湖南大学电气与信息工程学院

本科生课程设计

数字电压表设计报告

课 程：单片机综合实训

专 业：自动化

班 级：自动化2102班

姓 名：黄涛

学 号：202107030227

同组人姓名：朱富国

指 导 老师：荣 宏

时 间：2024.5

目录

[1、任务及目标要求 2](#_Toc48)

[2、本次课程设计实际所实现的功能 2](#_Toc5481)

[3、系统原理框图及模块的作用 3](#_Toc4453)

[4、硬件设计及说明 4](#_Toc679)

[4.1 学生设计所实际使用到的具体元器件清单 4](#_Toc30379)

[4.2元器件计算和选择说明 4](#_Toc7407)

[4.3 硬件电路模块图及功能说明 5](#_Toc22051)

[5、程序流程图设计 9](#_Toc19895)

[6、系统操作说明或使用说明 9](#_Toc18396)

[7、综合设计心得体会 10](#_Toc6013)

[8、参考文献 11](#_Toc3164)

[9、全部源程序代码及详细注释 11](#_Toc26788)

[//STC15F2K60S2.h 11](#_Toc12430)

[//DS1302.h 19](#_Toc23098)

[//HC595.h 21](#_Toc20971)

[//KEY.h 22](#_Toc25070)

[//LCD1602.h 23](#_Toc3348)

[//ADC.h 24](#_Toc30914)

[//MAIN.h 26](#_Toc12870)

[//MAIN.c 28](#_Toc1084)

[//DS1302.c 40](#_Toc30277)

[//LCD1602.c 46](#_Toc32286)

[//KEY.c 56](#_Toc19369)

[//ADC.c 60](#_Toc30451)

[//HC595.C 63](#_Toc12228)

**1、任务及目标要求**

本次课程设计任务是设计一个数字电压表。任务要求有：

1）电压测量范围为0—5V（也可扩展其它模块）；

2）测量精度0.02V；

3）可根据需要设定超限值，超范围可多方式报警；

4）具有简单电子表功能，可显示年、月、日、星期、时、分、秒等（根据实际显示方式的不同，可选择部分或全部功能）；

5）具有电子表的校准功能；

6）扩展功能：

a、可利用开发板上其他硬件模块资源增加功能；

b、可利用主板芯片的增强性功能进行开发和设计；

c、可外接扩展模块增加功能（模块可采用实验室已有模块，也可自备）；

d、自定义其他功能，要求具有一定难度、创新性和综合性；

**2、本次课程设计实际所实现的功能**

1）电压测量范围为0—5V；

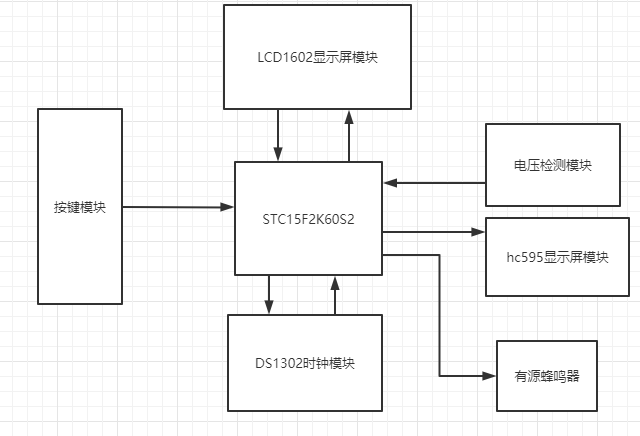
2）测量精度0.02V；

3）通过程序改变超限值，超范围可多方式报警；

4）具有简单电子表功能，可显示日、星期、时、分、秒等

5）具有电子表的校准功能；

**3、系统原理框图及模块的作用**

****

系统主要分为 DS1302 时钟模块，按键模块，单片机最小系统， LCD1602显示屏模块，hc595芯片及八位数码管显示屏模块和电压检测模块：

（1）按键模块：本模块可以通过按下不同的按键进行单片机的各个功能的切换，以及调试时间等。

（2）DS1302 时钟模块：产生时分秒数据并带有记忆功能，单片机可以直接读取

相应的数据送入显示，也可以对 DS1302 数据进行修改。使用该模块省去了对单

片机频率分频的麻烦，同时避免占用单片机内部的定时器。

（3）LCD1602显示屏模块：本模块可以显示万年历电子时钟，并可以通过按键模块对本模块进行设置。

（4）电压检测模块：利用ADC数模转换将采集多次的电压取平均值进行输出回单片机，以供其余模块进行利用。

（5）hc595芯片及八位数码管显示屏模块：将电压检测的模块输入的信号经单片机处理后先经过hc595芯片进行处理然后在八位数码管显示屏上显示。

**4、硬件设计及说明**

**4.1 学生设计所实际使用到的具体元器件清单**

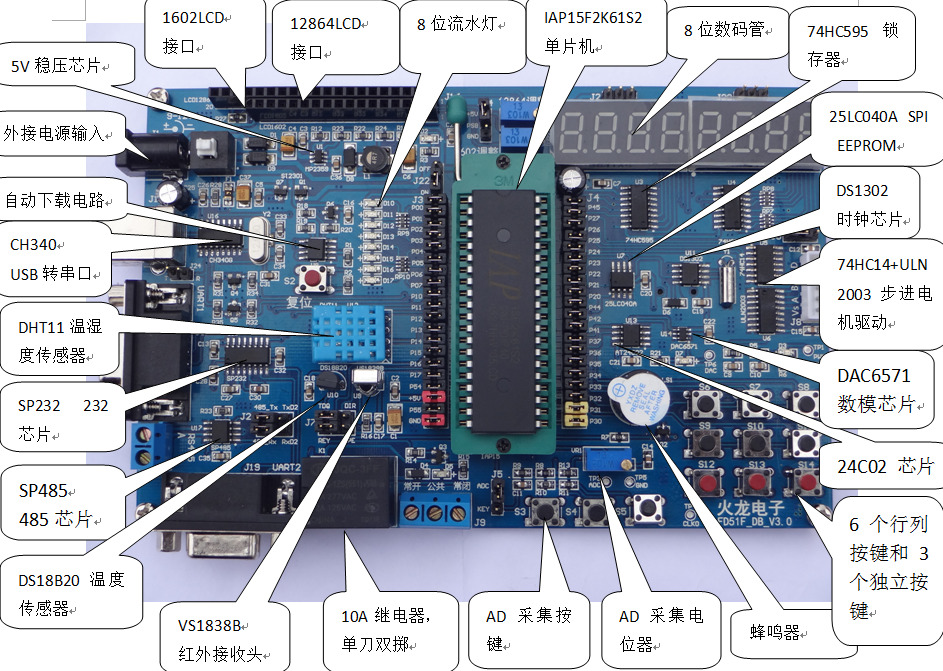
|  |  |  |
| --- | --- | --- |
| **元器件** | **数量** | **型号** |
| **单片机** | **1** | **STC15F2K60S2** |
| **按键** | **5** | **微动轻触按键** |
| **液晶显示屏** | **2** | **八位数码管、LCD1602** |
| **时钟芯片** | **1** | **DS1302** |
| **数模转换模块** | **1** | **DAC6571** |
| **电容、电阻** | **若干** | **各个规格** |
| **小型芯片** | **2** | **hc595** |

**4.2元器件计算和选择说明**

1. 因为需要时钟模块来进行频率的计算，所以采用DS1302.
2. 为了便于显示万年历时钟，遂采用LCD1602显示屏。
3. 因为要实现电压检测，所以选择数模转换模块DAC6571。
4. 为了便于显示所采集的电压，采用2块hc595芯片进行处理数据，并把处理好的数据发送到八位数码管显示屏来显示电压值。
5. 为了方便更换功能以及调试时钟，所以要选择按键模块来进行输入变量。

**4.3 硬件电路模块图及功能说明**

（1）单片机最小系统：



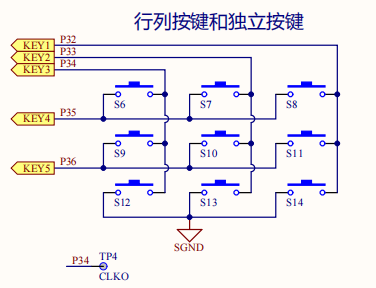
选用开发板1。

FD51F\_DB开发板采用的是宏晶科技（STC）公司推出的新一代增强型8051单片机IAP15F2K61S2芯片。该芯片隶属于STC15F2K60S2系列家族，具有STC15F2K60S2单片机的所有性能：不需要外部晶振和复位电路、可在线编程、大容量2K字节的SRAM、两个独立串口、8通道高速A/D转换器、1个时钟/机器周期8051等。同时IAP15具有自己的特色：可支持Keil中设置断点，在线单步调试等。待项目开发完毕后可更换成STC15系列进行生产，降低成本。

单片机：IAP15F2K61S2，DIP-40封装，是单时钟/机器周期（1T）的单片机，普通的51单片机是12T的单片机，1T的单片机比12T的单片机执行速度至少快12倍，是高速/高可靠/低功耗/超强抗干扰的新一代8051单片机，采用STC第八代加密技术，加密型强，指令代码完全兼容传统8051单片机。

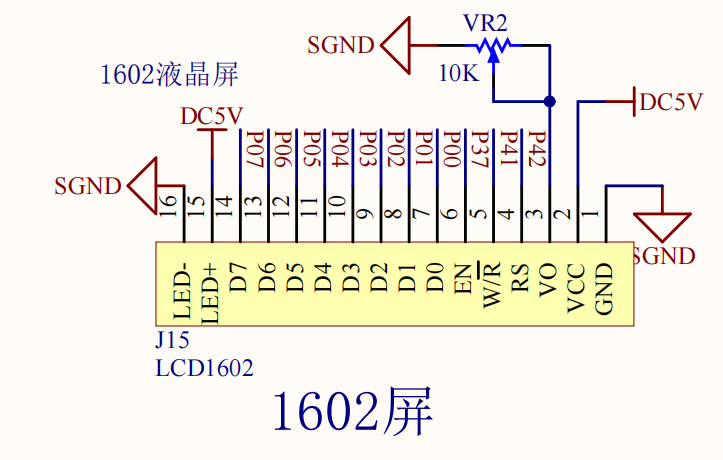
本次用到的有DS1302时钟芯片、DAC6571数模芯片、AD采集电位器、八位数码管显示屏、蜂鸣器、6个行列按键等内容。

