韶 关 学 院

毕 业 设 计

题　　目：基于单片机温度控制电路的设计与制作

学生姓名：

学　　号：

院　（系）：物理与机电工程学院　电子系

专　　业：电子信息科学与技术

班　　级：2007 级

指导老师姓名及职称：凌晟 讲师

起止时间： 2010 年 9 月—— 2011 年 5 月

基于单片机温度控制电路的设计与制作

摘　要：本设计以 STC89C52RC 单片机为控制核心，由实时时钟芯片 DS1302 和数字温度传感器 DS18B20 构成了一个高温和低温时,分别对相应的器件进行控制的系统。详细地介绍了整个系统的硬件组成结构、工作原理和系统的软件程序设计，重点阐述了时钟模块、显示模块、温度检测模块及相关控制模块等的模块化设计思路与制作。系统采用液晶 LCD1602 作为显示器，具有实时时间显示、环境温度显示，显示系统连续工作前 24 小时整点的温度值。在本设计中，软件程序均采用 C 语言编写，便于移植与升级。

关键词： STC89C52RC； 单片机； DS1302； LCD1602； 温度控制；

Based on single-chip microcomputer temperature control circuit design and production

Abstract： This design with STC89C52RC microcontroller as control core, by real time clock chip DS1302 and digital temperature sensor DS18B20 constitute a high temperature and low temperature respectively on the corresponding device to control system. Introduced the whole systems hardware structure, working principle and system software program design in detail, expounds the clock module, display module, temperature detecting module and related control module of modularization design and production. System adopts LCD monitor, LCD1602 as with real-time time display, environmental temperature display, display the system worked 24 hours before the temperature on the hour value. In this design, software program all use written in C language, for transplantation and upgrades.

Key words: STC89C52RC； single-chip； ds1302； lcd1602；Temperature control；

目录

基于单片机温度控制电路的设计与制作

专业班级：
指导教师：

1 背景与意义

1.1　背景

随着社会的发展,科技的进步,以及测温仪器在各个领域的应用,智能化已是现代温度控制系统发展的主流方向。特别是近年来，温度控制系统已应用到人们生活的各个方面，设计一个温度控制系统，具有广泛的应用前景与实际意义。温度是科学技术中最基本的物理量之一，物理，化学，生物等学科都离不开温度。在工业生产和实验研究中，像电力，化工，石油，冶金，航空航天，机械制造，粮食存储， 酒类生产等领域内，温度常常是表征对象和过程状态的最重要的参数之一。比如，发电厂锅炉的温度必须控制在一定的范围之内；许多化学反应的工艺过程必须在适当的温度下才能正常进行；炼油过程中，原油必须在不同的温度和压力条件下进行分馏才能得到汽油，柴油，煤油等产品。没有合适的温度环境，许多电子设备就不能正常工作，粮仓的储粮就会变质霉烂，酒类的品质就没有保障。因此，各行各业对温度控制的要求都越来越高。可见，温度的测量和控制是非常重要的。单片机在电子产品中的应用已经越来越广泛，在很多的电子产品中也用到了温度检测和温度控制。随着温度控制器应用范围的日益广泛和多样，各种适用于不同场合的智能温度控制器应运而生。比较成熟的产品就有水温测控温度控制系统和语音报警的温度监控仪等。其中水温测控温度控制系统的功能可以实现从常温开始对自来水加温，加热到人工设定的温度的恒温控制。而语音报警的温度监控仪可以通过控制空调对温度进行自动调节，使被控环境的温度达到要求的范围，并能实现对所控区域内环境温度的自动监控的远程智能调控系统。

1.2　意义

基于单片机温度控制的测试控制系统，控制对象是温度。温度控制在日常生活及工业领域应用相当广泛，比如温室，水池，发酵缸，电源等场所的温度控制。

而以往温度控制是由人工完成的而且不够重视，其实在很多场所温度都需要监控以防止发生意外。针对此问题，本系统设计的目的是实现可以根据外界环境温度控制电机的温度控制系统，它应用广泛，功能强大，小巧美观，便于携带，是一款既实用又廉价的控制系统。

1.3　功能介绍

1. 3. 1 按 Model 键可切换设置模式，液晶屏显示相应模式页面。

1. 3. 2 当液晶屏显示页面 1 时，即液晶屏的第一行显示年、月、日、星期，第二行显示时、分、秒、温度时，按 Set 键,可实现风扇的自动控制和手动控制工作方式的切换。

1. 3. 3 当液晶屏显示页面 2 时，是时间设置模式，在该模式下按一下 Set 键后，通过按左键、右键、加键和减键就可以对时间进行设置，设置完后再按 Set 键后就可以保存设定的时间。

1. 3. 4 当液晶屏显示页面 3 时，可查看系统连续工作时前 24 小时内整点的温度值，即系统连续工作 24 小时后，输入整点的时间，液晶屏显示对应该点时间的温度值。

1. 3. 5 当液晶屏显示页面 4 时，是继电器工作情况的显示，继电器用于控制加热设备的工作。当光标在 AuTo/Manuel 时，按加或减键可设置继电器的自动或手动工作方式，手动（Manuel）工作方式时，继电器一直打开，此时，风扇无论何种情况下都是按照风扇本身最大速度运转；自动时，由设定温度控制继电器的开或关，当环境温度低于设定温度时，继电器才打开。

1. 3. 6 当液晶屏显示页面 5 时，是显示风扇自动工作时不能打开的时间段显示，即是在液晶页面显示的时间范围内，自动工作时，风扇不允许打开，直到过了这个时间段，自动控制才能再次正常工作。这个时间段是通过程序设定的，而且设定好之后不允许再次修改。

2 方案比较与论证

2.1 设计任务与要求

2. 1. 1 当传感器检测出的环境温度偏低时,控制继电器,实现电暖炉的开与关的状态。

2. 1. 2 当传感器检测出的环境温度偏高时,随着温度的改变,控制电机的转速作出相应的改变。

2. 1. 3 通过时钟芯片 DS1302 自动控制电机,使其在某个时间段不工作。

2. 1. 4 当环境温度超出了设定值时,蜂鸣器发出声响报警。

2.2 方案比较与论证

2.2.1 方案比较与论证

方案一：单片机按照一定的控制算法对采集的温度数据进行处理，得到控制量，以控制电机的功率，从而实现风扇转速的控制。传感器采用集成的 AD599，但是这个方案的电路结构十分复杂，A/D 转换器的精度实现既定功能的困难很大，而且由于器件很多，使得单片机 89C51 的内部资源不能满足需要，调试和安装都十分不方便，同时实现扩展功能困难。方案组成方框图如图 1 所示：

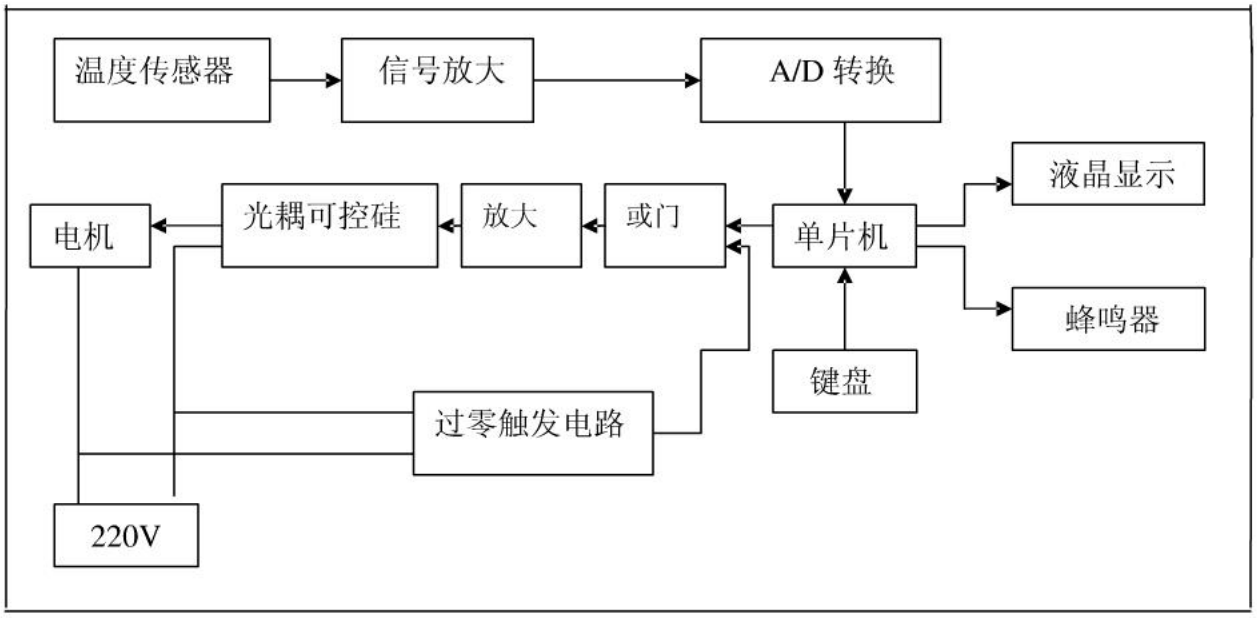


图 1 方案一组成方框图

方案二：采用数字式温度传感器 DS18B20，它能够将温度直接转换成数字信号，可以通过一根数据线与单片机进行通信，而且它不需要外部元件，在-10℃～85℃范围内可以精确到+0.5℃。完全满足设计要求。这样设计就可以不使用 A/D 转换器，从而使系统的精度得以提高，也能够大大节省单片机得系统资源，所以我又加了 DS1302 时钟模块电路，使时间能够实时显示。方案组成方框图如图 2 所示：

