# **LCD12864**

## LCD12864.C

#include"lcd12864.h"

extern uchar code CharCodeLine1[]; //第一行显示字符

extern uchar code CharCodeLine2[]; //第二行显示字符

void LCD12864\_Roll(void){ //卷动显示两行字

uchar i;

//为显示上半屏第一行字符做准备，地址0xa0

//详细参考文章：https://wenku.baidu.com/view/375ec764cc22bcd127ff0c9a.html

LCD12864\_WriteCmd(0xa0);

while(CharCodeLine1[i]!='\0'){

LCD12864\_WriteData(CharCodeLine1[i]);

i++;

}

i=0;

//为显示上半屏第二行字符做准备，地址0xb0

LCD12864\_WriteCmd(0xb0);

while(CharCodeLine2[i]!='\0')

{

LCD12864\_WriteData(CharCodeLine2[i]);

i++;

}

for(i=0;i<33;i++) //上半屏卷动显示

{

LCD12864\_VerticalRoll(i);

Delay100ms(); //每行高16个像素，两行32像素，0.1秒卷动1像素，两行字显示结束共需要3.2秒

}

LCD12864\_WriteCmd(0x30); //恢复基本指令集

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_Delay1ms

\* 函数功能 : 延时1MS

\* 输 入 : c

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_Delay1ms(uint c)

{

uchar a,b;

for(; c>0; c--)

{

for(b=199; b>0; b--)

{

for(a=1; a>0; a--);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_WriteCmd

\* 函数功能 : 写命令

\* 输 入 : cmd

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_WriteCmd(uchar cmd)

{

uchar i;

i = 0;

LCD12864\_Delay1ms(20); //等待忙完

LCD12864\_RS = 0; //选择命令

LCD12864\_RW = 0; //选择写入

LCD12864\_EN = 0; //初始化使能端

LCD12864\_DATAPORT = cmd; //放置数据

LCD12864\_EN = 1; //写时序

LCD12864\_Delay1ms(5);

LCD12864\_EN = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_WriteData

\* 函数功能 : 写数据

\* 输 入 : dat

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_WriteData(uchar dat)

{

uchar i;

i = 0;

LCD12864\_Delay1ms(20); //等待忙完

LCD12864\_RS = 1; //选择数据

LCD12864\_RW = 0; //选择写入

LCD12864\_EN = 0; //初始化使能端

LCD12864\_DATAPORT = dat; //放置数据

LCD12864\_EN = 1; //写时序

LCD12864\_Delay1ms(5);

LCD12864\_EN = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_ReadData

\* 函数功能 : 读取数据

\* 输 入 : 无

\* 输 出 : 读取到的8位数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar LCD12864\_ReadData(void)

{

uchar i, readValue;

i = 0;

LCD12864\_Delay1ms(20); //等待忙完

LCD12864\_RS = 1; //选择命令

LCD12864\_RW = 1;

LCD12864\_EN = 0;

LCD12864\_Delay1ms(1); //等待

LCD12864\_EN = 1;

LCD12864\_Delay1ms(1);

readValue = LCD12864\_DATAPORT;

LCD12864\_EN = 0;

return readValue;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_Init

\* 函数功能 : 初始化LCD12864

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_Init()

{

LCD12864\_WriteCmd(0x30); //选择基本指令操作

LCD12864\_WriteCmd(0x0c); //显示开，关光标

LCD12864\_WriteCmd(0x01); //清除LCD12864的显示内容

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_ClearScreen

\* 函数功能 : 在画图模式下，LCD12864的01H命令不能清屏，所以要自己写一个清

\* \* 屏函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_ClearScreen(void)

{

uchar i,j;

