序
入口参数: m
*****/
void DisplayMinute(unsigned char m)
{
    P2=0xf7; // P2.3 引脚输出低电平, DS3 点亮
    P0=Tab[m/10]; //显示个位
    delay();
    P2=0xef; // P2.4 引脚输出低电平, DS4 点亮
    P0=Tab[m%10];
    delay();
    P2=0xdf; //P2.5 引脚输出低电平, DS5 点亮
    P0=0xbf; //分隔符“-”的段码
    delay();
    P2=0xff; //关闭所有数码管

}

/*****
函数功能: 显示小时的子程序
入口参数: h
*****/
void DisplayHour(unsigned char h)
{
    P2=0xfe; //P2.0 引脚输出低电平, DS0 点亮
    P0=Tab[h/10]; //显示十位
    delay();

    P2=0xfd; //P2.1 引脚输出低电平, DS1 点亮
    P0=Tab[h%10]; //显示个位
    delay();
    P2=0xfb; //P2.2 引脚输出低电平, DS2 点亮

```

```

    P0=0xbf;      //分隔符“-”的段码
    delay();
    P2=0xff;     //关闭所有数码管

}
/*****
函数功能：主函数
*****/

void main(void)
{

    TMOD=0x01;      //使用定时器 T0
    EA=1;           //开中断总允许
    ET0=1;         //允许 T0 中断
    TH0=(65536-46083)/256; //定时器高八位赋初值
    TL0=(65536-46083)%256; //定时器低八位赋初值
    TR0=1;
    int_time=0;    //中断计数变量初始化
    second=0;     //秒计数变量初始化
    minute=0;     //分钟计数变量初始化
    hour=0;       //小时计数变量初始化

    while(1)
    {
        DisplaySecond(second); //调用秒显示子程序
        delay();
        DisplayMinute(minute); //调用分钟显示子程序
        delay();
        DisplayHour(hour);
        delay();
    }
}

/*****
函数功能：定时器 T0 的中断服务子程序
*****/

void interserve(void ) interrupt 1 using 1 //using Time0
{
    int_time++;
    if(int_time==20)
    {
        int_time=0; //中断计数变量清 0
        second++;  //秒计数变量加 1
    }
}

```

```

    }
    if(second==60)
    {
        second=0;    //如果秒计满 60，将秒计数变量清 0
        minute++;    //分钟计数变量加 1
    }
    if(minute==60)
    {
        minute=0;    //如果分钟计满 60，将分钟计数变量
清 0
        hour++;      //小时计数变量加 1
    }
    if(hour==24)
    {
        hour=0;      //如果小时计满 24，将小时计数变量
清 0
    }

    TH0=(65536-46083)/256; //定时器重新赋初值
    TL0=(65536-46083)%256;

}

```

实例 67：用 LED 数码管显示计数器 T0 的计数值

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit S=P3^2; //将 S 位定义为 P3.2 引脚
unsigned char Tab[ ]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//段码表
unsigned char x;
/*****
函数功能： 延时约 0.6ms
*****/
void delay(void)
{
    unsigned char j;
    for(j=0;j<200;j++)
        ;
}

/*****
函数功能： 显示计数次数的子程序
*****/

```

入口参数: x

```

*****/
void Display(unsigned char x)
{
    P2=0xf7;          //P2.6 引脚输出低电平, DS6 点亮
    P0=Tab[x/10];    //显示十位
    delay();
    P2=0xfb;          //P2.7 引脚输出低电平, DS7 点亮
    P0=Tab[x%10];    //显示个位
    delay();

}

```

函数功能: 主函数

*****/

```

void main(void)
{
    EA=1;  //开放总中断
    EX0=1; //允许使用外中断
    IT0=1; //选择负跳变来触发外中断
    x=0;

    while(1)
        Display(x);

}

```

函数功能: 外中断 T0 的中断服务程序

*****/

```

void int0(void) interrupt 0 using 0 //外中断 0 的中断编号为 0
{
    x++;
    if(x==100)
        x=0;

}

```

实例 68: 静态显示数字“59”

```
#include<reg51.h> //包含 51 单片机寄存器定义的头文件
```

函数功能: 主函数

```

*****/
void main(void)
{
    P0=0x92;    //将数字 5 的段码送 P0 口
    P1=0x90;    //将数字 9 的段码送 P1 口
    while(1)    //无限循环，防止程序跑飞
    ;
}

```

键盘控制

实例 69：无软件消抖的独立式键盘输入实验

```

#include<reg51.h>    // 包含 51 单片机寄存器定义的头文件
sbit S1=P1^4;        //将 S1 位定义为 P1.4 引脚
sbit LED0=P3^0;      //将 LED0 位定义为 P3.0 引脚
void main(void)     //主函数
{
    LED0=0;          //P3.0 引脚输出低电平
    while(1)
    {
        if(S1==0)    //P1.4 引脚输出低电平，按键 S1 被按下
            LED0=!LED0; //P3.0 引脚取反
    }
}

```

实例 70：软件消抖的独立式键盘输入实验

```

#include<reg51.h>    // 包含 51 单片机寄存器定义的头文件
sbit S1=P1^4;        //将 S1 位定义为 P1.4 引脚
sbit LED0=P3^0;      //将 LED0 位定义为 P3.0 引脚
/*****
函数功能：延时约 30ms
*****/
void delay(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)

```

```

        for(j=0;j<100;j++)
            ;
    }
    /*****
    函数功能：主函数
    *****/
    void main(void) //主函数
    {
        LED0=0;      //P3.0 引脚输出低电平
        while(1)
        {
            if(S1==0) //P1.4 引脚输出低电平，按键 S1 被按下
            {
                delay(); //延时一段时间再次检测
                if(S1==0) // 按键 S1 的确被按下
                    LED0=!LED0; //P3.0 引脚取反
            }
        }
    }
}

```

实例 71：CPU 控制的独立式键盘扫描实验

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit S1=P1^4;     //将 S1 位定义为 P1.4 引脚
sbit S2=P1^5;     //将 S2 位定义为 P1.5 引脚
sbit S3=P1^6;     //将 S3 位定义为 P1.6 引脚
sbit S4=P1^7;     //将 S4 位定义为 P1.7 引脚
unsigned char keyval; //储存按键值
/*****
函数功能：流水灯延时
*****/
void led_delay(void)
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}

/*****
函数功能：软件消抖延时
*****/
void delay30ms(void)

```

```

{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<100;j++)
            ;
}
/*****
函数功能：正向流水点亮 LED
*****/
void forward(void)
{
    P3=0xfe;           //第一个灯亮
    led_delay();
    P3=0xfd;           //第二个灯亮
    led_delay();
    P3=0xfb;           //第三个灯亮
    led_delay();
    P3=0xf7;           //第四个灯亮
    led_delay();
    P3=0xef;           //第五个灯亮
    led_delay();
    P3=0xdf;           //第六个灯亮
    led_delay();
    P3=0xbf;           //第七个灯亮
    led_delay();
    P3=0x7f;           //第八个灯亮
    led_delay();
    P3=0xff;
    P3=0xfe;           //第一个灯亮
    led_delay();
}
/*****
函数功能：反向流水点亮 LED
*****/
void backward(void)
{
    P3=0x7f;           //第八个灯亮
    led_delay();
    P3=0xbf;           //第七个灯亮
    led_delay();
    P3=0xdf;           //第六个灯亮
    led_delay();
    P3=0xef;           //第五个灯亮
    led_delay();
}

```

```

        P3=0xf7;           //第四个灯亮
        led_delay();
        P3=0xfb;           //第三个灯亮
        led_delay();
        P3=0xfd;           //第二个灯亮
        led_delay();
        P3=0xfe;           //第一个灯亮
        led_delay();

    }
/*****
函数功能：关闭所有 LED
*****/
void stop(void)
{
    P3=0xff;
}
/*****
函数功能：闪烁点亮 LED
*****/
void flash(void)
{
    P3=0xff;
    led_delay();
    P3=0x00;
    led_delay();
}
/*****
函数功能：键盘扫描子程序
*****/
void key_scan(void)
{
    if((P1&0xf0)!=0xf0) //第一次检测到有键按下
    {
        delay30ms(); //延时 20ms 再去检测
        if(S1==0) //按键 S1 被按下
            keyval=1;
        if(S2==0) //按键 S2 被按下
            keyval=2;
        if(S3==0) //按键 S3 被按下
            keyval=3;
        if(S4==0) //按键 S4 被按下
            keyval=4;
    }
}

```



```

}

/*****
函数功能：主函数
*****/
void main(void) //主函数
{
    keyval=0;           //按键值初始化为 0，什么也不做
    while(1)
    {
        key_scan();
        switch(keyval)
        {
            case 1:forward();
                break;
            case 2:backward();
                break;
            case 3:stop();
                break;
            case 4: flash();
                break;
        }
    }
}

```

实例 72：定时器中断控制的独立式键盘扫描实验

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit S1=P1^4;     //将 S1 位定义为 P1.4 引脚
sbit S2=P1^5;     //将 S2 位定义为 P1.5 引脚
sbit S3=P1^6;     //将 S3 位定义为 P1.6 引脚
sbit S4=P1^7;     //将 S4 位定义为 P1.7 引脚
unsigned char keyval; //储存按键值
/*****
函数功能：流水灯延时
*****/
void led_delay(void)
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}

```

```

}

/*****
函数功能：软件消抖延时
*****/
void delay20ms(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)
            ;
}

/*****
函数功能：正向流水点亮 LED
*****/
void forward(void)
{
    P3=0xfe;          //第一个灯亮
    led_delay();
    P3=0xfd;          //第二个灯亮
    led_delay();
    P3=0xfb;          //第三个灯亮
    led_delay();
    P3=0xf7;          //第四个灯亮
    led_delay();
    P3=0xef;          //第五个灯亮
    led_delay();
    P3=0xdf;          //第六个灯亮
    led_delay();
    P3=0xbf;          //第七个灯亮
    led_delay();
    P3=0x7f;          //第八个灯亮
    led_delay();
    P3=0xff;
    P3=0xfe;          //第一个灯亮
    led_delay();
}

/*****
函数功能：反向流水点亮 LED
*****/
void backward(void)
{
    P3=0x7f;          //第八个灯亮
    led_delay();

```

```

        P3=0xbf;           //第七个灯亮
    led_delay();
        P3=0xdf;           //第六个灯亮
    led_delay();
        P3=0xef;           //第五个灯亮
    led_delay();
        P3=0xf7;           //第四个灯亮
    led_delay();
        P3=0xfb;           //第三个灯亮
    led_delay();
        P3=0xfd;           //第二个灯亮
    led_delay();
        P3=0xfe;           //第一个灯亮
    led_delay();

}
/*****
函数功能：关闭所有 LED
*****/
void stop(void)
{
    P3=0xff;    //关闭 8 个 LED
}
/*****
函数功能：闪烁点亮 LED
*****/
void flash(void)
{
    P3=0xff;    //关闭 8 个 LED
    led_delay();
    P3=0x00;    //点亮 8 个 LED
    led_delay();
}

/*****
函数功能：主函数
*****/
void main(void) //主函数
{
    TMOD=0x01;    //使用定时器 T0 的模式 1
    EA=1;         //开总中断
    ET0=1;        //定时器 T0 中断允许
    TR0=1;        //启动定时器 T0

```

```

    TH0=(65536-1000)/256;    //定时器 T0 赋初值，每计数 200 次（217 微秒）
    发送一次中断请求
    TL0=(65536-1000)%256;    //定时器 T0 赋初值
    keyval=0;                //按键值初始化为 0，什么也不做
    while(1)
    {
        switch(keyval)
        {
            case 1:forward();
                break;
            case 2:backward();
                break;
            case 3:stop();
                break;
            case 4: flash();
                break;
        }
    }
}

/*****
函数功能：定时器 T0 的中断服务子程序
*****/
void Time0_serve(void) interrupt 1 using 1
{
    if((P1&0xf0)!=0xf0)    //第一次检测到有键按下
    {
        delay20ms();    //延时 20ms 再去检测
        if(S1==0)    //按键 S1 被按下
            keyval=1;
        if(S2==0)    //按键 S2 被按下
            keyval=2;
        if(S3==0)    //按键 S3 被按下
            keyval=3;
        if(S4==0)    //按键 S4 被按下
            keyval=4;
    }
    TH0=(65536-1000)/256;
    TL0=(65536-1000)%256;
}

```

实例 73：独立式键盘控制的 4 级变速流水灯

```

#include<reg51.h>           // 包含 51 单片机寄存器定义的头文件
unsigned char speed;       // 储存流水灯的流动速度
sbit S1=P1^4;             // 位定义 S1 为 P1.4
sbit S2=P1^5;             // 位定义 S2 为 P1.5
sbit S3=P1^6;             // 位定义 S3 为 P1.6
sbit S4=P1^7;             // 位定义 S4 为 P1.7
/*****

函数功能：延时 20ms 的子程序
*****/

void delay20ms(void)       //3*i*j+2*i=3*100*60+2*100=20000μs=20ms;
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)
            ;
}
/*****

函数功能：延时可调子程序
入口参数：x
*****/

void delay(unsigned char x)
{
    unsigned char k;
    for(k=0;k<x;k++)
        delay20ms();
}
/*****

函数功能：主函数
*****/

void main(void)
{
    TMOD=0x02;           //使用定时器 T0 的模式 2
    EA=1;                //开总中断
    ET0=1;               //定时器 T0 中断允许
    TR0=1;               //定时器 T0 开始运行
    TH0=256-200;         //定时器 T0 赋初值,每 200 微妙来 1 次中断请求
    TL0=256-200;
    speed=3;             //默认流水灯流水点亮延时 20ms×3=60ms
    while(1)
    {
        P3=0xfe;         //第一个灯亮
        delay(speed);    //调用延时可调子程序
        P3=0xfd;         //第二个灯亮
        delay(speed);
    }
}

```

```

        P3=0xfb;           //第三个灯亮
        delay(speed);
        P3=0xf7;         //第四个灯亮
        delay(speed);
        P3=0xef;         //第五个灯亮
        delay(speed);
        P3=0xdf;         //第六个灯亮
        delay(speed);
        P3=0xbf;         //第七个灯亮
        delay(speed);
        P3=0x7f;         //第八个灯亮
        delay(speed);
        P3=0xff;
    }
}
/*****
函数功能：定时器 T0 的中断服务子程序,进行键盘扫描
*****/
void intersev(void) interrupt 1 using 1
{
    TR0=0;                //关闭定时器 T0/
    P1=0xff;              //将 P1 口的均置高电平"1"
    if((P1&0xf0)!=0xf0)   //如果有键按下
    {
        delay20ms();      //延时 20ms,软件消抖
        if((P1&0xf0)!=0xf0) //确实有键按下
        {
            if(S1==0)     //如果是按键 S1 按下
                speed=5;  //流水灯流水点亮延时 20ms×5=100ms
            if(S2==0)     //如果是按键 S2 按下
                speed=10; //流水灯流水点亮延时 20ms×10=200ms
            if(S3==0)     //如果是按键 S3 按下
                speed=25; //流水灯流水点亮延时 20ms×25=500ms
            if(S4==0)     //如果是按键 S4 按下
                speed=50; //流水灯流水点亮延时 20ms×50=1000ms
        }
    }
    TR0=1;                //启动定时器 T0
}

```

实例 74：独立式键盘的按键功能扩展：“以一当四”

```
#include<reg51.h>           // 包含 51 单片机寄存器定义的头文件
```

```

unsigned char ID;    //储存流水灯的流动速度
sbit S1=P1^4;      //位定义 S1 为 P1.4

/*****
函数功能：延时子程序
*****/
void delay(void)    //因为仅对一个按键扫描，所以延时时间较长约 200ms
{
    unsigned char i,j;
    for(i=0;i<200;i++)
        for(j=0;j<100;j++)
            ;
}

/*****
函数功能：主函数
*****/
void main(void)
{
    TMOD=0x02;      //使用定时器 T0 的模式 2
    EA=1;          //开总中断
    ET0=1;         //定时器 T0 中断允许
    TR0=1;         //定时器 T0 开始运行
    TH0=256-200;   //定时器 T0 赋初值，每 200 微妙来 1 次中断请求
    TL0=256-200;

    ID=0;
    while(1)
    {
        switch(ID)
        {
            case 0: P3=0xfe;
                    break;
            case 1: P3=0xfd;
                    break;
            case 2: P3=0xfb;
                    break;
            case 3: P3=0xf7;
                    break;
        }
    }
}
/*****

```

函数功能：定时器 T0 的中断服务子程序，进行键盘扫描

```

*****/
void intersev(void) interrupt 1 using 1
{
    TR0=0; //关闭定时器 T0
    P1=0xff;
    if(S1==0) //如果是按键 S1 按下
    {
        delay(); //延时 20ms，软件消抖
        if(S1==0) //如果是按键 S1 按下
            ID=ID+1;
    }
    if(ID==4)
        ID=0;

    TR0=1; //启动定时器 T0
}

```

实例 75：独立式键盘调时的数码时钟实验

```

#include<reg51.h> // 包含 51 单片机寄存器定义的头文件
unsigned char code
Tab[ ]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90}; //数字 0~9 的
段码
unsigned char int_time ; //中断次数计数变量
unsigned char second; //秒计数变量
unsigned char minute; //分钟计数变量
unsigned char hour; //小时计数变量

sbit S1=P1^4; //将 S1 位定义为 P1.4
sbit S2=P1^5; //将 S2 位定义为 P1.5
sbit S3=P1^6; //将 S3 位定义为 P1.6
sbit S4=P1^7; //将 S4 位定义为 P1.7

```

函数功能：数码管扫描延时

```

*****/
void delay(void)
{
    unsigned char j;
    for(j=0;j<200;j++)
        ;
}

```



```

    }
    /*****
函数功能：键盘扫描延时
*****/

void delay60ms(void)
{
    unsigned char i,j;
    for(i=0;i<200;i++)
        for(j=0;j<100;j++)
            ;
}

/*****
函数功能：显示秒
入口参数：s
*****/

void DisplaySecond(unsigned char s)
{
    P2=0xbf;        //P2.6 引脚输出低电平， DS6 点亮
    P0=Tab[s/10];   //显示十位
    delay();

    P2=0x7f;        //P2.7 引脚输出低电平， DS7 点亮
    P0=Tab[s%10];   //显示个位
    delay();
    P2=0xff;        //关闭所有数码管
}

/*****
函数功能：显示分钟
入口参数：m
*****/

void DisplayMinute(unsigned char m)
{
    P2=0xf7;    // P2.3 引脚输出低电平， DS3 点亮
    P0=Tab[m/10];//显示个位
    delay();
    P2=0xef;    // P2.4 引脚输出低电平， DS4 点亮
    P0=Tab[m%10];
    delay();
    P2=0xdf;    //P2.5 引脚输出低电平， DS5 点亮
    P0=0xbf;    //分隔符“-”的段码
    delay();
}

```

```

        P2=0xff;    //关闭所有数码管
    }
    /*****
    函数功能：显示小时的子程序
    入口参数：h
    *****/
    void DisplayHour(unsigned char h)
    {

        P2=0xfe;    //P2.0 引脚输出低电平， DS0 点亮
        P0=Tab[h/10];    //显示十位
        delay();

        P2=0xfd;    //P2.1 引脚输出低电平， DS1 点亮
        P0=Tab[h%10];    //显示个位
        delay();
        P2=0xfb;    //P2.2 引脚输出低电平， DS2 点亮
        P0=0xbf;    //分隔符“-”的段码
        delay();
        P2=0xff;    //关闭所有数码管

    }
    /*****
    函数功能：键盘扫描
    *****/
    void key_scan(void)
    {
        P1=0xf0;    //将 P1 口高 4 位置高电平“1”
        if((P1&0xf0)!=0xf0)    //有键按下
        {
            delay60ms();    //延时 60ms 再检测
            if((P1&0xf0)!=0xf0)    //确实有键按下
            {
                if(S1==0)    //如果是 S1 键按下
                    second++;    //秒加 1
                if(S2==0)    //如果是 S2 键按下
                    minute++;    //分钟加 1
                if(S3==0)    //如果是 S3 键按下
                    hour++;    //小时加 1
                if(S4==0)    //如果是 S4 键按下
                {
                    second=0;    //秒清 0
                    minute=0;    //分钟清 0
                    hour=0;    //小时清 0
                }
            }
        }
    }

```



```

    if(second==60)    //如果秒计满 60
    {
        second=0;    //如果秒计满 60, 将秒计数变量清 0
        minute++;    //分钟计数变量加 1
    }
    if(minute==60)    //如果分钟计满 60
    {
        minute=0;    //如果分钟计满 60, 将分钟计数变量清 0
        hour++;    //小时计数变量加 1
    }
    if(hour==24)    //如果小时计满 24
    {
        hour=0;    //如果小时计满 24, 将小时计数变量清 0
    }
    key_scan();    //执行键盘扫描
    TH0=(65536-46083)/256;    //定时器 T0 高四位赋值
    TL0=(65536-46083)%256;    //定时器 T0 低四位赋值
    TR0=1;    //启动定时器 T0

}

```

实例 76: 独立式键盘控制步进电机实验

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
sbit S1=P1^4;    //将 S1 位定义为 P1.4 引脚
sbit S2=P1^5;    //将 S2 位定义为 P1.5 引脚
sbit S3=P1^6;    //将 S3 位定义为 P1.6 引脚
unsigned char keyval;    //储存按键值
unsigned char ID;    //储存功能标号
/*****
函数功能: 软件消抖延时 (约 50ms)
*****/
void delay(void)
{
    unsigned char i,j;
    for(i=0;i<150;i++)
        for(j=0;j<100;j++)
            ;
}
/*****
函数功能: 步进电机转动延时, 延时越长, 转速越慢
*****/
void motor_delay(void)

```

