扬州大学广陵学院

本科生毕业设计（毕业论文）

题　　目：带温度显示的电子闹钟的设计

学生姓名：葛文强

学　　号：052612210

专　　业：电子信息科学与技术

班　　级：信科 80502 班

指导教师：陈卫峰　老师

带温度显示的电子闹钟的设计

摘　要

电子时钟主要是利用电子技术将时钟电子化、数字化，拥有时钟精确、体积小、界面友好、可扩展性能强等特点，被广泛应用于生活和工作当中。另外，在生活和工农业生产中，也常常需要温度，这就需要电子时钟具有多功能性。

本设计主要为实现一款可正常显示时钟/日历、测量环境温度、带有定时闹铃的电子时钟。

本文采用单片机技术实现带温度显示的电子闹(时)钟。本设计应用 AT89C51 芯片作为核心，7 位 LED 数码管显示，使用 DS1302 实时时钟日历芯片完成时钟/日历的基本功能，同时利用 DS18B20 温度传感器测量环境温度。这种实现方法的优点是电路简单，性能可靠，实时性好，时间和温度精度高，操作简单，编程容易。

该电子时钟可以应用于一般的生活和工作中，也可通过改装，提高性能，增加新功能，从而给人们的生活和工作带来更多的方便。

关键词： 电子时钟；　温度；~~多功能；~~ AT89C51；　~~时钟日历芯片~~ DS1302；　~~温度传感器~~ DS18B20

~~With temperature shows that~~ the design of an electronic alarm clock ~~With temperature measurement~~

Abstract

The electronic clock mainly uses the electronic technology ~~to make the clock~~ . Electronic clock with the clock precision, small size, user-friendly, scalable performance and so on ~~,computerization, the digitization, with the clock precision, small size, friendly interface, scalable performance and other characteristics~~ ,It was widely used in life and work. Measuring temperature, in life, industry and agricultural production, so electronic clock need multi-function.

The design is an electronic clock, additional functionality in this alarm, the ambient temperature and other functions. ~~The design for the main implementing a clock/calendar can be displayed normal, collecting personal ambient temperature, with the timing alarm of the multi-function electronic clock.~~

The design determines to use the MCU technology to realize the multi-functional electron clock. This design ~~,application~~ ~~applicates~~use AT89C51 as a core chips, 7 LED digital displaying, using DS1302 real-time clock chip to complete the basic function of the clock/calendar. At the same time the design use of DS18B20 temperature sensors ~~is to for~~is for collect~~in~~ging the environmental temperature. The method has the advantage of being simple circuit, reliable performance, good real-time, high precision of the time and temperature, simply operation, easy programming.

The electronic clock can be applied to the general living and working ,can also be modified to improve performance, add new functions, and brings more convenient to people's life and work.

Key words: Electronic clock; Temperature; Multi-function; AT89C51; DS1302; DS18B20 Temperature pickup DS18B20