LCD12864\_WriteCmd(0x34); //开启拓展指令集

for(i=0;i<32;i++) //因为LCD有纵坐标32格所以写三十二次

{

LCD12864\_WriteCmd(0x80+i); //先写入纵坐标Y的值

LCD12864\_WriteCmd(0x80); //再写入横坐标X的值

for(j=0;j<32;j++) //横坐标有16位，每位写入两个字节的的数据，也

{ //就写入32次以为当写入两个字节之后横坐标会自

LCD12864\_WriteData(0x00); //动加1，所以就不用再次写入地址了。

}

}

LCD12864\_WriteCmd(0x36); //0x36扩展指令里面打开绘图显示

LCD12864\_WriteCmd(0x30); //恢复基本指令集

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LCD12864\_SetWindow

\* 函数功能 : 设置在基本指令模式下设置显示坐标。注意：x是设置行，y是设置列

\* 输 入 : x, y

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_SetWindow(uchar x, uchar y)

{

uchar pos;

if(x == 0) // 第一行的地址是80H

{

x = 0x80;

}

else if(x == 1) //第二行的地址是90H

{

x = 0x90;

}

else if(x == 2) //第三行的地址是88H

{

x = 0x88;

}

else if(x == 3)

{

x = 0x98;

}

pos = x + y;

LCD12864\_WriteCmd(pos);

}

/\*-----------------------------------------------------------------------------------------

函数原型：void LCD12864\_VerticalRoll(uchar N)

函数功能：将DDRAM内容垂直卷动N个像素的距离

入口参数：uchar N\_Pixel：卷动的距离大小（单位为像素），范围为0~33（实际上可以是0~63，但33~

63实际意义不大）

出口参数：无

返回参数：无

注意事项：将N\_Pixel设为33可将DDRAM地址0xa0~0xbf的内容完全切换到屏幕上，N为0则显示DDRAM地址

0x80~0x9f的内容（复位默认状态）

-----------------------------------------------------------------------------------------\*/

void LCD12864\_VerticalRoll(uchar N\_Pixel)

{

LCD12864\_WriteCmd(0x34);//允许绘图模式（开启扩展指令集模式）

LCD12864\_WriteCmd(0x03);//允许输入卷动位址

LCD12864\_WriteCmd(0x40|N\_Pixel);//上卷N行（像素）

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数功能 ：向液晶连续写入一段字符串

\*\* 函数说明 ：字符可以是任何字符，包括汉字，但是汉字必须是写在一个连续的16\*16的点阵中

\*\* 函数举例 ：Write\_12864\_String("LCD12864液晶实验")，这段字符串有8个英文字符，总共占4个16\*16的点阵，后面的四个同样占4个16\*16的点阵

\*\* 错误举例 ：Write\_12864\_String("LCD液晶显示"),前面的三个字符占了一个半的16\*16单元的点阵，会导致后面的汉字没法正常显示

\*\* 入口参数 ：待写入的字符串

\*\* 出口参数 ：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void print(uchar \*str)//写入字符串或者汉字

{

uchar \*p;

p = str;

while(\*p != 0)

{

LCD12864\_WriteData(\*p);

p = ++str;

}

}

//定位输出数字

//x: 0 - 3 (行)

//y: 0 - 15 (列)

//num: 0 - 65535 要显示的数字

//num\_bit: 0 - 5 数字的位数

void print\_n(unsigned char x, unsigned char y,unsigned int num, unsigned char num\_bit)

{

char i;

unsigned char ii;

unsigned char dat[6];

LCD12864\_SetWindow(x,y);

for(i = 0; i < 6; i++) dat[i] = 0; i = 0; //初始化数据

while(num / 10) //拆位

{

dat[i] = num % 10; //最低位

num /= 10; i++;

}

dat[i] = num; //最高位

ii = i; //保存dat的位数

for(; i >= 0; i--) dat[i] += 48; //转化成ASCII

for(i = 0; i < num\_bit; i++)

LCD12864\_WriteData(0x20); //清显示区域

LCD12864\_SetWindow(x,y);

for(i = ii; i >= 0; i--)

LCD12864\_WriteData(dat[i]); //输出数值

}

## LCD12864.H

#ifndef \_\_LCD12864\_H

#define \_\_LCD12864\_H

//---包含头文件---//

#include "stc15f2k60s2.h"