```

{
    unsigned int i;
    for(i=0;i<2000;i++)
        ;
}
/*****
函数功能：步进电机正转
*****/
void forward( )
{
    P0=0xfc;          //P0 口低四位脉冲 1100
    motor_delay();
    P0=0xf6;          //P0 口低四位脉冲 0110
    motor_delay();
    P0=0xf3;          //P0 口低四位脉冲 0011
    motor_delay();
    P0=0xf9;          //P0 口低四位脉冲 1001
    motor_delay();
}
/*****
函数功能：步进电机反转
*****/
void backward()
{
    P0=0xfc;          //P0 口低四位脉冲 1100
    motor_delay();
    P0=0xf9;          //P0 口低四位脉冲 1001
    motor_delay();
    P0=0xf3;          //P0 口低四位脉冲 0011
    motor_delay();
    P0=0xf6;          //P0 口低四位脉冲 0110
    motor_delay();
}
/*****
函数功能：步进电机停转
*****/
void stop(void)
{
    P0=0xff ;          //停止输出脉冲
}
/*****
函数功能：主函数
*****/
void main(void)

```

```

{
    TMOD=0x01;           //使用定时器 T0 的模式 1
    EA=1;                //开总中断
    ET0=1;              //定时器 T0 中断允许
    TR0=1;              //启动定时器 T0
    TH0=(65536-500)/256; //定时器 T0 赋初值，每计数 200 次（217 微秒）
    TL0=(65536-500)%256; //定时器 T0 赋初值
    keyval=0;           //按键值初始化为 0，什么也不做
    ID=0;
    while(1)
    {
        switch(keyval)   //根据按键值 keyval 选择待执行的功能
        {
            case 1:forward(); //按键 S1 按下,正转
                break;
            case 2:backward(); //按键 S2 按下，反转
                break;
            case 3:stop();    //按键 S3 按下，停转
                break;
        }
    }
}
/*****
函数功能：定时器 T0 的中断服务子程序
*****/
void Time0_serve(void) interrupt 1 using 1
{
    TR0=0;                //关闭定时器 T0
    if((P1&0xf0)!=0xf0)   //第一次检测到有键按下
    {
        delay();         //延时一段时间再去检测
        if((P1&0xf0)!=0xf0) //确实有键按下
        {
            if(S1==0)     //按键 S1 被按下
                keyval=1;
            if(S2==0)     //按键 S2 被按下
                keyval=2;
            if(S3==0)     //按键 S3 被按下
                keyval=3;
        }
    }
    TH0=(65536-200)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-200)%256; //定时器 T0 的低 8 位赋初值
}

```

```

    TR0=1;                //启动定时器 T0
}

```

实例 77：矩阵式键盘按键值的数码管显示实验

```

#include<reg51.h>    //包含 51 单片机寄存器定义的头文件
sbit P14=P1^4;      //将 P14 位定义为 P1.4 引脚
sbit P15=P1^5;      //将 P15 位定义为 P1.5 引脚
sbit P16=P1^6;      //将 P16 位定义为 P1.6 引脚
sbit P17=P1^7;      //将 P17 位定义为 P1.7 引脚
unsigned char code
Tab[ ]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};    //数字 0~9
的段码
unsigned char keyval;    //定义变量储存按键值
/*****
函数功能：数码管动态扫描延时
*****/
void led_delay(void)
{
    unsigned char j;
    for(j=0;j<200;j++)
        ;
}
/*****
函数功能：按键值的数码管显示子程序
*****/
void display(unsigned char k)
{
    P2=0xbf;          //点亮数码管 DS6
    P0=Tab[k/10];     //显示十位
    led_delay();      //动态扫描延时
    P2=0x7f;          //点亮数码管 DS7
    P0=Tab[k%10];    //显示个位
    led_delay();      //动态扫描延时
}
/*****
函数功能：软件延时子程序
*****/
void delay20ms(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)

```

```

        ;
    }
}
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1;           //开总中断
    ET0=1;         //定时器 T0 中断允许
    TMOD=0x01;     //使用定时器 T0 的模式 1
    TH0=(65536-500)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256; //定时器 T0 的低 8 位赋初值
    TR0=1;         //启动定时器 T0
    keyval=0x00;   //按键值初始化为 0

    while(1)       //无限循环
    {
        display(keyval); //调用按键值的数码管显示子程序
    }
}
/*****
函数功能：定时器 0 的中断服务子程序，进行键盘扫描，判断键位
*****/
void time0_interserve(void) interrupt 1 using 1 //定时器 T0 的中断编号为
1，使用第一组寄存器
{
    TR0=0;         //关闭定时器 T0
    P1=0xf0;       //所有行线置为低电平“0”，所有列线置为高
电平“1”
    if((P1&0xf0)!=0xf0) //列线中有一位为低电平“0”，说明有键按下
        delay20ms(); //延时一段时间、软件消抖
    if((P1&0xf0)!=0xf0) //确实有键按下
    {
        P1=0xfe;     //第一行置为低电平“0”（P1.0 输出低电
平“0”）
        if(P14==0) //如果检测到接 P1.4 引脚的列线为低电平
“0”
            keyval=1; //可判断是 S1 键被按下
            if(P15==0) //如果检测到接 P1.5 引脚的列线为低电平
“0”
                keyval=2; //可判断是 S2 键被按下
                if(P16==0) //如果检测到接 P1.6 引脚的列线为低电平
“0”

```



```

        keyval=3;           //可判断是 S3 键被按下
        if(P17==0)        //如果检测到接 P1.7 引脚的列线为低电平
“0”
        keyval=4;           //可判断是 S4 键被按下

        P1=0xfd;          //第二行置为低电平“0”（P1.1 输出低电
平“0”）
        if(P14==0)        //如果检测到接 P1.4 引脚的列线为低电平“0”
        keyval=5;           //可判断是 S5 键被按下
        if(P15==0)        //如果检测到接 P1.5 引脚的列线为低电平
“0”
        keyval=6;           //可判断是 S6 键被按下
        if(P16==0)        //如果检测到接 P1.6 引脚的列线为低电平
“0”
        keyval=7;           //可判断是 S7 键被按下
        if(P17==0)        //如果检测到接 P1.7 引脚的列线为低电平
“0”
        keyval=8;           //可判断是 S8 键被按下

        P1=0xfb;          //第三行置为低电平“0”（P1.2 输出低电
平“0”）
        if(P14==0)        //如果检测到接 P1.4 引脚的列线为低电平“0”
        keyval=9;           //可判断是 S9 键被按下
        if(P15==0)        //如果检测到接 P1.5 引脚的列线为低电平
“0”
        keyval=10;          //可判断是 S10 键被按下
        if(P16==0)        //如果检测到接 P1.6 引脚的列线为低电平“0”
        keyval=11;          //可判断是 S11 键被按下
        if(P17==0)        //如果检测到接 P1.7 引脚的列线为低电平“0”
        keyval=12;          //可判断是 S12 键被按下

        P1=0xf7;          //第四行置为低电平“0”（P1.3 输出低电
平“0”）
        if(P14==0)        //如果检测到接 P1.4 引脚的列线为低电平“0”
        keyval=13;          //可判断是 S13 键被按下
        if(P15==0)        //如果检测到接 P1.5 引脚的列线为低电平
“0”
        keyval=14;          //可判断是 S14 键被按下
        if(P16==0)        //如果检测到接 P1.6 引脚的列线为低电平“0”
        keyval=15;          //可判断是 S15 键被按下
        if(P17==0)        //如果检测到接 P1.7 引脚的列线为低电平“0”
        keyval=16;          //可判断是 S16 键被按下
    }
    TR0=1;                //开启定时器 T0

```

```

    TH0=(65536-500)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256; //定时器 T0 的高 8 位赋初值
}

```

实例 78：矩阵式键盘按键音

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit sound=P3^7; //将 sound 位定义为 P3.7

```

```

/*****
函数功能：蜂鸣器发声延时约 120ms
*****/
void delay_sound(void)
{
    unsigned char i;
    for(i=0;i<250;i++)
        ;
}
/*****
函数功能：软件延时子程序约 20ms
*****/
void delay20ms(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)
            ;
}
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1; //开总中断
    ET0=1; //定时器 T0 中断允许
    TMOD=0x01; //使用定时器 T0 的模式 1
    TH0=(65536-500)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256; //定时器 T0 的高 8 位赋初值
    TR0=1; //启动定时器 T0
    while(1) //无限循环，等待键盘按下
        ;
}

```

```

/*****
函数功能：定时器 0 的中断服务子程序，进行键盘扫描，判断键位
*****/
void time0_interserve(void) interrupt 1 using 1 //定时器 T0 的中断编号为
1，使用第一组寄存器
{
    unsigned char i;
    TR0=0; //关闭定时器 T0
    P1=0xf0; //所有行线置为低电平“0”，所有列线置为高
电平“1”
    if((P1&0xf0)!=0xf0) //列线中有一位为低电平“0”，说明有键按下
        delay20ms(); //延时一段时间、软件消抖
    if((P1&0xf0)!=0xf0) //确实有键按下
    {
        for(i=0;i<200;i++) //让 P3.7 引脚电平不断取反输出音频
        {
            sound=0;
            delay_sound();
            sound=1;
            delay_sound();
        }
    }
    TR0=1; //开启定时器 T0
    TH0=(65536-500)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256; //定时器 T0 的高 8 位赋初值
}

```

实例 79：简易电子琴

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件

sbit P14=P1^4; //将 P14 位定义为 P1.4 引脚
sbit P15=P1^5; //将 P15 位定义为 P1.5 引脚
sbit P16=P1^6; //将 P16 位定义为 P1.6 引脚
sbit P17=P1^7; //将 P17 位定义为 P1.7 引脚
unsigned char keyval; //定义变量储存按键值

sbit sound=P3^7; //将 sound 位定义为 P3.7
unsigned int C; //全局变量，储存定时器的定时常数
unsigned int f; //全局变量，储存音阶的频率

//以下是 C 调低音的音频宏定义
#define I_dao 262 //将“I_dao”宏定义为低音“1”的频率 262Hz

```

```

#define l_re 286 //将“l_re”宏定义为低音“2”的频率 286Hz
#define l_mi 311 //将“l_mi”宏定义为低音“3”的频率 311Hz
#define l_fa 349 //将“l_fa”宏定义为低音“4”的频率 349Hz
#define l_sao 392 //将“l_sao”宏定义为低音“5”的频率 392Hz
#define l_la 440 //将“l_a”宏定义为低音“6”的频率 440Hz
#define l_xi 494 //将“l_xi”宏定义为低音“7”的频率 494Hz

```

//以下是 C 调中音的音频宏定义

```

#define dao 523 //将“dao”宏定义为中音“1”的频率 523Hz
#define re 587 //将“re”宏定义为中音“2”的频率 587Hz
#define mi 659 //将“mi”宏定义为中音“3”的频率 659Hz
#define fa 698 //将“fa”宏定义为中音“4”的频率 698Hz
#define sao 784 //将“sao”宏定义为中音“5”的频率 784Hz
#define la 880 //将“la”宏定义为中音“6”的频率 880Hz
#define xi 987 //将“xi”宏定义为中音“7”的频率 53

```

//以下是 C 调高音的音频宏定义

```

#define h_dao 1046 //将“h_dao”宏定义为高音“1”的频率 1046Hz
#define h_re 1174 //将“h_re”宏定义为高音“2”的频率 1174Hz
#define h_mi 1318 //将“h_mi”宏定义为高音“3”的频率 1318Hz
#define h_fa 1396 //将“h_fa”宏定义为高音“4”的频率 1396Hz
#define h_sao 1567 //将“h_sao”宏定义为高音“5”的频率 1567Hz
#define h_la 1760 //将“h_la”宏定义为高音“6”的频率 1760Hz
#define h_xi 1975 //将“h_xi”宏定义为高音“7”的频率 1975Hz

```

/*

*/

函数功能：软件延时子程序

*/

```

void delay20ms(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)
            ;
}

```

/*

*/

函数功能：节拍的延时的基本单位，延时 200ms

*/

```

void delay()
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)

```

```

        ;
    }
}
/*****
函数功能：输出音频
入口参数：F
*****/
void Output_Sound(void)
{
    C=(46083/f)*10;    //计算定时常数
    TH0=(8192-C)/32;    //可证明这是 13 位计数器 TH0 高 8 位的赋初值方法
    TL0=(8192-C)%32;    //可证明这是 13 位计数器 TL0 低 5 位的赋初值方法
    TR0=1;            //开定时 T0
    delay();          //延时 200ms，播放音频
    TR0=0;            //关闭定时器
    sound=1;          //关闭蜂鸣器
    keyval=0xff;      //播放按键音频后，将按键值更改，停止播放
}
/*****
函数功能：主函数
*****/
void main(void)
{
    EA=1;            //开总中断
    ET0=1;          //定时器 T0 中断允许
    ET1=1;          //定时器 T1 中断允许
    TR1=1;          //定时器 T1 启动，开始键盘扫描
    TMOD=0x10;      //分别使用定时器 T1 的模式 1，T0 的模式 0
    TH1=(65536-500)/256; //定时器 T1 的高 8 位赋初值
    TL1=(65536-500)%256; //定时器 T1 的高 8 位赋初值

    while(1)        //无限循环
    {
        switch(keyval)
        {
            case 1:f=dao;            //如果第 1 个键按下，
将中音 1 的频率赋给 f
                Output_Sound();    //转去计算定时常数
                break;
            case 2:f=l_xi;          //如果第 2 个键按下，将
低音 7 的频率赋给 f
                Output_Sound();    //转去计算定时常数
                break;
        }
    }
}

```

```

        case 3:f=l_la;           //如果第 3 个键按下, 将低音
6 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 4:f=l_sao;        //如果第 4 个键按下, 将
低音 5 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 5:f=sao;          //如果第 5 个键按下,
将中音 5 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 6:f=fa;           //如果第 6 个键按下,
将中音 4 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 7:f=mi;           //如果第 7 个键按下, 将中
音 3 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 8:f=re;           //如果第 8 个键按下,
将中音 2 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 9:f=h_re;         //如果第 9 个键按下,
将高音 2 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 10:f=h_dao;       //如果第 10 个键按
下, 将高音 1 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 11:f=xi;          //如果第 11 个键按下, 将中
音 7 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 12:f=la;          //如果第 12 个键按下,
将中音 6 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;
        case 13:f=h_la;        //如果第 13 个键按下,
将高音 6 的频率赋给 f
            Output_Sound();    //转去计算定时常数
            break;

```

```

        case 14:f=h_sao;           //如果第 14 个键按下, 将高音 5 的频率赋给 f
            Output_Sound();       //转去计算定时常数
            break;
        case 15:f=h_fa;           //如果第 15 个键按下, 将高音 4 的频率赋给 f
            Output_Sound();       //转去计算定时常数
            break;
        case 16:f=h_mi;           //如果第 16 个键按下, 将高音 3 的频率赋给 f
            Output_Sound();       //转去计算定时常数
            break;
    }
}

```

```

/*****

```

函数功能：定时器 T0 的中断服务子程序，使 P3.7 引脚输出音频方波

```

*****/

```

```

void Time0_serve(void ) interrupt 1 using 1
{
    TH0=(8192-C)/32;           //可证明这是 13 位计数器 TH0 高 8 位的赋初值方法
    TL0=(8192-C)%32;          //可证明这是 13 位计数器 TL0 低 5 位的赋初值方法
    sound=!sound;             //将 P3.7 引脚取反，输出音频方波
}

```

```

/*****

```

函数功能：定时器 T1 的中断服务子程序，进行键盘扫描，判断键位

```

*****/

```

```

void time1_serve(void) interrupt 3 using 2 //定时器 T1 的中断编号为 3, 使用第 2 组寄存器
{
    TR1=0;                       //关闭定时器 T0
    P1=0xf0;                       //所有行线置为低电平“0”，所有列线置为高电平“1”
    if((P1&0xf0)!=0xf0)           //列线中有一位为低电平“0”，说明有键按下
    {
        delay20ms();              //延时一段时间、软件消抖
        if((P1&0xf0)!=0xf0)       //确实有键按下
        {

```

```

P1=0xfe;           //第一行置为低电平“0”（P1.0
输出低电平“0”）
if(P14==0)        //如果检测到接 P1.4 引脚的列线
为低电平“0”
    keyval=1;     //可判断是 S1 键被按下
if(P15==0)        //如果检测到接 P1.5 引脚的
列线为低电平“0”
    keyval=2;     //可判断是 S2 键被按下
if(P16==0)        //如果检测到接 P1.6 引脚的
列线为低电平“0”
    keyval=3;     //可判断是 S3 键被按下
if(P17==0)        //如果检测到接 P1.7 引脚的列
线为低电平“0”
    keyval=4;     //可判断是 S4 键被按下

P1=0xfd;           //第二行置为低电平“0”（P1.1
输出低电平“0”）
if(P14==0)        //如果检测到接 P1.4 引脚的列线
为低电平“0”
    keyval=5;     //可判断是 S5 键被按下
if(P15==0)        //如果检测到接 P1.5 引脚的
列线为低电平“0”
    keyval=6;     //可判断是 S6 键被按下
if(P16==0)        //如果检测到接 P1.6 引脚的
列线为低电平“0”
    keyval=7;     //可判断是 S7 键被按下
if(P17==0)        //如果检测到接 P1.7 引脚的列
线为低电平“0”
    keyval=8;     //可判断是 S8 键被按下

P1=0xfb;           //第三行置为低电平“0”（P1.2
输出低电平“0”）
if(P14==0)        //如果检测到接 P1.4 引脚的列线为低
电平“0”
    keyval=9;     //可判断是 S9 键被按下
if(P15==0)        //如果检测到接 P1.5 引脚的列
线为低电平“0”
    keyval=10;    //可判断是 S10 键被按下
if(P16==0)        //如果检测到接 P1.6 引脚的列线
为低电平“0”
    keyval=11;    //可判断是 S11 键被按下
if(P17==0)        //如果检测到接 P1.7 引脚的列线
为低电平“0”
    keyval=12;    //可判断是 S12 键被按下

```



```

        P1=0xf7;           //第四行置为低电平“0”（P1.3
输出低电平“0”）
        if(P14==0)       //如果检测到接 P1.4 引脚的列线为低
电平“0”
            keyval=13;    //可判断是 S13 键被按下
            if(P15==0)    //如果检测到接 P1.5 引脚的列
线为低电平“0”
                keyval=14; //可判断是 S14 键被按下
                if(P16==0) //如果检测到接 P1.6 引脚的列线
为低电平“0”
                    keyval=15; //可判断是 S15 键被按下
                    if(P17==0) //如果检测到接 P1.7 引脚的列线
为低电平“0”
                        keyval=16; //可判断是 S16 键被按下
                }
            }
        TR1=1;           //开启定时器 T1
        TH1=(65536-500)/256; //定时器 T1 的高 8 位赋初值
        TL1=(65536-500)%256; //定时器 T1 的高 8 位赋初值
    }

```

实例 80：矩阵式键盘实现的电子密码锁

```

#include<reg51.h> //包含 51 单片机寄存器定义的头文件
sbit P14=P1^4;    //将 P14 位定义为 P1.4 引脚
sbit P15=P1^5;    //将 P15 位定义为 P1.5 引脚
sbit P16=P1^6;    //将 P16 位定义为 P1.6 引脚
sbit P17=P1^7;    //将 P17 位定义为 P1.7 引脚
sbit sound=P3^7;  //将 sound 位定义为 P3.7
unsigned char keyval; //储存按键值
/*****
函数功能：延时输出音频
*****/
void delay(void)
{
    unsigned char i;
    for(i=0;i<200;i++)
        ;
}
/*****

```

函数功能：软件延时子程序

```

*****/
void delay20ms(void)
{
    unsigned char i,j;
    for(i=0;i<100;i++)
        for(j=0;j<60;j++)
            ;
}
/*****

```

函数功能：主函数

```

*****/
void main(void)
{
    unsigned char D[ ]={0,8,0,8,7,4,11};    //设定密码
    EA=1;                                    //开总中断
    ET0=1;                                   //定时器 T0 中断允许
    TMOD=0x01;                              //使用定时器 T0 的模式 1
    TH0=(65536-500)/256; //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256; //定时器 T0 的低 8 位赋初值
    TR0=1;                                   //启动定时器 T0
    keyval=0xff;                             //按键值初始化

    while(keyval!=D[0]) //第一位密码输入不正确，等待
        ;
    while(keyval!=D[1]) //第二位密码输入不正确，等待
        ;
    while(keyval!=D[2]) //第三位密码输入不正确，等待
        ;
    while(keyval!=D[3]) //第四位密码输入不正确，等待
        ;
    while(keyval!=D[4]) //第五位密码输入不正确，等待
        ;
    while(keyval!=D[5]) //第六位密码输入不正确，等待
        ;
    while(keyval!=D[6]) //没有输入“OK”，等待
        ;
    P3=0xfe; //P3.0 引脚输出低电平，点亮 LED
}
/*****

```

函数功能：定时器 0 的中断服务子程序，进行键盘扫描，判断键位

```

*****/

```

```

void time0_interserve(void) interrupt 1 using 1 //定时器 T0 的中断编号为
1, 使用第一组寄存器
{
    unsigned char i;
    TR0=0; //关闭定时器 T0
    P1=0xf0; //所有行线置为低电平“0”，所有列线置为高
电平“1”
    if((P1&0xf0)!=0xf0) //列线中有一位为低电平“0”，说明有键按下
        delay20ms(); //延时一段时间、软件消抖
    if((P1&0xf0)!=0xf0) //确实有键按下
    {
        P1=0xfe; //第一行置为低电平“0”（P1.0 输出低电
平“0”）
        if(P14==0) //如果检测到接 P1.4 引脚的列线为低电平
“0”
            keyval=1; //可判断是 S1 键被按下
            if(P15==0) //如果检测到接 P1.5 引脚的列线为低电平
“0”
                keyval=2; //可判断是 S2 键被按下
                if(P16==0) //如果检测到接 P1.6 引脚的列线为低电平
“0”
                    keyval=3; //可判断是 S3 键被按下
                    if(P17==0) //如果检测到接 P1.7 引脚的列线为低电平
“0”
                        keyval=4; //可判断是 S4 键被按下