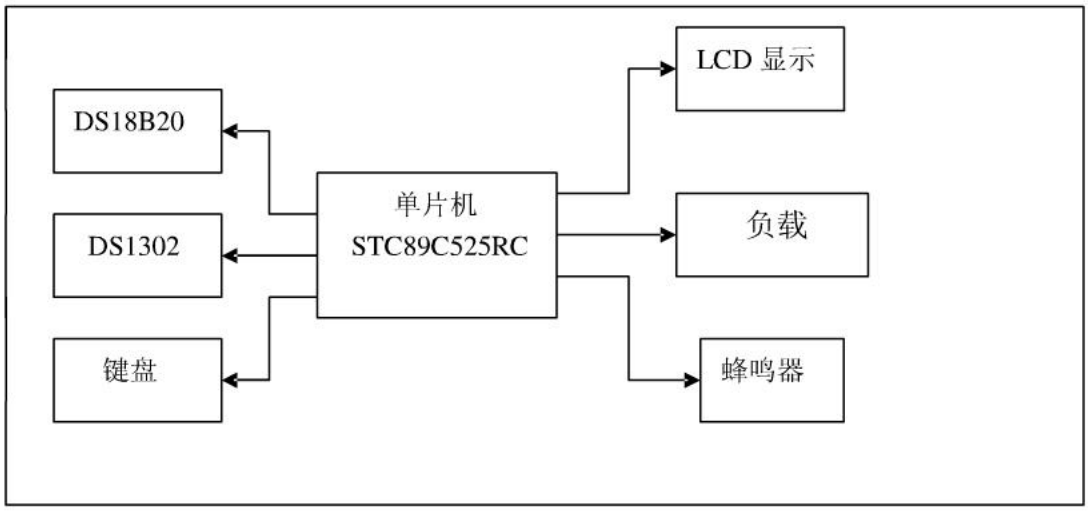


图 2 方案二组成方框图

2.2.2 方案的选择

通过以上两种方案的比较，我选择的是方案二作为设计方案，因为方案二与方案一相比，它的硬件系统更简单，但是功能却更强大，它本身的模块化设计又使它的系统通用性远远超过方案一，在现在的条件下我能够完成这个方案，所以最终选择了这个方案。

3 系统硬件设计

3.1 总体电路框图

本设计以 STC89C52RC 单片机为主控核心设计的一个温度控制系统，低温时可控制加热设备，高温时控制风扇，超出设定最高温度值时蜂鸣器发出声响报警。硬件方框图如图 3 所示：

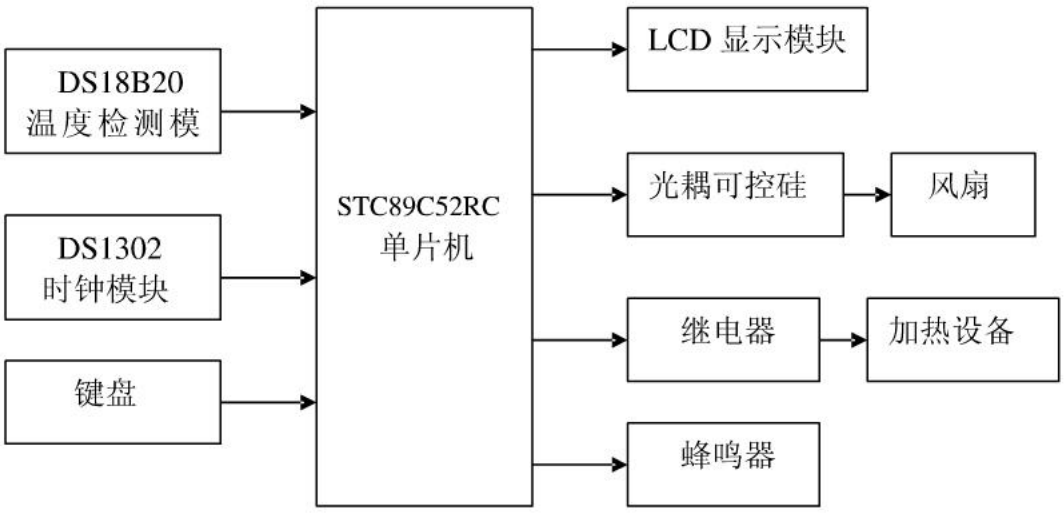


图 3 总体硬件方框图

3.2 单元电路设计

3.2.1 单片机模块

STC89C52RC 单片机为 40 引脚双列直插芯片,有四个 I/O 口 P0,P1,P2,P3,MCS-51 单片机共有 4 个 8 位的 I/O 口（P0、P1、P2、P3），每一条 I/O 线都能独立地作输出或输入。

单片机的最小系统如下图所示,18 引脚和 19 引脚接时钟电路,XTAL1 接外部晶振和微调电容的一端,在片内它是振荡器倒相放大器的输入,XTAL2 接外部晶振和微调电容的另一端,在片内它是振荡器倒相放大器的输出。第 9 引脚为复位输入端,接上电容,电阻及开关后够上电复位电路，20 引脚为接地端，40 引脚为电源端。31 引脚接电源端[9-11]，如图 4 所示：

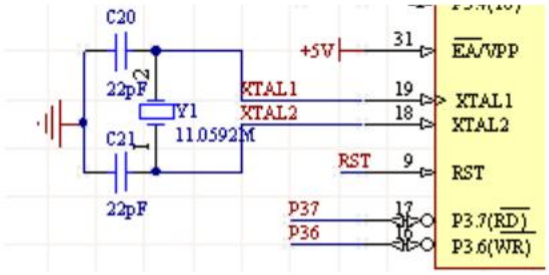


图 4 单片机电路

3.2.2 时钟模块

DS1302 是美国 DALLAS 公司推出的一种高性能，低功耗的实时时钟芯片，附加 31 字节静态 RAM，采用 SPI 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和 RAM 数据。实时时钟可提供秒，分，时，日，星期，月和年，一个月小于 31 天时可以自动调整，且具有闰年补偿功能。工作电压宽达 2.5V～5.5V[234-243]。采用双电源供电（主电源和备用电源），可设置备用电源供电方式，提供了对后背电源进行涓细电流充电的能力。DS1302 与单片机的连接仅需要 3 条线：RST 引脚、SCLK 串行时钟引脚、I/O 串行数据引脚，由 Y2 组成 DS1302 时钟振荡电路，提供计时脉冲，其中 Y2 为 32.768MHz。然后其中 SCLK, I/O, RST 分别接主控单片机的 P22, P23, P24 脚。电路原理图如图 5 所示：

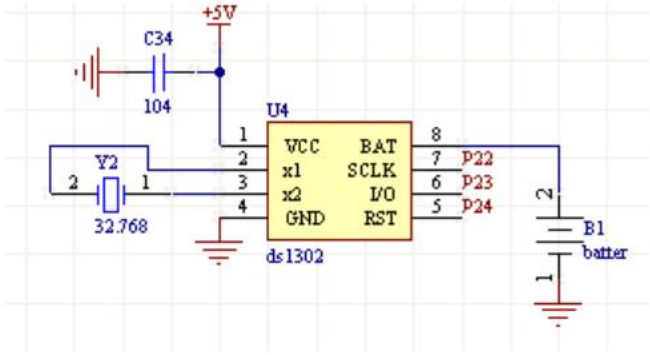


图 5　DS1302 原理图

3.2.3 温度采集模块

DS18B20 支持“一线总线”，故可以大大提高系统的抗干扰性，适合于恶劣的环境温度测量。全部传感元件及转换电路集成在形如一只三极管的集成电路内，体积小。同时 DS18B20 的测量温度范围为-55 ℃～+125 ℃, 在-10～+85 ℃ 范围内，精度为±0.5 ℃。同样的，DS18B20 还可以程序设定 9～12 位的分辨率，精度为±0.5

℃。同时设定的报警温度存储在 EEPROM 中，掉电后依然保存。并且支持 3V～5V 的电压范围。便于系统设计。

DS18B20 的主要特点：

适应电压范围更宽，电压范围：3.0～5.5V，在寄生电源方式下可由数据线供电；

独特的单线接口方式，DS18B20 在与微处理器连接时仅需要一条口线即可实现微处理器与 DS18B20 的双向通讯；

DS18B20 支持多点组网功能，多个 DS18B20 可以并联在唯一的三线上，实现组网多点测温；

DS18B20 在使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内；

温范围－55℃～＋125℃，在-10～+85℃时精度为±0.5℃；

可编程的分辨率为 9～12 位，对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃，可实现高精度测温；

在 9 位分辨率时最多在 93.75ms 内把温度转换为数字，12 位分辨率时最多在 750ms 内把温度值转换为数字，速度更快；

测量结果直接输出数字温度信号，以"一线总线"串行传送给 CPU，同时可传送 CRC 校验码，具有极强的抗干扰纠错能力；

负压特性：电源极性接反时，芯片不会因发热而烧毁，但不能正常工作[249-256]。

对 DS18B20 的设计外部供电方式单点测温。在这种外部电源供电方式下，DS18B20 工作电源由 VDD 引脚接入，因为由 VDD 接入电源不存在电源电流不足的问题，可以保证转换精度。不过要注意。在这种外部供电的方式下，DS18B20 的 GND 脚不能悬空，否则不能转换温度，读取的温度总是 80 ℃。DS18B20 的硬件电路连接如下图 6 所示：