目　录

摘　要.....I

Abstract.....II

第一章 绪论.....III

1.1 电子时钟研究的背景和意义.....1

1.2 电子时钟的功能.....2

第二章 电子时钟设计方案分析.....3

2.1 NE555 时基电路设计方案.....3

2.2 单片机设计方案.....4

2.2 单片机设计方案.....3

第三章 基于单片机的电子时钟硬件设计.....6

第三章 基于单片机的电子时钟硬件设计.....5

3.1 主要 IC 芯片选择.....6

3.1 主要 IC 芯片选择.....5

3.1.1 微处理器选择.....6

3.1.1 微处理器选择.....5

3.1.2. 常用时钟芯片的选择.....5

3.1.2.1 常用时钟日历芯片 DS1302 简介.....7

3.1.2.2 DS1302 引脚说明.....8

3.1.2.2 DS1302 引脚说明.....7

3.1.2.3 DS1302 的控制字和读写时序说明.9

3.1.2.3 DS1302 的控制字和读写时序说明.....8

3.2.1.4 DS1302 的片内寄存器..... 11
3.2.1.4 DS1302 的片内寄存器..... 10
3.1.3 环境温度传感器..... 13
3.1.3 环境温度传感器..... 12
3.1.3.1 常用温度传感器 DS18B20 简介..... 12
3.1.3.2 DS18B20 内部结构..... 14
3.1.3.2 DS18B20 内部结构..... 13
3.2 电子时钟硬件电路设计..... 16
3.2 电子时钟硬件电路设计..... 15
3.2.1 时钟电路设计..... 17
3.2.1 时钟电路设计..... 16
3.2.2 环境温度采集电路设计..... 17
3.2.2 环境温度采集电路设计..... 16
3.2.3 显示电路..... 18
3.2.3 显示电路..... 17
3.2.4 按键电路设计..... 19
3.2.4 按键电路设计..... 18
3.2.5 闹铃电路设计..... 21
3.2.5 闹铃电路设计..... 20
第四章 电子时钟软件设计..... 22
第四章 电子时钟软件设计..... 21
4.1 主程序设计..... 22
4.1 主程序设计..... 21

4.2 子程序设计.....22
4.2.1 实时时钟日历子程序设计.....23
4.2.2 环境温度采集子程序设计.....23
4.2.3 显示子程序设计.....27
4.2.4 键盘扫描子程序.....27
4.2.5 闹铃子程序设计.....27
第五章 系统调试.....30
5.1 硬件调试.....31
5.1.1 单片机基础电路调试.....31
5.1.2 显示电路调试.....32
5.1.3 DS1302 电路调试.....32
5.1.4 按键电路调试.....33
5.2 软件调试.....33
5.2.1 环境温度采集子程序调试.....34
5.2.2 键盘子程序调试.....34

结　论.....35
结　论 28
参考文献.....29
致　谢.....30 致　谢
.....36
参考文献.....37
附录 A　带温度显示的电子闹钟元器件一览表31 附录 A　带温度显示的电子闹钟元器件
一览表.....38
附录 B　带温度显示的电子闹钟硬件电路图..... 3239
附录 C　带温度显示的电子闹钟 程序..... 3340

第一章 绪论

时间是人类生活必不可少的重要元素，如果没有时间的概念，社会将不会有所发展和进步。从古代的水漏、十二天干地支，到后来的机械钟表以及当今的石英钟，都充分显现出了时间的重要，同时也代表着科技的进步。致力于计时器的研究和充分发挥时钟的作用，将有着重要的意义。

温度是一个和人们生活环境有着密切关系的物理量，温度的变化会给我们的生活、工作、生产等带来重大影响，因此对温度的测量至关重要。

1.1 电子时钟研究的背景和意义

现代电子产品几乎渗透到了社会的各个领域，有力的推动和提高了社会生产力的发展与信息化程度，同时也使现代电子产品性能进一步提升，产品更新换代的节奏也越来越快。

时间对人们来说总是那么宝贵，工作的忙碌性和繁杂容易使人忘记当前的时间。平时我们要求上班准时，约会或召开会议必然要提及时间；火车要准点到达，航班要准点起飞；工业生产中，很多环节都需要用时间来确定工序替换时刻。所以说能随时准确的知道时间并利用时间，是我们生活和工作中必不可少的。

想知道时间，手表当然是一个很好的选择，但是，在忙碌当中，我们还需要一个“助理” 及时的给我们提醒时间。所以，计时器最好能够拥有一个定时系统，随时提醒容易忘记时间的人。 最早能够定时、报时的时钟属于机械式钟表，但这种时钟受到机械结构、动力和体积的限制，在功能、性能以及造价上都没办法与电子时钟相比。

电子钟是采用电子电路实现对时、分、秒进行数字显示的计时装置，广泛应用于个人家庭，车站， 码头办公室等公共场所，成为人们日常生活中不可少的必需品。由于数字集成电路的发展和石英晶体振荡器的广泛应用，使得数字钟的精度，远远超过老式钟

表，钟表的数字化给人们生产生活带来了极大的方便，而且大大地扩展了钟表原先的报时功能。诸如定时自动报警、0 按时自动打铃、定时广播、自动起闭路灯、定时开关烘箱、通断动力设备、甚至各种定时电气的自动启用等，所有这些，都是以钟表数字化为基础的。因此，研究数字钟及扩大其应用，有着非常现实的意义。