                        P1=0xfd; //第二行置为低电平“0”（P1.1 输出低电
平“0”）
                        if(P14==0) //如果检测到接 P1.4 引脚的列线为低电平“0”
                            keyval=5; //可判断是 S5 键被按下
                            if(P15==0) //如果检测到接 P1.5 引脚的列线为低电平
“0”
                                keyval=6; //可判断是 S6 键被按下
                                if(P16==0) //如果检测到接 P1.6 引脚的列线为低电平
“0”
                                    keyval=7; //可判断是 S7 键被按下
                                    if(P17==0) //如果检测到接 P1.7 引脚的列线为低电平
“0”
                                        keyval=8; //可判断是 S8 键被按下

                                        P1=0xfb; //第三行置为低电平“0”（P1.2 输出低电
平“0”）
                                        if(P14==0) //如果检测到接 P1.4 引脚的列线为低电平“0”
                                            keyval=9; //可判断是 S9 键被按下

```

```

        if(P15==0)           //如果检测到接 P1.5 引脚的列线为低电平
“0”
        keyval=0;           //可判断是 S10 键被按下
        if(P16==0)         //如果检测到接 P1.6 引脚的列线为低电平“0”
        keyval=11;         //可判断是 S11 键被按下
        if(P17==0)         //如果检测到接 P1.7 引脚的列线为低电平“0”
        keyval=12;         //可判断是 S12 键被按下

        P1=0xf7;           //第四行置为低电平“0”（P1.3
输出低电平“0”）
        if(P14==0)         //如果检测到接 P1.4 引脚的列线为低
电平“0”
        keyval=13;         //可判断是 S13 键被按下
        if(P15==0)         //如果检测到接 P1.5 引脚的列
线为低电平“0”
        keyval=14;         //可判断是 S14 键被按下
        if(P16==0)         //如果检测到接 P1.6 引脚的列线
为低电平“0”
        keyval=15;         //可判断是 S15 键被按下
        if(P17==0)         //如果检测到接 P1.7 引脚的列线
为低电平“0”
        keyval=16;         //可判断是 S16 键被按下
        for(i=0;i<200;i++) //让 P3.7 引脚电平不断取反输出音频
        {
            sound=0;
            delay();
            sound=1;
            delay();
        }
    }
    TR0=1;                 //开启定时器 T0
    TH0=(65536-500)/256;   //定时器 T0 的高 8 位赋初值
    TL0=(65536-500)%256;  //定时器 T0 的高 8 位赋初值
}

```

液晶显示 LCD

实例 81：用 LCD 显示字符'A'

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit RS=P2^0;      //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;      //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;       //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;      //忙碌标志位，，将 BF 位定义为 P0.7 引脚
/*****

函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/

void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****

函数功能：延时若干毫秒
入口参数：n
*****/

void delay(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****

函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌;result=0，不忙
*****/

unsigned char BusyTest(void)
{
    bit result;
    RS=0;          //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;          //E=1，才允许读写
    _nop_();      //空操作
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    result=BF;    //将忙碌标志电平赋给 result
    E=0;
    return result;
}

```

```

}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0; //根据规定，RS 和 R/W 同时为低电平时，可以
写入指令
    RW=0;
    E=0; //E 置低电平(根据表 8-6，写指令时，E 为高脉
冲，
// 就是让 E 从 0 到 1 发生正跳变，所以应先置
"0"
    _nop_();
    _nop_(); //空操作两个机器周期，给硬件反应时间
    P0=dictate; //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_(); //空操作四个机器周期，给硬件反应时间
    E=1; //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_(); //空操作四个机器周期，给硬件反应时间
    E=0; //当 E 由高电平跳变成低电平时，液晶模块开始
执行命令
}

```

```

/*****
函数功能：指定字符显示的实际地址
入口参数：x
*****/
void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}

```

```

/*****
函数功能：将数据(字符的标准 ASCII 码)写入液晶模块
入口参数：y(为字符常量)
*****/
void WriteData(unsigned char y)
{

```

```

while(BusyTest()==1);
    RS=1;           //RS 为高电平, RW 为低电平时, 可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6, 写指令时, E 为高脉冲,
                  // 就是让 E 从 0 到 1 发生正跳变, 所以应先置"0"
    P0=y;         //将数据送入 P0 口, 即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=1;         //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=0;         //当 E 由高电平跳变成低电平时, 液晶模块开始执行命令
}
/*****
函数功能: 对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delay(15);      //延时 15ms, 首次写指令时应给 LCD 一段较长的
                  反应时间
    WriteInstruction(0x38); //显示模式设置: 16×2 显示, 5×7 点阵, 8 位数
                  据接口
    delay(5);      //延时 5ms
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x0f); //显示模式设置: 显示开, 有光标, 光标闪烁
    delay(5);
    WriteInstruction(0x06); //显示模式设置: 光标右移, 字符不移
    delay(5);
    WriteInstruction(0x01); //清屏幕指令, 将以前的显示内容清除
    delay(5);
}
void main(void)    //主函数
{
    LcdInitiate(); //调用 LCD 初始化函数
    WriteAddress(0x07); //将显示地址指定为第 1 行第 8 列
    WriteData('A');    //将字符常量'A'写入液晶模块
}

```

//字符的字形点阵读出和显示由液晶模块自动完成

}

实例 82：用 LCD 循环右移显示"Welcome to China"

```
#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h>  //包含_nop_()函数定义的头文件
sbit RS=P2^0;       //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;       //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;        //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;       //忙碌标志位，将 BF 位定义为 P0.7 引脚
unsigned char code string[ ]={"Welcome to China"};
/*****
```

函数功能：延时 1ms

$(3j+2)*i=(3 \times 33+2) \times 10=1010$ (微秒)，可以认为是 1 毫秒

*****/

```
void delay1ms()
```

```
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
```

函数功能：延时若干毫秒

入口参数：n

*****/

```
void delay(unsigned char n)
```

```
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
```

*****/

函数功能：判断液晶模块的忙碌状态

返回值：result。result=1，忙碌;result=0，不忙

*****/

```
unsigned char BusyTest(void)
```

```
{
    bit result;
    RS=0;        //根据规定，RS 为低电平，RW 为高电平时，可以读状态
```



```

    RW=1;
    E=1;          //E=1, 才允许读写
    _nop_();     //空操作
    _nop_();
    _nop_();
    _nop_();     //空操作四个机器周期, 给硬件反应时间
    result=BF;   //将忙碌标志电平赋给 result
    E=0;
    return result;
}
/*****
函数功能: 将模式设置指令或显示地址写入液晶模块
入口参数: dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                  //根据规定, RS 和 R/W 同时为低电平时, 可以
    写入指令
    RW=0;
    E=0;                  //E 置低电平(根据表 8-6, 写指令时, E 为高脉
    冲,
                          // 就是让 E 从 0 到 1 发生正跳变, 所以应先置
    "0"
    _nop_();
    _nop_();              //空操作两个机器周期, 给硬件反应时间
    P0=dictate;          //将数据送入 P0 口, 即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();              //空操作四个机器周期, 给硬件反应时间
    E=1;                  //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();              //空操作四个机器周期, 给硬件反应时间
    E=0;                  //当 E 由高电平跳变成低电平时, 液晶模块开始
    执行命令
}
/*****
函数功能: 指定字符显示的实际地址
入口参数: x
*****/
void WriteAddress(unsigned char x)

```

```

    {
        WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
    }
}
/*****
函数功能: 将数据(字符的标准 ASCII 码)写入液晶模块
入口参数: y(为字符常量)
*****/
void WriteData(unsigned char y)
{
    while(BusyTest()!=1);
    RS=1;           //RS 为高电平, RW 为低电平时, 可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6, 写指令时, E 为高脉冲,
                  // 就是让 E 从 0 到 1 发生正跳变, 所以应先置"0"
    P0=y;          //将数据送入 P0 口, 即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=1;          //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=0;          //当 E 由高电平跳变成低电平时, 液晶模块开始执行命令
}
/*****
函数功能: 对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delay(15);     //延时 15ms, 首次写指令时应给 LCD 一段较长的
                  //反应时间
    WriteInstruction(0x38); //显示模式设置: 16×2 显示, 5×7 点阵, 8 位数
                  //据接口
    delay(5);     //延时 5ms
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x0f); //显示模式设置: 显示开, 有光标, 光标闪烁
    delay(5);
    WriteInstruction(0x06); //显示模式设置: 光标右移, 字符不移
    delay(5);
}

```

```

    WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
    delay(5);
}
void main(void)           //主函数
{
    unsigned char i;
    LcdInitiate();        //调用 LCD 初始化函数
    delay(10);
    while(1)
    {
        WriteInstruction(0x01); //清显示：清屏幕指令
        WriteAddress(0x00); // 设置显示位置为第一行的第 5 个字
        i = 0;
        while(string[i] != '\0')
        {
            // 显示字符
            WriteData(string[i]);
            i++;
            delay(150);
        }
        for(i=0;i<4;i++)
            delay(250);
    }
}
}

```

实例 83：用 LCD 显示适时检测结果

```

#include<reg51.h> //包含单片机寄存器的头文件
#include<stdlib.h> //包含随机函数 rand()的定义文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit RS=P2^0; //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1; //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2; //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7; //忙碌标志位，将 BF 位定义为 P0.7 引脚
unsigned char code digit[ ]={"0123456789"}; //定义字符数组显示数字
unsigned char code string[ ]={"Test Result"}; //定义字符数组显示提示信息
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()

```

```

{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delay(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌;result=0，不忙
*****/
unsigned char BusyTest(void)
{
    bit result;
    RS=0;        //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;        //E=1，才允许读写
    _nop_();    //空操作
    _nop_();
    _nop_();
    _nop_();    //空操作四个机器周期，给硬件反应时间
    result=BF;  //将忙碌标志电平赋给 result
    E=0;        //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                //根据规定，RS 和 RW 同时为低电平时，可以
    写入指令
    RW=0;

```

```

E=0;           //E 置低电平(根据表 8-6, 写指令时, E 为高脉
冲,           // 就是让 E 从 0 到 1 发生正跳变, 所以应先置
"0"
_nop_();
_nop_();      //空操作两个机器周期, 给硬件反应时间
P0=dictate;   //将数据送入 P0 口, 即写入指令或地址
_nop_();
_nop_();
_nop_();
_nop_();      //空操作四个机器周期, 给硬件反应时间
E=1;          //E 置高电平
_nop_();
_nop_();
_nop_();
_nop_();      //空操作四个机器周期, 给硬件反应时间
E=0;          //当 E 由高电平跳变成低电平时, 液晶模块开始
执行命令
}

```

函数功能: 指定字符显示的实际地址

入口参数: x

*****/

```

void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}

```

函数功能: 将数据(字符的标准 ASCII 码)写入液晶模块

入口参数: y(为字符常量)

*****/

```

void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;       //RS 为高电平, RW 为低电平时, 可以写入数据
    RW=0;
    E=0;        //E 置低电平(根据表 8-6, 写指令时, E 为高脉冲,
                // 就是让 E 从 0 到 1 发生正跳变, 所以应先置"0"
    P0=y;       //将数据送入 P0 口, 即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();    //空操作四个机器周期, 给硬件反应时间
    E=1;        //E 置高电平
}

```

```

    _nop_();
    _nop_();
    _nop_();
    _nop_();           //空操作四个机器周期，给硬件反应时间
    E=0;              //当 E 由高电平跳变成低电平时，液晶模块开始执行命令
}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delay(15);        //延时 15ms，首次写指令时应给 LCD 一段较长的
    反应时间
    WriteInstruction(0x38); //显示模式设置：16×2 显示，5×7 点阵，8 位数
    据接口
    delay(5);         //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x38); //连续三次，确保初始化成功
    delay(5);
    WriteInstruction(0x0c); //显示模式设置：显示开，无光标，光标不闪烁
    delay(5);
    WriteInstruction(0x06); //显示模式设置：光标右移，字符不移
    delay(5);
    WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
    delay(5);
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i; //定义变量 i 指向字符串数组元素
    unsigned int x; //定义变量，储存检测结果
    unsigned char D1,D2,D3,D4,D5; //分别储存采集的个位、十位、百位、千
    位和万位数字
    LcdInitiate(); //调用 LCD 初始化函数
    delay(10);     //延时 10ms，给硬件一点反应时间
    WriteAddress(0x02); //从第 1 行第 3 列开始显示
    i = 0;         //指向字符串数组的第 1 个元素
    while(string[i] != '\0')
    {
        WriteData(string[i]);
        i++;      //指向字符串数组一个元素
    }
}

```

```

    }
while(1)    //无限循环
{
    x=rand();           //模拟数据采集
    D1=x%10;           //计算个位数字
    D2=(x%100)/10;     //计算十位数字
    D3=(x%1000)/100;   //计算百位数字
    D4=(x%10000)/1000; //计算千位数字
    D5=x/10000;        //计算万位数字
    WriteAddress(0x45); //从第2行第6列开始显示
    WriteData(digit[D5]); //将万位数字的字符常量写入LCD
    WriteData(digit[D4]); //将千位数字的字符常量写入LCD
    WriteData(digit[D3]); //将百位数字的字符常量写入LCD
    WriteData(digit[D2]); //将十位数字的字符常量写入LCD
    WriteData('.');     //将小数点的字符常量写入LCD
    WriteData(digit[D1]); //将个位数字的字符常量写入LCD
    for(i=0;i<4;i++)   //延时1s(每1s采集一次数据)
        delay(250);   //延时250ms
}
}

```

实例 84：液晶时钟设计

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<stdlib.h>   //包含随机函数 rand()的定义文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit RS=P2^0;       //寄存器选择位，将RS位定义为P2.0引脚
sbit RW=P2^1;       //读写选择位，将RW位定义为P2.1引脚
sbit E=P2^2;        //使能信号位，将E位定义为P2.2引脚
sbit BF=P0^7;       //忙碌标志位，将BF位定义为P0.7引脚
unsigned char code digit[ ]={"0123456789"}; //定义字符数组显示数字
unsigned char code string[ ]={"BeiJing Time"}; //定义字符数组显示提示信息
unsigned char count; //定义变量统计中断累计次数
unsigned char s,m,h; //定义变量储存秒、分钟和小时
/*****
函数功能：延时1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是1毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)

```

```

    ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delay(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌；result=0，不忙
*****/
unsigned char BusyTest(void)
{
    bit result;
    RS=0;          //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;          //E=1，才允许读写
    _nop_();      //空操作
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    result=BF;    //将忙碌标志电平赋给 result
    E=0;          //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                  //根据规定，RS 和 R/W 同时为低电平时，可以
写入指令
    RW=0;
    E=0;                  //E 置低电平(根据表 8-6，写指令时，E 为高脉
冲，
// 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"
    _nop_();

```



```

    _nop_();           //空操作两个机器周期，给硬件反应时间
    P0=dictate;       //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();           //空操作四个机器周期，给硬件反应时间
    E=1;              //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();           //空操作四个机器周期，给硬件反应时间
    E=0;              //当 E 由高电平跳变成低电平时，液晶模块开始
    执行命令
}
/*****
函数功能：指定字符显示的实际地址
入口参数：x
*****/
void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}
/*****
函数功能：将数据(字符的标准 ASCII 码)写入液晶模块
入口参数：y(为字符常量)
*****/
void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;           //RS 为高电平，RW 为低电平时，可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6，写指令时，E 为高脉冲，
                  // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"
    P0=y;         //将数据送入 P0 口，即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();       //空操作四个机器周期，给硬件反应时间
    E=1;           //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();       //空操作四个机器周期，给硬件反应时间
    E=0;           //当 E 由高电平跳变成低电平时，液晶模块开始执行命令

```

```

}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delay(15);           //延时 15ms，首次写指令时应给 LCD 一段较长的
    反应时间
    WriteInstruction(0x38); //显示模式设置：16×2 显示，5×7 点阵，8 位数
    据接口
    delay(5);           //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x38); //连续三次，确保初始化成功
    delay(5);
    WriteInstruction(0x0c); //显示模式设置：显示开，无光标，光标不闪烁
    delay(5);
    WriteInstruction(0x06); //显示模式设置：光标右移，字符不移
    delay(5);
    WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
    delay(5);
}
/*****
函数功能：显示小时
*****/
void DisplayHour()
{
    unsigned char i,j;
    i=h/10;           //取整运算，求得十位数字
    j=h%10;           //取余运算，求得各位数字
    WriteAddress(0x44); //写显示地址，将十位数字显示在第 2 行第 5 列
    WriteData(digit[i]); //将十位数字的字符常量写入 LCD
    WriteData(digit[j]); //将个位数字的字符常量写入 LCD
}
/*****
函数功能：显示分钟
*****/
void DisplayMinute()
{
    unsigned char i,j;
    i=m/10;           //取整运算，求得十位数字
    j=m%10;           //取余运算，求得各位数字

```

```

WriteAddress(0x47); //写显示地址, 将十位数字显示在第 2 行第 8 列
WriteData(digit[i]); //将十位数字的字符常量写入 LCD
WriteData(digit[j]); //将个位数字的字符常量写入 LCD

}
/*****
函数功能: 显示秒
*****/
void DisplaySecond()
{
    unsigned char i,j;
    i=s/10;           //取整运算, 求得十位数字
    j=s%10;          //取余运算, 求得各位数字
    WriteAddress(0x4a); //写显示地址, 将十位数字显示在第 2 行第 11 列
    WriteData(digit[i]); //将十位数字的字符常量写入 LCD
    WriteData(digit[j]); //将个位数字的字符常量写入 LCD
}

/*****
main function
*****/
void main(void)
{
    unsigned char i;
    LcdInitiate(); //调用 LCD 初始化函数

    TMOD=0x01;           //使用定时器 T0 的模式 1
    TH0=(65536-46083)/256; //定时器 T0 的高 8 位设置初值
    TL0=(65536-46083)%256; //定时器 T0 的低 8 位设置初值
    EA=1;                //开总中断
    ET0=1;                //定时器 T0 中断允许
    TR0=1;                //启动定时器 T0

    count=0;             //中断次数初始化为 0
    s=0;                 //秒初始化为 0
    m=0;                 //分钟初始化为 0
    h=0;                 //小时初始化为 0

    WriteAddress(0x03); //写地址, 从第 1 行第 4 列开始显示
    i=0;                //从字符数组的第 1 个元素开始显示
    while(string[i]!='\0') //只要没有显示到字符串的结束标志'\0', 就继续
    {
        WriteData(string[i]); //将第 i 个字符数组元素写入 LCD
        i++;                  //指向下一个数组元素
    }
}

```

```

}
WriteAddress(0x46);    //写地址，将第二个分号显示在第2行第7列
WriteData(':');       //将分号的字符常量写入LCD
WriteAddress(0x49);    //写地址，将第二个分号显示在第2行第10列
WriteData(':');       //将分号的字符常量写入LCD
while(1)              //无限循环
{
    DisplayHour();    //显示小时
    delay(5);         //给硬件一点反应时间
    DisplayMinute(); //显示分钟
    delay(5);         //给硬件一点反应时间
    DisplaySecond(); //显示秒
    delay(5);         //给硬件一点反应时间
}
}
/*****
函数功能：定时器 T0 的中断服务函数
*****/
void Time0(void ) interrupt 1 using 1 //定时器 T0 的中断编号为 1，使用第 1 组
工作寄存器
{
    count++; //每产生 1 次中断，中断累计次数加 1
    if(count==20) //如果中断次数计满 20 次
    {
        count=0; //中断累计次数清 0
        s++;     //秒加 1
    }
    if(s==60) //如果计满 60 秒
    {
        s=0;    //秒清 0
        m++;    //分钟加 1
    }
    if(m==60) //如果计满 60 分
    {
        m=0;   //分钟清 0
        h++;   //小时加 1
    }
    if(h==24) //如果计满 24 小时
    {
        h=0;  //小时清 0
    }
    TH0=(65536-46083)/256; //定时器 T0 高 8 位重新赋初值
    TL0=(65536-46083)%256; //定时器 T0 低 8 位重新赋初值
}

```

一些芯片的使用*****24c02 DS18B20 X5045

ADC0832 DAC0832 DS1302 红外遥控

实例 85：将数据"0x0f"写入 AT24C02 再读出送 P1 口显示

```
#include <reg51.h>           // 包含 51 单片机寄存器定义的头文件
#include <intrins.h>         //包含 _nop_()函数定义的头文件
#define OP_READ 0xa1        // 器件地址以及读取操作,0xa1 即为 1010
0001B
#define OP_WRITE 0xa0       // 器件地址以及写入操作,0xa1 即为 1010
0000B
sbit SDA=P3^4;              //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;              //将串行时钟总线 SDA 位定义在为 P3.3 引脚
```

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

函数功能：延时 1ms

$(3j+2)*i=(3 \times 33+2) \times 10=1010$ (微秒)，可以认为是 1 毫秒

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

```
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
```

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

函数功能：延时若干毫秒

入口参数：n

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

```
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
```

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

函数功能：开始数据传送

```

*****/
void start()
// 开始位
{
    SDA = 1;    //SDA 初始化为高电平“1”
    SCL = 1;    //开始数据传送时，要求 SCL 为高电平“1”
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA = 0;    //SDA 的下降沿被认为是开始信号
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SCL = 0;    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的数据
传递)
}
/*****
函数功能：结束数据传送
*****/
void stop()
// 停止位
{
    SDA = 0;    //SDA 初始化为低电平“0” _n
    SCL = 1;    //结束数据传送时，要求 SCL 为高电平“1”
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA = 1;    //SDA 的上升沿被认为是结束信号
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA=0;
    SCL=0;
}
/*****
函数功能：从 AT24Cxx 读取数据
出口参数：x
*****/
unsigned char ReadData()
// 从 AT24Cxx 移入数据到 MCU

```

