

附录 D 第 18 章嵌入式代码

系统包括：main.c、main.h、delay.c、delay.h、ks0108.c、ks0108.h、ds18b20.c、ds18b20.h、ds1302.c、ds1302.h、IIC.c、IIC.h、serial.c、serial.h 共 14 个文件。

```

/*****
* 文件名: main.c
* 说明: 用 AT89C51、DS18B20、DS1302、24C02、AMPIRE128X64、电机、灯泡、按键、
*      led 指示灯等实现一个简易温度控制自动管理系统。
*      主要功能是可以显示时间（年、月、日、时、分、秒）、当前温度、温度上下限、
*      温度是否在正常范围，可以通过按键控制风机和加热灯，用蜂鸣器报警温度超出
*      上下限、同时用 LED 指示灯指示，可以通过串口修改时间、温度上限限，控制风机
*      加热灯等。
* 作者: 老杨
* 时间: 2011.6.9 完成
*****/

#include "main.h"
#include "delay.h"
#include "ds1302.h"
#include "KS0108.h"
#include "IIC.h"
#include "ds18B20.h"
#include "serial.h"

//端口定义
sbit SPEAK = P2^2;           //报警器控制管脚
sbit LED_NOR = P2^3;         //正常指示灯控制管脚
sbit LED_OVER = P2^4;       //温度上限指示灯控制管脚
sbit LED_LOW = P2^5;        //温度下限指示灯控制管脚
sbit MOTO = P2^6;           //风机控制管脚
sbit HEAT = P2^7;           //加热设备控制管脚
//sbit RXD = P3^0;           //reg51.h 中已经定义
//sbit TXD = P3^1;           //reg51.h 中已经定义
sbit K_MOTO = P3^2;          //控制风机按键
sbit K_HEAT = P3^3;          //控制加热设备的按键

#define DAT_ADDR 0x03        //IIC EEPROM 中保存的温度上、下限的地址

uchar g_pc_cont;            //pc 控制，0 没控制，1 风扇开，2 风扇关，4 加热开，8 加热关；

/*****
* 函数: 外部 0 中断，设定键中断响应
* 参数: 空
*****/

void int0_inter() interrupt 0

```

```
{
    if(K_MOTO == 0)
    {
        MOTO = ~MOTO;
        if(MOTO == 0)
        {
            g_pc_cont &= 0x0C;
        }
    }
}

/*****
* 函 数：外部 1 中断，设定键中断响应
* 参 数：空
*****/
void int1_inter() interrupt 2
{
    if(K_HEAT == 0)
    {
        HEAT = ~HEAT;
        g_pc_cont &= 0x03;
    }
}

/*****
* 函 数：定时器 1 中断函数
* 参 数：空
* 返回值：无
*****/
void timer0_inter() interrupt 1
{
    //250us*4=1ms
    static uchar timer = 0;
    static uchar speaker = 0;
    if(speaker < 235) //控制蜂鸣器声音间隔
    {
        if(timer++ == 3) //控制报警器声音
        {
            SPEAK = ~SPEAK;
            timer = 0;
            speaker++;
        }
    }
    }else
    {
```

```
        if(timer++ == 4)
        {
            speaker++;
        }
    }
}

/*****
* 函 数: AT89C51 端口初始化
* 参 数: 空
* 返回值: 无
*****/

void port_init(void)
{
    LED_NOR = 1;
    LED_OVER = 0;
    LED_LOW = 0;
    SPEAK = 0;

    MOTO = 0;
    HEAT = 0;
}

/*****
* 函 数: 定时器 0 初始化
* 参 数: 空
* 返回值: 无
*****/

void timer0_init(void)
{
    TL0 = 0x06;          //定时器初始值
    TH0 = 0x06;
    TMOD |= 2;           //定时器模式, 工作方式 2
    ET0 = 1;             //打开 T0 中断
}

/*****
* 函 数: 外部中断初始化
* 参 数: 空
* 返回值: 无
*****/

void int_init(void)
{
    IT0 = 1;             //下降沿触发
```

```
    EX0 = 1;        //外部中断打开
    IT1 = 1;
    EX1 = 1;
}

/*****

* 函 数：控制排风扇（打开、关闭）
* 参 数：open=0 关闭，=1 打开
* 返回值：无
*****/

void open_moto(uchar open)
{
    if(open == 0)
    {
        MOTO = 0;
        g_pc_cont ^= 0x2;
    }else
    {
        MOTO = 1;
        g_pc_cont ^= 0x1;
    }
}

/*****

* 函 数：控制加热模块（打开、关闭）
* 参 数：open=0 关闭，=1 打开
* 返回值：无
*****/

void open_heat(uchar open)
{
    if(open == 0)
    {
        HEAT = 0;
        g_pc_cont ^= 0x8;
    }else
    {
        HEAT = 1;
        g_pc_cont ^= 0x4;
    }
}

/*****

* 函 数：设置温度上下限
* 参 数：down 温度下限值，up 温度上限值
*****/
```

* 返回值: 无

*****/

void set_temper_updown(uchar down, uchar up)

```
{
    write_IIC(DAT_ADDR, down);
    delay_ms(2);
    write_IIC(DAT_ADDR+1, up);
}
```

*****/

* 函 数: 设置系统时间

* 参 数: pBuff 保存系统时间: 年、月、日、时、分、秒、星期

* 返回值: 无

*****/

void set_system_time(uchar *pBuff)

```
{
    SYSTEM_TIME str_time;
    str_time.year = pBuff[0];
    str_time.month = pBuff[1];
    str_time.date = pBuff[2];
    str_time.hour = pBuff[3];
    str_time.min = pBuff[4];
    str_time.sec = pBuff[5];
    str_time.day = pBuff[6];

    set_time(str_time);
}
```

*****/

* 函 数: 获取当前温度

* 参 数: pBuff 保存温度整数、小数值

* 返回值: 无

*****/

void get_temper(uchar *pBuff)

```
{
    uchar pTmpBuff[2] = {0,0};
    read_temper(pTmpBuff);    //pBuff[0]整数、pBuff[1]小数部分
    *pBuff = pTmpBuff[0];
    *(pBuff+1) = pTmpBuff[1];
}
```