#include "sys.h"

//---重定义关键词---//

#ifndef uchar

#define uchar unsigned char

#endif

#ifndef uint

#define uint unsigned int

#endif

//---定义使用的IO口---//

#define LCD12864\_DATAPORT P0 //数据IO口

sbit LCD12864\_RS = P2^0; //（数据命令）寄存器选择输入 TFT\_D10 A8

sbit LCD12864\_RW = P2^1; //液晶读/写控制 TFT\_D8 A9

sbit LCD12864\_EN = P2^2; //液晶使能控制 TFT\_D9 A10

//---声明全局函数---//

void LCD12864\_Delay1ms(uint c);

void LCD12864\_WriteCmd(uchar cmd);

void LCD12864\_WriteData(uchar dat);

void LCD12864\_Init();

void LCD12864\_ClearScreen(void);

void LCD12864\_SetWindow(uchar x, uchar y);

void LCD12864\_VerticalRoll(uchar N\_Pixel); //卷动显示

void LCD12864\_Roll(void);

void LCD12864\_pixel(unsigned char x, unsigned char y, unsigned char attr);

void LCD\_line\_h(unsigned char y, unsigned char attr);

void print(uchar \*str);

void print\_n(unsigned char x, unsigned char y,unsigned int num, unsigned char num\_bit);

#endif

# **DS**1302

## DS1302.C

//DS1302部分

//整体 显示 时间 日期 年月 星期

//可不停更新秒时间

#include "stc15f2k60s2.h"

#include "ds1302.h"

#include "intrins.h"

#include "sys.h"

#include "lcd12864.h"

code u8 write\_rtc\_address[7]={0x80,0x82,0x84,0x86,0x88,0x8a,0x8c}; //秒分时日月周年 最低位读写位

code u8 read\_rtc\_address[7]={0x81,0x83,0x85,0x87,0x89,0x8b,0x8d};

u8 ds1302tmp[7]={50,03,02,20,5,6,23};//秒分时日月周年 2020年 4月 20日 8点20分 星期日 这个数是自己设定的

//写一个字节

//temp 要写入的字节 （地址或数据）

void Write\_Ds1302\_Byte(u8 temp)

{

u8 i;

for (i=0;i<8;i++) //循环8次 写入数据

{

SCK=0;

SDA=temp&0x01; //每次传输低字节

temp>>=1; //右移一位

SCK=1;

}

}

//写入DS1302数据

//address 写入的地址

//dat 写入的数据

void Write\_Ds1302( u8 address,u8 dat )

{

CE=0;

\_nop\_();

SCK=0;

\_nop\_();

CE=1; //启动

\_nop\_();

Write\_Ds1302\_Byte(address); //发送地址

Write\_Ds1302\_Byte(dat); //发送数据

CE=0; //恢复

}

//读出DS1302数据

//address 读取数据的地址

//返回 读取的数据值

u8 Read\_Ds1302 ( u8 address )

{

u8 i,temp=0x00;

CE=0;

\_nop\_();

\_nop\_();

SCK=0;

\_nop\_();

\_nop\_();

CE=1;

\_nop\_();

\_nop\_();

Write\_Ds1302\_Byte(address);

for (i=0;i<8;i++) //循环8次 读取数据

{

if(SDA)

temp|=0x80; //每次传输低字节

SCK=0;

temp>>=1; //右移一位

\_nop\_();

\_nop\_();

\_nop\_();

SCK=1;

}

CE=0;

\_nop\_(); //以下为DS1302复位的稳定时间

\_nop\_();

CE=0;

SCK=0;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

SCK=1;

\_nop\_();

\_nop\_();

SDA=0;

\_nop\_();

\_nop\_();

SDA=1;

\_nop\_();

\_nop\_();

return (temp); //返回

}

//写入时 DCB转换

//DS1302 只接收 DCB码 即高4位放10位值 低4位放个位值

//add 要转换的地址

//返回 转换后的数据

u8 bcd\_read(u8 add)