```

{
    unsigned char i;
    unsigned char x; //储存从 AT24Cxx 中读出的数据
    for(i = 0; i < 8; i++)
    {
        SCL = 1; //SCL 置为高电平
        x<<=1; //将 x 中的各二进位向左移一位
        x|=(unsigned char)SDA; //将 SDA 上的数据通过按位“或”运算存入 x
    }
    SCL = 0; //在 SCL 的下降沿读出数据
}
return(x); //将读取的数据返回
}
/*****
函数功能：向 AT24Cxx 的当前地址写入数据
入口参数：y (储存待写入的数据)
*****/
//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0
bit WriteCurrent(unsigned char y)
{
    unsigned char i;
    bit ack_bit; //储存应答位
    for(i = 0; i < 8; i++) // 循环移入 8 个位
    {
        SDA = (bit)(y&0x80); //通过按位“与”运算将最高位数据送到 S
        //因为传送时高位在前,低位在后
        _nop_(); //等待一个机器周期
        SCL = 1; //在 SCL 的上升沿将数据写入 AT24Cxx
        _nop_(); //等待一个机器周期
        _nop_(); //等待一个机器周期

        SCL = 0; //将 SCL 重新置为低电平,以在 SCL 线形成传送
        //数据所需的 8 个脉冲
        y <<= 1; //将 y 中的各二进位向左移一位
    }
    SDA = 1; // 发送设备（主机）应在时钟脉冲的高电平期间
    // (SCL=1)释放 SDA 线,
    //以让 SDA 线转由接收设备(AT24Cxx)控制
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    SCL = 1; //根据上述规定, SCL 应为高电平
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
}

```

```

    _nop_();          //等待一个机器周期
    ack_bit = SDA; //接受设备 (AT24Cxx)向 SDA 送低电平, 表示已经接收到
                    //若送高电平, 表示没有接收到, 传送异常
                    //SCL 为低电平时, SDA 上数据才允许变化(即允许以后的
                    //数据传递)
    SCL = 0;          //SCL 为低电平时, SDA 上数据才允许变化(即允许以后的
                    //数据传递)
    return  ack_bit;          // 返回 AT24Cxx 应答位
}
/*****
函数功能: 向 AT24Cxx 中的指定地址写入数据
入口参数: add (储存指定的地址); dat(储存待写入的数据)
*****/
void WriteSet(unsigned char add, unsigned char dat)
// 在指定地址 addr 处写入数据 WriteCurrent
{
    start();          //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片, 并告知要对其
    写入数据
    WriteCurrent(add);          //写入指定地址
    WriteCurrent(dat);          //向当前地址 (上面指定的地址) 写入数据
    stop();                    //停止数据传递
    delaynms(4);              //1 个字节的写入周期为 1ms, 最好延时 1ms 以上
}
/*****
函数功能: 从 AT24Cxx 中的当前地址读取数据
出口参数: x (储存读出的数据)
*****/
unsigned char ReadCurrent()
{
    unsigned char x;
    start();          //开始数据传递
    WriteCurrent(OP_READ); //选择要操作的 AT24Cxx 芯片, 并告知要读其
    数据
    x=ReadData();          //将读取的数据存入 x
    stop();                //停止数据传递
    return x;              //返回读取的数据
}
/*****
函数功能: 从 AT24Cxx 中的指定地址读取数据
入口参数: set_addr
出口参数: x
*****/
unsigned char ReadSet(unsigned char set_addr)
// 在指定地址读取

```



```

{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片，并告知
    要对其写入数据
    WriteCurrent(set_addr); //写入指定地址
    return(ReadCurrent());  //从指定地址读出数据并返回
}
/*****
函数功能：主函数
*****/
main(void)
{
    SDA = 1;                // SDA=1,SCL=1,使主从设备处于空闲状态
    SCL = 1;
    WriteSet(0x36,0x0f);   //在指定地址“0x36”中写入数据“0x0f”
    P1=ReadSet(0x36);      //从指定地址“0x36”中读取数据并送 P1 口显示
}

```

实例 86：将按键次数写入 AT24C02，再读出并用 1602LCD 显示

```

#include<reg51.h>          //包含单片机寄存器的头文件
#include<intrins.h>       //包含_nop_()函数定义的头文件
sbit RS=P2^0;             //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;             //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;              //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;             //忙碌标志位，将 BF 位定义为 P0.7 引脚
sbit S=P1^4;              //将 S 位定义为 P1.4 引脚
#define OP_READ 0xa1      // 器件地址以及读取操作,0xa1 即为 1010 0001B
#define OP_WRITE 0xa0     // 器件地址以及写入操作,0xa0 即为 1010 0000B
sbit SDA=P3^4;            //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;            //将串行时钟总线 SDA 位定义在为 P3.3 引脚
unsigned char code digit[ ]={"0123456789"}; //定义字符数组显示数字
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)

```

```

        for(j=0;j<33;j++)
            ;
    }
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/

void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
以下是对液晶模块的操作程序
*****/

/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌;result=0，不忙
*****/

unsigned char BusyTest(void)
{
    bit result;
    RS=0;        //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;        //E=1，才允许读写
    _nop_();    //空操作
    _nop_();
    _nop_();
    _nop_();    //空操作四个机器周期，给硬件反应时间
    result=BF;  //将忙碌标志电平赋给 result
    E=0;        //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/

void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                //根据规定，RS 和 RW 同时为低电平时，可以
    写入指令
    RW=0;

```

```

    E=0;                //E 置低电平(根据表 8-6, 写指令时, E 为高脉
冲,                    // 就是让 E 从 0 到 1 发生正跳变, 所以应先置
"0"
    _nop_();
    _nop_();            //空操作两个机器周期, 给硬件反应时间
    P0=dictate;        //将数据送入 P0 口, 即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();            //空操作四个机器周期, 给硬件反应时间
    E=1;                //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();            //空操作四个机器周期, 给硬件反应时间
    E=0;                //当 E 由高电平跳变成低电平时, 液晶模块开始
执行命令
}

```

函数功能: 指定字符显示的实际地址

入口参数: x

*****/

```

void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}

```

函数功能: 将数据(字符的标准 ASCII 码)写入液晶模块

入口参数: y(为字符常量)

*****/

```

void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;                //RS 为高电平, RW 为低电平时, 可以写入数据
    RW=0;
    E=0;                //E 置低电平(根据表 8-6, 写指令时, E 为高脉冲,
                        // 就是让 E 从 0 到 1 发生正跳变, 所以应先置"0"
    P0=y;                //将数据送入 P0 口, 即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();            //空操作四个机器周期, 给硬件反应时间
    E=1;                //E 置高电平
}

```

```

    _nop_();
    _nop_();
    _nop_();
    _nop_();           //空操作四个机器周期，给硬件反应时间
    E=0;              //当 E 由高电平跳变成低电平时，液晶模块开始执行命令
}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delaynms(15);      //延时 15ms，首次写指令时应给 LCD 一段
较长的反应时间
    WriteInstruction(0x38); //显示模式设置：16×2 显示，5×7 点阵，8
位数据接口
    delaynms(5);       //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38);
    delaynms(5);
    WriteInstruction(0x38); //连续三次，确保初始化成功
    delaynms(5);
    WriteInstruction(0x0c); //显示模式设置：显示开，无光标，光标不闪烁
    delaynms(5);
    WriteInstruction(0x06); //显示模式设置：光标右移，字符不移
    delaynms(5);
    WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
    delaynms(5);
}
/*****
函数功能：显示小时
*****/
void Display(unsigned char x)
{
    unsigned char i,j;
    i=x/10;           //取整运算，求得十位数字
    j=x%10;           //取余运算，求得各位数字
    WriteAddress(0x44); //写显示地址，将十位数字显示在第 2 行第 5 列
    WriteData(digit[i]); //将十位数字的字符常量写入 LCD
    WriteData(digit[j]); //将个位数字的字符常量写入 LCD
}
/*****
以下是对 AT24C02 的读写操作程序
*****/

```

```

/*****
函数功能：开始数据传送
*****/

void start()
// 开始位
{
    SDA = 1;    //SDA 初始化为高电平“1”
    SCL = 1;    //开始数据传送时，要求 SCL 为高电平“1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 0;    //SDA 的下降沿被认为是开始信号
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 0;    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的数据
传递)
}
/*****
函数功能：结束数据传送
*****/

void stop()
// 停止位
{
    SDA = 0;    //SDA 初始化为低电平“0”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 1;    //结束数据传送时，要求 SCL 为高电平“1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 1;    //SDA 的上升沿被认为是结束信号
}
/*****
函数功能：从 AT24Cxx 读取数据
出口参数：x
*****/

unsigned char ReadData()
// 从 AT24Cxx 移入数据到 MCU
{
    unsigned char i;
    unsigned char x;    //储存从 AT24Cxx 中读出的数据
    for(i = 0; i < 8; i++)

```

```

    {
        SCL = 1;           //SCL 置为高电平
        x<<=1;           //将 x 中的各二进位向左移一位
        x|=(unsigned char)SDA; //将 SDA 上的数据通过按位“或”运算存入 x
中
        SCL = 0;           //在 SCL 的下降沿读出数据
    }
    return(x);           //将读取的数据返回
}
/*****
函数功能：向 AT24Cxx 的当前地址写入数据
入口参数：y (储存待写入的数据)
*****/
//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0
bit WriteCurrent(unsigned char y)
{
    unsigned char i;
    bit ack_bit;           //储存应答位
    for(i = 0; i < 8; i++) // 循环移入 8 个位
    {
        SDA = (bit)(y&0x80); //通过按位“与”运算将最高位数据送到 S
                               //因为传送时高位在前,低位在后
        _nop_();           //等待一个机器周期
        SCL = 1;           //在 SCL 的上升沿将数据写入 AT24Cxx
        _nop_();           //等待一个机器周期
        _nop_();           //等待一个机器周期

        SCL = 0;           //将 SCL 重新置为低电平,以在 SCL 线形成传送
数据所需的 8 个脉冲
        y <<= 1;           //将 y 中的各二进位向左移一位
    }
    SDA = 1;           // 发送设备（主机）应在时钟脉冲的高电平期间
(SCL=1)释放 SDA 线,
                               //以让 SDA 线转由接收设备(AT24Cxx)控制
    _nop_();           //等待一个机器周期
    _nop_();           //等待一个机器周期
    SCL = 1;           //根据上述规定, SCL 应为高电平
    _nop_();           //等待一个机器周期
    _nop_();           //等待一个机器周期
    _nop_();           //等待一个机器周期
    _nop_();           //等待一个机器周期
    ack_bit = SDA; //接受设备（AT24Cxx)向 SDA 送低电平,表示已经接收到
一个字节
                               //若送高电平,表示没有接收到,传送异常
}

```

```

    SCL = 0;          //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的
数据传递)
    return  ack_bit;          // 返回 AT24Cxx 应答位
}
/*****
函数功能：向 AT24Cxx 中的指定地址写入数据
入口参数：add (储存指定的地址)；dat(储存待写入的数据)
*****/
void WriteSet(unsigned char add, unsigned char dat)
// 在指定地址 addr 处写入数据 WriteCurrent
{
    start();          //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片，并告知要对其
写入数据
    WriteCurrent(add);      //写入指定地址
    WriteCurrent(dat);      //向当前地址（上面指定的地址）写入数据
    stop();                //停止数据传递
    delaynms(4);          //1 个字节的写入周期为 1ms，最好延时 1ms 以上
}
/*****
函数功能：从 AT24Cxx 中的当前地址读取数据
出口参数：x (储存读出的数据)
*****/
unsigned char ReadCurrent()
{
    unsigned char x;
    start();          //开始数据传递
    WriteCurrent(OP_READ); //选择要操作的 AT24Cxx 芯片，并告知要读其
数据
    x=ReadData();      //将读取的数据存入 x
    stop();            //停止数据传递
    return x;          //返回读取的数据
}
/*****
函数功能：从 AT24Cxx 中的指定地址读取数据
入口参数：set_add
出口参数：x
*****/
unsigned char ReadSet(unsigned char set_add)
// 在指定地址读取
{
    start();          //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片，并告知
要对其写入数据

```

```

    WriteCurrent(set_add);        //写入指定地址
    return(ReadCurrent());       //从指定地址读出数据并返回
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char sum;          //储存计数值
    unsigned char x;           //储存从 AT24C02 读出的值
    LcdInitiate();             //调用 LCD 初始化函数
    sum=0;                      //将计数值初始化为 0
    while(1)                   //无限循环
    {
        if(S==0)               //如果该键被按下
        {
            delaynms(80);      //软件消抖，延时 80ms
            if(S==0)           //确实该键被按下
                sum++;          //计件值加 1
            if(sum==99)        //如果计满 99
                sum=0;         //清 0，重新开始计数
        }
        WriteSet(0x01,sum);    //将计件值写入 AT24C02 中的指定地址"0x01"
        x=ReadSet(0x01);       //从 AT24C02 中读出计件值
        Display(x);            //将计件值用 1602LCD 显示
    }
}

```

实例 87：对 I2C 总线上挂接多个 AT24C02 的读写操作

```

#include <reg51.h>              // 包含 51 单片机寄存器定义的头文件
#include <intrins.h>           //包含_nop_()函数定义的头文件
#define OP_READ1 0xa1         // 器件 1 地址以及读取操作,0xa1 即为 1010
                                0001B
#define OP_WRITE1 0xa0       // 器件 1 地址以及写入操作,0xa1 即为 1010
                                0000B
#define OP_READ2 0xaf        // 器件 2 地址以及读取操作,0xa1 即为 1010
                                1111B
#define OP_WRITE2 0xae       // 器件 2 地址以及写入操作,0xa1 即为 1010
                                1110B
sbit SDA=P3^4;                //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;                //将串行时钟总线 SDA 位定义在为 P3.3 引脚
/*****

```


函数功能：延时 1ms

$(3j+2)*i=(3\times 33+2)\times 10=1010$ (微秒)，可以认为是 1 毫秒

```

*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****

```

函数功能：延时若干毫秒

入口参数：n

```

*****/
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****

```

函数功能：开始数据传送

```

*****/
void start()
// 开始位
{
    SDA = 1;    //SDA 初始化为高电平“1”
    SCL = 1;    //开始数据传送时，要求 SCL 为高电平“1”
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA = 0;    //SDA 的下降沿被认为是开始信号
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SCL = 0;    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的数据
    传递)
    _nop_();    //等待一个机器周期
}
/*****

```

函数功能：结束数据传送

```

*****/
void stop()
// 停止位

```

```

{
    SDA = 0;      //SDA 初始化为低电平 “0”
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    SCL = 1;     //结束数据传送时，要求 SCL 为高电平 “1”
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    _nop_();     //等待一个机器周期
    SDA = 1;     //SDA 的上升沿被认为是结束信号
}
/*****
函数功能：从 AT24Cxx 读取数据
出口参数：x
*****/
unsigned char ReadData()
// 从 AT24Cxx 移入数据到 MCU
{
    unsigned char i;
    unsigned char x;          //储存从 AT24Cxx 中读出的数据
    for(i = 0; i < 8; i++)
    {
        SCL = 1;            //SCL 置为高电平
        x<<=1;              //将 x 中的各二进位向左移一位
        x|=(unsigned char)SDA; //将 SDA 上的数据通过按位“或”运算存入 x
中
        SCL = 0;            //在 SCL 的下降沿读出数据
    }
    return(x);              //将读取的数据返回
}
/*****
函数功能：向 AT24Cxx 的当前地址写入数据
入口参数：y (储存待写入的数据)
*****/
//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0
bit WriteCurrent(unsigned char y)
{
    unsigned char i;
    bit ack_bit;             //储存应答位
    for(i = 0; i < 8; i++)   // 循环移入 8 个位
    {
        SDA = (bit)(y&0x80); //通过按位“与”运算将最高位数据送到 S

```

```

        //因为传送时高位在前，低位在后
        //等待一个机器周期
        _nop_();
        SCL = 1; //在 SCL 的上升沿将数据写入 AT24Cxx
        //等待一个机器周期
        _nop_();
        //等待一个机器周期
        _nop_();

        SCL = 0; //将 SCL 重新置为低电平，以在 SCL 线形成传
送数据所需的 8 个脉冲
        y <<= 1; //将 y 中的各二进位向左移一位
    }
    SDA = 1; // 发送设备（主机）应在时钟脉冲的高电平期间
(SCL=1)释放 SDA 线，
        //以让 SDA 线转由接收设备(AT24Cxx)控制
        _nop_(); //等待一个机器周期
        _nop_(); //等待一个机器周期
        SCL = 1; //根据上述规定，SCL 应为高电平
        _nop_(); //等待一个机器周期
        _nop_(); //等待一个机器周期
        _nop_(); //等待一个机器周期
        _nop_(); //等待一个机器周期
        ack_bit = SDA; //接受设备（AT24Cxx）向 SDA 送低电平，表示已经接收到
一个字
        //若送高电平，表示没有接收到，传送异常
        SCL = 0; //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的
数据传递)
    return ack_bit; // 返回 AT24Cxx 应答位
}
/*****
函数功能：向第一个 AT24Cxx 中的指定地址写入数据
入口参数：add (储存指定的地址)；dat(储存待写入的数据)
*****/
void WriteSet1(unsigned char add, unsigned char dat)
// 在指定地址 addr 处写入数据 WriteCurrent
{
    start(); //开始数据传递
    WriteCurrent(OP_WRITE1); //选择要操作的第一个 AT24Cxx 芯片，并告
知要对其写入数据
    WriteCurrent(add); //写入指定地址
    WriteCurrent(dat); //向当前地址（上面指定的地址）写入数据
    stop(); //停止数据传递
    delaynms(4); //1 个字节的写入周期为 1ms，最好延时 1ms 以
上
}
/*****

```

函数功能：向第二个 AT24Cxx 中的指定地址写入数据

入口参数：add (储存指定的地址)；dat(储存待写入的数据)

*****/

```
void WriteSet2(unsigned char add, unsigned char dat)
```

```
// 在指定地址 addr 处写入数据 WriteCurrent
```

```
{
```

```
    start();                //开始数据传递
```

```
    WriteCurrent(OP_WRITE2); //选择要操作的 AT24Cxx 芯片，并告知要对其写入数据
```

```
    WriteCurrent(add);      //写入指定地址
```

```
    WriteCurrent(dat);     //向当前地址（上面指定的地址）写入数据
```

```
    stop();                //停止数据传递
```

```
    delaynms(4);          //1个字节的写入周期为 1ms，最好延时 1ms 以
```

上

```
}
```

函数功能：从第一个 AT24Cxx 中的当前地址读取数据

出口参数：x (储存读出的数据)

*****/

```
unsigned char ReadCurrent1()
```

```
{
```

```
    unsigned char x;
```

```
    start();                //开始数据传递
```

```
    WriteCurrent(OP_READ1); //选择要操作的第一个 AT24Cxx 芯片，并告知要读其数据
```

```
    x=ReadData();          //将读取的数据存入 x
```

```
    stop();                //停止数据传递
```

```
    return x;              //返回读取的数据
```

```
}
```

函数功能：从第二个 AT24Cxx 中的当前地址读取数据

出口参数：x (储存读出的数据)

*****/

```
unsigned char ReadCurrent2()
```

```
{
```

```
    unsigned char x;
```

```
    start();                //开始数据传递
```

```
    WriteCurrent(OP_READ2); //选择要操作的第二个 AT24Cxx 芯片，并告知要读其数据
```

```
    x=ReadData();          //将读取的数据存入 x
```

```
    stop();                //停止数据传递
```

```
    return x;              //返回读取的数据
```

```
}
```

```

/*****
函数功能：从第一个 AT24Cxx 中的指定地址读取数据
入口参数：set_addr
出口参数：x
*****/
unsigned char ReadSet1(unsigned char set_addr)
// 在指定地址读取
{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE1);    //选择要操作的第一个 AT24Cxx 芯片，
    并告知要对其写入数据
    WriteCurrent(set_addr);    //写入指定地址
    return(ReadCurrent1());    //从第一个 AT24Cxx 芯片指定地址读出数
    据并返回
}
/*****
函数功能：从第二个 AT24Cxx 中的指定地址读取数据
入口参数：set_addr
出口参数：x
*****/
unsigned char ReadSet2(unsigned char set_addr)
// 在指定地址读取
{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE2);    //选择要操作的第二个 AT24Cxx 芯片，
    并告知要对其写入数据
    WriteCurrent(set_addr);    //写入指定地址
    return(ReadCurrent2());    //从第二个 AT24Cxx 芯片指定地址读出数
    据并返回
}
/*****
函数功能：主函数
*****/
main(void)
{
    unsigned char x;
    SDA = 1;                // SDA=1,SCL=1,使主从设备处于空闲状态
    SCL = 1;
    WriteSet1(0x36,0xaa);    //将数据"0xaa"写入第一个 AT24C02 的指定地址
    "0x36"
    x=ReadSet1(0x36);        //从第二个 AT24C02 中的指定地址"0x36"读出
    数据
    WriteSet2(0x48,x);        //将读出的数据写入第二个 AT24C02 的指定地址
    "0x48"?
}

```

```

    P1=ReadSet2(0x48);    //将从第二个 AT24C02 的指定地址读出的数据
    送 P1 口显示验证
}

```

实例 88：基于 AT24C02 的多机通信 读取程序

```

#include <reg51.h>        // 包含 51 单片机寄存器定义的头文件
#include <intrins.h>     //包含_nop_()函数定义的头文件
#define OP_READ 0xa1     // 器件 1 地址以及读取操作,0xa1 即为 1010
                          0001B
#define OP_WRITE 0xa0    // 器件 1 地址以及写入操作,0xa1 即为 1010
                          0000B
sbit SDA=P3^4;          //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;          //将串行时钟总线 SDA 位定义在为 P3.3 引脚
sbit flag=P3^0;

```

```

/*****

```

函数功能：延时 1ms

$(3j+2)*i=(3 \times 33+2) \times 10=1010$ (微秒)，可以认为是 1 毫秒