*****/

* 函 数: 获取温度上下限

* 参 数: pDown 温度下限值, pUp 温度上限值

```
* 返回值: 无
*****/

void get_temper_updown(uchar *pDown, uchar *pUp)
{
    *pDown = read_IIC(DAT_ADDR);
    *pUp = read_IIC(DAT_ADDR+1);
}

/*****

* 函 数: 获取系统时间
* 参 数: pBuff 保存系统时间: 年、月、日、时、分、秒、星期
* 返回值: 无
*****/

void get_system_time(uchar *pBuff)
{
    SYSTEM_TIME str_time;
    str_time = read_time();
    *pBuff = str_time.year;
    *(pBuff+1) = str_time.month;
    *(pBuff+2) = str_time.date;
    *(pBuff+3) = str_time.hour;
    *(pBuff+4) = str_time.min;
    *(pBuff+5) = str_time.sec;
    *(pBuff+6) = str_time.day;
}

/*****

* 函 数: 获取风机状态
* 参 数: pStat 保存风机的状态
* 返回值: 无
*****/

void get_moto_stat(uchar *pStat)
{
    *pStat = MOTO;
}

/*****

* 函 数: 获取加热设备的状态
* 参 数: pStat 保存加热设备的状态
* 返回值: 无
*****/

void get_heat_stat(uchar *pStat)
{
    *pStat = HEAT;
```

```
}
```

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

```
* 函 数：检测温度是否在正常范围内
```

```
* 参 数：temper 当前温度，down_temper 温度下限，up_temper 温度上限
```

```
* 返回值：0 正常，1 超出下限，2 超出上限
```

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

```
uchar normal_temper(uchar temper, uchar down_temper, uchar up_temper)
```

```
{
```

```
    uchar state = 0;
```

```
    uchar symbol = 0;
```

```
    uchar symbol2 = 0, symbol3 = 0;
```

```
    if((temper&0x80) != 0)
```

```
    {
```

```
        symbol = 1;
```

```
    }
```

```
    if((down_temper&0x80) != 0)
```

```
    {
```

```
        symbol2 = 1;
```

```
    }
```

```
    if((temper <= down_temper) && (symbol2 == symbol))
```

```
    {
```

```
        state = 1;
```

```
    }
```

```
    if(symbol2 < symbol)
```

```
    {
```

```
        state = 1;
```

```
    }
```

```
    if((up_temper&0x80) != 0)
```

```
    {
```

```
        symbol3 = 1;
```

```
    }
```

```
    if((temper >= up_temper) && (symbol3 == symbol))
```

```
    {
```

```
        state = 2;
```

```
    }
```

```
    return(state);
```

```
}
```

```
/******  
* 函 数：控制调节温度  
* 参 数：state 温度范围：0 正常，1 超出下限，2 超出上限  
* 返回值：无  
*****/
```

```
void cont_temper(uchar state)  
{  
    //指示灯、报警定时器、加热、风机  
    if(state == 0)  
    {  
        LED_NOR = 1;  
        LED_OVER = 0;  
        LED_LOW = 0;  
        if((g_pc_cont&0x0F)==0)  
        {  
            MOTO = 0;  
            HEAT = 0;  
        }  
  
        TR0 = 0; //关闭 T0 定时器  
    }  
    else if(state == 1)  
    {  
        LED_NOR = 0;  
        LED_OVER = 0;  
        LED_LOW = 1;  
        if((g_pc_cont&0x0C)==0)  
        {  
            MOTO = 0;  
            HEAT = 1;  
        }  
  
        TR0 = 1; //启动 T0 定时器  
    }  
    else  
    {  
        LED_NOR = 0;  
        LED_OVER = 1;  
        LED_LOW = 0;  
  
        if((g_pc_cont&0x03)==0)  
        {  
            MOTO = 1;  
            HEAT = 0;  
        }  
    }  
}
```



```

    }

    TR0 = 1; //启动 T0 定时器
}
}

/*****
* 函 数：主函数：完成温控主系统功能
* 参 数：空
* 返回值：无
*****/

void main(void)
{
    SYSTEM_TIME str_time;
    uchar pbuff[2] = {0,0}; //温度整数、小数部分
    uchar down_temper, up_temper; //温度上、下限
    uchar state = 0; //当前温度状态，是否超温

    port_init(); //端口初始化
    int_init(); //外部中断初始化
    timer0_init(); //定时器 0 初始化

    //外围芯片初始化
    ds1302_init(); //DS1302 端口初始化
    ks0108_init(); //LCD 初始化
    init_IIC();
    timer1_init(); //uart 波特率
    serial_init(); //uart

    EA = 1; //打开全局中断
    //TR0 = 1; // 在 cont_temper 函数中控制 Timer0

    //读取 IIC 温度上下限
    down_temper = read_IIC(DAT_ADDR);
    up_temper = read_IIC(DAT_ADDR+1);
    //如果读取的值都是 0xFF（温度上下限没有被设置过），设定默认温度限值 0、35
    if(down_temper == 0xFF && up_temper == 0xFF)
    {
        down_temper = 0; //默认温度下限值
        up_temper = 35; //默认温度上限值
        write_IIC(DAT_ADDR, down_temper);
        delay_ms(2);
        write_IIC(DAT_ADDR+1, up_temper);
    }
}

```

```

temper_convert();    //启动温度转换
delay_ms(300);

while(1)
{
    str_time = read_time();    //读取时间
    show_time(str_time);      //显示年、月、日、时、分、秒、星期

    delay_ms(500);            //转换温度时间需要 750ms
    read_temper(pbuff);       //读取当前温度值

    //判断温度是否正常
    state = normal_temper(pbuff[0], down_temper, up_temper);    //判断温度是否正常
    cont_temper(state);    //控制温度： 风扇、加热、报警、指示灯

    temper_convert();    // 启动温度转换， 需要 750ms

    //显示温度、温度上、下限值
    show_temperature(pbuff[0], pbuff[1], state);    //显示当前温度、状态
    show_area(down_temper, up_temper);    //显示上、下限温度

    deal_protocol();    //串口通信处理
    delay_ms(200);

    //上下限有可能已经被修改
    down_temper = read_IIC(DAT_ADDR);
    up_temper = read_IIC(DAT_ADDR+1);
}
}

```

```

/*****

```

```

* 文件名： main.h

```

```

*****/

```

```

#include <reg51.h>

```

```

#include <intrins.h>

```

```

#ifndef _MAIN_H

```

```

#define _MAIN_H

```

```

#define uint    unsigned int

```

```

#define uchar    unsigned char

```

```

//自定义时间结构体

```

```

typedef struct __system_time{
    uchar year;
    uchar month;
    uchar date;
    uchar hour;
    uchar min;
    uchar sec;
    uchar day; //星期
}SYSTEM_TIME;

void get_temper(uchar *pBuff);
void get_temper_updown(uchar *pDown, uchar *pUp);
void get_system_time(uchar *pBuff);
void get_moto_stat(uchar *pStat);
void get_heat_stat(uchar *pStat);
void set_temper_updown(uchar down, uchar up);
void set_system_time(uchar *pBuff);
void open_moto(uchar open);
void open_heat(uchar open);

uchar normal_temper(uchar int_temper, uchar down_temper, uchar up_temper);

#endif