（2）按键模块：



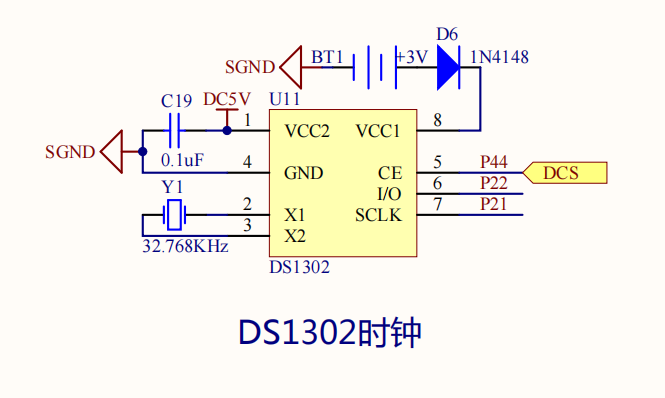
用于改变单片机模式以及调试万年历。

1. LCD1602显示屏模块：



用于显示万年历。

1. DS1302时钟模块：



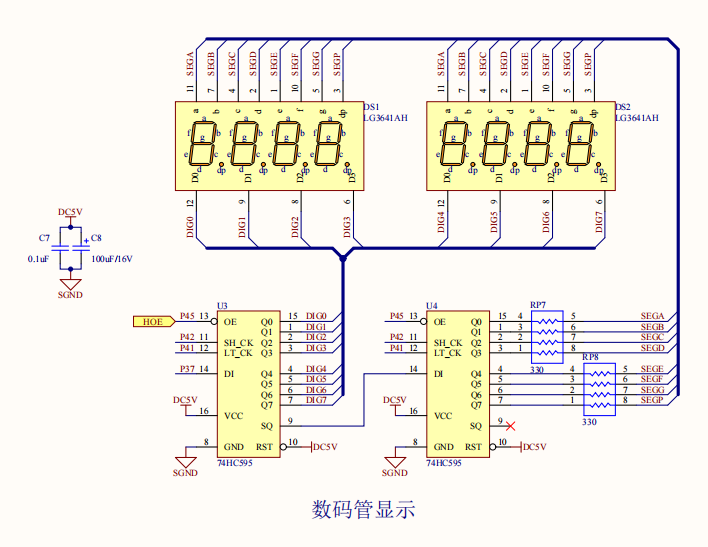
它可以对年、月、日、周、时、分、秒进行计时，具有闰年补偿功能，工作

电压为 2.0V～5.5V。采用三线接口与 CPU 进行同步通信，并可采用突发方式

一次传送多个字节的时钟信号或 RAM 数据。DS1302 内部有一个 31×8 的用于

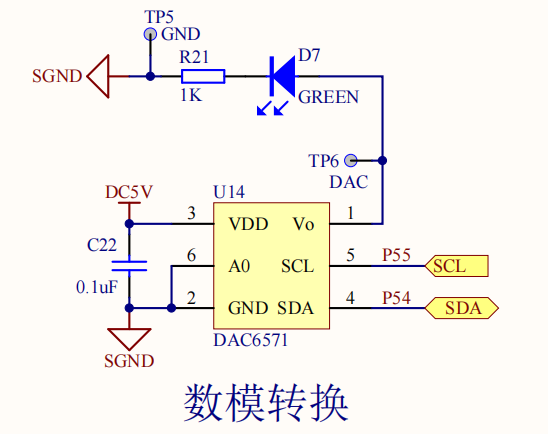
临时性存放数据的 RAM 寄存器。

1. hc595芯片及八位数码管显示屏模块：



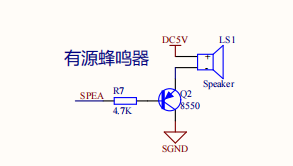
hc595用于处理数据，处理完毕后传输到八位数码管显示屏显示电压值。

1. 数模转换模块：



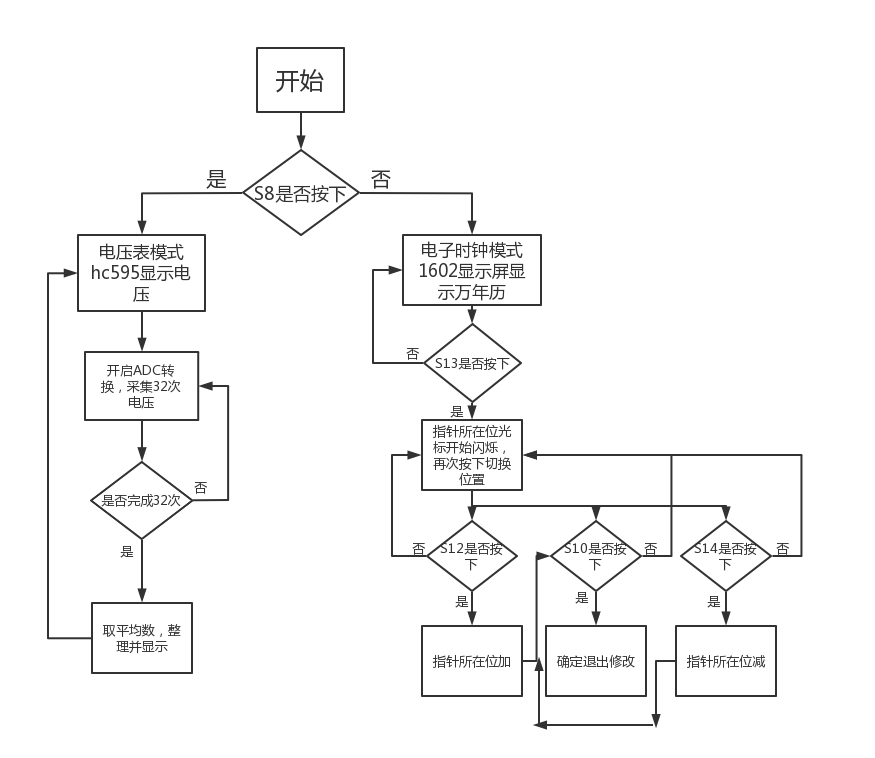
用于将采集到的电压转换成数字信号以便于处理。

（7）有源蜂鸣器：



电压表超限值报警

**5、程序流程图设计**



**6、系统操作说明或使用说明**

（1）打开 keil软件,进行编译，编译成功之后生成机器代码.hex 文件。

（2）打开单片机电源，用下载软件将机器代码载入单片机中，同时打开单片机的下载开关。

（3）等待单片机下载完毕后，单片机就会进入默认模式，按下键盘模块的S8可以选择万年历模式或电压检测模式。

（4）在万年历模式中，单片机会读取 DS1302 时钟模块的数据并写入 LCD1602 显示屏中显示年月日时间。当需要修改时间时 可以通过S13选择需要修改的项目，光标指向对应的项目后按动S12可以对数据进行加操作，按动S14可以对数据进行减操作，按下S10确认修改退出。

（5）在电压检测模式中，单片机会通过ADC数模转换将收集到的数据进行处理并在八位数码管上显示所检测到的电压，还可以设置电压阈值，当超过此阈值时，有源蜂鸣器会发出警报。

**7、综合设计心得体会**

1、透过这次课程设计使我懂得了理论与实际相结合是很重要的，只有理论知识是远远不够的，只有把所学的理论知识与实践相结合起来，从理论中得出结论，才能真正为社会服务，从而提高自己的实际动手潜力和独立思考的潜力。

2、在做本次课程设计的过程中，我感触最深的当属查阅大量的设计资料了。为了让自己的设计更加完善，查阅这方面的设计资料是十分必要的，同时也是必不可少的。

3、在编程过程中分别实现各部分功能，但是在将他们合在一起时却遇到了困难，这时我想到了老师课堂上讲过的分时复用，因此设计了一个切换模式按键，完美地解决了数码管和LCD1602同时使用时LCD1602乱码的现象。

**8、参考文献**

[1]陈忠平.《基于 Proteus 的 51 系列单片机设计与仿真》.电子工业出版社

[2]赵建领，崔昭霞.《精通 51 单片机开发技术与应用实例》.电子工业出版社

[3]肖明耀，程莉，刘平.《STC15增强型单片机应用技能实训》.北京：中国电力出版社，2016.4

[4]徐爱钧.《STC15增强型8051单片机C语言编程与应用》.北京：电子工业出社，2014

**9、全部源程序代码及详细注释**

**//STC15F2K60S2.h**

#ifndef \_\_STC15F2K60S2\_H\_\_

#define \_\_STC15F2K60S2\_H\_\_

//内核特殊功能寄存器 // 复位值 描述

sfr ACC = 0xE0; //0000,0000 累加器Accumulator

sfr B = 0xF0; //0000,0000 B寄存器

sfr PSW = 0xD0; //0000,0000 程序状态字

sbit CY = PSW^7;

sbit AC = PSW^6;

sbit F0 = PSW^5;

sbit RS1 = PSW^4;

sbit RS0 = PSW^3;

sbit OV = PSW^2;

sbit P = PSW^0;

sfr SP = 0x81; //0000,0111 堆栈指针

sfr DPL = 0x82; //0000,0000 数据指针低字节

sfr DPH = 0x83; //0000,0000 数据指针高字节

//I/O 口特殊功能寄存器

sfr P0 = 0x80; //1111,1111 端口0

sbit P00 = P0^0;

sbit P01 = P0^1;

sbit P02 = P0^2;

sbit P03 = P0^3;

sbit P04 = P0^4;

sbit P05 = P0^5;

sbit P06 = P0^6;

sbit P07 = P0^7;

sfr P1 = 0x90; //1111,1111 端口1

sbit P10 = P1^0;

sbit P11 = P1^1;

sbit P12 = P1^2;

sbit P13 = P1^3;

sbit P14 = P1^4;

sbit P15 = P1^5;

sbit P16 = P1^6;

sbit P17 = P1^7;