```

*****/

```

```

void delay1ms()

```

```

{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}

```

```

/*****

```

函数功能：延时若干毫秒

入口参数：n

```

*****/

```

```

void delaynms(unsigned char n)

```

```

{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}

```

```

/*****

```

函数功能：开始数据传送

```

*****/

```

```

void start()

```

// 开始位

```

{
    SDA = 1;    //SDA 初始化为高电平“1”
}

```

```

    SCL = 1;    //开始数据传送时, 要求 SCL 为高电平 “1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 0;   //SDA 的下降沿被认为是开始信号
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 0;   //SCL 为低电平时, SDA 上数据才允许变化(即允许以后的数据
传递)
    _nop_();   //等待一个机器周期
}
/*****
函数功能: 结束数据传送
*****/
void stop()
// 停止位
{
    SDA = 0;    //SDA 初始化为低电平 “0”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 1;    //结束数据传送时, 要求 SCL 为高电平 “1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 1;    //SDA 的上升沿被认为是结束信号
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
}
/*****
函数功能: 从 AT24Cxx 读取数据
出口参数: x
*****/
unsigned char ReadData()
// 从 AT24Cxx 移入数据到 MCU
{
    unsigned char i;
    unsigned char x;           //储存从 AT24Cxx 中读出的数据
    for(i = 0; i < 8; i++)
    {
        SCL = 1;              //SCL 置为高电平
        x<<=1;                 //将 x 中的各二进位向左移一位
    }
}

```

```

        x|=(unsigned char)SDA; //将 SDA 上的数据通过按位“或”运算存入 x
中
        SCL = 0;                //在 SCL 的下降沿读出数据
    }
    return(x);                  //将读取的数据返回
}
/*****
函数功能：向 AT24Cxx 的当前地址写入数据
入口参数：y (储存待写入的数据)
*****/
//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0
bit WriteCurrent(unsigned char y)
{
    unsigned char i;
    bit ack_bit;                //储存应答位
    for(i = 0; i < 8; i++)      //循环移入 8 个位
    {
        SDA = (bit)(y&0x80);    //通过按位“与”运算将最高位数据送到 S
                                //因为传送时高位在前，低位在后
        _nop_();                //等待一个机器周期
        SCL = 1;                //在 SCL 的上升沿将数据写入 AT24Cxx
        _nop_();                //等待一个机器周期
        _nop_();                //等待一个机器周期

        SCL = 0;                //将 SCL 重新置为低电平，以在 SCL 线形成传
送数据所需的 8 个脉冲
        y <<= 1;                //将 y 中的各二进位向左移一位
    }
    SDA = 1;                    //发送设备（主机）应在时钟脉冲的高电平期间
(SCL=1)释放 SDA 线，
                                //以让 SDA 线转由接收设备(AT24Cxx)控制
    _nop_();                    //等待一个机器周期
    _nop_();                    //等待一个机器周期
    SCL = 1;                    //根据上述规定，SCL 应为高电平
    _nop_();                    //等待一个机器周期
    _nop_();                    //等待一个机器周期
    _nop_();                    //等待一个机器周期
    _nop_();                    //等待一个机器周期
    ack_bit = SDA; //接受设备（AT24Cxx)向 SDA 送低电平，表示已经接收到
一个字节
                                //若送高电平，表示没有接收到，传送异常
    SCL = 0;                    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的
数据传递)
    return ack_bit; //返回 AT24Cxx 应答位
}

```



```

}

/*****
函数功能：从第一个 AT24Cxx 中的当前地址读取数据
出口参数：x (储存读出的数据)
*****/
unsigned char ReadCurrent()
{
    unsigned char x;
    start();                //开始数据传递
    WriteCurrent(OP_READ); //选择要操作的 AT24Cxx 芯片，并告知要读其
数据
    x=ReadData();          //将读取的数据存入 x
    stop();                //停止数据传递
    return x;              //返回读取的数据
}

/*****
函数功能：从 AT24Cxx 中的指定地址读取数据
入口参数：set_addr
出口参数：x
*****/
unsigned char ReadSet(unsigned char set_addr)
// 在指定地址读取
{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片，并告知
要对其写入数据
    WriteCurrent(set_addr); //写入指定地址
    return(ReadCurrent()); //从第一个 AT24Cxx 芯片指定地址读出数
据并返回
}

/*****
函数功能：主函数
*****/
main(void)
{
    SDA = 1;                // SDA=1,SCL=1,使主从设备处于空闲状态
    SCL = 1;
    while(1)
    {
        while(flag==1)
            ;
    }
}

```

```

        P1=ReadSet(0x36);        //从第二个 AT24C02 中的指定地址"0x36"读
    出数据
        delaynms(90);
    }
}

```

实例 88：基于 AT24C02 的多机通信 写入程序

```

#include <reg51.h>           // 包含 51 单片机寄存器定义的头文件
#include <intrins.h>        //包含_nop_()函数定义的头文件
#define OP_READ 0xa1        // 器件 1 地址以及读取操作,0xa1 即为 1010
0001B
#define OP_WRITE 0xa0       // 器件 1 地址以及写入操作,0xa1 即为 1010
0000B
sbit SDA=P3^4;              //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;              //将串行时钟总线 SDA 位定义在为 P3.3 引脚
sbit flag=P3^0;
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：开始数据传送
*****/
void start()
// 开始位

```

```

{
    SDA = 1;    //SDA 初始化为高电平 “1”
    SCL = 1;    //开始数据传送时，要求 SCL 为高电平 “1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 0;    //SDA 的下降沿被认为是开始信号
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 0;    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的数据
    传递)
    _nop_();   //等待一个机器周期
}

```

```

/*****

```

函数功能：结束数据传送

```

*****/

```

```

void stop()

```

```

// 停止位

```

```

{
    SDA = 0;    //SDA 初始化为低电平 “0”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SCL = 1;    //结束数据传送时，要求 SCL 为高电平 “1”
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    SDA = 1;    //SDA 的上升沿被认为是结束信号
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
    _nop_();   //等待一个机器周期
}

```

```

/*****

```

函数功能：向 AT24Cxx 的当前地址写入数据

入口参数：y (储存待写入的数据)

```

*****/

```

//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0

```

bit WriteCurrent(unsigned char y)

```

```

{
    unsigned char i;
    bit ack_bit;    //储存应答位

```

```

for(i = 0; i < 8; i++)    // 循环移入 8 个位
{
    SDA = (bit)(y&0x80);    //通过按位“与”运算将最高位数据送到 S
                            //因为传送时高位在前，低位在后
    _nop_();                //等待一个机器周期
    SCL = 1;                //在 SCL 的上升沿将数据写入 AT24Cxx
    _nop_();                //等待一个机器周期
    _nop_();                //等待一个机器周期

    SCL = 0;                //将 SCL 重新置为低电平，以在 SCL 线形成传
送数据所需的 8 个脉冲
    y <<= 1;                //将 y 中的各二进位向左移一位
}
SDA = 1;                    // 发送设备（主机）应在时钟脉冲的高电平期间
(SCL=1)释放 SDA 线，
                            //以让 SDA 线转由接收设备(AT24Cxx)控制
_nop_();                    //等待一个机器周期
_nop_();                    //等待一个机器周期
SCL = 1;                    //根据上述规定，SCL 应为高电平
_nop_();                    //等待一个机器周期
_nop_();                    //等待一个机器周期
_nop_();                    //等待一个机器周期
_nop_();                    //等待一个机器周期
ack_bit = SDA; //接受设备（AT24Cxx)向 SDA 送低电平，表示已经接收到
一个字节
                            //若送高电平，表示没有接收到，传送异常
SCL = 0;                    //SCL 为低电平时，SDA 上数据才允许变化(即允许以后的
数据传递)
return  ack_bit;// 返回 AT24Cxx 应答位
}
/*****
函数功能：向 AT24Cxx 中的指定地址写入数据
入口参数：add (储存指定的地址)；dat(储存待写入的数据)
*****/
void WriteSet(unsigned char add, unsigned char dat)
// 在指定地址 addr 处写入数据 WriteCurrent
{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的第一个 AT24Cxx 芯片，并告知
要对其写入数据
    WriteCurrent(add);      //写入指定地址
    WriteCurrent(dat);      //向当前地址（上面指定的地址）写入数据
    stop();                 //停止数据传递
}

```

```

    delaynms(4);           //1 个字节的写入周期为 1ms, 最好延时 1ms 以
上
}

/*****
函数功能: 主函数
*****/
main(void)
{
    TMOD=0x01;
    TH0=(65536-46083)/256;
    TL0=(65536-46083)%256;
    EA=1;
    ET0=1;
    TR0=1;
    flag=1;
    while(1)
    {
        while(flag==1)
        {
            WriteSet(0x36,0xf0); //将数据"0xf0"写入第一个 AT24C02 的指定地
址"0x36"
            delaynms(50);       //延时 50ms
        }
        while(flag==0)
        ;
    }
}
/*****
函数功能: 定时器 T0 的中断函数, 使 P3.0 引脚输出 100ms 方波
*****/
void Time0(void) interrupt 1 using 1
{
    TH0=(65536-46083)/256;
    TL0=(65536-46083)%256;
    flag=!flag;
}

```

//实例 89: 将"渴望"乐谱写入 AT24C02 并读出播放

```

#include <reg51.h>           // 包含 51 单片机寄存器定义的头文件
#include <intrins.h>         //包含_nop_()函数定义的头文件
#define OP_READ 0xa1        // 器件地址以及读取操作,0xa1 即为 1010
0001B

```

```

#define OP_WRITE 0xa0      // 器件地址以及写入操作,0xa1 即为 1010
                             0000B
sbit SDA=P3^4;           //将串行数据总线 SDA 位定义在为 P3.4 引脚
sbit SCL=P3^3;           //将串行时钟总线 SDA 位定义在为 P3.3 引脚
sbit sound=P3^7;         //将 sound 位定义为 P3.7,从该引脚输出音频
unsigned int C;          //储存定时器的定时常数

```

//以下是 C 调低音的音频宏定义

```

#define l_dao 262        //将“l_dao”宏定义为低音“1”的频率 262Hz
#define l_re 286         //将“l_re”宏定义为低音“2”的频率 286Hz
#define l_mi 311         //将“l_mi”宏定义为低音“3”的频率 311Hz
#define l_fa 349         //将“l_fa”宏定义为低音“4”的频率 349Hz
#define l_sao 392        //将“l_sao”宏定义为低音“5”的频率 392Hz
#define l_la 440         //将“l_a”宏定义为低音“6”的频率 440Hz
#define l_xi 494         //将“l_xi”宏定义为低音“7”的频率 494Hz

```

//以下是 C 调中音的音频宏定义

```

#define dao 523          //将“dao”宏定义为中音“1”的频率 523Hz
#define re 587           //将“re”宏定义为中音“2”的频率 587Hz
#define mi 659           //将“mi”宏定义为中音“3”的频率 659Hz
#define fa 698           //将“fa”宏定义为中音“4”的频率 698Hz
#define sao 784          //将“sao”宏定义为中音“5”的频率 784Hz
#define la 880           //将“la”宏定义为中音“6”的频率 880Hz
#define xi 987           //将“xi”宏定义为中音“7”的频率 523Hz

```

//以下是 C 调高音的音频宏定义

```

#define h_dao 1046       //将“h_dao”宏定义为高音“1”的频率 1046Hz
#define h_re 1174        //将“h_re”宏定义为高音“2”的频率 1174Hz
#define h_mi 1318        //将“h_mi”宏定义为高音“3”的频率 1318Hz
#define h_fa 1396        //将“h_fa”宏定义为高音“4”的频率 1396Hz
#define h_sao 1567       //将“h_sao”宏定义为高音“5”的频率 1567Hz
#define h_la 1760        //将“h_la”宏定义为高音“6”的频率 1760Hz
#define h_xi 1975        //将“h_xi”宏定义为高音“7”的频率 1975Hz

```

/******

函数功能：节拍的延时的基本单位，延时 200ms

*****/

```

void delay()
{
    unsigned char i,j;
    for(i=0;i<250;i++)
        for(j=0;j<250;j++)
            ;
}

```

/******

以下是对 AT24C02 进行读写操作的源程序

*****/

```

/*****
函数功能：延时 1ms
 $(3j+2)*i=(3\times 33+2)\times 10=1010$ (微秒),可以认为是 1 毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：开始数据传送
*****/
void start()
{
    SDA = 1;    //SDA 初始化为高电平"1"
    SCL = 1;    //开始数据传送时,要求 SCL 为高电平"1"
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA = 0;    //SDA 的下降沿被认为是开始信号
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SCL = 0;    //SCL 为低电平时,SDA 上数据才允许变化(即允许以后的数据
传递)
}
/*****
函数功能：结束数据传送
*****/
void stop()
{
    SDA = 0;    //SDA 初始化为低电平"0"

```

```

    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SCL = 1;    //结束数据传送时,要求 SCL 为高电平"1"
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    _nop_();    //等待一个机器周期
    SDA = 1;    //SDA 的上升沿被认为是结束信号
}
/*****
函数功能: 从 AT24Cxx 读取数据
出口参数: x
*****/
unsigned char ReadData()
{
    unsigned char i;
    unsigned char x;           //储存从 AT24Cxx 中读出的数据
    for(i = 0;i < 8;i++)
    {
        SCL = 1;             //SCL 置为高电平
        x<<=1;               //将 x 中的各二进位向左移一位
        x|=(unsigned char)SDA; //将 SDA 上的数据通过按位"或"运算存入 x 中
        SCL = 0;             //在 SCL 的下降沿读出数据
    }
    return(x);               //将读取的数据返回
}
/*****
函数功能: 向 AT24Cxx 的当前地址写入数据
入口参数: y (储存待写入的数据)
*****/
//在调用此数据写入函数前需首先调用开始函数 start(),所以 SCL=0
bit WriteCurrent(unsigned char y)
{
    unsigned char i;
    bit ack_bit;              //储存应答位
    for(i = 0; i < 8; i++)    // 循环移入 8 个位
    {
        SDA = (bit)(y&0x80); //通过按位"与"运算将最高位数据送到 S
                             //因为传送时高位在前,低位在后
        _nop_();              //等待一个机器周期
        SCL = 1;              //在 SCL 的上升沿将数据写入 AT24Cxx
        _nop_();              //等待一个机器周期
        _nop_();              //等待一个机器周期
    }
}

```



```

        SCL = 0; //将 SCL 重新置为低电平,以在 SCL 线形成传送数据所需的
8 个脉冲
        y <<= 1;          //将 y 中的各二进位向左移一位
    }
    SDA = 1; // 发送设备(主机)应在时钟脉冲的高电平期间(SCL=1)释放 SDA
线,
        //以让 SDA 线转由接收设备(AT24Cxx)控制
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    SCL = 1; //根据上述规定,SCL 应为高电平
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    _nop_(); //等待一个机器周期
    ack_bit = SDA; //接受设备(AT24Cxx)向 SDA 送低电平,表示已经接收到一个
字节
        //若送高电平,表示没有接收到,传送异常
    SCL = 0; //SCL 为低电平时,SDA 上数据才允许变化(即允许以后的数
据传递)
    return  ack_bit;// 返回 AT24Cxx 应答位
}
/*****
函数功能: 向 AT24Cxx 中的指定地址写入数据
入口参数: add (储存指定的地址);dat(储存待写入的数据)
*****/
void WriteSet(unsigned char add, unsigned char dat)
{
    start(); //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片,并告知要对其
写入数据
    WriteCurrent(add); //写入指定地址
    WriteCurrent(dat); //向当前地址(上面指定的地址)写入数据
    stop(); //停止数据传递
    delaynms(4); //1 个字节的写入周期为 1ms, 最好延时 1ms 以
上
}
/*****
函数功能: 从 AT24Cxx 中的当前地址读取数据
出口参数: x (储存读出的数据)
*****/
unsigned char ReadCurrent()
{
    unsigned char x;
    start(); //开始数据传递

```

```

    WriteCurrent(OP_READ); //选择要操作的 AT24Cxx 芯片,并告知要读其
数据
    x=ReadData();          //将读取的数据存入 x
    stop();                //停止数据传递
    return x;              //返回读取的数据
}
/*****
函数功能: 从 AT24Cxx 中的指定地址读取数据
入口参数: set_addr
出口参数: x
*****/
unsigned char ReadSet(unsigned char set_addr)
{
    start();                //开始数据传递
    WriteCurrent(OP_WRITE); //选择要操作的 AT24Cxx 芯片,并告知要对其写
入数据
    WriteCurrent(set_addr); //写入指定地址
    return(ReadCurrent()); //从指定地址读出数据并返回
}
/*****
函数功能: 主函数
*****/
main(void)
{
    unsigned char i,j;
    unsigned char temp; //储存压缩后的音频
    unsigned char Ji;   //储存音符节拍
    unsigned char N;    //储存音符的最大个数以在 AT24C02 中为音符和节
拍分配存储空间
    unsigned int fr;    //储存解压缩后的音频
    //以下是《渴望》片头曲的一段简谱
    unsigned int code f[]={re,mi,re,dao,l_la,dao,l_la,
                            l_sao,l_mi,l_sao,l_la,dao,
                            l_la,dao,sao,la,mi,sao,
                            re,
                            mi,re,mi,sao,mi,
                            l_sao,l_mi,l_sao,l_la,dao,
                            l_la,l_la,dao,l_la,l_sao,l_re,l_mi,
                            l_sao,
                            re,re,sao,la,sao,
                            fa,mi,sao,mi,
                            la,sao,mi,re,mi,l_la,dao,
                            re,
                            mi,re,mi,sao,mi,

```

```

        l_sao,l_mi,l_sao,l_la,dao,
        l_la,dao,re,l_la,dao,re,mi,
        re,
        l_la,dao,re,l_la,dao,re,mi,
        re,
        0x00}; //以频率 0x00 作为简谱的结束标
志
//以下是简谱中每个音符的节拍
unsigned char code JP[ ]={4,1,1,4,1,1,2,
        2,2,2,2,8,
        4,2,3,1,2,2,
        10,
        4,2,2,4,4,
        2,2,2,2,4,
        2,2,2,2,2,2,2,
        10,
        4,4,4,2,2,
        4,2,4,4,
        4,2,2,2,2,2,2,
        10,
        4,2,2,4,4,
        2,2,2,2,6,
        4,2,2,4,1,1,4,
        10,
        4,2,2,4,1,1,4,
        10
        };
EA=1; //开总中断
ET0=1; //定时器 T0 中断允许
TMOD=0x00; // 使用定时器 T0 的模式 1 (13 位计数器)
SDA = 1; // SDA=1,SCL=1,使主从设备处于空闲状态
SCL = 1;
while(1) //无限循环
{
    i=0; //从第 1 个音符频率 f[0]开始写入 AT24C02
    while(f[i]!=0x01) //只要没有读到结束标志就继续写
    入
        {
            temp=(unsigned char)(f[i]/8); //将音频压缩为较小的字符
            变量
            WriteSet(0x00+i,temp); //在指定地址写入数据压缩后
            的音频
            i++; //指向下一个音符音频
        }
}

```

```

        N=i;          //将音符的最大个数存于 N
        i=0;         //从第一个音符节拍 JP[0]开始写入 AT24C02
        while(f[i]!=0x00)
        {
            WriteSet(0x00+N+i,JP[i]); //在指定地址写入音符的节拍
            i++;                       //指向下一个音符音频
        }
        for(i=0;i<N;i++)
        {
            temp=ReadSet(0x00+i); //读出音频
            Ji=ReadSet(0x00+N+i); //读出节拍
            fr=8*temp;           //将音频解压
            C=460830/fr;         //定时常数的计算公式
            TH0=(8192-C)/32;     //可证明这是 13 位计数器 TH0 高
8 位的赋初值方法
            TL0=(8192-C)%32;     //可证明这是 13 位计数器 TL0
低 5 位的赋初值方法
            TR0=1;               //启动定时器 T0
            for(j=0;j<Ji;j++)   //控制节拍数
                delay();       //延时 1 个节拍单位
            TR0=0;              //关闭定时器 T0

        }
        sound=1;               //播放完毕后, 关闭蜂鸣器
        for(i=0;i<8;i++)       //播放完毕后, 停顿一段时间后继续
            delay();
    }
}

```

```

////////////////////////////////////

```

```

/*****

```

函数功能：定时器 T0 的中断服务子程序，使 P3.7 引脚输出音频的方波

```

*****/

```

```

void Time0(void ) interrupt 1 using 1
{
    TH0=(8192-C)/32; //可证明这是 13 位计数器 TH0 高 8 位的赋初值方
法
    TL0=(8192-C)%32; //可证明这是 13 位计数器 TL0 低 5 位的赋初值方
法

```

```

    sound=!sound;    //将 P3.7 引脚输出电平取反，形成方波
}

```

实例 90：DS18B20 温度检测及其液晶显示

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
unsigned char code digit[10]={"0123456789"};    //定义字符数组显示数字
unsigned char code Str[]{"Test by DS18B20"};    //说明显示的是温度
unsigned char code Error[]{"Error!Check!"};    //说明没有检测到 DS18B20
unsigned char code Temp[]{"Temp:"};    //说明显示的是温度
unsigned char code Cent[]{"Cent"};    //温度单位

```

```

/*****

```

以下是对液晶模块的操作程序

```

*****/

```

```

sbit RS=P2^0;    //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;    //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;    //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;    //忙碌标志位，将 BF 位定义为 P0.7 引脚

```

```

*****/

```

函数功能：延时 1ms

$(3j+2)*i=(3 \times 33+2) \times 10=1010$ (微秒)，可以认为是 1 毫秒

```

*****/

```

```

void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}

```

```

*****/

```

函数功能：延时若干毫秒

入口参数：n

```

*****/

```

```

void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}

```

```

*****/

```

函数功能：判断液晶模块的忙碌状态

返回值：**result**。**result=1**，忙碌;**result=0**，不忙

*****/

bit BusyTest(void)

```
{
    bit result;
    RS=0;          //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;          //E=1，才允许读写
    _nop_();      //空操作
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    result=BF;    //将忙碌标志电平赋给 result
    E=0;          //将 E 恢复低电平
    return result;
}
```