```

```

/*****
* 文件名: delay.c
* 说 明: 延时函数
*****/

```

```
#include "main.h"
```

```

/*****
* 函数: 延时 1ms 函数
* 参数: timer 要延时的毫秒数值
*****/

```

```

void delay_ms(uint timer)
{
    uchar j = 0;
    while(timer--)
    {
        for(j = 124; j>0; j--)
        {
            ;
        }
    }
}

```

```
}

```

```
/******

```

```
* 函数：延时约 9us*timer + 6um

```

```
* 参数：timer 要延时的 timer*9+6 微秒数值

```

```
*****/

```

```
void delay_us(uchar timer)

```

```
{

```

```
    for( ; timer>0; timer--)

```

```
    {

```

```
        _nop_();

```

```
    }

```

```
}

```

```
/******

```

```
* 文件名：delay.h

```

```
*****/

```

```
#ifndef _DELAY_H

```

```
#define _DELAY_H

```

```
void delay_ms(uint timer);

```

```
void delay_us(uchar timer);

```

```
#endif

```

```
/******

```

```
* 文件名：KS0108.c

```

```
* 说明：操作 KS0108 系列芯片驱动液晶的基本函数

```

```
*****/

```

```
#include "main.h"

```

```
#define DATA P0    //LCD12864 数据线

```

```
sbit RS = P3^7;    // 数据\指令 选择

```

```
sbit RW = P3^6;    // 读\写 选择

```

```
sbit EN = P2^0;    // 读\写使能

```

```
sbit CS1 = P1^6;    // 片选 1

```

```
sbit CS2 = P1^7;    // 片选 2

```

```
/******

```

```
* 定义中文字库

```

```
* 宋体、小四（12），点阵为：宽 x 高=16x16，纵向取模、字节倒序

```

```
*****/

```

```
uchar code PIC1616[]={

```

```
0x00,0x02,0x07,0xE7,0xFA,0x1C,0x06,0x02,0x02,0x02,0x02,0x06,0x1E,0x1E,0x00,    //°C

```

```

0x00,0x00,0x00,0x07,0x1F,0x38,0x60,0x40,0x40,0x40,0x40,0x60,0x38,0x18,0x00,

0x00,0x00,0x40,0x42,0x5E,0x5C,0x48,0x40,0x7F,0x7F,0x50,0x5E,0x4E,0xC4,0xC0,0x00, //当
0x00,0x00,0x20,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x7F,0x7F,0x00,

0x08,0x08,0xE8,0xE8,0xA9,0xAF,0xEE,0xEA,0x08,0xC8,0xCC,0x0F,0xEB,0xEA,0x08,0x08, //前
0x00,0x00,0x7F,0x7F,0x24,0x64,0x7F,0x3F,0x00,0x1F,0x5F,0xC0,0xFF,0x7F,0x00,0x00,

0x00,0x02,0x02,0xC2,0xC2,0x02,0x02,0x02,0xFE,0xFE,0x82,0x82,0x82,0x82,0x82,0x02, //正
0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x20,0x20,

0x20,0x38,0x18,0x09,0xEF,0xEE,0xAA,0xAF,0xAF,0xA8,0xEC,0xEF,0x2B,0x3A,0x18,0x08, //常
0x00,0x00,0x3E,0x3E,0x02,0x02,0x02,0xFF,0xFF,0x02,0x12,0x32,0x3E,0x1E,0x00,0x00,

0x40,0x48,0x48,0x48,0xFF,0xFF,0x48,0xCA,0xC2,0xFE,0xBE,0xA2,0xE2,0xFE,0xBE,0x00, //超
0x60,0x7F,0x3F,0x60,0x7F,0x7F,0x42,0x42,0x5F,0x5F,0x48,0x48,0x48,0x5F,0x5F,0x40,

0x10,0x31,0xA7,0xF6,0x70,0x7E,0x7E,0x4A,0x4A,0x4A,0x4A,0x7E,0x7E,0x00,0x00,0x00, //温
0x02,0xFE,0xFF,0x41,0x7F,0x7F,0x41,0x7F,0x7F,0x41,0x7F,0x7F,0x41,0x7F,0x7F,0x40,
};

/*****
*   定义数字、符号点阵
*   宋体、小四（12），点阵为：宽 x 高=8x16；
*   纵向取模、字节倒序（逆向、列行式）
*****/

uchar code PIC0816[]={
0x00,0xE0,0xF0,0x18,0x08,0x18,0xF0,0xE0,0x00,0x0F,0x1F,0x30,0x20,0x30,0x1F,0x0F, //0
0x00,0x10,0x10,0xF8,0xF8,0x00,0x00,0x00,0x20,0x20,0x3F,0x3F,0x20,0x20,0x00, //1
0x00,0x70,0x78,0x08,0x08,0x88,0xF8,0x70,0x00,0x30,0x38,0x2C,0x26,0x23,0x31,0x30, //2
0x00,0x30,0x38,0x88,0x88,0xC8,0x78,0x30,0x00,0x18,0x38,0x20,0x20,0x31,0x1F,0x0E, //3
0x00,0x00,0xC0,0xE0,0x30,0xF8,0xF8,0x00,0x00,0x07,0x07,0x24,0x24,0x3F,0x3F,0x24, //4
0x00,0xF8,0xF8,0x88,0x88,0x88,0x08,0x08,0x00,0x19,0x39,0x21,0x20,0x31,0x1F,0x0E, //5
0x00,0xE0,0xF0,0x98,0x88,0x98,0x18,0x00,0x00,0x0F,0x1F,0x31,0x20,0x31,0x1F,0x0E, //6
0x00,0x38,0x38,0x08,0xC8,0xF8,0x38,0x08,0x00,0x00,0x00,0x3F,0x3F,0x00,0x00,0x00, //7
0x00,0x70,0xF8,0x88,0x08,0x88,0xF8,0x70,0x00,0x1C,0x3E,0x23,0x21,0x23,0x3E,0x1C, //8
0x00,0xE0,0xF0,0x18,0x08,0x18,0xF0,0xE0,0x00,0x00,0x31,0x33,0x22,0x33,0x1F,0x0F, //9
//up
0x40,0x70,0x7C,0xFF,0xFF,0x7C,0x70,0x40,0x00,0x00,0x00,0x7F,0x7F,0x00,0x00,0x00, //10
//down
0x00,0x00,0x00,0xFE,0xFE,0x00,0x00,0x00,0x02,0x0E,0x3E,0xFF,0xFF,0x3E,0x0E,0x02, //11
//-
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01, //12
//+