{

u8 fla,fla2;

fla=Read\_Ds1302(add);

fla2=((fla/16)\*10)+(fla&0x0f);

return fla2;

}

//设定时钟数据

//主要设置时钟芯片里的 秒分时日月周年

void Set\_RTC(void) //设定 日历

{

u8 i;

for(i=0;i<7;i++) //BCD处理

ds1302tmp[i]=ds1302tmp[i]/10\*16+ds1302tmp[i]%10; //ds1302tmp[7]={0,20,8,20,4,7,20};

Write\_Ds1302(0x8E,0X00); //写使能

for(i=0;i<7;i++) //7次写入 秒分时日月周年

Write\_Ds1302(write\_rtc\_address[i],ds1302tmp[i]);

Write\_Ds1302(0x8E,0x80); //写禁止

}

//DS1302 时间 显示函数

//此函数需要循环 进行更新 不能自动更新

//例20:25

void ds1302\_scan(void)

{

u8 temp;

temp = bcd\_read(0x85);//时

LCD12864\_SetWindow(3,0); //第4行 显示

LCD12864\_WriteData(temp/10%10+0x30);

LCD12864\_WriteData(temp%10+0x30);

LCD12864\_WriteData(':');

temp = bcd\_read(0x83);//分

LCD12864\_WriteData(temp/10%10+0x30);

LCD12864\_WriteData(temp%10+0x30);

temp = bcd\_read(0x81);//秒

LCD12864\_WriteData(':');

LCD12864\_WriteData(temp/10%10+0x30);

LCD12864\_WriteData(temp%10+0x30);

print("你好！");

}

## DS1302.H

#ifndef \_\_DS1302\_H

#define \_\_DS1302\_H

#include "sys.h"

sbit SCK=P3^5; //ʱ��

sbit SDA=P3^6; //����

sbit CE=P5^4; //��λ

//void Set\_RTC(void); //set RTC

void ds1302\_scan(void); //ʱ����ʾ

void Write\_Ds1302(u8 address,u8 dat);

u8 bcd\_read(u8 add);

u8 Read\_Ds1302 ( u8 address );

void Set\_RTC();

#endif

# KEY

## KEY.c

#include "key.h"

#include "sys.h"

u8 KeyCode=0; //给用户使用的键码, 17~32有效

u8 IO\_KeyState=0;

u8 IO\_KeyState1=0;

u8 IO\_KeyHoldCnt=0; //行列键盘变量

u8 code T\_KeyTable[16] = {0,1,2,0,3,0,0,0,4,0,0,0,0,0,0,0};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

行列键扫描程序

使用XY查找4x4键的方法，只能单键，速度快

Y P74 P75 P76 P77

| | | |

X | | | |

P70 ---- K00 ---- K01 ---- K02 ---- K03 ----

| | | |

P71 ---- K04 ---- K05 ---- K06 ---- K07 ----

| | | |

P72 ---- K08 ---- K09 ---- K10 ---- K11 ----

| | | |

P73 ---- K12 ---- K13 ---- K14 ---- K15 ----

| | | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IO\_KeyDelay(void)

{

u8 i;

i = 60;

while(--i) ;

}

void IO\_KeyScan(void)

{

u8 j;

j = IO\_KeyState1; //保存上一次状态

P7 = 0xf0; //X低，读Y

IO\_KeyDelay();

IO\_KeyState1 = P7 & 0xf0;

P7 = 0x0f; //Y低，读X

IO\_KeyDelay();

IO\_KeyState1 |= (P7 & 0x0f);

IO\_KeyState1 ^= 0xff; //取反

if(j == IO\_KeyState1) //连续两次读相等

{

j = IO\_KeyState;

IO\_KeyState = IO\_KeyState1;

if(IO\_KeyState != 0) //有键按下

{

F0 = 0;

if(j == 0) F0 = 1; //第一次按下

else if(j == IO\_KeyState)