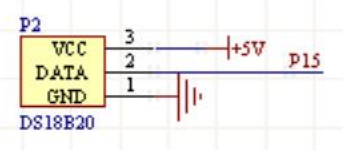


图 6　DS18B20 原理图

3.2.4 液晶显示模块

字符型液晶显示模块是一种专门用于显示字母、数字、版本号等的点阵式液晶显示模块。它是由若干个 5×7 或 5×11 等点阵符位组成的，第一个点阵字符位都可以显示一个字符。点阵字符位之间有一定点距的间隔，这样就起到了字符间距和行距的作用。本系统采用字符型液晶显示模块 1602，我设置单片机驱动 LCD1602 采用并行方式，RS,RW,EN 分别接主控单片机的 P25,P26,P27 脚，DB0～DB7 接到主控单片机的 P0 数据接口。BLA 接口通过一个+5V 电源,BLK 接地。LCD1602 的硬件连接原理图如图 7 所示：

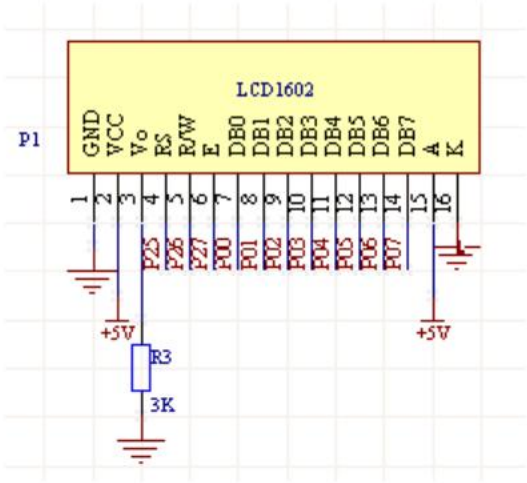


图 7 液晶显示 LCD1602 原理图

3.2.5 按键模块

我在本设计中加入了 5 个按键，其中 S1 为模式键，按一次，选择显示当前时间与温度模式,按第二次,选择显示日期和时间设置模式,按第三次,选择查看前 24 小时整点温度模式，按第四次，选择显示继电器工作情况模式，按第五次，显示风扇自动工作时不能工作的时间段。S2 为设置键,用于设置风扇手动跟自动工作方式的设置及锁定 S1 选择的模式。S3 是对选中位的数据进行加 1，S4 是对被选中位数据进行减 1，S5 是左移键设置数据时,若按一次则光标向当前所设数据左移一位,按两次,则再移一位,依此类推。S6 是右移键设置数据时,若按一次则光标向当前所设数据右移一位,按两次,则再移一位,依此类推。如图 8 所示：

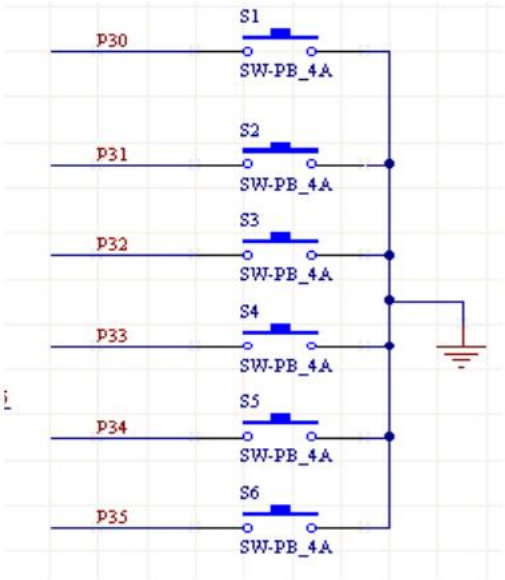


图 8　按键原理图

4　系统软件设计

4.1　总体软件框图

总体软件框图如图 9 所示：

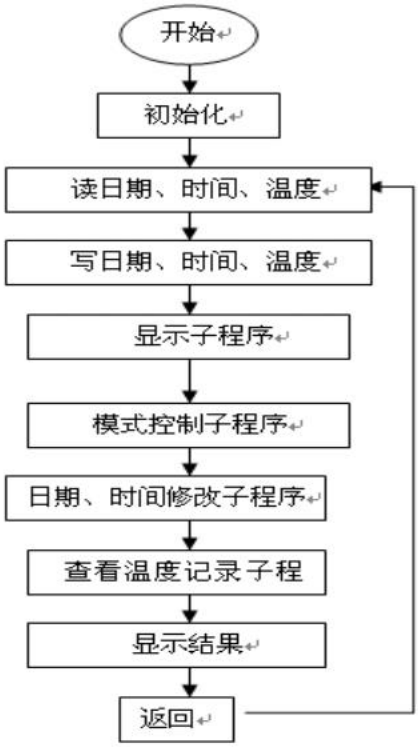


图 9 总体软件框图

4.2　各模块程序设计

4.2.1 时钟驱动程序：

DS1302 是 SPI 总线驱动方式。它不仅要向寄存器写入控制字，还需要读取相应寄存器的数据。下图图 10 是 DS1302 的控制字。

7	6	5	4	3	2	1	0
1	RAM $\overline{CK}$	A4	A3	A2	A1	A0	RD $\overline{WR}$

图 10 控制字（即地址和命令字节）

控制字的最高有效位（位 7）必须是逻辑 1，如果它为 0，则不能把数据写入到 DS1302 中。

位 6：如果为 0，则表示存取日历时钟数据，为 1 表示存取 RAM 数据；位 5 至位 1（A4～A0）：指示操作单元的地址；位 0（最低有效位）：如为 0，表示要进行写操作，为 1 表示进行读操作。控制字总是从最低位开始输出。在控制字指令输入后的下一个 SCLK 时钟的上升沿时，数据被写入 DS1302，数据输入从最低位（0 位）开始。同样，在紧跟 8 位的控制字指令后的下一个 SCLK 脉冲的下降沿，读出 DS1302 的数据，读出的数据也是从最低位到最高位。数据读写时序如下图 11 所示：

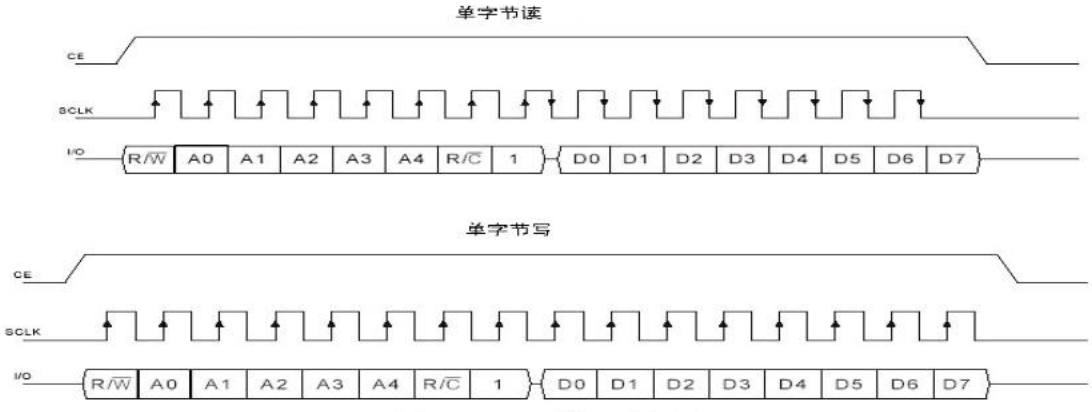


图 11 DS1302 读写时序图

4.2.2 温度数据采集：

根据 DS18B20 的通讯协议，单片机控制 DS18B20 完成温度转换必须经过三个步骤：每一次读写之前都要对 DS18B20 进行复位操作，复位成功后发送一条 ROM

指令，最后发送 RAM 指令，这样才能对 DS18B20 进行预定的操作，复位要求单片机将数据线下拉 50 微秒，然后释放，当 DS18B20 受到信号后等待 16～60 微秒左右，然后发出 60～240 微秒的存在低脉冲，主 CPU 收到此信号表示复位成功。

指令	约定代码	功能
读 ROM	33H	读 DS1820 温度传感器 ROM 中的编码（即 64 位地址）
符合 ROM	55H	发出此命令之后，接着发出 64 位 ROM 编码，访问单总线上与该编码相对应的 BS1820 使之作出响应，为下一步对该 DS1820 的读写作准备。
搜索 ROM	OFOH	用于确定挂接在同一总线上 DS1820 的个数和识别 64 位 ROM 地址。为操作各器件作好准备。
跳过 ROM	OCCH	忽略 64 位 ROM 地址，直接向 DS1820 发温度变换命令。适用与单片工作。
告警搜索命令	OECH	执行后只有温度超过设定值上限或下限的片子才做出响应。

指令	约定代码	功能
温度变换	44H	启动 DS1820 进行温度转换 12 位转换时最厂为 750ms(9 位为 93.75ms)。结果存入内部 9 字节 RAM 中。
读暂存器	OBEH	读内部 RAM 中 9 字节的内容。
写暂存器	4EH	发出向内部 RAM 的 3、4 字节写上、下限温度数据命令，紧跟该命令之后，是传送两字节的数据。
复制暂存器	48H	将 RAM 中第 3、4 字节的内容复制到 EEPROM 中
重调 EEPROM	OB8H	将 EEPROM 中内容恢复到 RAM 中的第 3、4 字节。
读供电方式	OB4H	读 DS1820 的供电模式。寄生东佃时 DS1820 发送“0”，外界电源供电 DS1820 发送“1”。