函数功能：将模式设置指令或显示地址写入液晶模块

入口参数：**dictate**

*****/

void WriteInstruction (unsigned char dictate)

```
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                  //根据规定，RS 和 R/W 同时为低电平时，可以
    写入指令
    RW=0;
    E=0;                  //E 置低电平(根据表 8-6，写指令时，E 为高脉
    冲，
    // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"

    _nop_();
    _nop_();              //空操作两个机器周期，给硬件反应时间
    P0=dictate;          //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();              //空操作四个机器周期，给硬件反应时间
    E=1;                  //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();              //空操作四个机器周期，给硬件反应时间
    E=0;                  //当 E 由高电平跳变成低电平时，液晶模块开始
    执行命令
}
```

```

}
/*****
函数功能：指定字符显示的实际地址
入口参数：x
*****/
void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}
/*****
函数功能：将数据(字符的标准 ASCII 码)写入液晶模块
入口参数：y(为字符常量)
*****/
void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;           //RS 为高电平，RW 为低电平时，可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6，写指令时，E 为高脉冲，
                  // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"
    P0=y;          //将数据送入 P0 口，即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    E=1;          //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    E=0;          //当 E 由高电平跳变成低电平时，液晶模块开始执行命令
}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delaynms(15); //延时 15ms，首次写指令时应给 LCD 一段
较长的反应时间
    WriteInstruction(0x38); //显示模式设置：16×2 显示，5×7 点阵，8
位数据接口
    delaynms(5); //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38);
    delaynms(5); //延时 5ms ，给硬件一点反应时间
}

```

```

WriteInstruction(0x38); //连续三次，确保初始化成功
delaynms(5); //延时 5ms ，给硬件一点反应时间
WriteInstruction(0x0c); //显示模式设置：显示开，无光标，光标不闪烁
delaynms(5); //延时 5ms ，给硬件一点反应时间
WriteInstruction(0x06); //显示模式设置：光标右移，字符不移
delaynms(5); //延时 5ms ，给硬件一点反应时间
WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
delaynms(5); //延时 5ms ，给硬件一点反应时间
}
/*****
以下是 DS18B20 的操作程序
*****/
sbit DQ=P3^3;
unsigned char time; //设置全局变量，专门用于严格延时
/*****
函数功能：将 DS18B20 传感器初始化，读取应答信号
出口参数：flag
*****/
bit Init_DS18B20(void)
{
    bit flag; //储存 DS18B20 是否存在的标志，flag=0，表示存在；flag=1，
    表示不存在
    DQ = 1; //先将数据线拉高
    for(time=0;time<2;time++) //略微延时约 6 微秒
        ;
    DQ = 0; //再将数据线从高拉低，要求保持 480~960us
    for(time=0;time<200;time++) //略微延时约 600 微秒
        ; //以向 DS18B20 发出一持续 480~960us 的低电平复位脉冲
    DQ = 1; //释放数据线（将数据线拉高）
    for(time=0;time<10;time++)
        ; //延时约 30us（释放总线后需等待 15~60us 让 DS18B20 输出存在脉冲）
    flag=DQ; //让单片机检测是否输出了存在脉冲（DQ=0 表示存在）
    for(time=0;time<200;time++) //延时足够长时间，等待存在脉冲输出完毕
        ;
    return (flag); //返回检测成功标志
}
/*****
函数功能：从 DS18B20 读取一个字节数据
出口参数：dat
*****/
unsigned char ReadOneChar(void)
{
    unsigned char i=0;

```



```

unsigned char dat; //储存读出的一个字节数据
for (i=0;i<8;i++)
{
    DQ = 1;        // 先将数据线拉高
    _nop_();      //等待一个机器周期
    DQ = 0;        //单片机从 DS18B20 读书据时,将数据线从高拉低即
启动读时序
    dat>>=1;
    _nop_();      //等待一个机器周期
    DQ = 1;        //将数据线"人为"拉高,为单片机检测 DS18B20 的输出
电平作准备
    for(time=0;time<2;time++)
        ;        //延时约 6us, 使主机在 15us 内采样
    if(DQ==1)
        dat|=0x80; //如果读到的数据是 1, 则将 1 存入 dat
    else
        dat|=0x00;//如果读到的数据是 0, 则将 0 存入 dat
        //将单片机检测到的电平信号 DQ 存入 r[i]
    for(time=0;time<8;time++)
        ;        //延时 3us,两个读时序之间必须有大于 1us 的
恢复期
}
return(dat);     //返回读出的十进制数据
}

```

```

/*****

```

函数功能：向 DS18B20 写入一个字节数据

入口参数：dat

```

*****/

```

```

WriteOneChar(unsigned char dat)

```

```

{
    unsigned char i=0;
    for (i=0; i<8; i++)
    {
        DQ = 1;        // 先将数据线拉高
        _nop_();      //等待一个机器周期
        DQ=0;         //将数据线从高拉低时即启动写时序
        DQ=dat&0x01;  //利用与运算取出要写的某位二进制数据,
                        //并将其送到数据线上等待 DS18B20 采样
        for(time=0;time<10;time++)
            ;        //延时约 30us, DS18B20 在拉低后的约 15~60us 期间从数据线上
上采样
        DQ=1;        //释放数据线
        for(time=0;time<1;time++)

```

```

        ;//延时 3us,两个写时序间至少需要 1us 的恢复期
        dat>>=1;      //将 dat 中的各二进制位数据右移 1 位
    }
    for(time=0;time<4;time++)
        ;//稍作延时,给硬件一点反应时间
}
/*****
以下是与温度有关的显示设置
*****/
/*****
函数功能: 显示没有检测到 DS18B20
*****/
void display_error(void)
{
    unsigned char i;
    WriteAddress(0x00); //写显示地址,将在第 1 行第 1 列开始显
示
    i = 0;              //从第一个字符开始显示
    while(Error[i] != '\0') //只要没有写到结束标志,就继续写
    {
        WriteData(Error[i]); //将字符常量写入 LCD
        i++;                 //指向下一个字符
        delaynms(100);       //延时 100ms 较长时间,以看清关
于显示的说明
    }
    while(1)             //进入死循环,等待查明原因
        ;
}
/*****
函数功能: 显示说明信息
*****/
void display_explain(void)
{
    unsigned char i;
    WriteAddress(0x00); //写显示地址,将在第 1 行第 1 列开始显
示
    i = 0;              //从第一个字符开始显示
    while(Str[i] != '\0') //只要没有写到结束标志,就继续写
    {
        WriteData(Str[i]); //将字符常量写入 LCD
        i++;                 //指向下一个字符
        delaynms(100);       //延时 100ms 较长时间,以看清关
于显示的说明
    }
}

```

```

}
/*****
函数功能：显示温度符号
*****/
void display_symbol(void)
{
    unsigned char i;
    WriteAddress(0x40); //写显示地址，将在第 2 行第 1 列开始显示
    i = 0; //从第一个字符开始显示
    while(Temp[i] != '\0') //只要没有写到结束标志，就继续写
    {
        WriteData(Temp[i]); //将字符常量写入 LCD
        i++; //指向下一个字符
        delaynms(50); //延时 1ms 给硬件一点反应时间
    }
}

/*****
函数功能：显示温度的小数点
*****/
void display_dot(void)
{
    WriteAddress(0x49); //写显示地址，将在第 2 行第 10 列开始显示

    WriteData('.'); //将小数点的字符常量写入 LCD
    delaynms(50); //延时 1ms 给硬件一点反应时间
}

/*****
函数功能：显示温度的单位(Cent)
*****/
void display_cent(void)
{
    unsigned char i;
    WriteAddress(0x4c); //写显示地址，将在第 2 行第 13 列
    开始显示
    i = 0; //从第一个字符开始显示
    while(Cent[i] != '\0') //只要没有写到结束标志，就继续写
    {
        WriteData(Cent[i]); //将字符常量写入 LCD
        i++; //指向下一个字符
        delaynms(50); //延时 1ms 给硬件一点反应时间
    }
}

```

```

/*****
函数功能：显示温度的整数部分
入口参数：x
*****/
void display_temp1(unsigned char x)
{
    unsigned char j,k,l;    //j,k,l 分别储存温度的百位、十位和个位
    j=x/100;                //取百位
    k=(x%100)/10;         //取十位
    l=x%10;                //取个位
    WriteAddress(0x46);    //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[j]);   //将百位数字的字符常量写入 LCD
    WriteData(digit[k]);   //将十位数字的字符常量写入 LCD
    WriteData(digit[l]);   //将个位数字的字符常量写入 LCD
    delaynms(50);         //延时 1ms 给硬件一点反应时间
}
/*****
函数功能：显示温度的小数数部分
入口参数：x
*****/
void display_temp2(unsigned char x)
{
    WriteAddress(0x4a);    //写显示地址,将在第 2 行第 11 列开始显示
    WriteData(digit[x]);   //将小数部分的第一位数字字符常量写入 LCD
    delaynms(50);         //延时 1ms 给硬件一点反应时间
}
/*****
函数功能：做好读温度的准备
*****/
void ReadyReadTemp(void)
{
    Init_DS18B20();        //将 DS18B20 初始化
    WriteOneChar(0xCC);    // 跳过读序号列号的操作
    WriteOneChar(0x44);    // 启动温度转换
    for(time=0;time<100;time++)
        ; //温度转换需要一点时间
    Init_DS18B20();        //将 DS18B20 初始化
    WriteOneChar(0xCC);    //跳过读序号列号的操作
    WriteOneChar(0xBE);    //读取温度寄存器,前两个分别是温度的低位和高
位
}
/*****

```

函数功能：主函数

```

*****/

void main(void)
{
    unsigned char TL;    //储存暂存器的温度低位
    unsigned char TH;    //储存暂存器的温度高位
    unsigned char TN;    //储存温度的整数部分
    unsigned char TD;    //储存温度的小数部分
    LcdInitiate();      //将液晶初始化
    delaynms(5);        //延时 5ms 给硬件一点反应时间
    if(Init_DS18B20()==1)
        display_error();
        display_explain();
    display_symbol();   //显示温度说明
    display_dot();      //显示温度的小数点
    display_cent();     //显示温度的单位
    while(1)            //不断检测并显示温度
    {
        ReadyReadTemp(); //读温度准备
        TL=ReadOneChar(); //先读的是温度值低位
        TH=ReadOneChar(); //接着读的是温度值高位
        TN=TH*16+TL/16;   //实际温度值=(TH*256+TL)/16,即:
        TH*16+TL/16
        //这样得出的是温度的整数部分,小数部分被丢
        弃了
        TD=(TL%16)*10/16; //计算温度的小数部分,将余数乘以 10 再除以 16
        取整,
        //这样得到的是温度小数部分的第一位数字(保
        留 1 位小数)
        display_temp1(TN); //显示温度的整数部分
        display_temp2(TD); //显示温度的小数部分
        delaynms(10);
    }
}

```

实例 91：将数据"0xaa"写入 X5045 再读出送 P1 口显示

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit SCK=P3^4;      //将 SCK 位定义为 P3.4 引脚

```

```

sbit SI=P3^5;           //将 SI 位定义为 P3.5 引脚
sbit SO=P3^6;           //将 SO 位定义为 P3.6 引脚
sbit CS=P3^7;           //将 SCK 位定义为 P3.7 引脚
#define WREN 0x06       //写使能锁存器允许
#define WRDI 0x04       //写使能锁存器禁止
#define WRSR 0x01       //写状态寄存器
#define READ 0x03       //读出
#define WRITE 0x02      //写入
/*****
函数功能：延时 1ms
 $(3j+2)*i=(3 \times 33+2) \times 10=1010$ (微秒)，可以认为是 1 毫秒
*****/

void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/

void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：从 X5045 的当前地址读出数据
出口参数：x
*****/

unsigned char ReadCurrent(void)
{
    unsigned char i;
    unsigned char x=0x00; //储存从 X5045 中读出的数据
    SCK=1;                //将 SCK 置于已知的高电平状态
    for(i = 0; i < 8; i++)
    {
        SCK=1;           //拉高 SCK
        SCK=0;           //在 SCK 的下降沿输出数据
        x<<=1; //将 x 中的各二进位向左移一位，因为首先读出的是字节的最高
    }
}

```

```

        x|=(unsigned char)SO; //将 SO 上的数据通过按位“或”运算存入 x
    }
    return(x); //将读取的数据返回
}
/*****
函数功能：写数据到 X5045 的当前地址
入口参数：dat
*****/
void WriteCurrent(unsigned char dat)
{
    unsigned char i;
    SCK=0; //将 SCK 置于已知的低电平状态
    for(i = 0; i < 8; i++) // 循环移入 8 个位
    {
        SI=(bit)(dat&0x80); //通过按位“与”运算将最高位数据送到 S
        //因为传送时高位在前，低位在后

        SCK=0;
        SCK=1; //在 SCK 上升沿写入数据
        dat<<=1; //将 y 中的各二进位向左移一位，因为首先写入的是字节的最高
位
    }
}

/*****
函数功能：写状态寄存器，可以设置看门狗的溢出时间及数据保护
入口参数：rs; //储存寄存器状态值
*****/
void WriteSR(unsigned char rs)
{
    CS=0; //拉低 CS，选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS=1; //拉高 CS
    CS=0; //重新拉低 CS，否则下面的写寄存器状态指令将
被丢弃
    WriteCurrent(WRSR); //写状态寄存器
    WriteCurrent(rs); //写入新设定的寄存器状态值
    CS=1; //拉高 CS
}

/*****
函数功能：写数据到 X5045 的指定地址
入口参数：addr
*****/
void WriteSet(unsigned char dat,unsigned char addr)

```

```

{
    SCK=0;           //将 SCK 置于已知状态
    CS=0;           //拉低 CS, 选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS=1;           //拉高 CS
    CS=0;           //重新拉低 CS, 否则下面的写入指令将被丢弃
    WriteCurrent(WRITE); //写入指令
    WriteCurrent(addr); //写入指定地址
    WriteCurrent(dat); //写入数据
    CS=1;           //拉高 CS
    SCK=0;           //将 SCK 置于已知状态
}

```

```

/*****

```

函数功能：从 X5045 的指定地址读出数据

入口参数：addr

出口参数：dat

```

*****/

```

```

unsigned char ReadSet(unsigned char addr)

```

```

{
    unsigned char dat;
    SCK=0;           //将 SCK 置于已知状态
    CS=0;           //拉低 CS, 选中 X5045
    WriteCurrent(READ); //开始读
    WriteCurrent(addr); //写入指定地址
    dat=ReadCurrent(); //读出数据
    CS=1;           //拉高 CS
    SCK=0;           //将 SCK 置于已知状态
    return dat;     //返回读出的数据
}

```

```

/*****

```

函数功能：看门狗复位程序

```

*****/

```

```

void WatchDog(void)

```

```

{
    CS=1; //拉高 CS
    CS=0; //CS 引脚的一个下降沿复位看门狗定时器
    CS=1; //拉高 CS
}

```

```

/*****

```

函数功能：主程序

```

*****/

```

```

void main(void)

```



```

{
  WriteSR(0x12);          //写状态寄存器（设定看门狗溢出时间为 600ms，写
  不保护）
  delaynms(10);          //X5045 的写入周期约为 10ms
  while(1)
  {
    WriteSet(0xaa,0x10); //将数据“0xaa”写入指定地址“0x10”
    delaynms(10);        //X5045 的写入周期约为 10ms
    P1=ReadSet(0x10);    //将数据读出送 P1 口显示
    WatchDog();          //复位看门狗
  }
}

```

实例 92：将流水灯控制码写入 X5045 并读出送 P1 口显示

```

#include<reg51.h>        //包含单片机寄存器的头文件
#include<intrins.h>     //包含_nop_()函数定义的头文件
sbit SCK=P3^4;          //将 SCK 位定义为 P3.4 引脚
sbit SI=P3^5;           //将 SI 位定义为 P3.5 引脚
sbit SO=P3^6;           //将 SO 位定义为 P3.6 引脚
sbit CS=P3^7;           //将 SCK 位定义为 P3.7 引脚
#define WREN 0x06       //写使能锁存器允许
#define WRDI 0x04       //写使能锁存器禁止
#define WRSR 0x01       //写状态寄存器
#define READ 0x03       //读出
#define WRITE 0x02      //写入
unsigned char lamp[ ]={0xFF,0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F,
                      0x7F,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,0xFE,0xFF,
                      0xFF,0xFE,0xFC,0xFB,0xF0,0xE0,0xC0,0x80,0x00,
                      0xE7,0xDB,0xBD,0x7E,0xFF,
                      0xFF,0x3C,0x18,0x00,
                      0x81,0xC3,0xE7,0xFF,0xFF,0x7E,0xBD,0xDB,0xE7,
                      0xBD,0xDB,0x7E,0xFF,0xAA}; //流水灯控制码
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()
{
  unsigned char i,j;

```

```

        for(i=0;i<10;i++)
            for(j=0;j<33;j++)
                ;
    }
    /*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
    /*****
函数功能：从 X5045 的当前地址读出数据
出口参数：x
*****/
unsigned char ReadCurrent(void)
{
    unsigned char i;
    unsigned char x=0x00;    //储存从 X5045 中读出的数据
    SCK=1;                  //将 SCK 置于已知的高电平状态
    for(i = 0; i < 8; i++)
    {
        SCK=1;              //拉高 SCK
        SCK=0;              //在 SCK 的下降沿输出数据
        x<<=1; //将 x 中的各二进位向左移一位，因为首先读出的是字节的最高
        位数据
        x|=(unsigned char)SO; //将 SO 上的数据通过按位“或”运算存入 x
    }
    return(x); //将读取的数据返回
}
    /*****
函数功能：写数据到 X5045 的当前地址
入口参数：dat
*****/
void WriteCurrent(unsigned char dat)
{
    unsigned char i;
    SCK=0;                  //将 SCK 置于已知的低电平状态
    for(i = 0; i < 8; i++) // 循环移入 8 个位
    {
        SI=(bit)(dat&0x80); //通过按位“与”运算将最高位数据送到 S

```

```

//因为传送时高位在前，低位在后
    SCK=0;
    SCK=1;           //在 SCK 上升沿写入数据
    dat<<=1;       //将 y 中的各二进位向左移一位，因为首先写入的是字节的最高
位
}
}

```

```

/*****

```

函数功能：写状态寄存器，可以设置看门狗的溢出时间及数据保护
入口参数：rs; //储存寄存器状态值

```

*****/

```

```

void WriteSR(unsigned char rs)

```

```

{
    CS=0;           //拉低 CS，选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS=1;           //拉高 CS
    CS=0;           //重新拉低 CS，否则下面的写寄存器状态指令将
被丢弃
    WriteCurrent(WRSR); //写状态寄存器
    WriteCurrent(rs);   //写入新设定的寄存器状态值
    CS=1;           //拉高 CS
}

```

```

/*****

```

函数功能：写数据到 X5045 的指定地址

入口参数：addr

```

*****/

```

```

void WriteSet(unsigned char dat,unsigned char addr)

```

```

{
    SCK=0;           //将 SCK 置于已知状态
    CS=0;           //拉低 CS，选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS=1;           //拉高 CS
    CS=0;           //重新拉低 CS，否则下面的写入指令将被丢弃
    WriteCurrent(WRITE); //写入指令
    WriteCurrent(addr); //写入指定地址
    WriteCurrent(dat);   //写入数据
    CS=1;           //拉高 CS
    SCK=0;           //将 SCK 置于已知状态
}

```

```

/*****

```

函数功能：从 X5045 的指定地址读出数据

入口参数: addr

出口参数: dat

*****/

```
unsigned char ReadSet(unsigned char addr)
```

```
{
    unsigned char dat;
    SCK=0;           //将 SCK 置于已知状态
    CS=0;           //拉低 CS, 选中 X5045
    WriteCurrent(READ); //开始读
    WriteCurrent(addr); //写入指定地址
    dat=ReadCurrent(); //读出数据
    CS=1;           //拉高 CS
    SCK=0;           //将 SCK 置于已知状态
    return dat;     //返回读出的数据
}
```

函数功能: 看门狗复位程序

*****/

```
void WatchDog(void)
```

```
{
    CS=1; //拉高 CS
    CS=0; //CS 引脚的一个下降沿复位看门狗定时器
    CS=1; //拉高 CS
}
```

函数功能: 主程序

*****/

```
void main(void)
```

```
{
    unsigned char i;
    WriteSR(0x12); //写状态寄存器 (设定看门狗溢出时间为 600ms, 写不保护)
    delaynms(10); //X5045 的写入周期约为 10ms
```

```
    for(i=0;i<50;i++)
    {
        WriteSet(lamp[i],0x00+i); //将数据 "0xaa" 写入指定地址 "0x10"
        delaynms(10); //X5045 的写入周期约为 10ms
    }
    while(1)
    {
        for(i=0;i<50;i++)
```

```

    {
        P1=ReadSet(0x00+i);    //将数据读出送 P1 口显示
        delaynms(100);
        WatchDog();
    }
}
}

```

实例 93：对 SPI 总线上挂接多个 X5045 的读写操作

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit SCK=P3^4;      //将 SCK 位定义为 P3.4 引脚
sbit SI=P3^5;       //将 SI 位定义为 P3.5 引脚
sbit SO=P3^6;       //将 SO 位定义为 P3.6 引脚
sbit CS1=P3^7;      //将 CS 定义为 P3.7 引脚
sbit CS2=P3^3;      //将 CS1 位定义为 P3.7 引脚
#define WREN 0x06    //写使能锁存器允许
#define WRDI 0x04    //写使能锁存器禁止
#define READ 0x03    //读出
#define WRITE 0x02   //写入
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****

```

函数功能：从 X5045 的当前地址读出数据

出口参数：x

*****/

```
unsigned char ReadCurrent(void)
```

```
{
    unsigned char i;
    unsigned char x=0x00;    //储存从 X5045 中读出的数据
    SCK=1;                  //将 SCK 置于已知的高电平状态
    for(i = 0; i < 8; i++)
    {
        SCK=1;              //拉高 SCK
        SCK=0;              //在 SCK 的下降沿输出数据
        x<<=1; //将 x 中的各二进位向左移一位，因为首先读出的是字节的最
```