{

if(++IO\_KeyHoldCnt >= 20) //1秒后重键

{

IO\_KeyHoldCnt = 18;

F0 = 1;

}

}

if(F0)

{

j = T\_KeyTable[IO\_KeyState >> 4];

if((j != 0) && (T\_KeyTable[IO\_KeyState& 0x0f] != 0))

{

KeyCode = (u8)(((j - 1) \* 4 )+ (T\_KeyTable[IO\_KeyState & 0x0f]) ); //计算键码，17~32 + 16

}

}

}

else IO\_KeyHoldCnt = 0;

}

P7 = 0xff;

}

## KEY.H

#ifndef \_\_KEY\_H

#define \_\_KEY\_H

#include "stc15f2k60s2.h"

void IO\_KeyDelay(void);

void IO\_KeyScan(void);

#endif

# DHT11

## DHT11.C

#include "stc15f2k60s2.h"

#include <stdio.h>

#include "string.h"

#include "sys.h"

sbit DHT\_IO = P1^1 ;

u8 U8FLAG,k;

u8 U8count,U8temp;

u8 U8T\_data\_H,U8T\_data\_L,U8RH\_data\_H,U8RH\_data\_L,U8checkdata;

u8 U8T\_data\_H\_temp,U8T\_data\_L\_temp,U8RH\_data\_H\_temp,U8RH\_data\_L\_temp,U8checkdata\_temp;

u8 U8comdata;

u8 outdata[5]; //定义发送的字节数

u8 indata[5];

u8 count, count\_r=0;

u8 str[5]={"RS232"};

u16 U16temp1,U16temp2;

void Delay\_10us(void)

{

delay\_us(10);

}

void COM(void)

{

u8 i;

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

{

U8FLAG=2;

while((!DHT\_IO)&&U8FLAG++);

Delay\_10us();

Delay\_10us();

Delay\_10us();

U8temp=0;

if(DHT\_IO)U8temp=1;

U8FLAG=2;

while((DHT\_IO)&&U8FLAG++);

//超时则跳出for循环

if(U8FLAG==1)break;

//判断数据位是0还是1

// 如果高电平高过预定0高电平值则数据位为 1

U8comdata<<=1;

U8comdata|=U8temp; //0

}//rof

}

//--------------------------------

//-----湿度读取子程序 ------------

//--------------------------------

//----以下变量均为全局变量--------

//----温度高8位== U8T\_data\_H------

//----温度低8位== U8T\_data\_L------

//----湿度高8位== U8RH\_data\_H-----

//----湿度低8位== U8RH\_data\_L-----

//----校验 8位 == U8checkdata-----

//----调用相关子程序如下----------

//---- delay1ms();, Delay\_10us();,COM();

//--------------------------------

void UpdateTemp(void)

{

//主机拉低18ms

DHT\_IO=0;

delay1ms(18);

DHT\_IO=1;

//总线由上拉电阻拉高 主机延时20us

Delay\_10us();

Delay\_10us();

// Delay\_10us();

// Delay\_10us();

//主机设为输入 判断从机响应信号

DHT\_IO=1;

//判断从机是否有低电平响应信号 如不响应则跳出，响应则向下运行

if(!DHT\_IO) //T !

{

U8FLAG=2;

//判断从机是否发出 80us 的低电平响应信号是否结束

while((!DHT\_IO)&&U8FLAG++);

U8FLAG=2;

//判断从机是否发出 80us 的高电平，如发出则进入数据接收状态

while((DHT\_IO)&&U8FLAG++);

//数据接收状态

COM();

U8RH\_data\_H\_temp=U8comdata;

COM();

U8RH\_data\_L\_temp=U8comdata;

COM();

U8T\_data\_H\_temp=U8comdata;

COM();

U8T\_data\_L\_temp=U8comdata;

COM();

U8checkdata\_temp=U8comdata;

DHT\_IO=1;