下图 12 是 DS18B20 的初始化和读写时序：

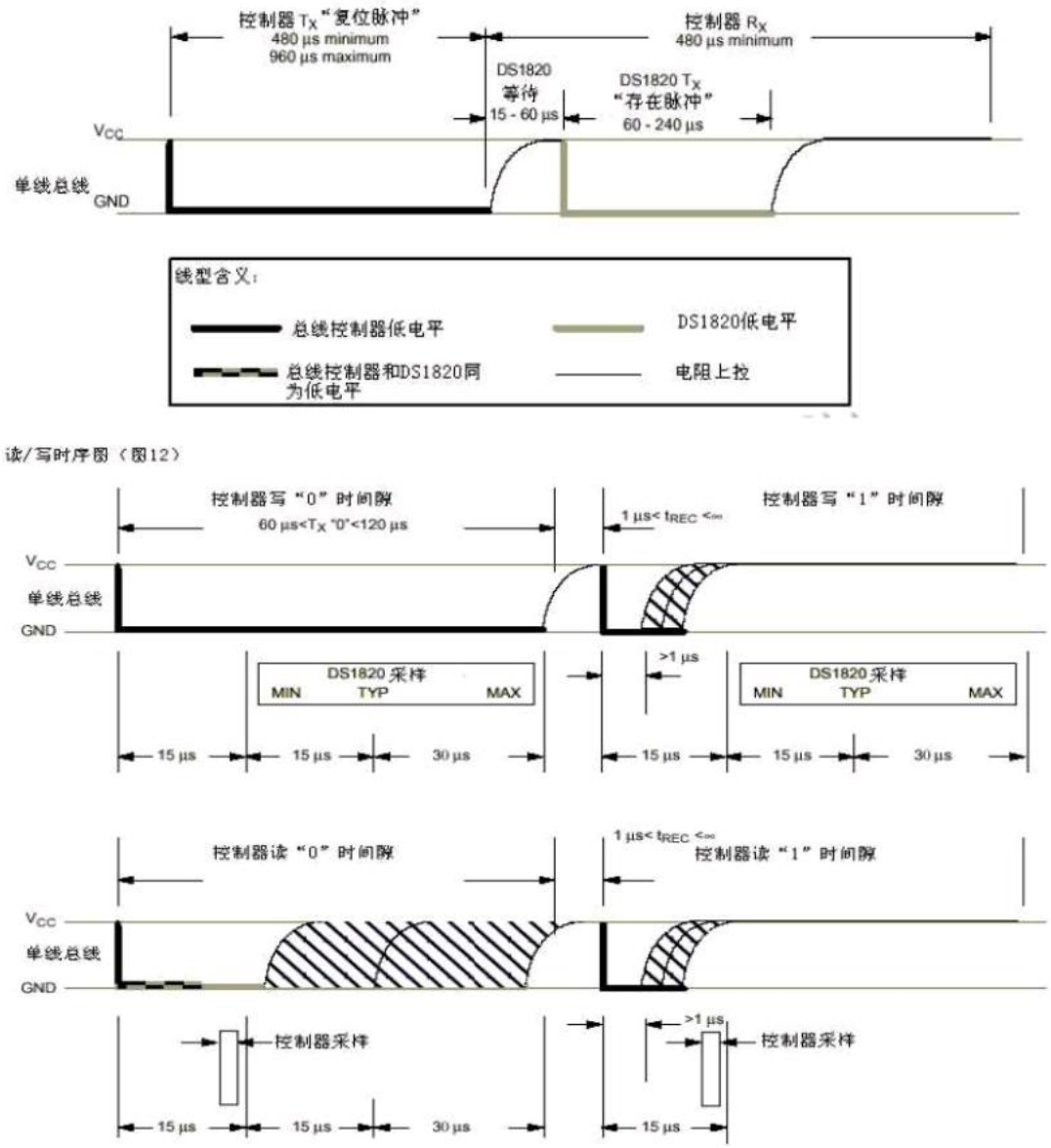


图 12 DS18B20 读写时序

4.2.3 液晶驱动程序：

LCD 使用之前须对它进行初始初始化可通过复位完成，也可在复位后完成，初始化过程如下：

(1)清屏。将显示缓冲区 DDRAM 的内容全部写入空格(ASCII20H)。

(2)功能设置。

(3)开/关显示设置。控制显示的开关，当 D=1 时显示，D=0 时不显示。控制光标开关，当 C=1 时光标显示，C=0 时光标不显示。控制字符是否闪烁，当 B=0 时字符闪烁，B=0 时字符不闪烁。

(4)输入方式设置。

初始化过程：（1）延时 15ms；（2）写指令 38H（不检测忙信号）；（3）延时 5 ms；（4）写指令 38H（不检测忙信号）；（5）延时 5ms；（6）写指令 38H（不检测忙信号）；（7）以后每次写指令、读/写数据操作之前均需检测忙信号；（8）写指令 38H：显示模式设置；（9）写指令 08H：显示关闭；（10）写指令 01H：显示清屏；（11）写指令 06H：显示光标移动设置；（12）写指令 0CH：显示开及光标设置。

本系统中液晶显示器的初始化程序流程如图 13 所示：

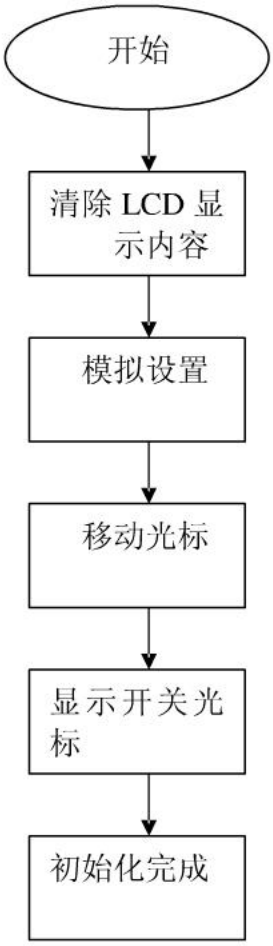


图 13　LCD 初始化程序流程图

5 系统调试与测试

5.1 硬件调试

在本温度控制电路的设计调试中遇到了很多的问题。回想这些问题只要认真多思考都是可以避免的，以下为主要的问题：

（1） 开始调试时发现时钟芯片 DS1302 发热很厉害,后来发现电路没有给它加上拉电阻，加上上拉电阻后问题就解决了。

（2） 双向可控硅 BTA12-600B 高压部分电路，加 104 电容时，发生击穿现象，导致光耦控制部分电路瘫痪，原因是耐压值不够。解决的方法有两种，第一种是更换耐压值更高的电容；由于该电容的作用是吸收可控硅元件的残余电量，使得可控硅能够正常导通截止，而根据使用的负载（交流电机），去掉该电容后，该电路能够正常的工作，所以第二种是在电路可以运行的状况下去掉该 104 电容相连的电路。我采取的是第二种方法。

5.2 软件调试

在软件调试时也出现了一些问题，其中主要的问题有以下两个方面：

（1） 程序读取温度值时，出现的现象，造成风扇温度在判断时出现错误，使风扇经常性突快突慢变化，蜂鸣器也不断的蜂鸣报警。解决的方法是在读取温度判断时加延时，并且多次判断，防止跳变。

（2） 调节系统参数时，液晶光标太快以致调节的时候观察困难，原因是刷新液晶太快。解决的方法是在相应数据更该时，才开始刷新液晶内容。

6 设计总结

在整个设计过程中，充分发挥人的主观能动性，自主学习，学到了许多没学到的知识。较好的完成了作品，达到了预期的目的，完了最初的设想。但是在做板时由于之前考虑的不够周全,时钟芯片没有加到上拉电阻,由于是做好了板把元件焊上去才发现的,只能在板的后面把 3 个上拉电阻焊上去,所以导致了整块板的看起来不是很美观。通过这一次的经验我意识到了对电路的设计、布局要先

有一个好的构思，而且要认真仔细的检查各个功能模块的具体情况，确保不遗留什么元件，才做出美观、大方的电路板。程序编写中，先研究各个功能模块的程序，包括时钟模块的程序，温度模块的程序，不懂的就通过查资料或者请教老师和同学来解决，然后再整理好这些程序，最终完成了能实现整个设计要求的程序。在此次设计中，知道了做凡事要有一颗平常的心，不要想着走捷径，也练就了我们的耐心和细心，做什么事都要认真仔细，因为细节决定成败。总之，这次设计使我的能力得到了全方位的提高。

致　谢

这次的设计和论文是在各位老师的悉心指导下完成的。你们严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到项目的最终完成，你们都始终给予我细心的指导和不懈的支持。在此谨向老师们致以诚挚的谢意和崇高的敬意。

在此，我还要感谢在一起愉快的度过大学四年的 07 级电子本科班的同学们，你们的帮助和支持，我才能克服一个一个的困难和疑惑，直至本文的顺利完成。

最后，衷心感谢在百忙之中抽出宝贵时间对此论文进行评阅与审议的老师们。感谢学院给我提供了一个展现自己的舞台，给我一次难得煅炼的机会，使得我的动手能力和专业技能都有了很大的提高。