高位数据

```
x|=(unsigned char)SO; //将 SO 上的数据通过按位“或”运算存
```

入 x

```
    }
    return(x); //将读取的数据返回
}
/*****
```

函数功能：写数据到 X5045 的当前地址

入口参数：dat

*****/

```
void WriteCurrent(unsigned char dat)
```

```
{
    unsigned char i;
    SCK=0;    //将 SCK 置于已知的低电平状态
    for(i = 0; i < 8; i++) // 循环移入 8 个位
    {
        SI=(bit)(dat&0x80); //通过按位“与”运算将最高位数据送到 S
        //因为传送时高位在前，低位在后

        SCK=0;
        SCK=1;    //在 SCK 上升沿写入数据
        dat<<=1; //将 y 中的各二进位向左移一位，因为首先写入的是字节的最高
```

位

```
    }
}
/*****
```

函数功能：写数据到第一个 X5045 的指定地址

入口参数：addr

*****/

```
void WriteSet1(unsigned char dat,unsigned char addr)
```

```

{
    CS2=1;           //使第二个 X5045 的片选无效
    SCK=0;          //将 SCK 置于已知状态
    CS1=0;          //拉低 CS, 选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS1=1;          //拉高 CS
    CS1=0;          //重新拉低 CS, 否则下面的写入指令将被丢弃
    WriteCurrent(WRITE); //写入指令
    WriteCurrent(addr); //写入指定地址
    WriteCurrent(dat);  //写入数据
    CS1=1;          //拉高 CS
    SCK=0;          //将 SCK 置于已知状态
}

```

```

/*****

```

函数功能: 写数据到第二个 X5045 的指定地址

入口参数: addr

```

*****/

```

```

void WriteSet2(unsigned char dat,unsigned char addr)

```

```

{
    CS1=1;           //使第一个 X5045 的片选无效
    SCK=0;          //将 SCK 置于已知状态
    CS2=0;          //拉低 CS, 选中 X5045
    WriteCurrent(WREN); //写使能锁存器允许
    CS2=1;          //拉高 CS
    CS2=0;          //重新拉低 CS, 否则下面的写入指令将被丢弃
    WriteCurrent(WRITE); //写入指令
    WriteCurrent(addr); //写入指定地址
    WriteCurrent(dat);  //写入数据
    CS2=1;          //拉高 CS
    SCK=0;          //将 SCK 置于已知状态
}

```

```

/*****

```

函数功能: 从第一个 X5045 的指定地址读出数据

入口参数: addr

出口参数: dat

```

*****/

```

```

unsigned char ReadSet1(unsigned char addr)

```

```

{
    unsigned char dat;
    CS2=1;           //使第二个 X5045 的片选无效
    SCK=0;          //将 SCK 置于已知状态
    CS1=0;          //拉低 CS, 选中 X5045
    WriteCurrent(READ); //开始读

```

```

WriteCurrent(addr); //写入指定地址
dat=ReadCurrent(); //读出数据
CS1=1;             //拉高 CS
SCK=0;            //将 SCK 置于已知状态
return dat;       //返回读出的数据
}
/*****
函数功能: 从第二个 X5045 的指定地址读出数据
入口参数: addr
出口参数: dat
*****/
unsigned char ReadSet2(unsigned char addr)
{
    unsigned char dat;
    CS1=1;           //使第一个 X5045 的片选无效
    SCK=0;          //将 SCK 置于已知状态
    CS2=0;          //拉低 CS, 选中 X5045
    WriteCurrent(READ); //开始读
    WriteCurrent(addr); //写入指定地址
    dat=ReadCurrent(); //读出数据
    CS2=1;          //拉高 CS
    SCK=0;          //将 SCK 置于已知状态
    return dat;     //返回读出的数据
}
/*****
函数功能: 看门狗复位程序
*****/
void WatchDog1(void)
{
    CS1=1; //拉高 CS
    CS1=0; //CS 引脚的一个下降沿复位看门狗定时器
    CS1=1; //拉高 CS
}
/*****
函数功能: 看门狗复位程序
*****/
void WatchDog2(void)
{
    CS2=1; //拉高 CS
    CS2=0; //CS 引脚的一个下降沿复位看门狗定时器
    CS2=1; //拉高 CS
}
/*****

```


函数功能：主程序

```

*****/
void main(void)
{
    unsigned char x;
    while(1)
    {
        WriteSet1(0xf0,0x10); //将数据“0xaa”写入第一个 X5045 的指定地
                               址“0x10”
        delaynms(10);         //X5045 的写入周期为约 10ms
        x=ReadSet1(0x10);     //将数据从第一个 X5045 中的指定地址读出来
        WriteSet2(x,0x20);   //将数据 x 写入第二个 X5045 的指定地址“0x20”
        delaynms(10);       //X5045 的写入周期为约 10ms
        P1=ReadSet2(0x20);   //将数据从第二个 X5045 中的指定地址读出来

        , 送 P1 口显示
        delaynms(100);       //延时 100ms
        WatchDog1();        //复位第一个 X5045 的看门狗
        WatchDog2();        //复位第二个 X5045 的看门狗
    }
}

```

实例 94：基于 ADC0832 的数字电压表

```

#include<reg51.h> //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
sbit CS=P3^4; //将 CS 位定义为 P3.4 引脚
sbit CLK=P1^0; //将 CLK 位定义为 P1.0 引脚
sbit DIO=P1^1; //将 DIO 位定义为 P1.1 引脚
////////////////////////////////////
unsigned char code digit[10]={"0123456789"}; //定义字符数组显示数字
unsigned char code Str[]{"Volt="}; //说明显示的是电压
/*****
以下是对液晶模块的操作程序
*****/
sbit RS=P2^0; //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1; //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2; //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7; //忙碌标志位，将 BF 位定义为 P0.7 引脚
/*****

```

函数功能：延时 1ms

$(3j+2)*i=(3\times 33+2)\times 10=1010$ (微秒)，可以认为是 1 毫秒

```

*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/

void delaynms(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌；result=0，不忙
*****/

bit BusyTest(void)
{
    bit result;
    RS=0;        //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;        //E=1，才允许读写
    _nop_();    //空操作
    _nop_();
    _nop_();
    _nop_();    //空操作四个机器周期，给硬件反应时间
    result=BF;  //将忙碌标志电平赋给 result
    E=0;        //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/

void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
}

```

```

    RS=0;                //根据规定，RS 和 R/W 同时为低电平时，可以
写入指令
    RW=0;
    E=0;                //E 置低电平(根据表 8-6，写指令时，E 为高脉
冲，
                        // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"

    _nop_();
    _nop_();            //空操作两个机器周期，给硬件反应时间
    P0=dictate;        //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();            //空操作四个机器周期，给硬件反应时间
    E=1;                //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();            //空操作四个机器周期，给硬件反应时间
    E=0;                //当 E 由高电平跳变成低电平时，液晶模块开始

```

执行命令

```
}

```

```
/******

```

函数功能：指定字符显示的实际地址

入口参数：x

```
*****/

```

```

void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}

```

```
/******

```

函数功能：将数据(字符的标准 ASCII 码)写入液晶模块

入口参数：y(为字符常量)

```
*****/

```

```

void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;                //RS 为高电平，RW 为低电平时，可以写入数据
    RW=0;
    E=0;                //E 置低电平(根据表 8-6，写指令时，E 为高脉冲，
                        // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"
    P0=y;                //将数据送入 P0 口，即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
}

```

```

    _nop_();          //空操作四个机器周期，给硬件反应时间
    E=1;             //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();          //空操作四个机器周期，给硬件反应时间
    E=0;             //当 E 由高电平跳变成低电平时，液晶模块开始执行命令
}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delaynms(15);    //延时 15ms，首次写指令时应给 LCD 一段
较长的反应时间
    WriteInstruction(0x38); //显示模式设置：16×2 显示，5×7 点阵，8
位数据接口
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38);
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x38); //连续三次，确保初始化成功
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x0c); //显示模式设置：显示开，无光标，光标不闪烁
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x06); //显示模式设置：光标右移，字符不移
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
    WriteInstruction(0x01); //清屏幕指令，将以前的显示内容清除
    delaynms(5);      //延时 5ms ，给硬件一点反应时间
}
/*****
以下是电压显示的说明
*****/
/*****
函数功能：显示电压符号
*****/
void display_volt(void)
{
    unsigned char i;
    WriteAddress(0x03); //写显示地址，将在第 2 行第 1 列开始显示
    i = 0;              //从第一个字符开始显示
    while(Str[i] != '\0') //只要没有写到结束标志，就继续写
    {
        WriteData(Str[i]); //将字符常量写入 LCD
    }
}

```

```

        i++;                //指向下一个字符
    }
}

/*****
函数功能：显示电压的小数点
*****/
void display_dot(void)
{
    WriteAddress(0x09);    //写显示地址，将在第 1 行第 10 列开始显示

    WriteData('.');        //将小数点的字符常量写入 LCD
}

/*****
函数功能：显示电压的单位(V)
*****/
void display_V(void)
{
    WriteAddress(0x0c);    //写显示地址，将在第 2 行第 13 列开始显示
    WriteData('V');        //将字符常量写入 LCD
}

/*****
函数功能：显示电压的整数部分
入口参数：x
*****/
void display1(unsigned char x)
{
    WriteAddress(0x08);    //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[x]);    //将百位数字的字符常量写入 LCD
}

/*****
函数功能：显示电压的小数数部分
入口参数：x
*****/
void display2(unsigned char x)
{
    unsigned char i,j;
    i=x/10;                //取十位（小数点后第一位）
    j=x%10;                //取个位（小数点后第二位）
    WriteAddress(0x0a);    //写显示地址,将在第 1 行第 11 列开始显示
    WriteData(digit[i]);    //将小数部分的第一位数字字符常量写入 LCD
}

```

```

        WriteData(digit[j]);    //将小数部分的第一位数字字符常量写入 LCD
    }
}
/*****
函数功能：将模拟信号转换成数字信号
*****/
unsigned char  A_D()
{
    unsigned char i,dat;
    CS=1;    //一个转换周期开始
    CLK=0;   //为第一个脉冲作准备
    CS=0;    //CS 置 0，片选有效

    DIO=1;   //DIO 置 1，规定的起始信号
    CLK=1;   //第一个脉冲
    CLK=0;   //第一个脉冲的下降沿，此前 DIO 必须是高电平
    DIO=1;   //DIO 置 1，通道选择信号
    CLK=1;   //第二个脉冲，第 2、3 个脉冲下沉之前，DI 必须跟别输入两位数据用于选择通道，这里选通道 CH0
    CLK=0;   //第二个脉冲下降沿
    DIO=0;   //DI 置 0，选择通道 0
    CLK=1;   //第三个脉冲
    CLK=0;   //第三个脉冲下降沿
    DIO=1;   //第三个脉冲下沉之后，输入端 DIO 失去作用，应置 1
    CLK=1;   //第四个脉冲
    for(i=0;i<8;i++) //高位在前
    {
        CLK=1;       //第四个脉冲
        CLK=0;
        dat<<=1;     //将下面储存的低位数据向右移
        dat|=(unsigned char)DIO; //将输出数据 DIO 通过或运算储存在 dat
    }
    CS=1;           //片选无效
    return dat;    //将读书的数据返回
}
/*****
函数功能：主函数
*****/
main(void)
{
    unsigned int AD_val;    //储存 A/D 转换后的值
    unsigned char Int,Dec;  //分别储存转换后的整数部分与小数部分
    LcdInitiate();        //将液晶初始化
    delaynms(5);          //延时 5ms 给硬件一点反应时间
}

```

```

display_volt();           //显示温度说明
display_dot();           //显示温度的小数点
display_V();             //显示温度的单位
while(1)
{
    AD_val= A_D();        //进行 A/D 转换
    Int=(AD_val)/51;     //计算整数部分
    Dec=(AD_val%51)*100/51; //计算小数部分
    display1(Int);        //显示整数部分
    display2(Dec);        //显示小数部分
    delaynms(250);        //延时 250ms
}
}

```

实例 95：用 DAC0832 产生锯齿波电压

```

#include<reg51.h>         //包含单片机寄存器的头文件
#include<absacc.h>        //包含对片外存储器地址进行操作的头文件
sbit CS=P2^7;            //将 CS 位定义为 P2.7 引脚
sbit WR12=P3^6;          //将 WR12 位定义为 P3.6 引脚
void main(void)
{
    unsigned char i;
    CS=0;                 //输出低电平以选中 DAC0832
    WR12=0;               //输出低电平以选中 DAC0832
    while(1)
    {
        for(i=0;i<255;i++)
            XBYTE[0x7fff]=i; //将数据 i 送入片外地址 07FFFH ，实际上就是通过 P0 口将数据送入 DAC0832
    }
}

```

实例 96：用 P1 口显示红外遥控器的按键值

```

#include<reg51.h>         //包含单片机寄存器的头文件
sbit IR=P3^2;            //将 IR 位定义为 P3.2 引脚
unsigned char a[4];       //储存用户码、用户反码与键数据码、键数据反码
unsigned int LowTime,HighTime; //储存高、低电平的宽度
/*****
函数功能：对 4 个字节的用户码和键数据码进行解码

```

说明：解码正确，返回 1，否则返回 0

出口参数：dat

*****/

```

bit DeCode(void)
{
    unsigned char i,j;
    unsigned char temp;    //储存解码出的数据
    for(i=0;i<4;i++)      //连续读取 4 个用户码和键数据码
    {
        for(j=0;j<8;j++) //每个码有 8 位数字
        {
            temp=temp>>1; //temp 中的各数据位右移一位，因为先读出的是高位数据
            TH0=0;        //定时器清 0
            TL0=0;        //定时器清 0
            TR0=1;        //开启定时器 T0
            while(IR==0) //如果是低电平就等待
                ;        //低电平计时
            TR0=0;        //关闭定时器 T0
            LowTime=TH0*256+TL0; //保存低电平宽度
            TH0=0;        //定时器清 0
            TL0=0;        //定时器清 0
            TR0=1;        //开启定时器 T0
            while(IR==1) //如果是高电平就等待
                ;
            TR0=0;        //关闭定时器 T0
            HighTime=TH0*256+TL0; //保存高电平宽度
            if((LowTime<370)||((LowTime>640)))
                return 0; //如果低电平长度不在合理范围，则认为出错，停止解码
            if((HighTime>420)&&(HighTime<620)) //如果高电平时间在 560 微秒左右，即计数 560 / 1.085=516 次
                temp=temp&0x7f; //((520-100=420, 520+100=620)，则该位是 0
            if((HighTime>1300)&&(HighTime<1800)) //如果高电平时间在 1680 微秒左右，即计数 1680 / 1.085=1548 次
                temp=temp|0x80; //((1550-250=1300,1550+250=1800)，则该位是 1
        }
        a[i]=temp; //将解码出的字节值储存在 a[i]
    }
}
if(a[2]==~a[3]) //验证键数据码和其反码是否相等，一般情况下不必验证用户码

```



```

    return 1;    //解码正确，返回 1
}
/*****
函数功能：执行遥控功能
*****/
void Function(void)
{
    P1=a[2];    //将按键数据码送 P1 口显示
}
/*****
函数功能：主函数
*****/
void main()
{
    EA=1;        //开启总中断
    EX0=1;       //开外中断 0
    ET0=1;       //定时器 T0 中断允许
    IT0=1;       //外中断的下降沿触发
    TMOD=0x01;   //使用定时器 T0 的模式 1
    TR0=0;       //定时器 T0 关闭
    while(1)     //等待红外信号产生的中断
        ;
}
/*****
函数功能：红外线触发的外中断处理函数
*****/
void Int0(void) interrupt 0 using 0
{
    EX0=0;       //关闭外中断 0，不再接收二次红外信号的中断，只解码当前红外信号
    TH0=0;       //定时器 T0 的高 8 位清 0
    TL0=0;       //定时器 T0 的低 8 位清 0
    TR0=1;       //开启定时器 T0
    while(IR==0) //如果是低电平就等待，给引导码低电平计时
        ;
    TR0=0;       //关闭定时器 T0
    LowTime=TH0*256+TL0; //保存低电平时间
    TH0=0;       //定时器 T0 的高 8 位清 0
    TL0=0;       //定时器 T0 的低 8 位清 0
    TR0=1;       //开启定时器 T0
    while(IR==1) //如果是高电平就等待，给引导码高电平计时
        ;
    TR0=0;       //关闭定时器 T0
    HighTime=TH0*256+TL0; //保存引导码的高电平长度
}

```

```

if((LowTime>7800)&&(LowTime<8800)&&(HighTime>3600)&&(HighTime<4700))
{
    //如果是引导码,就开始解码,否则放弃,引导码的低电平计时
    //次数=9000us/1.085=8294, 判断区间:8300-500=7800, 8300
+500=8800.
    if(DeCode()==1)
        Function(); //如果满足条件, 执行遥控功能
}
EX0=1; //开启外中断 EX0
}

```

实例 97: 用红外遥控器控制继电器

```

#include<reg51.h> //包含单片机寄存器的头文件
sbit IR=P3^2; //将 IR 位定义为 P3.2 引脚
unsigned char a[4]; //储存用户码、用户反码与键数据码、键数据反码
unsigned int LowTime,HighTime; //储存高、低电平的宽度
sbit Relay=P1^3; //将 Relay 位定义为 P1.3 引脚
/*****
函数功能: 对 4 个字节的用户码和键数据码进行解码
说明: 解码正确, 返回 1, 否则返回 0
出口参数: dat
*****/
bit DeCode(void)
{
    unsigned char i,j;
    unsigned char temp; //储存解码出的数据
    for(i=0;i<4;i++) //连续读取 4 个用户码和键数据码
    {
        for(j=0;j<8;j++) //每个码有 8 位数字
        {
            temp=temp>>1; //temp 中的各数据位右移一位, 因为先读出的是高位数据
            TH0=0; //定时器清 0
            TL0=0; //定时器清 0
            TR0=1; //开启定时器 T0
            while(IR==0) //如果是低电平就等待
                ; //低电平计时
            TR0=0; //关闭定时器 T0
            LowTime=TH0*256+TL0; //保存低电平宽度

```

```

        TH0=0;          //定时器清 0
        TL0=0;          //定时器清 0
        TR0=1;          //开启定时器 T0
        while(IR==1)   //如果是高电平就等待
            ;
        TR0=0;          //关闭定时器 T0
        HighTime=TH0*256+TL0; //保存高电平宽度
        if((LowTime<370)||((LowTime>640)))
            return 0;    //如果低电平长度不在合理范围,则认
为出错, 停止解码
        if((HighTime>420)&&(HighTime<620)) //如果高电平时间在
560 微秒左右, 即计数 560 / 1.085=516 次
            temp=temp&0x7f;    //(520-100=420,
520+100=620), 则该位是 0
        if((HighTime>1300)&&(HighTime<1800)) //如果高电平时间在
1680 微秒左右, 即计数 1680 / 1.085=1548 次
            temp=temp|0x80;
        //(1550-250=1300,1550+250=1800),则该位是 1
    }
    a[i]=temp; //将解码出的字节值储存在 a[i]

}
if(a[2]=~a[3]) //验证键数据码和其反码是否相等,一般情况下不必验证用户码
    return 1;    //解码正确, 返回 1
}
/*****
函数功能: 执行遥控功能
*****/
void Function(void)
{
    Relay=!Relay; //对 P1.3 引脚取反, 控制继电器的吸合、释放
}
/*****
函数功能: 主函数
*****/
void main()
{
    EA=1;          //开启总中断
    EX0=1;         //开外中断 0
    ET0=1;         //定时器 T0 中断允许
    IT0=1;         //外中断的下降沿触发
    TMOD=0x01;     //使用定时器 T0 的模式 1
    TR0=0;         //定时器 T0 关闭

```

```

while(1)    //等待红外信号产生的中断
;
}
/*****
函数功能：红外线触发的外中断处理函数
*****/
void Int0(void) interrupt 0 using 0
{
    EX0=0;    //关闭外中断 0，不再接收二次红外信号的中断，只解码当前红外信号
    TH0=0;    //定时器 T0 的高 8 位清 0
    TL0=0;    //定时器 T0 的低 8 位清 0
    TR0=1;    //开启定时器 T0
    while(IR==0)    //如果是低电平就等待，给引导码低电平计时
        ;
    TR0=0;    //关闭定时器 T0
    LowTime=TH0*256+TL0; //保存低电平时间
    TH0=0;    //定时器 T0 的高 8 位清 0
    TL0=0;    //定时器 T0 的低 8 位清 0
    TR0=1;    //开启定时器 T0
    while(IR==1) //如果是高电平就等待，给引导码高电平计时
        ;
    TR0=0;    //关闭定时器 T0
    HighTime=TH0*256+TL0; //保存引导码的高电平长度

    if((LowTime>7800)&&(LowTime<8800)&&(HighTime>3600)&&(HighTime<4700))
    {
        //如果是引导码,就开始解码,否则放弃,引导码的低电平计时
        //次数=9000us/1.085=8294, 判断区间:8300-500=7800, 8300+500=8800.
        if(DeCode()==1)
            Function(); //如果满足条件，执行遥控功能
    }
    EX0=1;    //开启外中断 EX0
}

```

实例 98：基于 DS1302 的日历时钟

```

#include<reg51.h>    //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件
/*****
以下是 DS1302 芯片的操作程序