sfr P2 = 0xA0; //1111,1111 端口2

sbit P20 = P2^0;

sbit P21 = P2^1;

sbit P22 = P2^2;

sbit P23 = P2^3;

sbit P24 = P2^4;

sbit P25 = P2^5;

sbit P26 = P2^6;

sbit P27 = P2^7;

sfr P3 = 0xB0; //1111,1111 端口3

sbit P30 = P3^0;

sbit P31 = P3^1;

sbit P32 = P3^2;

sbit P33 = P3^3;

sbit P34 = P3^4;

sbit P35 = P3^5;

sbit P36 = P3^6;

sbit P37 = P3^7;

sfr P4 = 0xC0; //1111,1111 端口4

sbit P40 = P4^0;

sbit P41 = P4^1;

sbit P42 = P4^2;

sbit P43 = P4^3;

sbit P44 = P4^4;

sbit P45 = P4^5;

sbit P46 = P4^6;

sbit P47 = P4^7;

sfr P5 = 0xC8; //xxxx,1111 端口5

sbit P50 = P5^0;

sbit P51 = P5^1;

sbit P52 = P5^2;

sbit P53 = P5^3;

sbit P54 = P5^4;

sbit P55 = P5^5;

sbit P56 = P5^6;

sbit P57 = P5^7;

sfr P6 = 0xE8; //0000,0000 端口6

sbit P60 = P6^0;

sbit P61 = P6^1;

sbit P62 = P6^2;

sbit P63 = P6^3;

sbit P64 = P6^4;

sbit P65 = P6^5;

sbit P66 = P6^6;

sbit P67 = P6^7;

sfr P7 = 0xF8; //0000,0000 端口7

sbit P70 = P7^0;

sbit P71 = P7^1;

sbit P72 = P7^2;

sbit P73 = P7^3;

sbit P74 = P7^4;

sbit P75 = P7^5;

sbit P76 = P7^6;

sbit P77 = P7^7;

sfr P0M0 = 0x94; //0000,0000 端口0模式寄存器0

sfr P0M1 = 0x93; //0000,0000 端口0模式寄存器1

sfr P1M0 = 0x92; //0000,0000 端口1模式寄存器0

sfr P1M1 = 0x91; //0000,0000 端口1模式寄存器1

sfr P2M0 = 0x96; //0000,0000 端口2模式寄存器0

sfr P2M1 = 0x95; //0000,0000 端口2模式寄存器1

sfr P3M0 = 0xB2; //0000,0000 端口3模式寄存器0

sfr P3M1 = 0xB1; //0000,0000 端口3模式寄存器1

sfr P4M0 = 0xB4; //0000,0000 端口4模式寄存器0

sfr P4M1 = 0xB3; //0000,0000 端口4模式寄存器1

sfr P5M0 = 0xCA; //0000,0000 端口5模式寄存器0

sfr P5M1 = 0xC9; //0000,0000 端口5模式寄存器1

sfr P6M0 = 0xCC; //0000,0000 端口6模式寄存器0

sfr P6M1 = 0xCB; //0000,0000 端口6模式寄存器1

sfr P7M0 = 0xE2; //0000,0000 端口7模式寄存器0

sfr P7M1 = 0xE1; //0000,0000 端口7模式寄存器1

//系统管理特殊功能寄存器

sfr PCON = 0x87; //0001,0000 电源控制寄存器

sfr AUXR = 0x8E; //0000,0000 辅助寄存器

sfr AUXR1 = 0xA2; //0000,0000 辅助寄存器1

sfr P\_SW1 = 0xA2; //0000,0000 外设端口切换寄存器1

sfr CLK\_DIV = 0x97; //0000,0000 时钟分频控制寄存器

sfr BUS\_SPEED = 0xA1; //xx10,x011 总线速度控制寄存器

sfr P1ASF = 0x9D; //0000,0000 端口1模拟功能配置寄存器

sfr P\_SW2 = 0xBA; //xxxx,x000 外设端口切换寄存器

//中断特殊功能寄存器

sfr IE = 0xA8; //0000,0000 中断控制寄存器

sbit EA = IE^7;

sbit ELVD = IE^6;

sbit EADC = IE^5;

sbit ES = IE^4;

sbit ET1 = IE^3;

sbit EX1 = IE^2;

sbit ET0 = IE^1;

sbit EX0 = IE^0;

sfr IP = 0xB8; //0000,0000 中断优先级寄存器

sbit PPCA = IP^7;

sbit PLVD = IP^6;

sbit PADC = IP^5;

sbit PS = IP^4;

sbit PT1 = IP^3;

sbit PX1 = IP^2;

sbit PT0 = IP^1;

sbit PX0 = IP^0;

sfr IE2 = 0xAF; //0000,0000 中断控制寄存器2

sfr IP2 = 0xB5; //xxxx,xx00 中断优先级寄存器2

sfr INT\_CLKO = 0x8F; //0000,0000 外部中断与时钟输出控制寄存器

//定时器特殊功能寄存器

sfr TCON = 0x88; //0000,0000 T0/T1控制寄存器

sbit TF1 = TCON^7;

sbit TR1 = TCON^6;

sbit TF0 = TCON^5;

sbit TR0 = TCON^4;

sbit IE1 = TCON^3;

sbit IT1 = TCON^2;

sbit IE0 = TCON^1;

sbit IT0 = TCON^0;

sfr TMOD = 0x89; //0000,0000 T0/T1模式寄存器

sfr TL0 = 0x8A; //0000,0000 T0低字节

sfr TL1 = 0x8B; //0000,0000 T1低字节

sfr TH0 = 0x8C; //0000,0000 T0高字节

sfr TH1 = 0x8D; //0000,0000 T1高字节

sfr T4T3M = 0xD1; //0000,0000 T3/T4模式寄存器

sfr T3T4M = 0xD1; //0000,0000 T3/T4模式寄存器

sfr T4H = 0xD2; //0000,0000 T4高字节

sfr T4L = 0xD3; //0000,0000 T4低字节

sfr T3H = 0xD4; //0000,0000 T3高字节

sfr T3L = 0xD5; //0000,0000 T3低字节

sfr T2H = 0xD6; //0000,0000 T2高字节

sfr T2L = 0xD7; //0000,0000 T2低字节

sfr WKTCL = 0xAA; //0000,0000 掉电唤醒定时器低字节

sfr WKTCH = 0xAB; //0000,0000 掉电唤醒定时器高字节

sfr WDT\_CONTR = 0xC1; //0000,0000 看门狗控制寄存器

//串行口特殊功能寄存器

sfr SCON = 0x98; //0000,0000 串口1控制寄存器

sbit SM0 = SCON^7;

sbit SM1 = SCON^6;

sbit SM2 = SCON^5;

sbit REN = SCON^4;

sbit TB8 = SCON^3;

sbit RB8 = SCON^2;

sbit TI = SCON^1;

sbit RI = SCON^0;

sfr SBUF = 0x99; //xxxx,xxxx 串口1数据寄存器

sfr S2CON = 0x9A; //0000,0000 串口2控制寄存器

sfr S2BUF = 0x9B; //xxxx,xxxx 串口2数据寄存器

sfr S3CON = 0xAC; //0000,0000 串口3控制寄存器

sfr S3BUF = 0xAD; //xxxx,xxxx 串口3数据寄存器

sfr S4CON = 0x84; //0000,0000 串口4控制寄存器

sfr S4BUF = 0x85; //xxxx,xxxx 串口4数据寄存器

sfr SADDR = 0xA9; //0000,0000 从机地址寄存器

sfr SADEN = 0xB9; //0000,0000 从机地址屏蔽寄存器

//ADC 特殊功能寄存器

sfr ADC\_CONTR = 0xBC; //0000,0000 A/D转换控制寄存器

sfr ADC\_RES = 0xBD; //0000,0000 A/D转换结果高8位

sfr ADC\_RESL = 0xBE; //0000,0000 A/D转换结果低2位

//SPI 特殊功能寄存器

sfr SPSTAT = 0xCD; //00xx,xxxx SPI状态寄存器

sfr SPCTL = 0xCE; //0000,0100 SPI控制寄存器

sfr SPDAT = 0xCF; //0000,0000 SPI数据寄存器

//IAP/ISP 特殊功能寄存器

sfr IAP\_DATA = 0xC2; //0000,0000 EEPROM数据寄存器

sfr IAP\_ADDRH = 0xC3; //0000,0000 EEPROM地址高字节

sfr IAP\_ADDRL = 0xC4; //0000,0000 EEPROM地址第字节

sfr IAP\_CMD = 0xC5; //xxxx,xx00 EEPROM命令寄存器

sfr IAP\_TRIG = 0xC6; //0000,0000 EEPRPM命令触发寄存器

sfr IAP\_CONTR = 0xC7; //0000,x000 EEPROM控制寄存器

//PCA/PWM 特殊功能寄存器

sfr CCON = 0xD8; //00xx,xx00 PCA控制寄存器

sbit CF = CCON^7;

sbit CR = CCON^6;

sbit CCF2 = CCON^2;

sbit CCF1 = CCON^1;

sbit CCF0 = CCON^0;

sfr CMOD = 0xD9; //0xxx,x000 PCA 工作模式寄存器

sfr CL = 0xE9; //0000,0000 PCA计数器低字节

sfr CH = 0xF9; //0000,0000 PCA计数器高字节

sfr CCAPM0 = 0xDA; //0000,0000 PCA模块0的PWM寄存器

sfr CCAPM1 = 0xDB; //0000,0000 PCA模块1的PWM寄存器

sfr CCAPM2 = 0xDC; //0000,0000 PCA模块2的PWM 寄存器

sfr CCAP0L = 0xEA; //0000,0000 PCA模块0的捕捉/比较寄存器低字节

sfr CCAP1L = 0xEB; //0000,0000 PCA模块1的捕捉/比较寄存器低字节

sfr CCAP2L = 0xEC; //0000,0000 PCA模块2的捕捉/比较寄存器低字节

sfr PCA\_PWM0 = 0xF2; //xxxx,xx00 PCA模块0的PWM寄存器

sfr PCA\_PWM1 = 0xF3; //xxxx,xx00 PCA模块1的PWM寄存器

sfr PCA\_PWM2 = 0xF4; //xxxx,xx00 PCA模块1的PWM寄存器

sfr CCAP0H = 0xFA; //0000,0000 PCA模块0的捕捉/比较寄存器高字节

sfr CCAP1H = 0xFB; //0000,0000 PCA模块1的捕捉/比较寄存器高字节

sfr CCAP2H = 0xFC; //0000,0000 PCA模块2的捕捉/比较寄存器高字节

#endif

**//DS1302.h**

# ifndef \_\_DS1302\_H\_\_

# define \_\_DS1302\_H\_\_

#include "stc15f2k60s2.h"