另外，温度实时显示系统应用同样越来越广泛，比如空调遥控器上当前室温的显示、热水器温度的显示等等。医药卫生、工农业生产上也有很多场合需要测量环境温度。

如果能够在电子时钟上附加温度采集功能，将使电子时钟的应用更加广泛。

1.2 电子时钟的功能

电子时钟主要是利用电子技术将时钟电子化、数字化，拥有时间精确、体积小、界面友好、可扩展性能强等特点，被广泛应用于生活和工作当中。当今市场上的电子时钟品类繁多，外形小巧别致。也有体型较大的，诸如公共场所的大型电子报时器等。电子时钟首先是数字化了的时间显示或报时器，在此基础上，人们可以根据不同场合的要求，在时钟上加置其他功能，比如定时闹铃，万年历，环境温度、湿度检测，环境空气质量检测，USB 扩展口功能等。

本设计电子时钟主要功能为：

1. 具有时间显示和手动校对功能，24 小时制；
2. 具有年、月、日显示和手动校对功能；
3. 具有闹铃功能；
4. 具有贪睡功能；
5. 具有环境温度采集和显示功能；
6. 掉电后无需重新设置时间和日期；
7. 采用交直流供电电源。交流供电为主，直流电源为备用电源，并能自动切换。

第二章 电子时钟设计方案分析

电子闹钟既可以通过纯硬件实现，也可以通过软硬件结合实现，根据电子时钟里的核心部件——秒信号的产生原理，通常有以下两种形式：

2.1 NE555 时基电路设计方案

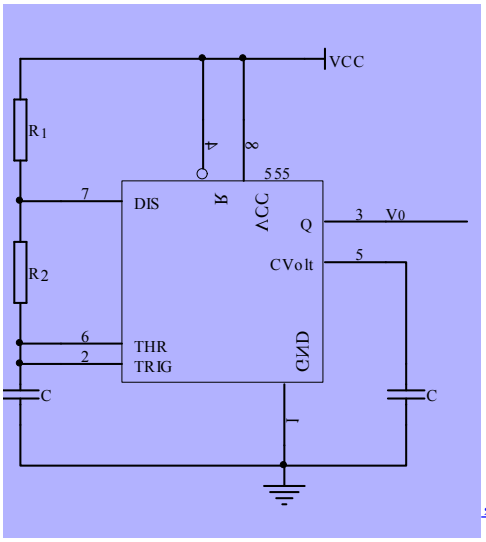
555 定时器是美国 Signetics 公司 1972 年研制的用于取代机械式定时器的中规模集成电路，因输入端设计有三个 5KΩ的电阻而得名。目前，流行的产品主要有 4 种：BJT 两个：555，556（含有两个 555）；CMOS 两个：7555，7556（含有两个 7555）。

555 定时器是一种数字与模拟混合型的集成电路，应用广泛。成本较低，外加电阻、电容等元件就可以构成多谐振荡器、单稳电路、施密特触发器等，常作为定时器广泛应用于仪器仪表、家用电器、电子测量及自动控制等领域。

采用 NE555 时基电路或其他振荡电路产生秒脉冲信号，作为秒加法电路的时钟信号或微处理器的外部中断输入信号，可构成电子钟~~。由 555 构成的秒脉冲发生器电路见图 2.1。输出的脉冲信号 V0 的频率 F 为：~~

~~$$F = 1.443/(R_1 + 2R_2) \times C$$ 式(2.1)~~

~~可通过调节式 2.1 中的 3 个参数，使输出 V_0 的频率为精确的 1Hz。~~



，

图 2.1 基于 5 5 5 的秒脉冲发生器

采用 555 定时器设计电子时钟，成本低，容易实现。但是受芯片引脚数量和功能限制，不容易实现电子时钟的多功能性。

2.2 单片机设计方案

单片机是微型机的一个主要分支，它在结构上的最大特点使把 CPU、存储器、定时器和多种输入/输出接口电路集成在一块超大规模集成电路芯片上。就其组成和功能而言，一块单片机芯片就是一台计算机。