参考文献

[1]沙占友 王彦明 孟志永. 单片机外围电路设计[M]. 北京: 电子工业出版社, 2003, 1

[2]李建忠. 单片机原理及应用(第二版) [M]. 西安: 西安电子科技大学出版社, 2008, 2

[3]谭浩强. C 程序设计(第三版) [M]. (清华大学出版社) 2006. 11

[4]求是科技. 单片机典型模块设计实例导航[M]. 北京: 人民邮电出版社, 2004

[5]华成英 童诗白. 模拟电子技术基础(第四版) [M]. (高等教育出版社) 2006. 1

[6]缪家鼎 徐文娟 牟同升. 光电技术[M]. 浙江大学出版社, 1996. 3

[7]吴大正. 电路基础（第二版）(西安电子科技大学出版社）2000. 7

[8]袁小平. 电子技术综合设计教程（第一版）(机械工业出版社）2008. 4

[9]康华光， 邹寿彬，电子技术基础数字部分(第四版) 北京：高等教育出版社 1999

[10]杜肤生， 数字集成电路应用精粹, 北京：人民邮电出版社 2001

[11]陈大钦，电子技术基础实验(第二版), 北京：高等教育出版社 2000

[12]全国大学生电子设计竞赛组委会, 第五界全国大学生电子设计竞赛获奖作品选编. 2001. 北京理工大学出版社， 2003

[13]中国计量出版社组编，新编电子电路大全，北京：中国计量出版社， 2001. 1

[14]葛汝明,《电子技术实验与课程设计》, 山东： 山东大学出版社 2004

[15]周永金,《 模拟电子技术及应用 》, 西安：陕西国防学院电子教研室 2005

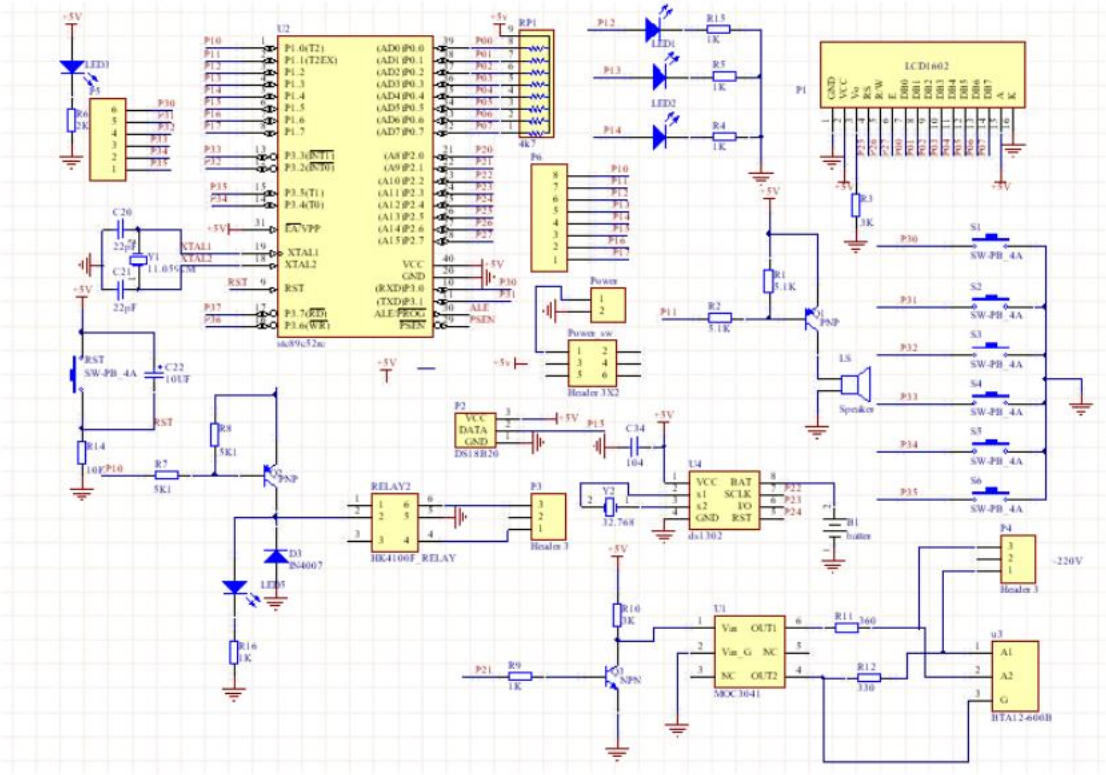
[16]吴玮玮， PROTEL 99 简明应用教程 , 西安：陕西国防学院电子教研室 2006