//DS1302接口定义

sbit SCE = P4^4; //DS1302选择端

sbit SIO = P2^2; //DS1302数据口

sbit SCK = P2^1; //DS1302时钟口

//定义端口操作

#define DS1302\_SCE\_Clr() {SCE=0;}

#define DS1302\_SCE\_Set() {SCE=1;}

#define DS1302\_SIO\_Clr() {SIO=0;}

#define DS1302\_SIO\_Set() {SIO=1;}

#define DS1302\_SCK\_Clr() {SCK=0;}

#define DS1302\_SCK\_Set() {SCK=1;}

//定义日期结构

typedef struct datetime

{

unsigned char Year; //年

unsigned char Week; //星期

unsigned char Month; //月

unsigned char Day; //日

unsigned char Hour; //时

unsigned char Minute; //分

unsigned char Second; //秒

}DAYTIME;

//设定数据范围

#define Year\_TOP 99 //最大值

#define Year\_BOTTOM 0 //最小值

#define Week\_TOP 7

#define Week\_BOTTOM 1

#define Month\_TOP 12

#define Month\_BOTTOM 1

#define Day\_TOP 31

#define Day\_BOTTOM 1

#define Hour\_TOP 23

#define Hour\_BOTTOM 0

#define Minute\_TOP 59

#define Minute\_BOTTOM 0

#define Second\_TOP 59

#define Second\_BOTTOM 0

//函数定义

unsigned char DS1302\_ReadData(unsigned char addr);

void DS1302\_WriteData(unsigned char addr, unsigned char dat);

void DS1302\_GetTime(DAYTIME \*pClock);

void DS1302\_SetTime(DAYTIME \*pClock);

void DS1302\_Init(DAYTIME \*pClock);

# endif

**//HC595.h**

#ifndef \_\_HC595\_H\_\_

#define \_\_HC595\_H\_\_

#include "stc15f2k60s2.h"

//HC595接口定义

sbit HC595\_DAT = P3^7; //HC595串行数据线

sbit HC595\_SCK = P4^2; //HC595移位时钟线

sbit HC595\_RCK = P4^1; //HC595锁存时钟线

sbit HC595\_OE = P4^5; //HC595输出允许线

//数码管位定义，从左往右依次为1~8位

#define DIG1 0

#define DIG2 1

#define DIG3 2

#define DIG4 3

#define DIG5 4

#define DIG6 5

#define DIG7 6

#define DIG8 7

//端口操作定义

#define SET 1 //高电平

#define CLR 0 //低电平

//端口操作

#define HC595\_DAT\_Clr() {HC595\_DAT=CLR;} //端口置低

#define HC595\_DAT\_Set() {HC595\_DAT=SET;} //端口置高

#define HC595\_SCK\_Clr() {HC595\_SCK=CLR;}

#define HC595\_SCK\_Set() {HC595\_SCK=SET;}

#define HC595\_RCK\_Clr() {HC595\_RCK=CLR;}

#define HC595\_RCK\_Set() {HC595\_RCK=SET;}

#define HC595\_OE\_Clr() {HC595\_OE=CLR;}

#define HC595\_OE\_Set() {HC595\_OE=SET;}

//外部函数

void HC595\_Init(void);

void LED\_Display(void);

void ToDisplayBuf(unsigned int Low\_Dat);

#endif

**//KEY.h**

#ifndef \_\_KEY\_H\_\_

#define \_\_KEY\_H\_\_

#include "stc15f2k60s2.h"

//行列按键所在端口的序号

//例如Line1在端口P35，则对应的序号为5

#define Line1 5 //行1的序号

#define Line2 6 //行2的序号

#define Column1 4 //列1的序号

#define Column2 3 //列2的序号

#define Column3 2 //列3的序号

//独立按键所在端口的序号

//例如KEY12在端口P34，则对应的序号为4

#define K12 4 //按键12序号

#define K13 3 //按键13序号

#define K14 2 //按键14序号

//行列按键所在的端口

#define KEYPORT P3

//按键定义 定义值可以任意，六个值不相同就可以

#define KEY6 6

#define KEY7 7

#define KEY8 8

#define KEY9 9

#define KEY10 10

#define KEY11 11

#define KEY12 12

#define KEY13 13

#define KEY14 14

//函数声明

unsigned char Key\_Scan(void);

#endif

**//LCD1602.h**

# ifndef \_\_LCD1602\_H\_\_

# define \_\_LCD1602\_H\_\_

#include "stc15f2k60s2.h"

#include "ds1302.h"

//LCD1602操作位定义

#define DataPort P0

sbit RS = P4^2; //寄存器选择 0:指令寄存器 1:数据寄存器

sbit RW = P4^1; //读写控制 0：写 1：读

sbit EN = P3^7; //读写数据使能 0：停止 1：启动

//定义端口操作

#define SET 1 //置高

#define CLR 0 //置低

#define RS\_Set() {RS=SET;} //端口置高

#define RS\_Clr() {RS=CLR;} //端口置低

#define RW\_Set() {RW=SET;} //端口置高

#define RW\_Clr() {RW=CLR;} //端口置低

#define EN\_Set() {EN=SET;} //端口置高

#define EN\_Clr() {EN=CLR;} //端口置低

#define LINE1 0x80 //第一行地址

#define LINE2 0xc0 //第二行地址

void LCD1602\_Init(void);

void LCD1602\_Display\_Clock(DAYTIME \*pClock);

void ToDateNonDispBuf(unsigned char set);

void ToDateOrgDispBuf(unsigned char set,DAYTIME \*pClock);

void LCD1602\_WriteCMD(unsigned char cmd);

void LCD1602\_Display\_Str(unsigned char\* str);

# endif

**//ADC.h**

#include "stc15f2k60s2.h"

//包含该文件可以使用\_nop\_()函数

#include <intrins.h>

//定义电源电压,如果用户开发板实际电源电压为5.110v，可将5000该成5110

#define VCC 5110

//ADC控制位定义

#define ADC\_POWER 0x80 //ADC电源控制位

#define ADC\_FLAG 0x10 //ADC转换完成标记

#define ADC\_START 0x08 //ADC开始转换标记

#define ADC\_SPEEDLL 0x00 //ADC转换速率540时钟

#define ADC\_SPEEDL 0x20 //ADC转换速率360时钟

#define ADC\_SPEEDH 0x40 //ADC转换速率180时钟

#define ADC\_SPEEDHH 0x60 //ADC转换速率90时钟

//ADC通道定义

#define ADC\_CHANNEL0 0x00 //转换通道P1.0

#define ADC\_CHANNEL1 0x01 //转换通道P1.1

#define ADC\_CHANNEL2 0x02 //转换通道P1.2

#define ADC\_CHANNEL3 0x03 //转换通道P1.3

#define ADC\_CHANNEL4 0x04 //转换通道P1.4

#define ADC\_CHANNEL5 0x05 //转换通道P1.5

#define ADC\_CHANNEL6 0x06 //转换通道P1.6

#define ADC\_CHANNEL7 0x07 //转换通道P1.7

//ADC端口模拟功能设置

#define P1ASF\_0 0x01 //设置P1.0口为ADC端口

#define P1ASF\_1 0x02 //设置P1.1口为ADC端口

#define P1ASF\_2 0x04 //设置P1.2口为ADC端口

#define P1ASF\_3 0x08 //设置P1.3口为ADC端口

#define P1ASF\_4 0x10 //设置P1.4口为ADC端口

#define P1ASF\_5 0x20 //设置P1.5口为ADC端口

#define P1ASF\_6 0x40 //设置P1.6口为ADC端口

#define P1ASF\_7 0x80 //设置P1.7口为ADC端口

//函数声明

void ADC\_Init(void);

void ADC\_Process(void);

void Delay\_ms(unsigned int ms);

unsigned int ADC\_GetResult(unsigned char ch);

void alarm(void);

//定义位变量

sbit ALARM=P1^6;

//关和开的宏定义

#define ALARM\_ON() {ALARM=0;} //喇叭响

#define ALARM\_OFF() {ALARM=1;} //喇叭不响

**//MAIN.h**

#ifndef \_\_MAIN\_H\_\_

#define \_\_MAIN\_H\_\_

//文件包含

#include "stc15f2k60s2.h"

#include "ds1302.h"

#include "lcd1602.h"

#include "key.h"

#include "hc595.h"

#include "ADC.h"

#define uchar unsigned char

#define uint unsigned int

//时间更新标记

bit UpdateTimeFlag=1;

//在设置中标记

bit InSet=0;

//闪烁标记

bit Flash\_Flag=0;

//定义日期对象

DAYTIME Clock;

//定义菜单结构

typedef struct

{

uchar MENU\_UP; //上翻

uchar MENU\_DOWN; //下翻

uchar MENU\_SET; //设置

uchar MENU\_CERTAIN; //确定

uchar page;

}MENU;

//定义菜单对象

MENU menu;

//菜单个数

#define MenuSizes 7

//定义加减键

#define UP 1

#define DOWN 0

//十六进制转换成十进制

#define Hex2Int(x) ( (x/16)\*10+(x%16) )

//十进制转换成十六进制

#define Int2Hex(x) ( (x/10)\*16+(x%10) )

//函数声明

void Delay\_ms(uint ms);

void Timer0\_Init(void);

void Key\_Process(void);

void Add\_Process(bit add);

void IO\_Init(void);

#endif

**//MAIN.c**

#include "main.h"

//主方法

void main(void)