单片机具有如下特点：

有优异的性能价格比；

1　集成度高、体积小、有很高的可靠性；

2　控制功能强；

3　低功耗、低电压，便于生产便携式产品；

4　外部总线增加了 I^2C、SPI 等串行总线方式，进一步缩小了体积，简化了结构；

5　单片机的系统扩展、系统配置较典型、规范，容易构成各种规模的应用系统。

所以单片机的应用非常广泛，在智能仪表、机电一体化、实时控制、分布式多机系统以及人们的生活中均有用武之地。单片机应用的重要意义还在于，它从根本上改变了传统的控制系统设计思路和设计方法。从前必须由模拟电路或数字电路实现的大部分功能，现在已能用单片机通过软件方法来实现了。这种用软件代替硬件的控制技术，是对生产控制技术的一次革命。

利用单片机的智能性，可方便地实现具有智能的电子钟设计。单片机均具有时钟振荡系统，利用系统时钟借助微处理器的定时器/计数器可实现电子钟功能。然而系统时钟误差较大，电子钟的积累误差也可能较大，所以可以通过误差修正软件加以修正，或者

在设计中加入高精度时钟日历芯片，以精确时间。另外很多功能不同的单片机是兼容的，这就更便于实现产品的多功能性。

第三章 基于单片机的电子时钟硬件设计

在比较了第二章的两种实现方案之后，考虑单片机货源充足、价格低廉，可软硬件结合使用，能够较方便的实现系统的多功能性，故采用单片机作为本设计的硬件基础。

3.1 主要 IC 芯片选择

3.1.1 微处理器选择

目前在单片机系统中，应用比较广泛的微处理器芯片主要为 8XC5X 系列单片机。该系列单片机均采用标准 MCS-51 内核，硬件资源相互兼容，品类齐全，功能完善，性能稳定，体积小，价格低廉，货源充足，调试和编程方便，所以应用极为广泛。

例如比较常用的 AT89C2051 单片机，带有 2KB Flash 可编程、可擦除只读存储器（E^2PROM）的低压、高性能 8 位 CMOS 微型计算机。拥有 15 条可编程 I/O 引脚，2 个 16 位定时器/计数器，6 个中断源，可编程串行 UART 通道，并能直接驱动 LED 输出。

仅仅是为了完成时钟设计或者是环境温度采集设计，应用 AT89C2051 单片机完全可以实现。但是将两种功能结合在一片单片机上，就需要更多的 I/O 引脚，故本设计采用具有 32 根 I/O 引脚的 AT89C51 单片机。

AT89C51 单片机是一款低功耗，低电压，高性能 CMOS 8 位单片机，片内含 4KB（可经受 1000 次擦写周期）的 FLASH 可编程可反复擦写的只读程序存储器（EPROM），器件采用 CMOS 工艺和 ATMEIL 公司的高密度、非易失性存储器（NURAM）技术制造，其输出引脚和指令系统都与 MCS-51 兼容。片内的 FLASH 存储器允许在系统内可改编程序或用常规的非易失性存储器编程器来编程。因此，AT89C51 是一种功能强，灵活性高且价格合理的单片机，可方便的应用在各个控制领域。

AT89C51 具有以下主要性能：

1. 4KB 可改编程序 Flash 存储器；

2. 全静态工作：0——24Hz；

3. 128×8 字节内部 RAM；

4. 32 个外部双向输入/输出（I/O）口；

5. 6 个中断优先级； 2 个 16 位可编程定时计数器；

6. 可编程串行通道；

7. 片内时钟振荡器。

此外，AT89C51 是用静态逻辑来设计的，其工作频率可下降到 0Hz，并提供两种可用软件来选择的省电方式——空闲方式（Idle Mode）和掉电方式（Power Down Mode）。在空闲方式中，CPU 停止工作，而 RAM、定时器/计数器、串行口和中断系统都继续工作。在掉电方式中，片内振荡器停止工作，由于时钟被“冻结”，使一切功能都暂停，只保存片内 RAM 中的内容，直到下一次硬件复位为止。