```

```

0x00,0x00,0x00,0xF0,0xF0,0x00,0x00,0x00,0x01,0x01,0x01,0x1F,0x01,0x01,0x01, //13
//:
0x00,0x00,0x00,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x00,0x00, //14
//.
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x00,0x00,0x00, //15
};

/*****
* 函 数: 检查 LCD 状态, 是否忙
* 参 数: 空
* 返回值: 0 空闲, 非 0 忙
*****/

uchar check_state(void)
{
    uchar i = 0;
    uchar tmp = 0;      //状态信息 (判断是否忙)

    RS = 0;  //数据/指令选择, D/I (RS) = "L", 表示 DB7~DB0 为显示数据
    RW = 1;  //R/W= "H", E= "H" 数据被读到 DB7~DB0
    for(i=0; i<6; i++)
    {
        DATA = 0x00;
        EN = 1;      //拉高 EN 管脚为高电平
        _nop_();      //一个时钟延时
        tmp = DATA;
        EN = 0;
        tmp = 0x80 & tmp;  //仅当第 7 位为 0 时才可操作(判别 busy 信号)
        if(tmp == 0)
        {
            break;
        }
    }
    return(tmp);
}

/*****
* 函 数: 写命令到 LCD
* 参 数: cmd 要写的命令
* 返回值: 无
*****/

void write_cmd(uchar cmd)
{
    uchar stat = 0;
    stat = check_state();  //状态检查, LCD 是否忙

```

```

    RS = 0;           //向 LCD 发送命令。RS=0 写指令，RS=1 写数据
    RW = 0;           //R/W= “L” ， E= “H→L” 数据被写到 IR 或 DR
    DATA = cmd;       //com :命令
    EN = 1;           //拉高 EN 管脚，高电平
    _nop();_nop();
    EN = 0;           //拉低 EN 管脚，形成下降沿
}

/*****
* 函 数：写数据到 LCD
* 参 数：dat 要写的的数据
* 返回值：无
*****/

void write_data(uchar dat)
{
    uchar stat = 0;
    stat = check_state(); //状态检查，LCD 是否忙
    RS = 1;               //RS=0 写指令，RS=1 写数据
    RW = 0;               //R/W= “L” ， E= “H→L” 数据被写到 IR 或 DR
    DATA = dat;          //dat:显示数据
    EN = 1;               //拉高 EN 管脚，高电平
    _nop();_nop();
    EN = 0;               //拉低 EN 管脚，形成下降沿
}

/*****
* 函 数：设置页地址：0xB8
* 参 数：page 页地址
* 返回值：无
*****/

void set_page(uchar page)
{
    page = 0xB8|page;     //1011 1xxx 0<=page<=7 设定页地址--X 0-7,8 行为一页 64/8=8，共 8 页
    write_cmd(page);
}

/*****
* 函 数：设定行地址--(X:0-63)：0xC0
* 参 数：line 行地址
* 返回值：无
*****/

void set_line(uchar line)
{
    line = 0xC0|line;     //1100 0000

```

```

        write_cmd(line);    //设置从哪行开始：0--63，一般从0行开始显示
    }

/*****

* 函 数：设定列地址--(Y:0-63)：0x40
* 参 数：column 列地址
* 返回值：无
*****/

void set_column(uchar column)
{
    column = column&0x3f;    //column 最大值为 64，越出 0=<column<=63
    column = 0x40|column;    //01xx xxxx
    write_cmd(column);
}

/*****

* 函 数：开关显示：0x3E
* 参 数：onoff 开关，0x3E 关显示，0x3F 开显示
* 返回值：无
*****/

void set_on(uchar onoff)
{
    onoff = 0x3E|onoff;    //0011 111x,onoff 只能为 0 或者 1
    write_cmd(onoff);
}

/*****

* 函 数：选择屏幕
* 参 数：screen: 0-全屏,1-左屏,2-右屏
* 返回值：无
*****/

void select_screen(uchar screen)
{
    switch(screen)
    {
    case 0:
        CS1=0;    //全屏
        _nop_();_nop_();_nop_();
        CS2=0;
        _nop_();_nop_();_nop_();
        break;
    case 1:
        CS1=1;    //左屏
        _nop_();_nop_();_nop_();

```



```

        CS2=0;
        _nop_(); _nop_(); _nop_();
        break;
    case 2:
        CS1=0;          //右屏
        _nop_(); _nop_(); _nop_();
        CS2=1;
        _nop_(); _nop_(); _nop_();
        break;
    }
}

/*****
* 函 数：清屏
* 参 数：screen: 0-全屏,1-左屏,2-右屏
* 返回值：无
*****/
void clear_screen(uchar screen)
{
    uchar i,j;

    select_screen(screen);
    for(i=0; i<8; i++)          //控制页数 0-7，共 8 页
    {
        set_page(i);
        set_column(0);
        for(j=0; j<64; j++)      //控制列数 0-63，共 64 列
        {
            write_data(0x00);    //写点内容，列地址自动加 1
        }
    }
}

/*****
* 函 数：初始化 LCD
* 参 数：空
* 返回值：无
*****/
void ks0108_init(void)
{
    check_state();

    select_screen(0);

```

```

    set_on(0);      //关显示

    select_screen(0);
    set_on(1);      //开显示

    select_screen(0);
    clear_screen(0); //清屏

    set_line(0);    //开始行:0
}

/*****
* 函 数: 显示全角汉字 (16x16 图片)
* 参 数: cs 选屏参数, page 选页参数,
*        column 选列参数, number 图片序号 (第几个汉字)
* 返回值: 无
*****/
void show_1616(uchar cs, uchar page, uchar column, uchar number)
{
    int i;

    select_screen(cs);
    column = column & 0x3f;

    set_page(page);      //写上半页
    set_column(column);  //控制列
    for(i=0; i<16; i++)  //控制 16 列的数据输出
    {
        write_data(PIC1616[i+32*number]); //i+32*number 汉字的前 16 个数据输出
    }

    set_page(page+1);    //写下半页
    set_column(column);  //控制列
    for(i=0; i<16; i++)  //控制 16 列的数据输出
    {
        write_data(PIC1616[i+32*number+16]); //i+32*number+16 汉字的后 16 个数据输出
    }
}

/*****
* 函 数: 显示半角汉字、数字、字母 (08x16 图片)
* 参 数: cs 选屏参数, page 选页参数,
*        column 选列参数, number 图片序号 (第几个数字)
*****/