{

Delay\_ms(10); //上电延时10ms

IO\_Init(); //IO口初始化

Timer0\_Init(); //定时器初始化

DS1302\_Init(&Clock); //时钟初始化

LCD1602\_Init(); //液晶屏初始化

ADC\_Init(); //ADC初始化

HC595\_Init(); //HC595初始化

while(1)

{

Key\_Process(); //按键扫描

//定时1s更新时间，且不在设置时间中

if( (UpdateTimeFlag==1)&&(InSet==0) )

{

UpdateTimeFlag=0;

DS1302\_GetTime(&Clock); //读取时钟

if(menu.page==0)

{

LCD1602\_WriteCMD(0x01); //清屏

LCD1602\_Display\_Clock(&Clock);

} //显示时钟

if(menu.page==1)

{

LCD1602\_WriteCMD(0x01); //清屏

ADC\_Process(); //ADC数据采集并且处理

}

}

}

}

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

函数名称：Delay\_ms

功 能：STC 1T单片机1ms延时程序

入口参数：ms:延时的毫秒数

返 回 值：无

备 注：示波器实测1.05ms 时钟11.0592MHz

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

void Delay\_ms(unsigned int ms)

{

unsigned int De\_Cnt;

while( (ms--) != 0)

{

for(De\_Cnt = 0; De\_Cnt < 600; De\_Cnt++);

}

}

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

函数名称：Timer0\_Init

功 能：定时器0初始化函数

入口参数：无

返 回 值：无

备 注：定时初值可以使用stc下载软件中的

定时初值自动生成功能。

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

void Timer0\_Init(void)

{

AUXR &= 0x7f; //T0时钟12T模式

TMOD &= 0xf0; //T0工作于十六位定时方式

TMOD |=0x01;

//使用位操作，避免对其他定时器产生影响

TL0=0x00; //定时初值50ms 11.0592MHz

TH0=0x4c;

TF0 = 0; //清除TF0标记

ET0=1; //使能T0中断

EA=1; //使能总中断

TR0=1; //运行T0

}

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

函数名称：Timer0\_ISR

功 能：定时器0中断服务函数

入口参数：无

返 回 值：无

备 注：无

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

void Timer0\_ISR(void) interrupt 1

{

static unsigned char T0\_Cnt=0; //静态计数器

static unsigned char T1\_Cnt=0; //静态计数器

static bit Flash\_bit=0; //闪烁转换

//使用静态计数器,每次调用该中断函数时，

//静态计数器都能保持上一次的计数值。

//如果不使用静态计数器，每次调用该中断函数时

//该计数器初值都是0，计数值就无法累加。

TL0=0x00; //重装定时初值定时初值50ms

TH0=0x4c;

//如果闪烁标记有效

if(Flash\_Flag==1)

{

T1\_Cnt++;

if(T1\_Cnt==10) //定时500ms

{

T1\_Cnt=0;

//用于设置部分消隐

if(Flash\_bit==0)

{

ToDateNonDispBuf(menu.MENU\_SET); //不显示

}

else

{

ToDateOrgDispBuf(menu.MENU\_SET,&Clock); //显示

}

//标记取反

Flash\_bit=!Flash\_bit;

}

}

//定时读取时钟

T0\_Cnt++;

if(T0\_Cnt==20) //定时1000ms

{

T0\_Cnt=0;

UpdateTimeFlag=1;//更新标记有效

}

}

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

函数名称：Key\_Process

功 能：按键处理函数

入口参数：无

返 回 值：无

备 注：无

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

void Key\_Process(void)

{

unsigned char KeyNum;

if( (KeyNum=Key\_Scan())!=0 )//检测是否有键按下

{

if(KeyNum==KEY8)

{

LCD1602\_WriteCMD(0x01); //清屏

if (menu.page==1)

{

menu.page=0;

ALARM\_OFF();

}

else if(menu.page==0)

{menu.page=1;

}

}

if(menu.page==0)

{

switch(KeyNum)

{

case KEY12: //S12: 按下 "上翻“的功能

{

//如果在时钟设置中

if(InSet==1)

{

Add\_Process(UP); //处理按键的“上翻”功能

}

}break;

case KEY13: //S13：按下 "设置“的功能

{

//时钟设置有效

InSet=1;

//闪烁无效

Flash\_Flag=0;

//每次按下该键就显示下一个设置量

menu.MENU\_SET=(menu.MENU\_SET+1) % (MenuSizes+1);

if(menu.MENU\_SET==0)

{

menu.MENU\_SET=1;

}

//更新显示

LCD1602\_Display\_Clock(&Clock);

//闪烁有效

Flash\_Flag=1;

//菜单显示

}break;

case KEY14: //S14：按下 "下翻“的功能

{

//如果在菜单设置中

if(InSet==1)

{

Add\_Process(DOWN);//处理按键的“下翻”功能

}

}break;

case KEY10: //S10：按下 "确定“的功能

{

//如果在菜单设置中

if(InSet==1)

{

InSet=0; //菜单设置无效

Flash\_Flag=1; //闪烁无效

menu.MENU\_SET=0;//清零菜单项

//保存时钟

DS1302\_SetTime(&Clock);

}

}break;

}

}

}

if(menu.page==1)

{

ADC\_Process(); //ADC数据采集并且处理

LED\_Display();

alarm();

}

}

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

函数名称：Add\_Process

功 能：设置按键处理函数

入口参数：add：功能分类 UP:为上翻按键，DOWM:为下翻按键，

返 回 值：无

备 注：无

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

void Add\_Process(bit add)

{

unsigned char Temp;

if(add==UP) //处理上翻按键

{

switch(menu.MENU\_SET)

{

case 1: //年设定

{

//将读出的时钟年十六进制码转换成十进制

Temp=Hex2Int(Clock.Year);

//长度值加1

Temp = (Temp + 1) % (Year\_TOP+1) ;

if(Temp==0)

{

Temp=Year\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Year=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 2: //月设定

{

//将读出的时钟月十六进制码转换成十进制

Temp=Hex2Int(Clock.Month);

//长度值加1

Temp = (Temp + 1) % (Month\_TOP+1) ;

if(Temp==0)

{

Temp=Month\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Month=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 3: //日设定

{

//将读出的时钟日十六进制码转换成十进制

Temp=Hex2Int(Clock.Day);

//长度值加1

Temp = (Temp + 1) % (Day\_TOP+1) ;

if(Temp==0)

{

Temp=Day\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Day=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 4: //星期设定

{

//将读出的时钟星期十六进制码转换成十进制

Temp=Hex2Int(Clock.Week);

//长度值加1

Temp = (Temp + 1) % (Week\_TOP+1) ;

if(Temp==0)

{

Temp=Week\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Week=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 5: //时设定

{

//将读出的时钟时十六进制码转换成十进制

Temp=Hex2Int(Clock.Hour);

//长度值加1

Temp = (Temp + 1) % (Hour\_TOP+1) ;

if(Temp==0)

{

Temp=Hour\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Hour=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 6: //分设定

{

//将读出的时钟分十六进制码转换成十进制

Temp=Hex2Int(Clock.Minute);

//长度值加1

Temp = (Temp + 1) % (Minute\_TOP+1) ;

if(Temp==0)

{

Temp=Minute\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Minute=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 7: //秒设定

{

//将读出的时钟秒十六进制码转换成十进制

Temp=Hex2Int(Clock.Second);

//长度值加1

Temp = (Temp + 1) % (Second\_TOP+1) ;

if(Temp==0)

{

Temp=Second\_BOTTOM;

}

//将十进制码转换回十六进制

Clock.Second=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

}

}

else if(add==DOWN) //处理下翻按键

{

switch(menu.MENU\_SET)

{

case 1: //年设定

{

//将读出的时钟年十六进制码转换成十进制

Temp=Hex2Int(Clock.Year);

//长度值减1

if(Temp == Year\_BOTTOM)

{

Temp = Year\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Year=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 2: //月设定

{

//将读出的时钟月十六进制码转换成十进制

Temp=Hex2Int(Clock.Month);

//长度值减1

if(Temp == Month\_BOTTOM)

{

Temp = Month\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Month=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 3: //日设定

{

//将读出的时钟日十六进制码转换成十进制

Temp=Hex2Int(Clock.Day);

//长度值减1

if(Temp == Day\_BOTTOM)

{

Temp = Day\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Day=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 4: //星期设定

{

//将读出的时钟星期十六进制码转换成十进制

Temp=Hex2Int(Clock.Week);

//长度值减1

if(Temp == Week\_BOTTOM)

{

Temp = Week\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Week=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 5: //时设定

{

//将读出的时钟时十六进制码转换成十进制

Temp=Hex2Int(Clock.Hour);

//长度值减1

if(Temp == Hour\_BOTTOM)

{

Temp = Hour\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Hour=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 6: //分设定

{

//将读出的时钟分十六进制码转换成十进制

Temp=Hex2Int(Clock.Minute);

//长度值减1

if(Temp == Minute\_BOTTOM)

{

Temp = Minute\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Minute=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

case 7: //秒设定

{

//将读出的时钟秒十六进制码转换成十进制

Temp=Hex2Int(Clock.Second);

//长度值减1

if(Temp == Second\_BOTTOM)

{

Temp = Second\_TOP ;

}

else

{

Temp--;

}

//将十进制码转换回十六进制

Clock.Second=Int2Hex(Temp);

//更新显示

LCD1602\_Display\_Clock(&Clock);

}break;

}

}

}

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

函数名称：IO\_Init

功 能：单片机IO端口模式初始化

入口参数：无

返 回 值：无

备 注：无

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

void IO\_Init(void)

{

//将P32 P33 P34 P35 P36设置为准双向口

P3M1 &=~( (1<<2) | (1<<3) | (1<<4) | (1<<5) | (1<<6) );

P3M0 &=~( (1<<2) | (1<<3) | (1<<4) | (1<<5) | (1<<6) );

//分析

// 1<<0等价于0x01 即 0000 0001

// 1<<1等价于0x02 即 0000 0010

// 1<<2等价于0x04 即 0000 0100

// 1<<3等价于0x08 即 0000 1000

// 以此类推1<<n 即第n位为1，其余位是0

// x |=(1<<n) 即对x执行按位取或

// 即x中的第n位置为1，不改变其他位状态

// y &=~(1<<n) 即将1<<n按位取反，然后对y按位取与

// 即y中的第n位置为0，不改变其他位状态

}

**//DS1302.c**

#include "ds1302.h"