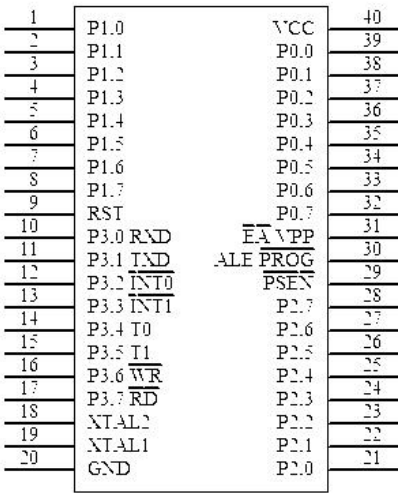


图 3.1 AT89C51 芯片 PDIP 封装引脚图

~~AT89C51 为适应不同的产品需求，采用 PDIP、TQFP、PLCC 三种封装形式，本系统采用双列直插 PDIP 封装形式，如图 3.1。~~ 3.1.2. 常用时钟芯片的选择

在电子时钟设计中，常用的实时时钟芯片有 DS12887、DS1216、DS1643、DS1302。每种芯片的主要时钟功能基本相同，只是在引脚数量、备用电池的安装方式、计时精度和扩展功能等方面略有不同。DS12887 与 DS1216 芯片都有内嵌式锂电池作为备用电池；

X1203 引脚少，没有嵌入式锂电池，跟 DS1302 芯片功能相似，只是相比较之下，X1203 与 AT89S51 搭配使用时占用 I/O 口较多。DS1643 为带有全功能实时时钟的 8K×8 非易失性 SRAM，集成了非易失性 SRAM、实时时钟、晶振、电源掉电控制电路和锂电池电源，BCD 码表示的年、月、日、星期、时、分、秒，带闰年补偿。同样，DS1643 拥有 28 只管脚，硬件连接起来占用微处理器 I/O 口较多，不方便系统功能拓展和维护。故而从性价比和货源上考虑，本设计采用实时时钟日历芯片 DS1302。

~~时钟日历芯片选择~~

3.1.2.1 常用时钟日历芯片 DS1302 简介

DS1302 是美国 DALLAS 公司推出的一种高性能、低功耗的实时时钟日历芯片，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 2.5～5.5V。采用双电源供电（主电源和备用电源），可设置备用电源充电方式，提供了对后备电源进行涓细电流充电的能力。有主电源和备份电源双引脚，而且备份电源可由大容量电容（＞1F）来替代。需要强调的是，DS1302 需要使用 32.768KHz 的晶振。

3.1.2.2 DS1302 引脚说明

DS1302 引脚图参照图 3.2。

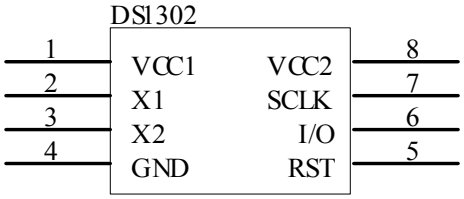


图 3.2 DS1302 芯片引脚图

其的引脚功能参照表 3.1。

表 3.1 DS1302 引脚功能说明

引脚号	名称	功能
1	V_{CC1}	备份电源输入
2	X1	32.768KHz 晶振输入
3	X2	32.768KHz 晶振输出
4	GND	地
5	RST	~~控制移位寄存器~~/复位
6	I/O	数据输入/输出
7	SCLK	串行时钟
8	V_{CC2}	主电源输入

3.1.2.3 DS1302 的控制字和读写时序说明

在编程过程中要注意 DS1302 的读写时序。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。要想与 DS1302 通信，首先要先了解 DS1302 的控制字。DS1302 的控制字如表 3.2。

表 3.2 DS1302 控制字（即地址及命令字节）

BIT7	BIT6	BIT5	BIT4	BIT 3	BIT 2	BIT 1	BIT 0
1	RAM	A4	A3	A2	A1	A0	RD
	$\overline{CK}$						$\overline{WR}$

控制字的作用是设定 DS1302 的工作方式、传送字节数等。每次数据的传输都是由控制字开始。控制字各位的含义和作用如下：

1. BIT7：控制字的最高有效位，必须是逻辑 1，如果它为 0，则不能把数据写入到 DS1302 中。
2. BIT 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；

3. BIT 5 至 BIT 1（A4～A0）：用 A4～A0 表示，定义片内寄存器和 RAM 的地址。定义如下：

当 BIT 6 位=0 时，定义时钟和其他寄存器的地址。A4～A0=0～6，顺序为秒、分、时、日、月、星期、年的寄存器。当 A4～A0=7，为芯片写保护寄存器地址。当 A4～A0=8，为慢速充电参数选择寄存器。当 A4～A0=31，为时钟多字节方式选择寄存器。