//数据校验

U8temp=(U8T\_data\_H\_temp+U8T\_data\_L\_temp+U8RH\_data\_H\_temp+U8RH\_data\_L\_temp);

if(U8temp==U8checkdata\_temp)

{

U8RH\_data\_H=U8RH\_data\_H\_temp;

U8RH\_data\_L=U8RH\_data\_L\_temp;

U8T\_data\_H=U8T\_data\_H\_temp;

U8T\_data\_L=U8T\_data\_L\_temp;

U8checkdata=U8checkdata\_temp;

}//fi

}//fi

}

## DHT11.H

#ifndef DHT11\_H

#define DHT11\_H

#include "sys.h"

#include "stc15f2k60s2.h"

extern u8 U8T\_data\_H,U8T\_data\_L,U8RH\_data\_H,U8RH\_data\_L,U8checkdata;

void COM(void);

void UpdateTemp(void);

#endif

# AD

## AD.C

#include "stc15f2k60s2.h"

#include "sys.h"

#include "intrins.h"

#include "ad.h"

/\*Define ADC operation const for ADC\_CONTR\*/

#define ADC\_POWER 0x80 //ADC power control bit

#define ADC\_FLAG 0x10 //ADC complete flag 模数转换结束标志位

#define ADC\_START 0x08 //ADC start control bit 模数转换启动控制位

//转换速度控制位SPEED0和SPEED1，共四种状态，对应四种转换速度

#define ADC\_SPEEDLL 0x00 //540 clocks

#define ADC\_SPEEDL 0x20 //360 clocks

#define ADC\_SPEEDH 0x40 //180 clocks

#define ADC\_SPEEDHH 0x60 //90 clocks

/\*----------------------------

Get ADC result

----------------------------\*/

u16 GetADCResult(u8 ch) //

{

u16 Vo;

P1ASF = 0x01; //选择P1口的哪一口 这里的口和ch要对应才能达到选择该口

ADC\_CONTR = ADC\_POWER | ADC\_SPEEDLL | ch | ADC\_START;//0x00|0x00|ch|0x08:选择A/D输入通道，开始A/D转换

// 这么用语句的主要原因就是不能位寻址

// 通道选择在后3位所以直接用一个整数表示ch

//例如ch=6 那么对应的后三位就是110

\_nop\_(); //Must wait before inquiry ，

\_nop\_(); //设置ADC\_CONTR寄存器后需加4个CPU时钟周期的延时，才能保证值被写入ADC\_CONTR寄存器

\_nop\_();

\_nop\_();

while (!(ADC\_CONTR & ADC\_FLAG)); //Wait complete flag

ADC\_CONTR &= ~ADC\_FLAG; //Close ADC 将标志位清零等待下次硬件置1

//也可以写成 ADC\_CONTR= ADC\_CONTR & ( ~ADC\_FLAG)

Vo=(ADC\_RES<<2)+ADC\_RESL; //打开10位AD采集功能 如果用8位AD Vo=ADC\_RESL 即可

//10位AD采集 即2的10次方 满值为1024 这里用1024表示5伏的电压

//8位AD采集 即 2的8次方 满值为256 用256表示5伏

return Vo;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## AD.H

#ifndef AD\_H

#define AD\_H

#include "sys.h"

u16 GetADCResult(u8 ch); //读取P1.0口AD值

#endif

# TIMER

## TIMER.C

#include "sys.h"

#include "timer.h"

#include "stc15f2k60s2.h"

#include "ad.h"

//定制器和步进电机控制代码

u16 TimerCount = 0;

u8 PulseNum = 0; //脉冲计数

u8 MoterCount = 0; //步进电机控制计数

u16 AdcCount = 0 ; //adc采样计数

u16 WaiteTimeCount = 0; //静止状态等待时间计数

extern u8 MoterSpeed; //定制器2控制步进电机转速

extern u8 AdcFlag ; //adc采样标志

extern u16 ad; //ADC采样值

// 步进电机逆时针旋转相序表

u8 code CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};