[17]任元，吴勇,《 常用电子元器件简明手册 》, 北京：工业出版社 2000

[18]程路，郑毅，向先波，PROTEL 99SE 电路板设计与制作, 人民邮电出版社 2007

附录

附录 A：电路图



附录 B：程序清单

```
#include<reg52.h>
#include<intrins.h>
#include<ds1302.h>
#define uchar unsigned char
#define uint unsigned int
#define LCD_dat P0 //定义 lcd 数据口

sbit lcd_E=P2^7;   //定义 lcd 控制口
sbit lcd_RW=P2^6; //定义 lcd 控制口
sbit lcd_RS=P2^5; //定义 lcd 控制口

sbit DS18B20_DQ =P1^5;   //定义 DS18B20 通信端口

sbit LED1=P1^2;
sbit LED2=P1^3;
sbit speaker=P1^1; //蜂鸣器接口
sbit fan=P2^1;//风扇 pwm 输出控制口
bit fan_flag=0;
sbit key_model=P3^1;//模式键
sbit key_set=P3^2;//设置键
sbit key_add=P3^5;//加键
```

```
sbit key_sub=P3^4;//减键
sbit key_left=P3^0;//左移键
sbit key_right=P3^3;//右移键
sbit E_sw=P1^0;//继电器控制
uchar sw_flag=0;//继电器自动手动切换标志
uchar switch_flag=0;//继电器液晶设置切换标志
int sw_open_temperature=100;//继电器开启的温度值设置
uchar restrict_hour=0;//设定时间风扇不能转
uint temp=0; //温度缓存
uint fan_count=0;//风扇计数
bit fan_AUTO=0;//自动风扇自动或手动控制标志
uchar model_flag=0;     //总的模式标志

uchar code num_to_char[] =        {"0123456789ABCDEF"}; /*定义数字跟 ASCII 码的关系*/
uchar data lcd1602_line1[]=       {" 2010/00/00 TUE "}; //时间显示缓存数组
uchar data lcd1602_line2[]=       {"00:00:00 00.0C    "}; //时间显示缓存数组
uchar data lcd1602_line3[]=       {"00:00:00set time"}; //时间设置缓存数组
uchar idata temp_record_line1[]={"Temperature note"};//温度记录显示缓存数组
uchar idata temp_record_line2[]={"time 00 T:00.0c "};//温度记录显示缓存数组
uchar xdata temp_highter_line1[]={"sw T on/off       "};//继电器手动自动切换和自动时开启
电压
uchar xdata temp_highter_line2[]={"AUTO     T=00.0C "};//继电器手动自动切换和自动时开
启电压
uchar xdata temp_time_unable_line1[]={"fan time unable"}; //显示风扇禁止打开的时间段
uchar xdata temp_time_unable_line2[]={"F:4 T:7 o'clock"}; //显示风扇禁止打开的时间段
uchar code temp_AUTO_switch[][6]={{"AUTO    "},{"Manuel"}}; //自动，手动切换值
uchar                                                                    idata
temp_record[]={"01000000000000000000000000000000000000000000000000000000000
000000000"}; //保存每天对应钟点的温度值
uchar code Weeks[][4]={{"SUN "},{"MON "},{"TUE "},{"WED "},{"THU "},{"FRI "},{"SAT
"},{"SUN "}}; //星期数组
//时钟标志
uint data year=0;         //时间设置标志，数据
uchar                                                                    data
set_time=0,time_flag=0,day_flag=0,hour=0,minute=0,second=0,day=0,month=0,week=0;// 时 间
设置标志，数据
uint speaker_count=0;     //喇叭蜂鸣计数
uchar record_hour,record_minute,record_second,record_hour_count=0;    //温度记录的时间设
置


//小延时程序
void delay(uint x)
      {while(x--);
      }
/*1MS 为单位的延时程序*/
void delay_1ms(uint x)
{
      uchar j;
      while(x--){
            for(j=0;j<125;j++)
                  {;}
            }
}

//lcd 写命令字写入
void write_instruction(uchar x)
       {delay(100);
```

```
        lcd_E=0;
        lcd_RW=0;
        lcd_RS=0;
        LCD_dat=x;
        lcd_E=1;
        delay(20);
        lcd_E=0;
       }

//LCD 写数据（单个字符）
void write_data(uchar x)
      {delay(100);
       lcd_E=0;
       lcd_RW=0;
       lcd_RS=1;
       lcd_E=1;
       LCD_dat=x;
       lcd_E=0;
      }

//LCD 写数据（字符串）
void write_string(uchar x,uchar *p)
      {write_instruction(x);
       while(*p!=0x00)
       {write_data(*p);
        p++;
        x++;
        if(x==0x8f)
          write_instruction(0xc0);
       }
      }
     /*清屏函数*/
void cls(void)
{
   write_instruction(0x01);
}

/*初始化函数*/
void init_LCD(void)
{write_instruction(0x38);
 delay(200);
 write_instruction(0x0c);
 delay(200);
 write_instruction(0x06);
 delay(200);
}
//////////////////以下是 DS18B20 驱动程序/////////////////
//延时函数
/*void delay(uint i)
{
      while(i--);
} */

//初始化函数
void Init_DS18B20(void)
{
      uchar x=0;
      DS18B20_DQ = 1; //DQ 复位
```

```
    delay(8); //稍做延时
    DS18B20_DQ = 0; //单片机将 DQ 拉低
    delay(80); //精确延时 大于 480us
    DS18B20_DQ = 1; //拉高总线
    delay(14);
    x=DS18B20_DQ; //稍做延时后 如果 x=0 则初始化成功 x=1 则初始化失败
    delay(20);
}

//读一个字节
ReadOneChar(void)
{
    uchar i=0;
    uchar dat = 0;
    for (i=8;i>0;i--){
        DS18B20_DQ = 0; // 给脉冲信号
        dat>>=1;
        DS18B20_DQ = 1; // 给脉冲信号
        if(DS18B20_DQ)  dat|=0x80;
        delay(4);
        }
    return(dat);
}

//写一个字节
void WriteOneChar(uchar dat)
{
    uchar i=0;
    for (i=8; i>0; i--){
        DS18B20_DQ = 0;
        DS18B20_DQ= dat&0x01;
        delay(5);
        DS18B20_DQ = 1;
        dat>>=1;
        }
}
//读取温度
ReadTemperature(void)
{
    uchar a=0;
    uchar b=0;
    uint t=0;
    float tt=0;
    Init_DS18B20();
    WriteOneChar(0xCC); // 跳过读序号列号的操作
    WriteOneChar(0x44); // 启动温度转换
    Init_DS18B20();
    WriteOneChar(0xCC); //跳过读序号列号的操作
    WriteOneChar(0xBE); //读取温度寄存器等（共可读 9 个寄存器） 前两个就是温度
    a=ReadOneChar();
    b=ReadOneChar();
    t=b;
    t<<=8;
    t=t|a;
    tt=t*0.0625; //将温度的高位与低位合并
    t= tt*10+0.5; //对结果进行 4 舍 5 入
    return(t);
}
```

```
//////////////////以上是 DS18B20 驱动程序////////////////

///以下是温度高时处理程序
void temp_hight(void)
{if(fan_AUTO==0)   //自动
     {LED1=1;                                          //亮灯说明是自动控制
      if(temp<=230)              {fan_count=400;}//判断温度值，返回相应的 PWM 值
      if(temp>230&&temp<=270){fan_count=150;}//判断温度值，返回相应的 PWM 值
      if(temp>270&&temp<=320){fan_count=30;}   //判断温度值，返回相应的 PWM 值
      if(temp>320&&temp<=370){fan_count=1;} //判断温度值，返回相应的 PWM 值
      if(temp>400)
        {fan_count=1;
         temp=ReadTemperature();                    //防止跳变干扰，再读一次温度
         delay_1ms(5000);                          //防止跳变干扰，再读一次温度
         if(temp>400){TR1=1;}                      //防止跳变干扰，再读一次温度
        }
      if(fan_flag==1)                  //这部分作为 PWM 控制程序
      {                                    //这部分作为 PWM 控制程序
       fan=1;                                  //这部分作为 PWM 控制程序
       delay_1ms(fan_count);//控制风扇关闭的时间，以达到控制风扇转速的目的
       fan=0;                                  //这部分作为 PWM 控制程序
      }
    }
 if(fan_AUTO==1)//手动
    {LED1=0;   //灭灯说明是手动控制
     fan=0;      //表示风扇一直开着
    }
}
///以上是温度高时处理程序

/////以下是温度低时处理程序
void temp_low(void)
{      if(sw_flag==0)    //自动的时候继电器根据温度值判断开启
        {LED2=1;         //亮灯说明是自动控制
         if(temp<sw_open_temperature)
         E_sw=0;         //开继电器
         else E_sw=1; //关继电器
        }
       if(sw_flag==1)    //手动的时候继电器一直开着
        {LED2=0;         //灭灯说明是手动控制
         E_sw=0;       //开继电器
        }
}
///以上是温度低时处理程序

//以下程序用于记录一天温度
void deal_record(uchar record_hour_num)
{     uchar i=0;
      i=record_hour_num*3;
      temp_record[i]=num_to_char[temp/100];              //记录温度值到对应的数组位置
      temp_record[i+1]=num_to_char[temp%100/10];     //记录温度值到对应的数组位置
      temp_record[i+2]=num_to_char[temp%10];           //记录温度值到对应的数组位置
}

//以下程序用于处理对应时间的温度显示
void deal_record_dis(uchar record_hour_num)
{     uchar i=0;
      i=record_hour_num*3;
```

```
    temp_record_line2[10]=temp_record[i];                          //把温度放到数组中去方
便显示
    temp_record_line2[11]=temp_record[i+1];                        //把温度放到数组中去方
便显示
    temp_record_line2[13]=temp_record[i+2];                        //把温度放到数组中去方
便显示
    temp_record_line2[5]=num_to_char[record_hour_num/10];  //把对应时间放到数组中去
方便显示
    temp_record_line2[6]=num_to_char[record_hour_num%10];  //把对应时间放到数组中
去方便显示

}

//以下程序是用于记录一天的温度
void record(void)
{    // uchar record_hour,record_minute,record_second;
      record_hour=(time_data_buff[2]/0x10)*10+time_data_buff[2]%0x10;   //读取时间，用
于判断
      record_minute=(time_data_buff[1]/0x10)*10+time_data_buff[1]%0x10; //读取时间，用
于判断
      record_second=time_data_buff[0]/0x10*10+time_data_buff[0]%0x10;   //读取时间，用
于判断
      if(record_second==0&&record_minute==0)                                  //以下用
于判断对应整点的时候，保存对应时间的温度
      {switch(record_hour)
            {
             case 0:deal_record(0);break;     case 1:deal_record(1);break;   //0 表示在 0 点时
的温度，1 表示在 1 点时的温度，以下以此类推
             case 2:deal_record(2);break;     case 3:deal_record(3);break;
             case 4:deal_record(4);break;     case 5:deal_record(5);break;
             case 6:deal_record(6);break;     case 7:deal_record(7);break;
             case 8:deal_record(8);break;     case 9:deal_record(9);break;
             case 10:deal_record(10);break;    case 11:deal_record(11);break;
             case 12:deal_record(12);break;    case 13:deal_record(13);break;
             case 14:deal_record(14);break;    case 15:deal_record(15);break;
             case 16:deal_record(16);break;    case 17:deal_record(17);break;
             case 18:deal_record(18);break;    case 19:deal_record(19);break;
             case 20:deal_record(20);break;    case 21:deal_record(21);break;
             case 22:deal_record(22);break;    case 23:deal_record(23);break;
             default:break;
            }
      }
}

//自动情况下设定的时间内风扇不能转动，时间为早上 4 点到 7 点
void time_restrict(void)
{     restrict_hour=time_data_buff[2]/0x10*10+time_data_buff[2]%0x10; //读取温度
      if(restrict_hour>=4&&restrict_hour<7)      //限制的时间为早上 4 点到 7 点
          {fan_flag=0;}//允许标志，0 表示不允许
      else {fan_flag=1;fan=1;}
}


//T1 用于蜂鸣警报
void T1_set(void)
{
    TMOD=0X11;
    TH1=(65536-1000)/256;
```

```
    TL1=(65536-1000)%256;
    EA=1;
    ET1=1;
//  TR1=0;
    return;
}
//T1 中断，用于蜂鸣警报
void T1_()interrupt 3
{   uchar i;
    TH1=(65536-1000)/256;
    TL1=(65536-1000)%256;
    i++;
    if(i==150)
    {
    speaker=~speaker;
    i=0;
    }
    speaker_count++;
    if(speaker_count==1000)
      {speaker=1;
       speaker_count=0;
       TR1=0;
      }
}

//主函数
void main()
{     uchar i;
      bit dis_flag=0;
      T1_set();//初始化定时器，用于高温警报
      fan=1;
      init_LCD();//初始化 lcd
      //Set1302(time_data_buff);   //设置时间
      fan=1;
      while(1)
            {
//模式设置
              if(key_model==0)
                {delay_1ms(20);
                 while(key_model==0);
                 model_flag++;
                 time_flag=0;
                 if(model_flag==5){model_flag=0;write_instruction(0x0c);cls();}
                 if(model_flag==1)
                   {write_string(0xc8,"set time");
                    write_instruction(0x0c);
                   }
                 if(model_flag==3){write_instruction(0x0e);//这部分是用于设置继电器的
开启状态

for(i=0;i<6;i++)temp_highter_line2[i]=temp_AUTO_switch[sw_flag][i];//读取缓存值

temp_highter_line2[9]=num_to_char[sw_open_temperature/100];

temp_highter_line2[10]=num_to_char[sw_open_temperature%100/10];

temp_highter_line2[12]=num_to_char[sw_open_temperature%10];
                                        write_string(0x80,temp_highter_line1);
```

```
                                        write_string(0xc0,temp_highter_line2);
                                        write_instruction(0xc0);
                                       }
                    }
//显示时间
                if(model_flag==0)
                  {time_restrict();
                   temp_hight();
                   temp_low();
                   if(key_set==0)              //风扇自动手动切换
                      {delay_1ms(20);
                       while(key_set==0);
                       fan_AUTO=~fan_AUTO;
                      }

                   Get1302(time_data_buff); //读取当前时间
                   temp=ReadTemperature();
                   record();
                   lcd1602_line1[3]    = num_to_char[time_data_buff[6]/0x10];
                   lcd1602_line1[4]    = num_to_char[time_data_buff[6]%0x10];    /*年*/
                   lcd1602_line1[6]    = num_to_char[time_data_buff[4]/0x10];
                   lcd1602_line1[7]    = num_to_char[time_data_buff[4]%0x10];    /*月*/
                   lcd1602_line1[9]    = num_to_char[time_data_buff[3]/0x10];
                   lcd1602_line1[10] = num_to_char[time_data_buff[3]%0x10];    /*日*/
                   for(i=0;i<4;i++)  lcd1602_line1[i+12]=Weeks[time_data_buff[5]&0x07][i];  /*
星期*/
                   lcd1602_line2[0]    = num_to_char[time_data_buff[2]/0x10];
                   lcd1602_line2[1]    = num_to_char[time_data_buff[2]%0x10]; /*时*/
                   lcd1602_line2[3]    = num_to_char[time_data_buff[1]/0x10];
                   lcd1602_line2[4]    = num_to_char[time_data_buff[1]%0x10]; /*分*/
                   lcd1602_line2[6] = num_to_char[time_data_buff[0]/16];
                   lcd1602_line2[7] = num_to_char[time_data_buff[0]%16]; /*秒*/
                   lcd1602_line2[9] = num_to_char[(temp)/100]; /*温度*/
                   lcd1602_line2[10] = num_to_char[(temp)%100/10]; /*温度*/
                   lcd1602_line2[12] = num_to_char[(temp)%10]; /*温度*/
                   delay_1ms(100);
                   write_string(0x80,lcd1602_line1);
                   write_string(0xc0,lcd1602_line2);

                  }
//以上显示正常运行的时间
//以下设置时间
                if(model_flag==1)
                        {
                         if(key_set==0)
                            {delay_1ms(20);
                             while(key_set==0);
                             set_time++;
                             if(set_time==3)set_time=0;
                             if(set_time==1){write_instruction(0xc6);
                                             write_instruction(0x0e);//显示光标
                                            }
                            }
                         if(set_time==0)
                            {
                             write_instruction(0x0c);              //关掉光标
                             write_string(0xc8,"set time");
                            }
```

```
                    if(set_time==1)
                      {
                       if(key_left==0)      //设置时分秒星期日月年的标志
                         {delay_1ms(20);
                          while(key_left==0);
                          time_flag++;          //设置时分秒星期日月年的标志
                          dis_flag=1;
                          if(time_flag==7)time_flag=0;
                         }
                       if(key_right==0)       //设置时分秒星期日月年的标志
                           {delay_1ms(20);
                            while(key_right==0);
                            dis_flag=1;
                            time_flag--;              //设置时分秒星期日月年的标志
                            if(time_flag==-1)time_flag=6;
                           }
                       second= (time_data_buff[0]/0x10)*10+time_data_buff[0]%0x10;      //
读取时间，用于下面的操作
                       minute= (time_data_buff[1]/0x10)*10+time_data_buff[1]%0x10;      //
读取时间，用于下面的操作
                       hour   = (time_data_buff[2]/0x10)*10+time_data_buff[2]%0x10;      //
读取时间，用于下面的操作
                       day     = (time_data_buff[3]/0x10)*10+time_data_buff[3]%0x10;      //
读取时间，用于下面的操作
                       month = (time_data_buff[4]/0x10)*10+time_data_buff[4]%0x10;      //
读取时间，用于下面的操作
                       week   = time_data_buff[5]%0x10;