```

* 返回值: 无

*****/

void show_0816(uchar cs, uchar page, uchar column, uchar number)

```
{
    uint i;

    select_screen(cs);
    column = column & 0x3f;

    set_page(page);      //写上半页
    set_column(column);
    for(i=0; i<8; i++)
    {
        write_data(PIC0816[i+16*number]);
    }

    set_page(page+1);    //写下半页
    set_column(column);
    for(i=0; i<8; i++)
    {
        write_data(PIC0816[i+16*number+8]);
    }
}
```

*****/

* 函 数: 显示当前时间

* 参 数: str_time 当前时间

* 返回值: 无

*****/

void show_time(SYSTEM_TIME str_time)

```
{
    uchar tmp = 0;
    show_0816(2, 0, 24, 2);    //年
    show_0816(2, 0, 32, 0);
    tmp = str_time.year/10;
    show_0816(2, 0, 40, tmp);
    tmp = str_time.year%10;
    show_0816(2, 0, 48, tmp);
    show_0816(2, 0, 56, 12);    //-

    tmp = str_time.month/10;
    show_0816(1, 0, 0, tmp);    //月
    tmp = str_time.month%10;
    show_0816(1, 0, 8, tmp);
}
```

```

show_0816(1, 0, 16, 12);    //-

tmp = str_time.date/10;
show_0816(1, 0, 24, tmp);    //日
tmp = str_time.date%10;
show_0816(1, 0, 32, tmp);

tmp = str_time.hour/10;
show_0816(2, 2, 32, tmp);    //时
tmp = str_time.hour%10;
show_0816(2, 2, 40, tmp);
show_0816(2, 2, 48, 14);     //:

tmp = str_time.min/10;
show_0816(2, 2, 56, tmp);    //分
tmp = str_time.min%10;
show_0816(1, 2, 0, tmp);
show_0816(1, 2, 8, 14);      //:

tmp = str_time.sec/10;
show_0816(1, 2, 16, tmp);    //秒
tmp = str_time.sec%10;
show_0816(1, 2, 24, tmp);
}

/*****
* 函 数：显示当前温度（含符号）
* 参 数：up 上限，down 下限,state 状态 0 正常，非 0 超温
* 返回值：无
*****/

void show_temperature(uchar int_temper, uchar dec_temper, uchar state)
{
    uchar tmp = 0;
    uchar symbol = 0;
    uchar int_tmp = 0;
    uint dec_tmp = 0;

    show_1616(2,4,0*16, 1);    //当
    show_1616(2,4,1*16, 2);    //前

    int_tmp = int_temper;
    if((int_tmp & 0x80) != 0)
    {
        symbol = 1;             //符号为负

```

```

    int_tmp = ~int_tmp;
    dec_temper = (0x0F-dec_temper) + 1;    //负数取反加 1
    if(dec_temper == 0x10)                //小数部分进位，则整数进位
    {
        int_tmp += 1;
        dec_temper = 0;
    }
}

show_0816(2, 4, 32, 13-symbol);          //+/-
tmp = int_tmp/10;
show_0816(2, 4, 40, tmp);
tmp = int_tmp%10;
show_0816(2, 4, 48, tmp);
show_0816(2, 4, 56, 15);                //.

dec_tmp = dec_temper;
dec_tmp = dec_tmp * 625 + 500;            //对百分位进行四舍五入
dec_temper = dec_tmp/1000;               //只显示十分位

show_0816(1, 4, 0, dec_temper);
show_1616(1, 4, 8, 0);

//是否正常
if(state == 0)
{
    show_1616(1, 4, 32, 3);    //正常
    show_1616(1, 4, 48, 4);
}
else
{
    show_1616(1, 4, 32, 5);    //超温
    show_1616(1, 4, 48, 6);
}
}

/*****
* 函 数：显示温度上、下限（含符号）
* 参 数：up 上限，down 下限
* 返回值：无
*****/

void show_area(uchar down, uchar up)
{
    uchar tmp = 0;
    uchar symbol = 0;

```

```

if((down&0x80) != 0)
{
    down = 0xFF - down + 1;    //取反，加 1
    symbol = 1;
}
show_0816(2, 6, 8+0, 11);    //下限显示
show_0816(2, 6, 8+8, 13-symbol);    //+/-
tmp = down/10;
show_0816(2, 6, 8+16, tmp);
tmp = down%10;
show_0816(2, 6, 8+24, tmp);
show_1616(2, 6, 8+32, 0);    //°C

symbol = 0;
if((up&0x80) != 0)
{
    up = 0xFF - up + 1;    //取反，加 1
    symbol = 1;
}
show_0816(1, 6, 0, 10);    //上限显示
show_0816(1, 6, 8, 13-symbol);    //+/-
tmp = up/10;
show_0816(1, 6, 16, tmp);
tmp = up%10;
show_0816(1, 6, 24, tmp);
show_1616(1, 6, 32, 0);    //°C
}

```

```

/*****

```

```

* 文件名: ks0108.h

```

```

*****/

```

```

#ifndef _KS0108_H

```

```

#define _KS0108_H

```

```

void ks0108_init(void);

```

```

void show_time(SYSTEM_TIME str_time);

```

```

void show_temperature(uchar int_temper, uchar dec_temper, uchar state);

```

```

void show_area(uchar down_temper, uchar up_temper);

```

```

#endif

```

```

/*****

```

```

* 文件名: DS18B20.c

```

```

* 说 明: 操作以 DS18B20 为代表的单总线通信的基本函数

```

```

/*****

```

```

#include "main.h"

```

```

#include "delay.h"

```

```

sbit DQ = P1^5;          // 数据通信线 DQ

```

```

/*****

```

```

* 函 数： 初始化 DS18B20（产生复位脉冲、等待应答信号）

```

```

* 参 数： 空

```

```

* 返回值： 无

```

```

*****/

```

```

void master_reset(void)

```

```

{

```

```

    uint i = 0;

```

```

    DQ = 1;

```

```

    DQ = 0;

```

```

    delay_us(80); //拉低 480~960us,这里大该 80*9+8us

```

```

    //for(i=100; i>0; i--);

```

```

    DQ = 1;          // 产生上升沿

```

```

    i = 20;

```

```

    while(DQ == 1)    //上升沿，等待应答信号（15~60us）

```

```

    {

```

```

        if(i-- == 0)