// 步进电机四相的引脚定义

sbit A1 = P1^1;

sbit B1 = P1^2;

sbit C1 = P1^3;

sbit D1 = P1^4;

// 给步进电机每个相赋值

void Give(unsigned char dat)

{

A1 = dat & 0x08;

B1 = dat & 0x04;

C1 = dat & 0x02;

D1 = dat & 0x01;

}

//定时器0初始化

void Timer0Init(void)//10毫秒@24.000MHz

{

AUXR &= 0x7F; //定时器时钟12T模式

TMOD &= 0xF0; //设置定时器模式

TL0 = 0xE0; //设置定时初值

TH0 = 0xB1; //设置定时初值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

}

//定时器2初始化

void Timer2Init(void)//250微秒@24.000MHz

{

AUXR &= 0xFB; //定时器时钟12T模式

T2L = 0x0C; //设置定时初值

T2H = 0xFE; //设置定时初值

AUXR |= 0x10; //定时器2开始计时

}

void tm0() interrupt 1 using 1 //定时器0中断

{

WaiteTimeCount ++ ; //10毫秒加一

}

void tm2() interrupt 12 //定时器2中断

{

#if DRIVING\_SIMULATION

TimerCount ++ ; //定时器2脉冲模拟效果

if(TimerCount>4000) //定时超过 1秒

{

PulseNum ++ ;

TimerCount = 0;

}

#else

TimerCount ++ ;

if(TimerCount >MoterSpeed) //定时超过 MoterSpeed为步进电机速度控制

{

MoterCount ++ ;

TimerCount = 0;

Give(CCW[MoterCount%8]); //控制电机

}

if( MoterCount%64==63 ) //定时ADC采样标志位

{

MoterCount = 0;

AdcFlag = 1;

}

#endif

}

## TIMER.H

#ifndef \_\_TIMER\_H

#define \_\_TIMER\_H

void Give(unsigned char dat); //�����������

void Timer2Init(void);

void Timer0Init(void);

void tm2();

void tm0();

#endif

# SYSTEM\SYS

## SYS.C

#include "sys.h"

/\*

24MHz主频下

在示波器下看

一个语句的时间大概是 0.8us

\*/

//2微秒 延时

//此函数是以2us的倍数增加

//t 2us的倍数

void delay2us(u16 us)

{

u16 i;

u8 m;

for(i=0;i<us;i++)

for(m=0;m<3;m++);

}

void delay\_ms(u16 ms)

{

u16 m;

for(m=0;m<ms;m++)

delay4us(250);

}

//为兼容原程序

void delay\_us(u16 us)

{

u16 i;

u8 m;

for(i=0;i<us;i++)

for(m=0;m<1;m++);

}

//4微秒 延时

//此函数是以4us的倍数增加

//t 4us的倍数

void delay4us(u16 us)

{

u16 i;

u8 m;

for(i=0;i<us;i++)

for(m=0;m<5;m++);

}

//1毫秒 延时

//ms 1ms的倍数

void delay1ms(u16 ms)

{

u16 m;

for(m=0;m<ms;m++)

delay4us(250);

}

void Delay100ms(void) //@24.000MHz

{

unsigned char i, j, k;

\_nop\_();

\_nop\_();

i = 10;

j = 31;

k = 147;

do

{

do

{

while (--k);

} while (--j);

} while (--i);

}

## SYS.H

#ifndef \_\_SYS\_H

#define \_\_SYS\_H

#include "stc15f2k60s2.h"

#include <intrins.h>

#define DRIVING\_SIMULATION 0 //是否是定时器脉冲模拟计价/码盘计数

#define u8 unsigned char

#define u16 unsigned int

#define u32 unsigned long

void delay2us(u16 us); //2us倍数延时

void delay4us(u16 us); //4us倍数延时

void delay1ms(u16 ms); //毫秒延时

void delay\_us(u16 us); //为兼容源程序

void delay\_ms(u16 ms);

void Delay100ms(void);

#endif