    //读取时间，用于下面的操作
                       year                                                              =
2000+(time_data_buff[6]/0x10)*10+time_data_buff[6]%0x10;//读取时间，用于下面的操作
                       if(time_flag==0)                 //设秒
                         {if(key_add==0)          //秒加
                             {delay_1ms(20);
                              while(key_add==0);
                              second++;
                              if(second==60)second=0;
                              dis_flag=1;
                             }
                          if(key_sub==0)      //秒减
                             {delay_1ms(20);
                             while(key_sub==0);
                              second--;
                              if(second==-1)second=59;
                              dis_flag=1;
                             }
                          if(dis_flag==1)            //显示和保存修改值
                             {time_data_buff[0]=(second/10)*0x10+second%10;
                              write_instruction(0xc6);
                              write_data(num_to_char[second/10]);
                              write_data(num_to_char[second%10]);
                              write_instruction(0xc6);
                              dis_flag=0;
                             }
                         }
                       if(time_flag==1)                 //设分
                         {if(key_add==0)                //分加
                             {delay_1ms(20);
                              while(key_add==0);
```

```
            minute++;
            if(minute==60)minute=0;
            dis_flag=1;
           }
       if(key_sub==0)              //分减
          {delay_1ms(20);
           while(key_sub==0);
           minute--;
           if(minute==-1)minute=59;
           dis_flag=1;
          }
       if(dis_flag==1)                        //显示和保存修改值
          {
           time_data_buff[1]=(minute/10)*0x10+minute%10;
           write_instruction(0xc3);
           write_data(num_to_char[minute/10]);
           write_data(num_to_char[minute%10]);
           write_instruction(0xc3);
           dis_flag=0;
          }
     }
  if(time_flag==2)                 //设时
     {if(key_add==0)             //时加
          {delay_1ms(20);
           while(key_add==0);
           hour++;
           if(hour==24)hour=0;
           dis_flag=1;
          }
       if(key_sub==0)             //时减
          {delay_1ms(20);
           while(key_sub==0);
           hour--;
           if(hour==-1)hour=23;
           dis_flag=1;
          }
       if(dis_flag==1)                        //显示和保存修改值
          {time_data_buff[2]=(hour/10)*0x10+hour%10;
           write_instruction(0xc0);
           write_data(num_to_char[hour/10]);
           write_data(num_to_char[hour%10]);
           write_instruction(0xc0);
           dis_flag=0;
          }
     }
  if(time_flag==3)                 //设星期
     {if(key_add==0)      //星期加
          {delay_1ms(20);
           while(key_add==0);
           week++;
           if(week==8) week=1;
           dis_flag=1;
          }
       if(key_sub==0)      //星期减
          {delay_1ms(20);
           while(key_sub==0)
           week--;
           if(week==-1)hour=7;
```

```
                    dis_flag=1;
                   }
                  if(dis_flag==1)                    //显示和保存修改值
                    {time_data_buff[5]=week;
                     for(i=0;i<4;i++)
lcd1602_line1[i+13]=Weeks[time_data_buff[5]&0x07][i];
                     write_instruction(0x8c);
                     write_data(lcd1602_line1[13]);
                     write_data(lcd1602_line1[14]);
                     write_data(lcd1602_line1[15]);
                     write_instruction(0x8c);
                     dis_flag=0;
                    }
                 }
               if(time_flag==4)          //设日
                 {  if(key_add==0)                                    //日加
                     {delay_1ms(20);
                      while(key_add==0);
                      dis_flag=1;
                      day++;
                      switch(month)
                            {case 1:if(day==32)day=1;  break;
                             case 2:if ((year % 100 != 0) && (year % 4 == 0) ||
(year % 400 == 0))
                                    {if(day>29)
                                        {day=1;}
                                    }
                                    else
                                     { if(day>28)
                                        {day=1;
                                        }
                                         }break;
                            case 3:if(day==32)day=1;break;
                            case 4:if(day==31)day=1;break;
                            case 5:if(day==32)day=1;break;
                            case 6:if(day==31)day=1;break;
                            case 7:if(day==32)day=1;break;
                            case 8:if(day==32)day=1;break;
                            case 9:if(day==31)day=1;break;
                            case 10:if(day==32)day=1;break;
                            case 11:if(day==31)day=1;break;
                            case 12:if(day==32)day=1;break;
                            default:break;
                           }
                     }
                   if(key_sub==0)                                     //日减
                     {delay_1ms(20);
                      while(key_sub==0);
                      dis_flag=1;
                      day--;
                      switch(month)
                            {case 1:if(day==0)day=31;  break;
                             case 2:if ((year % 100 != 0) && (year % 4 == 0) ||
(year % 400 == 0))
                                    {if(day==0)
                                        {day=29;}
                                    }
                                    else
```

```
                                { if(day==0)
                                    {day=28;
                                    }
                                }break;
                            case 3:if(day==0)day=31;break;
                            case 4:if(day==0)day=30;break;
                            case 5:if(day==0)day=31;break;
                            case 6:if(day==0)day=30;break;
                            case 7:if(day==0)day=31;break;
                            case 8:if(day==0)day=31;break;
                            case 9:if(day==0)day=30;break;
                            case 10:if(day==0)day=31;break;
                            case 11:if(day==0)day=30;break;
                            case 12:if(day==0)day=31;break;
                            default:break;
                          }
                    }
        if(dis_flag==1)              //显示和保存修改值
           {time_data_buff[3]=(day/10)*0x10+day%10;
            write_instruction(0x89);
            write_data(num_to_char[day/10]);
            write_data(num_to_char[day%10]);
            write_instruction(0x89);
            dis_flag=0;
           }
      }
   if(time_flag==5&&set_time==1)     //设月
      {if(key_add==0)        //月加
           {delay_1ms(20);
            while(key_add==0);
            month++;
            if(month==13)month=1;
            dis_flag=1;
           }
        if(key_sub==0)         //月减
            {delay_1ms(20);
             while(key_sub==0);
             month--;
             if(month==0)month=12;
             dis_flag=1;
            }
        if(dis_flag==1)              //显示和保存修改值
           {time_data_buff[4]=(month/10)*0x10+month%10;
            write_instruction(0x86);
            write_data(num_to_char[month/10]);
            write_data(num_to_char[month%10]);
            write_instruction(0x86);
            dis_flag=0;
           }
       }
   if(time_flag==6&&set_time==1) //设年
      {
       if(key_add==0)              //年加
           {delay_1ms(20);
            while(key_add==0);
            year++;
            dis_flag=1;
            if(year==2100)year=2010;
```

```
                    }
                  if(key_sub==0)              //年减
                    {delay_1ms(20);
                     while(key_sub==0);
                     year--;
                     dis_flag=1;
                     if(year==2009)year=2099;
                    }
                  if(dis_flag==1)           //显示和保存修改值

{time_data_buff[6]=year%1000%100/10*0x10+year%1000%100%10;
                     write_instruction(0x81);
                     write_data(num_to_char[2]);
                     write_data(num_to_char[0]);
                     write_data(num_to_char[((year%1000)%100)/10]);
                     write_data(num_to_char[((year%1000)%100)%10]);
                     write_instruction(0x83);
                     dis_flag=0;
                    }
                 }

                }
               //以下是保存设置的时间
                if(set_time==2)
                   {write_instruction(0x0c);
                    Set1302(time_data_buff);
                    set_time=0;
                    time_flag=0;
                   }
            }
//以上设置时间
//以下是用于查看一天的温度记录
          if(model_flag==2)
            {
             if(key_add==0)
                {while(key_add==0);
                 record_hour_count++;
                 if(record_hour_count==24)record_hour_count=0;
                }
             if(key_sub==0)
                {while(key_sub==0);
                 record_hour_count--;
                 if(record_hour_count==-1)record_hour_count=23;
                }
             switch(record_hour_count)
                {
                  case 0:deal_record_dis(0);break;      case 1:deal_record_dis(1);break;
                  case 2:deal_record_dis(2);break;      case 3:deal_record_dis(3);break;
                  case 4:deal_record_dis(4);break;      case 5:deal_record_dis(5);break;
                  case 6:deal_record_dis(6);break;      case 7:deal_record_dis(7);break;
                  case 8:deal_record_dis(8);break;      case 9:deal_record_dis(9);break;
                  case          10:deal_record_dis(10);break;                    case
11:deal_record_dis(11);break;
                  case          12:deal_record_dis(12);break;                    case
13:deal_record_dis(13);break;
                  case          14:deal_record_dis(14);break;                    case
15:deal_record_dis(15);break;
                  case          16:deal_record_dis(16);break;                    case
```

```
17:deal_record_dis(17);break;
                        case    18:deal_record_dis(18);break;                    case
19:deal_record_dis(19);break;
                        case    20:deal_record_dis(20);break;                    case
21:deal_record_dis(21);break;
                        case    22:deal_record_dis(22);break;                    case
23:deal_record_dis(23);break;
                        default:break;
                    }
                   write_string(0x80,temp_record_line1);
                   write_string(0xc0,temp_record_line2);
                  }
//以上是用于查看一天的温度记录
//以下是用于设置继电器的开启状态
              if(model_flag==3)
                {
                  if(key_left==0)
                    {
                     while(key_left==0);
                     switch_flag++;
                     if(switch_flag==2)switch_flag=0;
                     dis_flag=1;
                    }
                  if(key_right==0)
                    {
                     while(key_right==0);
                     switch_flag--;
                     if(switch_flag==-1)switch_flag=1;
                     dis_flag=1;
                    }
                  if(switch_flag==0)
                    {
                     if(key_add==0)
                       {while(key_add==0);
                              {dis_flag=1;
                               sw_flag++;
                               if(sw_flag==2)sw_flag=0;
                              }
                       }
                     if(key_sub==0)
                       {while(key_sub==0);
                              {dis_flag=1;
                               sw_flag--;
                               if(sw_flag==-1)sw_flag=1;
                              }
                       }
                     write_instruction(0xc0);
                     if(dis_flag==1)
                     {
                     for(i=0;i<6;i++)temp_highter_line2[i]=temp_AUTO_switch[sw_flag][i];
                     }
                    }
                  if(switch_flag==1)
                    {
                     if(key_add==0)
                       {while(key_add==0);
                              {
                               sw_open_temperature=sw_open_temperature+10;
```

```
                                if(sw_open_temperature>=1000)sw_open_temperature=0;
                                dis_flag=1;
                               }
                        }
                      if(key_sub==0)
                        {while(key_sub==0);
                               {
                                sw_open_temperature=sw_open_temperature-10;
                                if(sw_open_temperature<0)sw_open_temperature=990;
                                dis_flag=1;
                               }
                        }
                       write_instruction(0xc9);
                       temp_highter_line2[9]=num_to_char[sw_open_temperature/100];
                       temp_highter_line2[10]=num_to_char[sw_open_temperature%100/10];
                       temp_highter_line2[12]=num_to_char[sw_open_temperature%10];
                     }
                  if(dis_flag==1)
                     {
                      write_string(0x80,temp_highter_line1);
                      write_string(0xc0,temp_highter_line2);
                      dis_flag=0;
                     }
                  temp_low();
                 }
//以上是用于设置继电器的开启状态
//以下是用于显示风扇禁止打开的时间
               if(model_flag==4)
                  {
                   write_string(0x80,temp_time_unable_line1);
                   write_string(0xc0,temp_time_unable_line2);
                  }
//以上是用于显示风扇禁止打开的时间
               }
}
```

```
#ifndef __DS1302_H_REVISION_FIRST__
#define __DS1302_H_REVISION_FIRST__

/*****************************************************************

程序名称:     LCD1602 显示时间

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