```

```

            break;

```

```

    }

```

```

    i = 100;

```

```

    while(DQ == 0)    //应答信号 60~240us

```

```

    {

```

```

        if(i-- == 0)

```

```

            break;

```

```

    }

```

```

    delay_us(1);      //短暂延时

```

```

}

```

```

/*****

```

```

* 函 数： 读取数据的一个位（满足读时隙要求）

```

```

* 参 数： 空

```

```

* 返回值： 读到的数据位

```

```

*****/

```

```

bit sing_read_bit(void)

```

```

{

```

```

    bit b = 0;

    DQ = 0;        //把总线拉低，低电平延时大于 1us
    _nop_();
    DQ = 1;        //释放低电平，等待 DS18B20 输出数据
    delay_us(1);    //延时 15us 以上，读时隙下降沿后 15us，DS18B20 输出数据才有效
    b = DQ;
    delay_us(5);    //延时大于 45us(5*9+8us)
    return (b);
}

```

```

/*****

```

```

* 函 数：读取数据的一个字节

```

```

* 参 数：空

```

```

* 返回值：读到的数据

```

```

*****/

```

```

uchar sing_read_byte(void)

```

```

{
    uchar i,j,b;
    b = 0;
    for (i=1; i<=8; i++)
    {
        j = sing_read_bit();
        b = (j<<7)|(b>>1);
    }
    return(b);
}

```

```

/*****

```

```

* 函 数：写一个字节数据（注意写 0 和写 1 的时序）

```

```

* 参 数：b 要写的数据

```

```

* 返回值：无

```

```

*****/

```

```

void sing_write_byte(uchar b)

```

```

{
    uint i = 0;
    uchar j = 0;
    bit bTmp = 0;
    for(j=1; j<=8; j++)
    {
        bTmp = b&0x01;
        b = b>>1;        //取下一位（由低位向高位）
        if (bTmp)
        {

```



```

        /* 写 1 */
        DQ = 0;
        i++;i++;      //延时，使得 15us 以内拉高
        DQ = 1;
        i = 8;
        while(i>0) i--; //整个写 1 时限不低于 60us
    }
    else
    {
        /* 写 0 */
        DQ = 0;
        i = 8;
        while(i>0) i--; //保持低电平在 60us 到 120us 之间
        DQ = 1;
        i++;
        i++;
    }
}

}

/*****
* 函 数：启动温度转换
* 参 数：空
* 返回值：无
*****/

void temper_convert(void)
{
    EA = 0;
    master_reset();      //初始化 DS18B20（产生复位脉冲、等待应答信号）
    sing_write_byte(0xCC); // skip rom 命令
    sing_write_byte(0x44); // convert T 命令
    EA = 1;
}

/*****
* 函 数：读取温度值
* 参 数：pbuff[0]保存整数位，pbuff[1]保存小数
* 返回值：无
*****/

void read_temper(uchar *pbuff)
{
    uchar tpls, tpmsb;

    EA = 0;

```

```

master_reset();           //初始化 DS18B20 (产生复位脉冲、等待应答信号)
sing_write_byte(0xCC);    // skip rom 命令
sing_write_byte(0xBE);    // read scratchpad 命令
tpls = sing_read_byte();  // 温度值低位字节 (其中低 4 位为二进制的“小数”部分)
tpms = sing_read_byte();  // 高位值高位字节 (其中高 5 位为符号位)
EA = 1;

*pbuff = (tpms<<4)|(tpls>>4);
*(pbuff+1) = (tpls&0x0F);
}

```

```

/*****
* 文件名: DS18B20.h
*****/

```

```

#ifndef _DS18B20_H
#define _DS18B20_H

void read_temper(uchar *pbuff);
void temper_convert(void);

#endif

```

```

/*****
* 文件名: ds1302.c
* 说 明: 操作 DS1302 的基本函数
*****/

```

```

#include "main.h"
#include "ds1302.h"

```

//DS1302 的引脚定义

```

sbit DS_RST = P1^2;      //RST 管脚
sbit DS_SCLK = P1^3;     //SCLK 管脚
sbit DS_IO = P1^4;       //IO 管脚

```

```

#define BCD2DEC(X) (((X&0x70)>>4)*10 + (X&0x0F))    //用于将 BCD 码转成十进制
#define DEC2BCD(X) ((X/10)<<4 | (X%10))           //用于将十进制转成 BCD 码

```

```

/*****
* 函 数: 初始化 DS1302 的引脚
* 参 数: 空
*****/

```

```

void ds1302_init(void)
{
    DS_RST = 0;

```

```
    DS_SCLK = 0;
    DS_IO = 0;
}

/*****
* 函 数：写单字节到 DS1302
* 参 数：dat 要写的数据
*****/
void write_byte(uchar dat)
{
    uchar i = 0;

    for(i=0; i<8; i++)
    {
        DS_IO = (dat&1);
        DS_SCLK = 0;
        _nop_(),_nop_();
        DS_SCLK = 1;    //上升沿发出数据
        _nop_(),_nop_();
        dat >>= 1;
    }
}

/*****
* 函 数：从 DS1302 读单字节
* 参 数：空
* 返回值：返回读取的字节（10 进制）
*****/
uchar read_byte(void)
{
    uchar i = 0;
    uchar dat = 0;
    uchar tmp = 0;

    for(i=0; i<8; i++)
    {
        DS_SCLK = 1;
        _nop_(),_nop_();
        DS_SCLK = 0;    //下降沿读出数据
        _nop_(),_nop_();
        tmp = DS_IO;
        dat >>= 1;        //先读的是低位，移位为低位
        dat |= (tmp<<7);
    }
}
```

```
    dat = BCD2DEC(dat);    //BCD 转换
    return(dat);
}

/*****
* 函 数：从 DS1302 的指定位置读数据
* 参 数：addr 要读取数据的控制字（地址/命令）
* 返回值：返回读取的数字（10 进制）
*****/

uchar read_ds1302(uchar addr)
{
    uchar tmp;

    DS_RST = 0;
    DS_SCLK = 0;
    DS_RST = 1;
    write_byte(addr);
    tmp = read_byte();
    DS_RST = 0;
    DS_SCLK = 1;

    return(tmp);
}

/*****
* 函 数：写数据到 DS1302 的指定位置
* 参 数：addr 要写入数据的控制字（地址/命令）
*          dat 要写的数字
*****/

void write_ds1302(uchar addr, uchar dat)
{
    uchar tmp = 0;
    tmp = DEC2BCD(dat);

    DS_RST = 0;
    DS_SCLK = 0;
    DS_RST = 1;
    write_byte(addr);
    write_byte(tmp);
    DS_RST = 0;
    DS_SCLK = 1;
}
```

```

/*****