#include <intrins.h>

//秒 分 时 日 月 星期 年

//初始化时间

//0 30 8 2 1 1 17

unsigned char code Init[7] = {0x00, 0x30, 0x08, 0x02, 0x01, 0x01, 0x17};

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

函数名称：Delay\_us

功 能：STC 1T单片机1us延时程序

入口参数：us:延时的微秒数

返 回 值：无

备 注：外部时钟11.0592MHz

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

void Delay\_us(unsigned int us)

{

while( us--)

{

\_nop\_();

}

}

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

函数名称：DS1302\_ReadByte

功 能：从DS1302读一个字节数据

入口参数：无

返 回 值：unsigned char：读出的数据

备 注：无

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

unsigned char DS1302\_ReadByte(void)

{

unsigned char i;

unsigned char dat = 0;

for (i=0; i<8; i++) //8位计数器

{

DS1302\_SCK\_Clr(); //时钟线拉低

Delay\_us(1); //延时等待

dat >>= 1; //数据右移一位

if (SIO)

{

dat |= 0x80; //读取数据

}

DS1302\_SCK\_Set(); //时钟线拉高

Delay\_us(1); //延时等待

}

return dat;

}

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

函数名称：DS1302\_WriteByte

功 能：向DS1302写一个字节数据

入口参数：dat：写入的数据

返 回 值：无

备 注：无

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

void DS1302\_WriteByte(unsigned char dat)

{

unsigned char i;

unsigned char temp;

for (i=0; i<8; i++) //8位计数器

{

temp=dat;

DS1302\_SCK\_Clr(); //时钟线拉低

Delay\_us(1); //延时等待

if( (temp&0x01)==0x01 ) //先发送低字节

{

DS1302\_SIO\_Set();

}

else

{

DS1302\_SIO\_Clr();

}

DS1302\_SCK\_Set(); //时钟线拉高

dat >>= 1; //移出数据

Delay\_us(1); //延时等待

}

}

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

函数名称：DS1302\_ReadData

功 能：读DS1302某地址的的数据

入口参数：addr:地址。

返 回 值：读出的数据。

备 注：无

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

unsigned char DS1302\_ReadData(unsigned char addr)

{

unsigned char dat;

DS1302\_SCE\_Clr(); //片选线拉低

Delay\_us(1);

DS1302\_SCK\_Clr(); //时钟线拉低

Delay\_us(1);

DS1302\_SCE\_Set(); //片选线拉高

Delay\_us(1);

DS1302\_WriteByte(addr); //写地址

dat = DS1302\_ReadByte();//读数据

DS1302\_SCK\_Set(); //时钟线拉高

DS1302\_SCE\_Clr(); //片选线拉低

return dat;

}

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

函数名称：DS1302\_WriteData

功 能：往DS1302的某个地址写入数据

入口参数：addr:地址。dat：写入的数据。

返 回 值：无

备 注：无

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

void DS1302\_WriteData(unsigned char addr, unsigned char dat)

{

DS1302\_SCE\_Clr(); //片选线拉低

Delay\_us(1);

DS1302\_SCK\_Clr(); //时钟线拉低

Delay\_us(1);

DS1302\_SCE\_Set(); //时钟线拉高

Delay\_us(1);

DS1302\_WriteByte(addr); //写地址

DS1302\_WriteByte(dat); //写数据

DS1302\_SCK\_Set(); //时钟线拉高

DS1302\_SCE\_Clr(); //片选线拉低

}

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

函数名称：DS1302\_WriteRAM

功 能：向DS1302的RAM中写入数据函数

入口参数：pBuf：指向设置值数据的指针

返 回 值：无

备 注：详见数据手册时间格式。

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

void DS1302\_WriteRAM(unsigned char \* pBuf)

{

unsigned char addr = 0xc0; //写ram开始地址

unsigned char n = 7; //使用7个字节的ram

DS1302\_WriteData(0x8e, 0x00); //允许写操作

while (n--) //连续写7个RAM寄存器，地址增2.

{

DS1302\_WriteData(addr, \*pBuf++);

addr += 2; //按数据手册定义，地址以2递增

}

DS1302\_WriteData(0x8e, 0x80); //写保护

}

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

函数名称：DS1302\_ReadRAM

功 能：从DS1302的RAM中读出数据函数

入口参数：pBuf：指向保存时间数据的指针

返 回 值：无

备 注：无

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

void DS1302\_ReadRAM(unsigned char \*pBuf)

{

unsigned char addr = 0xc1; //读ram开始地址

unsigned char n = 7; //使用7个字节的ram

while (n--) //连续读7个RAM寄存器，地址增2.

{

\*pBuf++ = DS1302\_ReadData(addr);

addr += 2;

}

}

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

函数名称：DS1302\_GetTime

功 能：DS1302读取时钟函数

入口参数：pClock：指向时间对象的指针

返 回 值：无

备 注：读取到的时钟数据格式为BCD格式。

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

void DS1302\_GetTime(DAYTIME \*pClock)

{

//开始读取的地址

unsigned char addr = 0x81;

//读取秒数

pClock->Second=DS1302\_ReadData(addr);

addr += 2;

//读取分钟

pClock->Minute=DS1302\_ReadData(addr);

addr += 2;

//读取小时

pClock->Hour=DS1302\_ReadData(addr);

addr += 2;

//读取日

pClock->Day=DS1302\_ReadData(addr);

addr += 2;

//读取月

pClock->Month=DS1302\_ReadData(addr);

addr += 2;

//读取星期

pClock->Week=DS1302\_ReadData(addr);

addr += 2;

//读取年

pClock->Year=DS1302\_ReadData(addr);

addr += 2;

}

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

函数名称：DS1302\_SetTime

功 能：DS1302时间设置函数

入口参数：pClock：指向时间对象的指针

返 回 值：无

备 注：详见数据手册时间格式。

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

void DS1302\_SetTime(DAYTIME \*pClock)

{

//开始写入的地址

unsigned char addr = 0x80;

DS1302\_WriteData(0x8e, 0x00); //允许写操作

//写入秒数

DS1302\_WriteData(addr, pClock->Second);

addr += 2;

//写入分钟

DS1302\_WriteData(addr, pClock->Minute);

addr += 2;

//写入时

DS1302\_WriteData(addr, pClock->Hour);

addr += 2;

//写入日

DS1302\_WriteData(addr, pClock->Day);

addr += 2;

//写入月

DS1302\_WriteData(addr, pClock->Month);

addr += 2;

//写入星期

DS1302\_WriteData(addr, pClock->Week);

addr += 2;

//写入年

DS1302\_WriteData(addr, pClock->Year);

addr += 2;

DS1302\_WriteData(0x8e, 0x80); //写保护

}

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

函数名称：DS1302\_Init

功 能：DS1302初始化函数

入口参数：无

返 回 值：无

备 注：

DS1302具体命令详见数据手册

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

void DS1302\_Init(DAYTIME \*pClock)

{

unsigned char cnt;

//保存RAM数据

unsigned char RAMData[7];

//将P21 P22设置为准双向口

P2M1 &=~( (1<<1) | (1<<2) );

P2M0 &=~( (1<<1) | (1<<2) );

//初始化P44口为准双向口

P4M1 &=~(1<<4);

P4M0 &=~(1<<4);

DS1302\_SCE\_Clr();

DS1302\_SCK\_Clr();

DS1302\_WriteData(0x8e, 0x00); //允许写操作

//读取振荡器工作状态

if( (DS1302\_ReadData(0x81)&0x80)==0x80)

{

//如果振荡器停止，则启动

DS1302\_WriteData(0x80, 0x00); //时钟启动

}

DS1302\_WriteData(0x90, 0xa6); //一个二极管＋4K电阻充电

DS1302\_WriteData(0x8e, 0x80); //写保护

Delay\_us(1000);

DS1302\_ReadRAM(RAMData); //读取内部RAM数据

for(cnt=0;cnt<7;cnt++)

{

if(RAMData[cnt]!=Init[cnt]) //如果不同，则认为未初始化

{

//初始化时钟

pClock->Second=Init[0];

pClock->Minute=Init[1];

pClock->Hour=Init[2];

pClock->Day=Init[3];

pClock->Month=Init[4];

pClock->Week=Init[5];

pClock->Year=Init[6];

DS1302\_SetTime(pClock); //重新初始化

DS1302\_WriteRAM(Init);

}

}

}

**//LCD1602.c**

#include "lcd1602.h"

//包含该函数可以使用\_nop\_()函数

#include <intrins.h>

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

函数名称：LCD1602\_Delay1us

功 能：STC 1T单片机1us延时程序

入口参数：us:延时的微秒数

返 回 值：无

备 注：外部时钟11.0592MHz

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

void LCD1602\_Delay1us(unsigned int us)

{

while( us--)

{

\_nop\_();

}

}

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

函数名称：LCD1602\_Delay1ms

功 能：STC 1T单片机1ms延时程序

入口参数：ms:延时的毫秒数

返 回 值：无

备 注：示波器实测1.05ms 时钟11.0592MHz

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

void LCD1602\_Delay1ms(unsigned int ms)

{

unsigned int De\_Cnt;

while( (ms--) != 0)

{

for(De\_Cnt = 0; De\_Cnt < 600; De\_Cnt++);

}

}

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

\* 函 数 名: LCD1602\_WriteCMD

\* 函数功能: 向lcd1602写指令

\* 入口参数: cmd:指令

\* 返 回: 无

\* 备 注：无

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

void LCD1602\_WriteCMD(unsigned char cmd)

{

EN\_Clr();

RS\_Clr();

RW\_Clr();

LCD1602\_Delay1us(10);

EN\_Set();

LCD1602\_Delay1us(10);

DataPort=cmd;

LCD1602\_Delay1us(10);

EN\_Clr();

}

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

函数名称：LCD1602\_WriteDAT

功 能：向lcd1602写数据

入口参数：dat：数据

返 回 值：无

备 注：无

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

void LCD1602\_WriteDAT(unsigned char dat)