当 BIT 6=1 时，定义 RAM 的地址，A4～A0=0～30，对应各子地址的 RAM，地址 31 对应的是 RAM 多字节方式选择寄存器。

4. BIT 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。

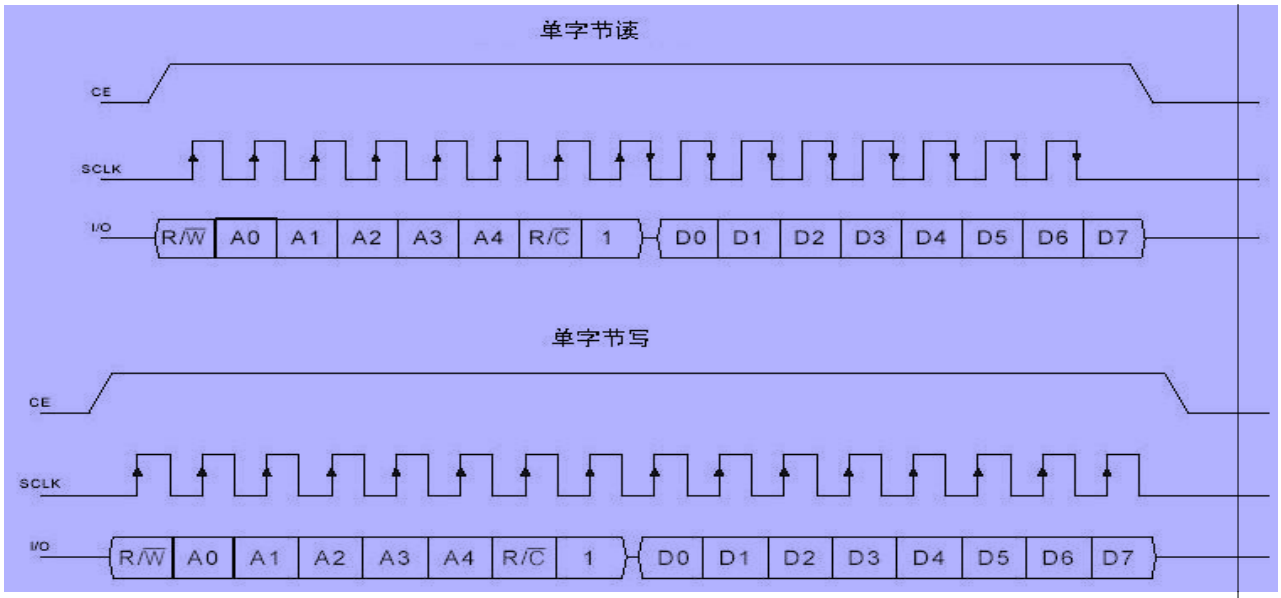


图 3.3 DS1302 数据读写时序

DS1302 的数据读写方式有两种，一种是单字节操作方式，一种是多字节操作方式。每次仅写入或读出一个字节数据称为单字节操作，每次对时钟/日历的 8 字节或 31 字节 RAM 进行全体写入或读出的操作，称其为多字节操作方式。当以多字节方式写时钟寄

存器时，必须按数据传送的次序依次写入 8 个寄存器。但是，当以多字节方式写 RAM 时，不必写所有 31 字节。不管是否写了全部 31 字节，所写的每一个字节都将传送至 RAM。