```

```

* 函 数：读取年、月、日、时、分、秒、星期

```

```

* 参 数：无

```

```

* 返回值：str_time 读取到的时间

```

```

*****/

```

```

SYSTEM_TIME read_time(void)

```

```

{

```

```

    SYSTEM_TIME str_time;

```

```

    str_time.year = read_ds1302(DS1302_YEAR|0x01);

```

```

    str_time.month = read_ds1302(DS1302_MONTH|0x01);

```

```

    str_time.date = read_ds1302(DS1302_DATE|0x01);

```

```

    str_time.hour = read_ds1302(DS1302_HOUR|0x01);

```

```

    str_time.min = read_ds1302(DS1302_MIN|0x01);

```

```

    str_time.sec = read_ds1302(DS1302_SEC|0x01);

```

```

    str_time.day = read_ds1302(DS1302_DAY|0x01);

```

```

    return(str_time);

```

```

}

```

```

/*****

```

```

* 函 数：设置 DS1302 的年、月、日、时、分、秒、星期

```

```

* 参 数：str_time 要设定的时间

```

```

*****/

```

```

void set_time(SYSTEM_TIME str_time)

```

```

{

```

```

    write_ds1302(DS1302_YEAR, str_time.year);

```

```

    write_ds1302(DS1302_MONTH, str_time.month);

```

```

    write_ds1302(DS1302_DATE, str_time.date);

```

```

    write_ds1302(DS1302_HOUR, str_time.hour);

```

```

    write_ds1302(DS1302_MIN, str_time.min);

```

```

    write_ds1302(DS1302_SEC, str_time.sec);

```

```

    write_ds1302(DS1302_DAY, str_time.day);

```

```

}

```

```

/*****

```

```

* 文件名：ds1302.h

```

```

*****/

```

```

#ifndef _DS1302_H

```

```

#define _DS1302_H

```

```

#include "main.h"

```

```

//DS1302 控制字（地址/命令）

```

```

#define DS1302_SEC      0x80
#define DS1302_MIN      0x82
#define DS1302_HOUR     0x84
#define DS1302_DATE     0x86
#define DS1302_MONTH    0x88
#define DS1302_DAY      0x8A
#define DS1302_YEAR     0x8C
#define DS1302_WRITE    0x8E
#define DS1302_POWER    0x90

//DS1302 写保护
#define DS1302_ENABLE    0x00    //写使能
#define DS1302_DISABLE   0x80    //写保护

void ds1302_init(void);
void write_ds1302(uchar addr, uchar dat);
uchar read_ds1302(uchar addr);

SYSTEM_TIME read_time(void);
void set_time(SYSTEM_TIME str_time);

#endif

```

```

/*****
* 文件名: iic.c
* 说 明: 操作 IIC 串行通信芯片的基本函数
*****/

```

```
#include "main.h"
```

```

sbit SDA = P1^1;    //SDA 管脚
sbit SCL = P1^0;    //SCL 管脚

```

```

/*****

```

```
* 函 数: 短暂延时函数, 大概 12um
```

```
* 参 数: 空
```

```
* 返回值: 无
```

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

```
void delay(void)
```

```

{
    _nop();_nop();
    _nop();_nop();
    _nop();_nop();
    // _nop();_nop();
}

```

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

```
* 函 数：初始化 IIC
```

```
* 参 数：空
```

```
* 返回值：无
```

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

```
void init_IIC(void)
```

```
{
```

```
    SDA = 1;
```

```
    delay();
```

```
    SCL = 1;
```

```
    delay();
```

```
}
```

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

```
* 函 数：IIC 起始信号
```

```
* 参 数：空
```

```
* 返回值：无
```

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

```
void start(void)
```

```
{
```

```
    SDA = 1;
```

```
    delay();
```

```
    SCL = 1;
```

```
    delay();
```

```
    SDA = 0;
```

```
    delay();
```

```
}
```

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

```
* 函 数：IIC 终止信号
```

```
* 参 数：空
```

```
* 返回值：无
```

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

```
void stop(void)
```

```
{
```

```
    SDA = 0;
```

```
    delay();
```

```
    SCL = 1;
```

```
    delay();
```

```
    SDA = 1;
```

```
    delay();
```

```
}
```

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

```
* 函 数: IIC ACK 应答信号
```

```
* 参 数: 空
```

```
* 返回值: 无
```

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

```
void ack(void)
```

```
{
```

```
    uchar i=0;
```

```
    SCL = 1;
```

```
    delay();
```

```
    while((SDA==1)&&(i<250))
```

```
    {
```

```
        i++;
```

```
    }
```

```
    SCL = 0;
```

```
    delay();
```

```
}
```

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

```
* 函 数: IIC NOACK 应答信号
```

```
* 参 数: 空
```

```
* 返回值: 无
```

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

```
void noack(void)
```

```
{
```

```
    SDA = 1;
```

```
    delay();
```

```
    SCL = 1;
```

```
    delay();
```

```
    SCL = 0;
```

```
    delay();
```

```
}
```

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

```
* 函 数: IIC 写数据
```

```
* 参 数: dat 要写的的数据
```

```
* 返回值: 无
```

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

```
void IIC_write_byte(uchar dat)
```

```
{
```

```
    uchar i,tmp;
```

```
    tmp = dat;
```

```
    for(i=0;i<8;i++)
```

```
    {
```



```
        SCL = 0;
        delay();
        if(tmp&0x80)
        {
            SDA = 1;
        }
        else
        {
            SDA = 0;
        }
        tmp = tmp<<1;
        delay();
        SCL = 1;
        delay();
    }
    SCL = 0;
    delay();
    SDA = 1;
    delay();
}
```

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

```
* 函 数: IIC 读数据
```

```
* 参 数: 空
```

```
* 返回值: dat 读取到的数据
```