{

EN\_Clr();

RS\_Set();

RW\_Clr();

LCD1602\_Delay1us(10);

EN\_Set();

LCD1602\_Delay1us(10);

DataPort=dat;

LCD1602\_Delay1us(10);

EN\_Clr();

}

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

函数名称：LCD1602\_CheckBusy

功 能：检测lcd1602忙信号

入口参数：无

返 回 值：无

备 注：无

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

void LCD1602\_CheckBusy(void)

{

unsigned char temp;

DataPort=0xff; //做输入先置高,51系列单片机读之前需要置高

while(1)

{

EN\_Clr();

RS\_Clr();

RW\_Set();

LCD1602\_Delay1us(10);

EN\_Set();

LCD1602\_Delay1us(10);

temp=DataPort; //读取忙通道数据

LCD1602\_Delay1us(10);

EN\_Clr();

if((temp&0x80)!=0x80)

{

break;

}

}

}

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

函数名称：HEX2ASCII

功 能：十六进制转ASCII函数

入口参数：dat:十六进制数

返 回 值：转换成的ASCII值。

备 注：无

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

unsigned char HEX2ASCII(unsigned char dat)

{

dat &= 0x0f;

if(dat <= 9) return (dat + '0'); //数字0~9

return (dat - 10 + 'A'); //字母A~F

}

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

函数名称：LCD1602\_Display\_Str

功 能：lcd1602显示字符函数

入口参数：str指针地址

返 回 值：无

备 注：无

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

void LCD1602\_Display\_Str(unsigned char\* str)

{

while(\*str!='\0') //未到字符串末尾

{

LCD1602\_CheckBusy(); //检测忙信号

LCD1602\_WriteDAT(\*str);//发送数据

str++; //移动指针

}

}

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

函数名称：LCD1602\_Init

功 能：LCD1602初始化

入口参数：无

返 回 值：无

备 注：详见数据手册定义。

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

void LCD1602\_Init(void)

{

//初始化P0口为准双向口

P0M1 =0x00;

P0M0 =0x00;

//初始化P41,P42口为准双向口

P4M1 &=~( (1<<1) | (1<<2) );

P4M0 &=~( (1<<1) | (1<<2) );

//初始化P37口为准双向口

P3M1 &=~(1<<7);

P3M0 &=~(1<<7);

//分析

// 1<<0等价于0x01 即 0000 0001

// 1<<1等价于0x02 即 0000 0010

// 1<<2等价于0x04 即 0000 0100

// 1<<3等价于0x08 即 0000 1000

// 以此类推1<<n 即第n位为1，其余位是0

// x |=(1<<n) 即对x执行按位取或

// 即x中的第n位置为1，不改变其他位状态

// y &=~(1<<n) 即将1<<n按位取反，然后对y按位取与

// 即y中的第n位置为0，不改变其他位状态

LCD1602\_Delay1ms(15); //上电延时15ms

LCD1602\_WriteCMD(0x38); //写显示指令(不检测忙信号)

LCD1602\_Delay1ms(5);

LCD1602\_WriteCMD(0x38); //写显示指令(不检测忙信号)

LCD1602\_Delay1ms(5);

LCD1602\_WriteCMD(0x38); //写显示指令(不检测忙信号)

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(0x38); //写显示指令

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(0x08); //显示关闭

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(0x01); //清屏

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(0x06); //显示光标移动设置

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(0x0c); //显示开及光标设置

}

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

函数名称：LCD1602\_Display\_Clock

功 能：lcd1602显示时钟

入口参数：pClock:指针地址

返 回 值：无

备 注：1602LCD显示的为字符，必须将整数转换成

相应的字符，否则显示乱码。

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

void LCD1602\_Display\_Clock(DAYTIME \*pClock)

{

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE1+2);

LCD1602\_Display\_Str("20");

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII(((pClock->Year)&0xf0)>>4)); //年高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Year)&0x0f)); //年低位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT('-'); //“-”

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Month&0xf0)>>4)); //月高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Month)&0x0f)); //月低位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT('-'); //“-”

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII(((pClock->Day)&0xf0)>>4)); //日高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Day)&0x0f)); //日低位

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE1+14);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Week)&0x0f)); //星期

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE2+4);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Hour&0xf0)>>4)); //时高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Hour)&0x0f)); //时低位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(':'); //“：”

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Minute&0xf0)>>4)); //分高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Minute)&0x0f)); //分低位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(':'); //“：”

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Second)>>4)); //秒高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Second)&0x0f)); //秒低位

}

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

函数名称：ToDateNonDispBuf

功 能：时钟设置部分消隐

入口参数：无

返 回 值：无

备 注：无

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

void ToDateNonDispBuf(unsigned char set)

{

switch(set)

{

case 1: //年

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+4);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 2: //月

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+7);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 3: //日

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+10);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 4: //星期

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+14);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 5: //时

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE2+4);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 6: //分

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE2+7);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

case 7: //秒

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE2+10);

LCD1602\_CheckBusy(); //显示空格

LCD1602\_WriteDAT(' ');

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(' ');

}break;

}

}

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

函数名称：ToTimeOriDispBuf

功 能：显示消隐前的时间值

入口参数：无

返 回 值：无

备 注：无

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

void ToDateOrgDispBuf(unsigned char set,DAYTIME \*pClock)

{

switch(set)

{

case 1: //年

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+4);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII(((pClock->Year)&0xf0)>>4)); //年高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Year)&0x0f)); //年低位

}break;

case 2: //月

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+7);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Month&0xf0)>>4)); //月高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Month)&0x0f)); //月低位

}break;

case 3: //日

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+10);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII(((pClock->Day)&0xf0)>>4)); //日高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Day)&0x0f)); //日低位

}break;

case 4: //星期

{

LCD1602\_CheckBusy(); //显示地址

LCD1602\_WriteCMD(LINE1+14);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Week)&0x0f)); //星期

}break;

case 5: //时

{

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE2+4);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Hour&0xf0)>>4)); //时高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Hour)&0x0f)); //时低位

}break;

case 6: //分

{

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE2+7);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Minute&0xf0)>>4)); //分高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Minute)&0x0f)); //分低位

}break;

case 7: //秒

{

LCD1602\_CheckBusy();

LCD1602\_WriteCMD(LINE2+10);

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Second)>>4)); //秒高位

LCD1602\_CheckBusy();

LCD1602\_WriteDAT(HEX2ASCII((pClock->Second)&0x0f)); //秒低位

}break;

}

}

**//KEY.c**

#include "key.h"

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

函数名称：Key\_Delayms

功 能：STC15系列单片机1ms延时程序

入口参数：ms:延时的毫秒数

返 回 值：无

备 注：示波器实测：1.05ms，内部时钟：11.0592MHz

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

void Key\_Delayms(unsigned int ms)

{

unsigned int i;

while( (ms--) != 0)

{

for(i = 0; i < 600; i++);

}

}

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

函数名称：Key\_Scan

功 能：按键扫描函数

入口参数：无

返 回 值：键值

备 注：无

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

unsigned char Key\_Scan(void)

{

unsigned char KeyTemp1,KeyTemp2;

unsigned char KeyValue;

//以下为S12，S13，S14独立按键扫描

//读入的端口先置高,准双向口做输入必须先置高

//1<<K12相当于1<<4 = (0001 0000) 即K12所在的P34口置高

KEYPORT |= (1<<K12) | (1<<K13) |(1<<K14);

//将读入端口不用的位屏蔽掉

KeyTemp1=KEYPORT | (~((1<<K12)|(1<<K13)|(1<<K14)));

if(KeyTemp1!=0xff) //如果有键按下

{

Key\_Delayms(10); //延时，防抖动

KeyTemp1=KEYPORT | (~((1<<K12)|(1<<K13)|(1<<K14)));//再次读取

if(KeyTemp1!=0xff)

{

while(KeyTemp1!=0xff) //等待按键释放

{

KeyTemp2=KeyTemp1;

//重新读取

KeyTemp1=KEYPORT | (~((1<<K12)|(1<<K13)|(1<<K14)));

}

switch(KeyTemp2)

{

case ~(1<<K12): //S12按下

{

KeyValue=KEY12;

}break;

case ~(1<<K13): //S13按下

{

KeyValue=KEY13;

}break;

case ~(1<<K14): //S14按下

{

KeyValue=KEY14;

}break;

}

return KeyValue;

}

else

{

return 0;

}

}

//以下为S6~S11行列按键扫描

//所有列置低

KEYPORT &=(~((1<<Column1)|(1<<Column2)|(1<<Column3)));

//读入端口先置高

//1<<Line1相当于1<<5 = (0010 0000) 即Line1所在的P35口置高

KEYPORT |= (1<<Line1) | (1<<Line2) ;

//读取按键，将读入端口不用的位屏蔽掉

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2)));

if(KeyTemp1!=0xff) //如果有键按下

{

Key\_Delayms(10); //延时，防抖动

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //再次读取

if(KeyTemp1!=0xff) //确实有按键按下

{

//将第一列拉低，其他列置高,扫描第一列

KEYPORT &=(~(1<<Column1));

KEYPORT |= (1<<Column2)|(1<<Column3);

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2)));

if(KeyTemp1!=0xff) //第一列有键按下

{

while(KeyTemp1!=0xff) //等待按键释放

{

KeyTemp2=KeyTemp1;

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //重新读取

}

switch(KeyTemp2)

{

case ~(1<<Line1): //S6按下

{

KeyValue=KEY6;

}break;

case ~(1<<Line2): //S9按下

{

KeyValue=KEY9;

}break;

}

}

//将第二列拉低，其他列置高,扫描第二列

KEYPORT &=(~(1<<Column2));

KEYPORT |= (1<<Column1)|(1<<Column3);

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //读取行

if(KeyTemp1!=0xff) //第二列有键按下

{

while(KeyTemp1!=0xff) //等待按键释放

{

KeyTemp2=KeyTemp1;

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //重新读取

}

switch(KeyTemp2)

{

case ~(1<<Line1): //S7按下

{

KeyValue=KEY7;

}break;

case ~(1<<Line2): //S10按下

{

KeyValue=KEY10;

}break;