```

```

*****/
unsigned char code digit[10]={"0123456789"}; //定义字符数组显示数字
sbit DATA=P1^1; //位定义 1302 芯片的接口, 数据输出端定义在 P1.1 引脚
sbit RST=P1^2; //位定义 1302 芯片的接口, 复位端口定义在 P1.1 引脚
sbit SCLK=P1^0; //位定义 1302 芯片的接口, 时钟输出端口定义在 P1.1 引脚
/*****
函数功能: 延时若干微秒
入口参数: n
*****/
void delaynus(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        ;
}
/*****
函数功能: 向 1302 写一个字节数据
入口参数: x
*****/
void Write1302(unsigned char dat)
{
    unsigned char i;
    SCLK=0; //拉低 SCLK, 为脉冲上升沿写入数据做好准备
    delaynus(2); //稍微等待, 使硬件做好准备
    for(i=0;i<8;i++) //连续写 8 个二进制位数据
    {
        DATA=dat&0x01; //取出 dat 的第 0 位数据写入 1302
        delaynus(2); //稍微等待, 使硬件做好准备
        SCLK=1; //上升沿写入数据
        delaynus(2); //稍微等待, 使硬件做好准备
        SCLK=0; //重新拉低 SCLK, 形成脉冲
        dat>>=1; //将 dat 的各数据位右移 1 位, 准备写入下一个数
据位
    }
}
/*****
函数功能: 根据命令字, 向 1302 写一个字节数据
入口参数: Cmd, 储存命令字; dat, 储存待写的数
据
*****/
void WriteSet1302(unsigned char Cmd,unsigned char dat)
{
    RST=0; //禁止数据传递
    SCLK=0; //确保写数居前 SCLK 被拉低

```

```

    RST=1;           //启动数据传输
    delaynus(2);    //稍微等待，使硬件做好准备
    Write1302(Cmd); //写入命令字
    Write1302(dat); //写数据
    SCLK=1;         //将时钟电平置于已知状态
    RST=0;         //禁止数据传递
}
/*****
函数功能：从 1302 读一个字节数据
入口参数：x
*****/
unsigned char Read1302(void)
{
    unsigned char i,dat;
    delaynus(2); //稍微等待，使硬件做好准备
    for(i=0;i<8;i++) //连续读 8 个二进制位数据
    {
        dat>>=1; //将 dat 的各数据位右移 1 位，因为先读出的是字节的
        最低位
        if(DATA==1) //如果读出的数据是 1
            dat|=0x80; //将 1 取出，写在 dat 的最高位
        SCLK=1; //将 SCLK 置于高电平，为下降沿读出
        delaynus(2); //稍微等待
        SCLK=0; //拉低 SCLK，形成脉冲下降沿
        delaynus(2); //稍微等待
    }
    return dat; //将读出的数据返回
}
/*****
函数功能：根据命令字，从 1302 读取一个字节数据
入口参数：Cmd
*****/
unsigned char ReadSet1302(unsigned char Cmd)
{
    unsigned char dat;
    RST=0; //拉低 RST
    SCLK=0; //确保写数居前 SCLK 被拉低
    RST=1; //启动数据传输
    Write1302(Cmd); //写入命令字
    dat=Read1302(); //读出数据
    SCLK=1; //将时钟电平置于已知状态
    RST=0; //禁止数据传递
    return dat; //将读出的数据返回
}

```

```

/*****
函数功能： 1302 进行初始化设置
*****/
void Init_DS1302(void)
{
    WriteSet1302(0x8E,0x00);           //根据写状态寄存器命令字，
    写入不保护指令
    WriteSet1302(0x80,((0/10)<<4|(0%10))); //根据写秒寄存器命令字，写入
    秒的初始值
    WriteSet1302(0x82,((0/10)<<4|(0%10))); //根据写分寄存器命令字，写入
    分的初始值
    WriteSet1302(0x84,((12/10)<<4|(12%10))); //根据写小时寄存器命令字，写
    入小时的初始值
    WriteSet1302(0x86,((16/10)<<4|(16%10))); //根据写日寄存器命令字，写入
    日的初始值
    WriteSet1302(0x88,((11/10)<<4|(11%10))); //根据写月寄存器命令字，写入
    月的初始值
    WriteSet1302(0x8c,((8/10)<<4|(8%10))); //根据写小时寄存器命令字，写
    入小时的初始值
}
/*****
以下是对液晶模块的操作程序
*****/
sbit RS=P2^0;           //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;           //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;           //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;          //忙碌标志位，将 BF 位定义为 P0.7 引脚
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/
void delaynms(unsigned char n)
{

```

```

    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}
/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌;result=0，不忙
*****/
bit BusyTest(void)
{
    bit result;
    RS=0;          //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;          //E=1，才允许读写
    _nop_();      //空操作
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期，给硬件反应时间
    result=BF;    //将忙碌标志电平赋给 result
    E=0;          //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                  //根据规定，RS 和 R/W 同时为低电平时，可以
    写入指令
    RW=0;
    E=0;                  //E 置低电平(根据表 8-6，写指令时，E 为高脉
    冲，
    // 就是让 E 从 0 到 1 发生正跳变,所以应先置"0"

    _nop_();
    _nop_();              //空操作两个机器周期，给硬件反应时间
    P0=dictate;          //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();              //空操作四个机器周期，给硬件反应时间
    E=1;                  //E 置高电平
    _nop_();

```



```

    _nop_();
    _nop_();
    _nop_();           //空操作四个机器周期，给硬件反应时间
    E=0;              //当 E 由高电平跳变成低电平时，液晶模块开始
执行命令
}
/*****
函数功能：指定字符显示的实际地址
入口参数：x
*****/
void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}
/*****
函数功能：将数据(字符的标准 ASCII 码)写入液晶模块
入口参数：y(为字符常量)
*****/
void WriteData(unsigned char y)
{
    while(BusyTest()==1);
    RS=1;           //RS 为高电平，RW 为低电平时，可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6，写指令时，E 为高脉冲，
// 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"
    P0=y;         //将数据送入 P0 口，即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();       //空操作四个机器周期，给硬件反应时间
    E=1;          //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();       //空操作四个机器周期，给硬件反应时间
    E=0;          //当 E 由高电平跳变成低电平时，液晶模块开始执行命令
}
/*****
函数功能：对 LCD 的显示模式进行初始化设置
*****/
void LcdInitiate(void)
{
    delaynms(15);   //延时 15ms，首次写指令时应给 LCD 一段
较长的反应时间

```

```

    WriteInstruction(0x38);    //显示模式设置: 16×2 显示, 5×7 点阵, 8
    位数据接口
    delaynms(5);             //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x38);
    delaynms(5);             //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x38);   //连续三次, 确保初始化成功
    delaynms(5);             //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x0c);   //显示模式设置: 显示开, 无光标, 光标不闪烁
    delaynms(5);             //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x06);   //显示模式设置: 光标右移, 字符不移
    delaynms(5);             //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x01);   //清屏幕指令, 将以前的显示内容清除
    delaynms(5);             //延时 5ms , 给硬件一点反应时间

```

```

}

```

```

/*****

```

以下是 1302 数据的显示程序

```

*****/

```

```

/*****

```

函数功能: 显示秒

入口参数: x

```

*****/

```

```

void DisplaySecond(unsigned char x)

```

```

{

```

```

    unsigned char i,j;      //j,k,l 分别储存温度的百位、十位和个位

```

```

    i=x/10;//取十位

```

```

    j=x%10;//取个位

```

```

    WriteAddress(0x49);    //写显示地址,将在第 2 行第 7 列开始显示

```

```

    WriteData(digit[i]);   //将百位数字的字符常量写入 LCD

```

```

    WriteData(digit[j]);   //将十位数字的字符常量写入 LCD

```

```

    delaynms(50);         //延时 1ms 给硬件一点反应时间

```

```

}

```

```

/*****

```

函数功能: 显示分钟

入口参数: x

```

*****/

```

```

void DisplayMinute(unsigned char x)

```

```

{

```

```

    unsigned char i,j;      //j,k,l 分别储存温度的百位、十位和个位

```

```

    i=x/10;//取十位

```

```

    j=x%10;//取个位

```

```

    WriteAddress(0x46);    //写显示地址,将在第 2 行第 7 列开始显示

```

```

    WriteData(digit[i]);   //将百位数字的字符常量写入 LCD

```

```

        WriteData(digit[j]);    //将十位数字的字符常量写入 LCD
        delaynms(50);          //延时 1ms 给硬件一点反应时间
    }
}
/*****
函数功能：显示小时
入口参数：x
*****/
void DisplayHour(unsigned char x)
{
    unsigned char i,j;        //j,k,l 分别储存温度的百位、十位和个位
    i=x/10;//取十位
    j=x%10;//取个位
    WriteAddress(0x43);      //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[i]);     //将百位数字的字符常量写入 LCD
    WriteData(digit[j]);     //将十位数字的字符常量写入 LCD
    delaynms(50);           //延时 1ms 给硬件一点反应时间
}
/*****
函数功能：显示日
入口参数：x
*****/
void DisplayDay(unsigned char x)
{
    unsigned char i,j;        //j,k,l 分别储存温度的百位、十位和个位
    i=x/10;//取十位
    j=x%10;//取个位
    WriteAddress(0x0c);      //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[i]);     //将百位数字的字符常量写入 LCD
    WriteData(digit[j]);     //将十位数字的字符常量写入 LCD
    delaynms(50);           //延时 1ms 给硬件一点反应时间
}
/*****
函数功能：显示月
入口参数：x
*****/
void DisplayMonth(unsigned char x)
{
    unsigned char i,j;        //j,k,l 分别储存温度的百位、十位和个位
    i=x/10;//取十位
    j=x%10;//取个位
    WriteAddress(0x09);      //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[i]);     //将百位数字的字符常量写入 LCD
    WriteData(digit[j]);     //将十位数字的字符常量写入 LCD
    delaynms(50);           //延时 1ms 给硬件一点反应时间
}

```

```

}
/*****
函数功能：显示年
入口参数：x
*****/
void DisplayYear(unsigned char x)
{
    unsigned char i,j;    //j,k,l 分别储存温度的百位、十位和个位
    i=x/10;//取十位
    j=x%10;//取个位
    WriteAddress(0x06);    //写显示地址,将在第 2 行第 7 列开始显示
    WriteData(digit[i]);    //将百位数字的字符常量写入 LCD
    WriteData(digit[j]);    //将十位数字的字符常量写入 LCD
    delaynms(50);    //延时 1ms 给硬件一点反应时间
}

/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char second,minute,hour,day,month,year;    //分别储存苗、分、
    小时,日,月,年
    unsigned char ReadValue;    //储存从 1302 读取的数据
    LcdInitiate();    //将液晶初始化
    WriteAddress(0x01);    //写 Date 的显示地址,将在第 1 行第 2 列开始显示
    WriteData('D');    //将字符常量写入 LCD
    WriteData('a');    //将字符常量写入 LCD
    WriteData('t');    //将字符常量写入 LCD
    WriteData('e');    //将字符常量写入 LCD
    WriteData(':');    //将字符常量写入 LCD
    WriteAddress(0x08);    //写年月分隔符的显示地址, 显示在第 1 行第 9 列
    WriteData('-');    //将字符常量写入 LCD
    WriteAddress(0x0b);    //写月日分隔符的显示地址, 显示在第 1 行第 12 列
    WriteData('-');    //将字符常量写入 LCD
    WriteAddress(0x45);    //写小时与分钟分隔符的显示地址, 显示在第 2 行第 6
    列
    WriteData(':');    //将字符常量写入 LCD
    WriteAddress(0x48);    //写分钟与秒分隔符的显示地址, 显示在第 2 行第 9
    列
    WriteData(':');    //将字符常量写入 LCD
    Init_DS1302();    //将 1302 初始化
    while(1)
    {

```

```

    ReadValue = ReadSet1302(0x81); //从秒寄存器读数据
    second=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F);//将读出数据
转化
    DisplaySecond(second); //显示秒
    ReadValue = ReadSet1302(0x83); //从分寄存器读
    minute=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F); //将读出数据
转化
    DisplayMinute(minute); //显示分
    ReadValue = ReadSet1302(0x85); //从分寄存器读
    hour=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F); //将读出数据转
化
    DisplayHour(hour); //显示小时
    ReadValue = ReadSet1302(0x87); //从分寄存器读
    day=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F); //将读出数据转
化
    DisplayDay(day); //显示日
    ReadValue = ReadSet1302(0x89); //从分寄存器读
    month=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F); //将读出数据
转化
    DisplayMonth(month); //显示月
    ReadValue = ReadSet1302(0x8d); //从分寄存器读
    year=((ReadValue&0x70)>>4)*10 + (ReadValue&0x0F); //将读出数据转
化
    DisplayYear(year); //显示年
}
}

```

实例 99：单片机数据发送程序

```

#include<reg51.h> //包含单片机寄存器的头文件
/*****
函数功能：向 PC 发送一个字节数据
*****/
void Send(unsigned char dat)
{
    SBUF=dat;
    while(TI==0)
        ;
    TI=0;
}
/*****
函数功能：延时 1ms
(3j+2)i=(3×33+2) × 10=1010(微秒)，可以认为是 1 毫秒

```

```

*****/
void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
*****/
void delaynms(unsigned char x)
{
    unsigned char i;
    for(i=0;i<x;i++)
        delay1ms();
}
/*****
函数功能：主函数
*****/
void main(void)
{
    unsigned char i;
    TMOD=0x20; //定时器 T1 工作于方式 2
    TH1=0xfd; //根据规定给定时器 T1 赋初值
    TL1=0xfd; //根据规定给定时器 T1 赋初值
    PCON=0x00; //波特率 9600
    TR1=1; //启动定时器 t1
    SCON=0x40; //串口工作方式 1
    while(1)
    {
        for(i=0;i<200;i++) //模拟检测数据
        {
            Send(i); //发送数据 i
            delaynms(100); //100ms 发送一次检测数据
        }
    }
}

```

实例 100：电机转速表设计

```

#include<reg51.h> //包含单片机寄存器的头文件
#include<intrins.h> //包含_nop_()函数定义的头文件

```

```

sbit RS=P2^0;    //寄存器选择位，将 RS 位定义为 P2.0 引脚
sbit RW=P2^1;    //读写选择位，将 RW 位定义为 P2.1 引脚
sbit E=P2^2;     //使能信号位，将 E 位定义为 P2.2 引脚
sbit BF=P0^7;    //忙碌标志位，将 BF 位定义为 P0.7 引脚
unsigned char code digit[ ]={"0123456789"}; //定义字符数组显示数字
unsigned int v;   //储存电机转速
unsigned char count; //储存定时器 T0 中断次数
bit flag;        //计满 1 秒钟标志位
/*****
函数功能：延时 1ms
(3j+2)*i=(3×33+2)×10=1010(微秒)，可以认为是 1 毫秒
*****/

void delay1ms()
{
    unsigned char i,j;
    for(i=0;i<10;i++)
        for(j=0;j<33;j++)
            ;
}
/*****
函数功能：延时若干毫秒
入口参数：n
*****/

void delay(unsigned char n)
{
    unsigned char i;
    for(i=0;i<n;i++)
        delay1ms();
}

/*****
函数功能：判断液晶模块的忙碌状态
返回值：result。result=1，忙碌;result=0，不忙
*****/

unsigned char BusyTest(void)
{
    bit result;
    RS=0;        //根据规定，RS 为低电平，RW 为高电平时，可以读状态
    RW=1;
    E=1;        //E=1，才允许读写
    _nop_();    //空操作
    _nop_();
    _nop_();
    _nop_();    //空操作四个机器周期，给硬件反应时间

```

```

    result=BF; //将忙碌标志电平赋给 result
    E=0;      //将 E 恢复低电平
    return result;
}
/*****
函数功能：将模式设置指令或显示地址写入液晶模块
入口参数：dictate
*****/
void WriteInstruction (unsigned char dictate)
{
    while(BusyTest()==1); //如果忙就等待
    RS=0;                 //根据规定，RS 和 R/W 同时为低电平时，可以
    写入指令
    RW=0;
    E=0;                 //E 置低电平(根据表 8-6，写指令时，E 为高脉
    冲，
    // 就是让 E 从 0 到 1 发生正跳变，所以应先置"0"

    _nop_();
    _nop_();             //空操作两个机器周期，给硬件反应时间
    P0=dictate;         //将数据送入 P0 口，即写入指令或地址
    _nop_();
    _nop_();
    _nop_();
    _nop_();             //空操作四个机器周期，给硬件反应时间
    E=1;                 //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();             //空操作四个机器周期，给硬件反应时间
    E=0;                 //当 E 由高电平跳变成低电平时，液晶模块开始
    执行命令
}
/*****
函数功能：指定字符显示的实际地址
入口参数：x
*****/
void WriteAddress(unsigned char x)
{
    WriteInstruction(x|0x80); //显示位置的确定方法规定为"80H+地址码 x"
}
/*****
函数功能：将数据(字符的标准 ASCII 码)写入液晶模块
入口参数：y(为字符常量)
*****/

```



```

void WriteData(unsigned char y)
{
    while(BusyTest()!=1);
    RS=1;           //RS 为高电平, RW 为低电平时, 可以写入数据
    RW=0;
    E=0;           //E 置低电平(根据表 8-6, 写指令时, E 为高脉冲,
                  // 就是让 E 从 0 到 1 发生正跳变, 所以应先置"0"
    P0=y;          //将数据送入 P0 口, 即将数据写入液晶模块
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=1;          //E 置高电平
    _nop_();
    _nop_();
    _nop_();
    _nop_();      //空操作四个机器周期, 给硬件反应时间
    E=0;          //当 E 由高电平跳变成低电平时, 液晶模块开始执行命令
}

```

函数功能: 对 LCD 的显示模式进行初始化设置

*****/

```

void LcdInitiate(void)
{
    delay(15);      //延时 15ms, 首次写指令时应给 LCD 一段较长的
                  //反应时间
    WriteInstruction(0x38); //显示模式设置: 16×2 显示, 5×7 点阵, 8 位数
                  //据接口
    delay(5);       //延时 5ms , 给硬件一点反应时间
    WriteInstruction(0x38);
    delay(5);
    WriteInstruction(0x38); //连续三次, 确保初始化成功
    delay(5);
    WriteInstruction(0x0c); //显示模式设置: 显示开, 无光标, 光标不闪烁
    delay(5);
    WriteInstruction(0x06); //显示模式设置: 光标右移, 字符不移
    delay(5);
    WriteInstruction(0x01); //清屏幕指令, 将以前的显示内容清除
    delay(5);
}

```

函数功能: 显示速度提示符

*****/

```

void display_sym(void)
{
    WriteAddress(0x00); //写显示地址,将在第 1 行第 1 列开始显示
    WriteData('v');    //将字符常量 v 写入 LCD
    WriteData('=');    //将字符常量=写入 LCD

}
/*****
函数功能: 显示速度数值
*****/

void display_val(unsigned int x)
{
    unsigned char i,j,k,l;    //j,k,l 分别储存温度的百位、十位和个位
    i=x/1000;                //取千位
    j=(x%1000)/100;         //取百位
    k=(x%100)/10;           //取十位
    l=x%10;                 //取个位
    WriteAddress(0x02);     //写显示地址,将在第 1 行第 3 列开始显示
    WriteData(digit[i]);    //将千位数字的字符常量写入 LCD
    WriteData(digit[j]);    //将百位数字的字符常量写入 LCD
    WriteData(digit[k]);    //将十位数字的字符常量写入 LCD
    WriteData(digit[l]);    //将个位数字的字符常量写入 LCD

}
/*****
函数功能: 显示速度单位 “r/min”
*****/

void display_unit(void)
{
    WriteAddress(0x06);    //写显示地址,将在第 2 行第 7 列开始显示
    WriteData('r');       //将字符常量 r 写入 LCD
    WriteData('/');       //将字符常量/写入 LCD
    WriteData('m');       //将字符常量 m 写入 LCD
    WriteData('i');       //将字符常量 i 写入 LCD
    WriteData('n');       //将字符常量 n 写入 LCD

}
/*****
函数功能: 主函数
*****/

void main(void)
{
    LcdInitiate();        //调用 LCD 初始化函数
    TMOD=0x51;           //定时器 T1 工作于计数模式 1, 定时器 T0
    工作于计时模式 1;

```

```

    TH0=(65536-46083)/256;    //定时器 T0 的高 8 位设置初值, 每 50ms 产生一次中断
    TL0=(65536-46083)%256;    //定时器 T0 的低 8 位设置初值, 每 50ms 产生一次中断
    EA=1;                      //开总中断
    ET0=1;                      //定时器 T0 中断允许
    TR0=1;                      //启动定时器 T0
    count=0;                    //将 T0 中断次数初始化为 0
    display_sym();              //显示速度提示符
    display_val(0000);          //显示器工作正常标志
    display_unit();             //显示速度单位
    while(1)                    //无限循环
    {
        TR1=1;                  //定时器 T1 启动
        TH1=0;                  //定时器 T1 高 8 位赋初值 0
        TL1=0;                  //定时器 T1 低 8 位赋初值 0
        flag=0;                 //时间还未满 1 分钟
        while(flag==0)          //时间未等待
            ;
        v=(TH1*256+TL1)*60/16; //计算速度, 每周产生 16 个脉冲
        display_val(v);         //显示速度
    }
}
/*****

```

函数功能: 定时器 T0 的中断服务函数

*****/

void Time0(void) interrupt 1 using 1 //定时器 T0 的中断编号为 1, 使用第 1 组工作寄存器

```

{
    count++;                    //T0 每中断 1 次, count 加 1
    if(count==20)               //若累计满 20 次, 即计满 1 秒钟
    {
        flag=1;                 //计满 1 秒钟标志位置 1
        count=0;                //清 0, 重新统计中断次数
    }
    TH0=(65536-46083)/256;     //定时器 T0 高 8 位重新赋初值
    TL0=(65536-46083)%256;     //定时器 T0 低 8 位重新赋初值
}

```

//模拟霍尔脉冲

```
#include<reg51.h>
```

```
sbit cp=P3^2; //将 cp 位定义为 P3.2 引脚, 从此脚输出脉冲信号
```

*****/

函数功能：延时约 600 微秒

```
*****/
void delay()
{
    unsigned char i;
    for(i=0;i<200;i++)
        ;
}
/*****
```

函数功能：主函数

```
*****/
void main(void)
{
    while(1)
    {
        cp=1;    //置高电平
        delay(); //等待 600 微秒
        cp=0;    //置低电平
        delay(); //等待 600 微秒
    }
}
```