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

```
uchar IIC_read_byte(void)
```

```
{
    uchar i,dat;
    SCL = 0;
    delay();
    SDA = 1;
    delay();
    for(i=0;i<8;i++)
    {
        SCL = 1;
        delay();
        dat = dat<<1;
        if(SDA)
        {
            dat++;
        }
        SCL = 0;
        delay();
    }
}
```

```
    }
    return dat;
}

/*****
* 函 数：指定地址写数据
* 参 数：addr 指定的地址，dat 要写的数据
* 返回值：无
*****/
void write_IIC(uchar addr, uchar dat)
{
    start();
    IIC_write_byte(0xa0);
    ack();
    IIC_write_byte(addr);
    ack();
    IIC_write_byte(dat);
    ack();
    stop();
}

/*****
* 函 数：指定地址读数据
* 参 数：addr 指定的地址
* 返回值：temp 读取的数据
*****/
uchar read_IIC(uchar add)
{
    uchar temp;
    start();
    IIC_write_byte(0xa0);
    ack();
    IIC_write_byte(add);
    ack();
    start();
    IIC_write_byte(0xa1);
    ack();
    temp = IIC_read_byte();
    noack();
    stop();
    return temp;
}

/*****/
```

```
* 文件名: iic.h
```

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

```
#ifndef _IIC_H
```

```
#define _IIC_H
```

```
#include "main.h"
```

```
void init_IIC(void);
```

```
void write_IIC(uchar add,uchar dat);
```

```
uchar read_IIC(uchar add);
```

```
#endif
```

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

```
* 文件名: serial.c
```

```
* 说 明: 串口通信有关的函数
```

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

```
#include "main.h"
```

```
uchar g_pSendBuff[13];          //发送缓存
```

```
uchar g_pRecvBuff[14];         //接收缓存
```

```
uchar g_ucRecvNum;             //接收字节数
```

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

```
* 函 数: 定时器 1 初始化
```

```
* 参 数: 空
```

```
* 返回值: 无
```

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

```
void timer1_int(void)
```

```
{
```

```
    TMOD |= 0x20;          //定时器 T1, 工作方式 2
```

```
    TL1 = 0xFD;            //定时器 T1 初始值 253 (0xFD)
```

```
    TH1 = 0xFD;
```

```
    TR1 = 1;               //定时器 T1 启动工作 (TCON | 0x40)
```

```
}
```

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

```
* 函 数: 串口初始化, 设定串口工作方式、中断
```

```
* 参 数: 空
```

```
* 返回值: 无
```

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

```
void serial_int(void)
```

```
{
```

```
    PCON = 0x80;          //设定 SMOD 位为 1
```

```
    SCON = 0x51;        //设定 SM0=0,SM1=1,串口工作方式 1, REN=1 接收数据使能
    ES = 1;              //使能串口中断
}

/*****
* 函 数: 串口中断函数
* 参 数: 空
* 返回值: 无
*****/
void serial_inter() interrupt 4
{
    static uchar sSendNum = 0;
    uchar tmp = 0;

    //串口接收中断
    if(RI == 1)
    {
        RI = 0;
        tmp = SBUF;
        if(g_ucRecvNum==0 && tmp == 0xEB)
        {
            g_pRecvBuff[0] = tmp;
            g_ucRecvNum ++;
        }else
        {
            if(g_ucRecvNum > 0)
            {
                g_pRecvBuff[g_ucRecvNum] = tmp;
                g_ucRecvNum ++;
            }
        }
    }

    //串口发送中断
    if(TI == 1)
    {
        TI = 0;
        sSendNum ++;
        if(sSendNum < g_pSendBuff[1])
        {
            SBUF = g_pSendBuff[sSendNum];
        }else
        {
            sSendNum = 0;
        }
    }
}
```

```
    }
}
}

/*****
* 函 数：通信协议处理函数
* 参 数：空
* 返回值：无
*****/

void deal_protocol(void)
{
    uchar chk_sum = 0;
    uchar len = 0;
    uchar cmd = 0;
    uchar i = 0;

    if(g_ucRecvNum >= 6)          //协议长度最短是 6 字节
    {
        len = g_pRecvBuff[1];    //数据长度
        if(g_ucRecvNum < len)
        {
            return;
        }
        g_ucRecvNum = 0;          //已经接收完整一帧数据

        if(g_pRecvBuff[len-1] != 0x90) //检验协议尾
        {
            return;
        }

        //检验校验和
        for(i=1; i<len-2; i++)
        {
            chk_sum += g_pRecvBuff[i];
        }
        if(chk_sum != g_pRecvBuff[len-2])
        {
            return;
        }

        if(g_pRecvBuff[3] == 0x01) //控制命令
        {
            cmd = g_pRecvBuff[2];
        }
    }
}
```

```
switch(cmd)
{
case 2:          //温度上下限
    set_temper_updown(g_pRecvBuff[4], g_pRecvBuff[5]);//
    break;
case 3:          //时间
    set_system_time((uchar*)(g_pRecvBuff+4));
    break;
case 4:          //风扇
    open_moto(g_pRecvBuff[4]);
    break;
case 5:          //加热灯
    open_heat(g_pRecvBuff[4]);
    break;
default :
    break;
}
g_pSendBuff[1] = 0x07;
g_pSendBuff[2] = cmd;
g_pSendBuff[3] = 0x01;    //控制
g_pSendBuff[4] = 0x00;    //成功
}else //查询命令字
{
    cmd = g_pRecvBuff[2];
    switch(cmd)
    {
case 1:          //温度
        get_temper((uchar*)(g_pSendBuff+5));
        g_pSendBuff[1] = 9;
        break;
case 2:          //温度上下限
        get_temper_updown((uchar*)(g_pSendBuff+5),(uchar*)(g_pSendBuff+6));//
        g_pSendBuff[1] = 9;
        break;
case 3:          //时间
        get_system_time((uchar*)(g_pSendBuff+5));
        g_pSendBuff[1] = 14;
        break;
case 4:          //风扇
        get_moto_stat((uchar*)(g_pSendBuff+5));
        g_pSendBuff[1] = 8;
        break;
case 5:          //加热灯
        get_heat_stat((uchar*)(g_pSendBuff+5));
```

```

        g_pSendBuff[1] = 8;
        break;
    default :
        break;
    }
    g_pSendBuff[2] = cmd;
    g_pSendBuff[3] = 0x00;    //读取
    g_pSendBuff[4] = 0x00;    //成功
}

//应答数据
g_pSendBuff[0] = 0xEB;
//g_pSendBuff[1] = 0x06;
len = g_pSendBuff[1];
chk_sum = 0;
for(i=1; i<len-2; i++)
{
    chk_sum += g_pSendBuff[i];
}
g_pSendBuff[len-2] = chk_sum;
g_pSendBuff[len-1] = 0x90;
SBUF = g_pSendBuff[0];
}
}

```

```

/*****

```

```

* 文件名: serial.h

```

```

*****/

```

```

#ifndef _SERIAL_H

```

```

#define _SERIAL_H

```

```

void serial_int(void);

```

```

void timer1_int(void);

```

```

void deal_protocol(void);

```

```

#endif

```