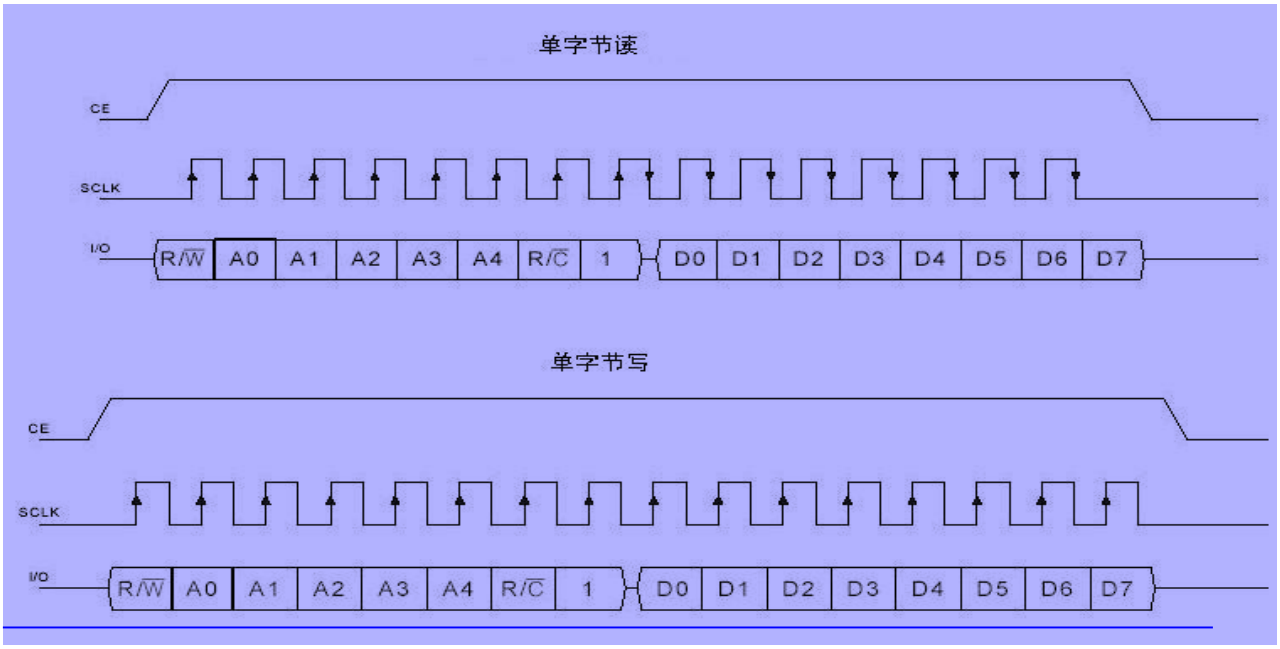


图 3.3 DS1302 数据读写时序

为了启动数据的传输，CE 引脚信号应由低变高，当把 CE 驱动至逻辑 1 的状态时，SCLK 必须为逻辑 0，数据在 SCLK 的上升沿串行输入。无论是读周期还是写周期，也无论送方式是单字节传送还是多字节传送，都要通过控制字指定 40 字节中的哪个将被访问。在开始 8 个时钟周期把命令字（具有地址和控制信息的 8 位数据）装入移位寄存器之后，另外的时钟在读操作时输出数据，在写操作时输入数据，所有的数据在时钟的下降沿变化。所有写入或读出操作都是先向芯片发送一个命令字节。对于单字节操作，包括命令字节在内，每次为 2 个字节，需要 16 个时钟；对于时钟/日历多字节模式操作，每次为 7 个字节，需要 72 个时钟；而对于 RAM 多字节模式操作，每次则为 32 字节，需要多达 256 个时钟。这里仅给出单字节读写时序，如图 3.3。多字节操作方式与其类似，只是后面跟的字节数不止一个。

3.1.2.4 DS1302 的片内寄存器

表 3.3 DS1302 有关日历、时间的寄存器

读寄存器	写寄存器	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	范围
81H	80H	CH	10 秒			秒				00-59
83H	82H		10 分			分				00-59
85H	84H	12/$\overline{24}$	0	10 AM/PM	时	时				1-12 0-23
87H	86H	0	0	10 日		日				1-31
89H	88H	0	0	10 月		月				1-12
8BH	8AH	0	0	0	0	0	周日			1-7
8DH	8CH	10 年				年				00-99
8FH	8EH	WP	0	0	0	0	0	0	0	—