```
/*头文件*/
//#include <reg52.h>
#include<reg52.h>
#include <intrins.h>
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint    unsigned int
#endif

#define _Nop() _nop_()

sbit    DS1302_SCLK = P2^2;     /*实时时钟时钟线引脚 */
sbit    DS1302_IO = P2^3;     /*实时时钟数据线引脚 */
sbit    DS1302_RST = P2^4;     /*实时时钟复位线引脚 */
sbit    ACC0 = ACC^0;
sbit    ACC7 = ACC^7;
uchar data time_data_buff[7]={0x00,0x20,0x13,0x27,0x11,0x06,0x10};/*格式为: 秒 分 时 日
月 星期 年 */



/*************************************************************************

 函 数 名：RTInputByte()
 功      能：实时时钟写入一字节
 说      明：往 DS1302 写入 1Byte 数据 (内部函数)
 入口参数：d 写入的数据
 返 回 值：无
*************************************************************************/
void RTInputByte(uchar d)
{
      uchar i;
      ACC = d;
      for(i=8; i>0; i--)
      {
            DS1302_IO = ACC0;                 /*相当于汇编中的 RRC */
            DS1302_SCLK = 1;
            DS1302_SCLK = 0;
            ACC = ACC >> 1;
      }
}
/*************************************************************************

 函 数 名：RTOutputByte()
 功      能：实时时钟读取一字节
 说      明：从 DS1302 读取 1Byte 数据 (内部函数)
 入口参数：无
 返 回 值：ACC
*************************/
uchar RTOutputByte(void)
{
      uchar i;
      for(i=8; i>0; i--)
      {
            ACC = ACC >>1;              /*相当于汇编中的 RRC */
```

```
            ACC7 = DS1302_IO;
            DS1302_SCLK = 1;
            DS1302_SCLK = 0;
        }
        return(ACC);
}
/*********************************************************************

 函 数 名：W1302()
 功　　能：往 DS1302 写入数据
 说　　明：先写地址，后写命令/数据 (内部函数)
 调　　用：RTInputByte() , RTOutputByte()
 入口参数：ucAddr: DS1302 地址, ucData: 要写的数据
 返 回 值：无
*********************************************************************/
void W1302(uchar ucAddr, uchar ucDa)
{
    DS1302_RST = 0;
    DS1302_SCLK = 0;
    DS1302_RST = 1;
    RTInputByte(ucAddr);          /* 地址，命令 */
    RTInputByte(ucDa);          /* 写 1Byte 数据*/
    DS1302_SCLK = 1;
    DS1302_RST = 0;
}
/*********************************************************************

 函 数 名：R1302()
 功　　能：读取 DS1302 某地址的数据
 说　　明：先写地址，后读命令/数据 (内部函数)
 调　　用：RTInputByte() , RTOutputByte()
 入口参数：ucAddr: DS1302 地址
 返 回 值：ucData :读取的数据
*********************************************************************/
uchar R1302(uchar ucAddr)
{
    uchar ucData;
    DS1302_RST = 0;
    DS1302_SCLK = 0;
    DS1302_RST = 1;
    RTInputByte(ucAddr);          /* 地址，命令 */
    ucData = RTOutputByte();          /* 读 1Byte 数据 */
    DS1302_SCLK = 1;
    DS1302_RST = 0;
    return(ucData);
}


/*********************************************************************

 函 数 名：Set1302()
 功　　能：设置初始时间
 说　　明：先写地址，后读命令/数据(寄存器多字节方式)
 调　　用：W1302()
 入口参数：pClock: 设置时钟数据地址 格式为: 秒 分 时 日 月 星期 年
                                    7Byte (BCD 码)1B 1B 1B 1B 1B  1B  1B
 返 回 值：无
*********************************************************************/
```

```
void Set1302(uchar *pClock)
{
    uchar i;
    uchar ucAddr = 0x80;
    EA = 0;
    W1302(0x8e,0x00);                 /* 控制命令,WP=0,写操作?*/
    for(i =7; i>0; i--)
    {
        W1302(ucAddr,*pClock);   /* 秒 分 时 日 月 星期 年 */
        pClock++;
        ucAddr +=2;
    }
    W1302(0x8e,0x80);                 /* 控制命令,WP=1,写保护?*/
    EA = 1;
}
/*********************************************************************

 函 数 名：Get1302()
 功　　能：读取 DS1302 当前时间
 说　　明：
 调　　用：R1302()
 入口参数：ucCurtime: 保存当前时间地址。当前时间格式为: 秒 分 时 日 月 星期 年
                                                    7Byte (BCD 码)    1B 1B 1B 1B 1B  1B
1B
 返 回 值：无
*********************************************************************/
void Get1302(uchar ucCurtime[])
{
    uchar i;
    uchar ucAddr = 0x81;
    EA = 0;
    for (i=0; i<7; i++)
    {
        ucCurtime[i] = R1302(ucAddr);/*格式为: 秒 分 时 日 月 星期 年 */
        ucAddr += 2;
    }
    EA = 1;
}
```