}

}

//将第三列拉低，其他列置高,扫描第三列

KEYPORT &=(~(1<<Column3));

KEYPORT |= (1<<Column1)|(1<<Column2);

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //读取行

if(KeyTemp1!=0xff) //第三列有键按下

{

while(KeyTemp1!=0xff) //等待按键释放

{

KeyTemp2=KeyTemp1;

KeyTemp1=KEYPORT | (~((1<<Line1)|(1<<Line2))); //重新读取

}

switch(KeyTemp2)

{

case ~(1<<Line1): //S8按下

{

KeyValue=KEY8;

}break;

case ~(1<<Line2): //S11按下

{

KeyValue=KEY11;

}break;

}

}

return KeyValue;

}

else

{

return 0;

}

}

else

{

return 0;

}

}

**//ADC.c**

//文件包含

#include "stc15f2k60s2.h"

#include "hc595.h"

#include "ADC.h"

#include "lcd1602.h"

//包含该文件可以使用\_nop\_()函数

#include <intrins.h>

//全局变量

unsigned long int ADCSum=0;

unsigned int ADCResult=0;

unsigned char ADCCnt=0;

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

函数名称：ADC\_Process

功 能：ADC数据处理

入口参数：无

返 回 值：无

备 注：采集32个数据进行求平均

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

void ADC\_Process(void)

{

//从AD通道采集数据，并且进行累加

ADCSum+=ADC\_GetResult(ADC\_CHANNEL7);

//计数器加1

ADCCnt++;

//如果累加到32个数据，则开始处理

if(ADCCnt==32)

{

ADCCnt=0;

//(ADCSum>>5)等价于(ADCSum/32)

//对32个数据取平均

ADCSum=ADCSum>>5;

//ADC=(Vin/Vref)\*1024

//Vin为输入的电压值，Vref这里为电源电压VCC

//根据ADC计算公式进行转换

ADCSum=ADCSum\*VCC/1024;

//保存转换结果并进行类型转换，方便显示

ADCResult=(unsigned int)ADCSum;

//清除AD暂存变量

ADCSum=0;

//显示测量结果

ToDisplayBuf(ADCResult);

// LCD1602\_Display\_ADC(ADCResult);

}

}

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

函数名称：ADC\_Init

功 能：ADC初始化函数

入口参数：无

返 回 值：无

备 注：无

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

void ADC\_Init(void)

{

//初始化P17为输入

P1M1 |=( 1<<7);

P1M0 &=~( 1<<7);

P1ASF |=P1ASF\_7; //P1.7口做为AD转换通道

ADC\_RES=0; //ADC数据寄存器清零

ADC\_CONTR=ADC\_POWER | ADC\_SPEEDLL;

//打开AD转换器电源，设置转换速率

Delay\_ms(2); //延时2ms，等待ADC上电稳定

}

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

函数名称：ADC\_GetResult

功 能：获取ADC转换的结果

入口参数：ch:转换的通道

返 回 值：unsigned int：转换得到的数据

备 注：使用查询方式

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

unsigned int ADC\_GetResult(unsigned char ch)

{

unsigned int ADC\_Value;

ADC\_CONTR =ADC\_POWER | ADC\_SPEEDLL| ch | ADC\_START; //启动ADC

\_nop\_(); //延时

\_nop\_();

\_nop\_();

\_nop\_();

while (!(ADC\_CONTR & ADC\_FLAG));//等待AD转换完成

ADC\_CONTR &= ~ADC\_FLAG; //清除转换完成标记

ADC\_Value=ADC\_RES; //读取ADC高八位

ADC\_Value=(ADC\_Value<<2)|ADC\_RESL;//读取ADC低两位，并且数据合并

return ADC\_Value;

}

void alarm(void)

{

if(ADCResult>5000)

{

ALARM\_ON();

}

else

{

ALARM\_OFF();

}

}

**//HC595.C**

#include "hc595.h"  
//共阴数码管段码表  
 unsigned char code DispCode[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,   
//   0  1  2  3  4  5  6   7   8   9     
0x77,0x7C,0x39,0x5E,0x79,0x71,0x3D,0x76,0x74,0x30,   
//   A  b  C   d  E  F   G   H    h   I

0x10,0x1E,0x38,0x54,0x5C,0x73,0x67,0x50,0x31,0x78,  
//    i   J   L   n   o   P  q  r   R  t     
0x3E,0x1C,0x40,0x48,0x08,0x00};  
//      U    V   一    二   \_    灭  
//显示缓冲区  
unsigned char DispBuf[8];  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：HC595\_Delayms  
功    能：STC 1T单片机1ms延时程序  
入口参数：ms:延时的毫秒数  
返 回 值：无   
备    注：示波器实测1.05ms 时钟11.0592MHz  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void HC595\_Delayms(unsigned int ms)  
{  
  unsigned int i;  
  while( (ms--) != 0)  
  {  
    for(i = 0; i < 600; i++);   
  }               
}   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：HC595\_SendWord  
功    能：HC595发送两个字节数据。  
入口参数：dat:显示的字型码  
dig:点亮的数码管位 dig中为0的位，则对应的数码管点亮  
          例如:dig=0x1111 1110,将dig送入U3芯片HC595后，最低位的0数据  
          将出现在原理图的Q0管脚，则该脚对应的数码管点亮，其他数码管熄灭。  
返 回 值：无   
备    注：两片HC595，发送两个字节数据，高字节  
          为数码管段码，低字节为选择点亮哪个数码管。  
该程序点亮数码管可以结合原理图来看，  
否则不好理解，和电路原理关系密切。  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void HC595\_SendWord(unsigned char dat,unsigned char dig)  
{  
unsigned char i;        //定义计数器  
unsigned int temp=0;    //定义数据变量  
//将两个8位变量合并成一个16字节变量  
temp=dat;  
temp=temp<<8  | dig;  
//送出数据  
for(i=0;i<16;i++)     
  {   
HC595\_SCK\_Clr();      //置低时钟线  
if( (temp&0x8000)==0x8000 )//先发送高字节  
  {  
    HC595\_DAT\_Set();    //置高信号线  
  }  
  else  
  {  
HC595\_DAT\_Clr();    //置低信号线  
  }   
HC595\_SCK\_Set();    //置高时钟线  
  
  temp<<=1;      //数据左移   
}  
//锁存信号上升沿锁存数据 数据输出到端口   
HC595\_RCK\_Clr();     
HC595\_RCK\_Set();   
}   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：ToDisplayBuf  
功    能：把待显示的数据放入显示缓冲区  
入口参数：  
Low\_Dat： 低四位数码管数据  
返 回 值：无   
备    注：无  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void ToDisplayBuf(unsigned int Low\_Dat)  
{  
  DispBuf[4]=Low\_Dat%10000/1000; //低四位千位  
DispBuf[5]=Low\_Dat%1000/100;  //低四位百位  
DispBuf[6]=Low\_Dat%100/10;  //低四位十位  
DispBuf[7]=Low\_Dat%10;  //低四位个位   
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：LED\_Display  
功    能：数码管显示函数  
入口参数：无  
返 回 值：无   
备    注：无  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void LED\_Display(void)  
{  
  unsigned char i;   
  for(i=0;i<8;i++)  
{   
//将待显示的字型码和点亮的数码管位送至HC595。  
//由于是共阴数码管，将八段数码管共阴极拉低，显示数据。  
//DispBuf[i]:为显示的内容  
//DispCode[DispBuf[i]:为将显示的内容转换成对应的字形码  
//数码管按字形码显示出来就是我们看到的显示内容  
if(i==4)  
{  
//增加小数点  
HC595\_SendWord(DispCode[DispBuf[i]]|0x80,(~(0x01<<i)));   
}  
else  
{  
//~(0x01<<i),当i=0时等价于:0x1111 1110 当i=1时等价于：0x1111 1101  
//以此类推，依次有一位为0，为0的位对应的数码管点亮  
HC595\_SendWord(DispCode[DispBuf[i]],(~(0x01<<i)));   
}  
HC595\_OE\_Clr();   //打开显示  
HC595\_Delayms(1); //适当延时        
HC595\_OE\_Set();   //关闭显示  
}     
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：HC595\_Clear  
功    能：清除显示函数  
入口参数：无  
返 回 值：无   
备    注：无  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void HC595\_Clear(void)  
{   
unsigned char i;  
//清除显示缓冲区内容  
for(i=0;i<8;i++)  
{  
//35为DispCode[35]中定义的数码管熄灭的字形码  
DispBuf[i]=35;  
}  
//数据0x00为字形码，0xff为数码管位码  
//由原理图可知该开发板使用共阴极数码管  
//共阴极数码管特点是字形码端高电平，位码端低电平，数码管才能点亮  
//现在发送的字形码全为低电平，位码全为高电平，所以全部熄灭  
HC595\_SendWord(0x00,0xff);     
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
函数名称：HC595\_Init  
功    能：显示初始化函数  
入口参数：无  
返 回 值：无   
备    注：无  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void HC595\_Init(void)  
{   
//等待HC595上电稳定  
HC595\_Delayms(10);   
//初始化P41,P42,P45口为准双向口  
P4M1 &=~( (1<<1) | (1<<2) | (1<<5));    
P4M0 &=~( (1<<1) | (1<<2) | (1<<5));   
//初始化P37口为准双向口  
P3M1 &=~(1<<7);    
P3M0 &=~(1<<7);   
//分析  
//  1<<0等价于0x01 即 0000 0001  
//  1<<1等价于0x02 即 0000 0010  
//  1<<2等价于0x04 即 0000 0100  
//  1<<3等价于0x08 即 0000 1000  
//  以此类推1<<n 即第n位为1，其余位是0  
//  x |=(1<<n)  即对x执行按位取或   
//  即x中的第n位置为1，不改变其他位状态    
//  y &=~(1<<n)  即将1<<n按位取反，然后对y按位取与   
//  即y中的第n位置为0，不改变其他位状态   
  
//将端口置低，赋予端口固定电平  
HC595\_DAT\_Clr();  
HC595\_SCK\_Clr();  
HC595\_RCK\_Clr();  
//关闭显示  
HC595\_OE\_Set();   
//清除显示内容  
HC595\_Clear();   
}