通过控制字对 DS1302 片内寄存器进行寻址之后，即可就所选中寄存器的各位进行操作。片内各寄存器及各位的功能定义如表 3.3。

DS1302 有关日历、时间的寄存器共有 10 个，时钟/日历包含在其中的 7 个写/读寄存器内，这 7 个寄存器分别是秒、分、小时、日、月、星期和年。

小时寄存器（85H、84H）的位 7 用于定义 DS1302 是运行于 12 小时模式还是 24 小时模式。当为 12 小时制式时，位 5 为“0”表示 AM；为“1”表示 PM。在 24 小时制式下，位 5 是第二个 10 小时位（20～23 时）。

秒寄存器（81H、80H）的位 7 定义为时钟暂停标志（CH）。当该位置为 1 时，时钟振荡器停止，DS1302 处于低功耗状态；当该位置为 0 时，时钟开始运行。一般在设置时钟时，可以停止其工作，设定完之后，再启动其工作。

控制寄存器（8FH、8EH）的位 7 是写保护位（WP），其它 7 位均置为 0。在任何片

内时钟/日历寄存器和 RAM，在写操作之前，WP 位必须为 0，否则将不可写入。当 WP 位为 1 时，写保护位防止对任一寄存器的写操作。因此，通过置写保护位，可以提高数据的安全性。另外，还有慢速充电控制寄存器和 RAM 寄存器。如表 3.4。

表 3.4 充电控制寄存器和 RAM 寄存器各位定义

	BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
充电控制寄存器	TCS	TCS	TCS	TCS	DS	DS	RS	RS
RAM 寄存器	—	—	—	—	—	—	—	—

慢速充电寄存器控制着 DS1302 的慢速充电特性。寄存器的 BIT4～BIT7（TCS）决定是否具备充电性能：仅在编码为 1010 的条件下才具备充电性能，其他编码组合不允许充电。

BIT2 和 BIT3 选择在 V_{CC2} 和 V_{CC1} 之间是一个还是两个二极管串入其中。如果编码 DS 是 01，选择一个二极管；如果编码是 10，选择两个二极管；其他编码将不允许充电。该寄存器的 BIT0 和 BIT1 用于选择与二极管相串联的电阻值。其中编码 RS=01 为 2 KΩ，RS=10 为 4 KΩ，RS=11 为 8 KΩ，而 RS=00 将不允许进行充电。因此，根据慢速充电寄存器的不同编码可得到不同的充电电流。其具体计算如公式 3.1：

$$I_{充电}=（V_0-V_D-V_E）/R \qquad （3.1）$$

式中：

V_0——所接入的 5.0V 工作电压；

V_D——二极管压降，一个按 0.7V 计算；

R——慢速充电控制寄存器 0 和 1 位编码决定的电阻值；

V_E——V_{CC1} 脚所接入的电池电压。

RAM 寄存器寻址空间一次排列的 31 字节静态 RAM 可为用户使用，备用电源位 RAM 提供了掉电保护功能。寄存器和 RAM 的操作通过命令字节的 BIT6 加以区别。当

BIT6 为“0”时对 RAM 区进行寻址；否则将对时钟/日历寄存器寻址。其操作方法与前述相同。

3.1.3 环境温度传感器

3.1.3.1 常用温度传感器 DS18B20 简介

DS18B20 是美国 Dallas 公司生产的基于单线（1-wire）技术的数字温度传感器芯片。其管脚分布如图 3.4。

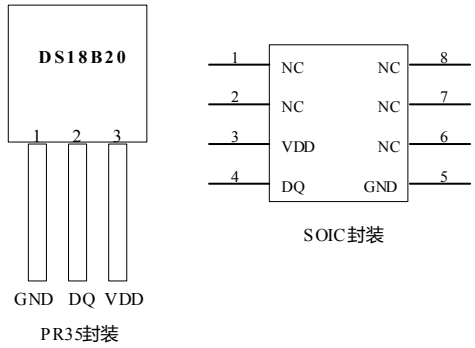


图 3.4 DS18B20 引脚分布图

每片 DS18B20 在出厂时都设有唯一的产品序列号，此序列号存放在它的内部 ROM 中，微处理器通过简单协议，就能识别这些序列号，因此多个 DS18B20 可以挂接于同一条单总线上，这允许在许多不同的地方放置温度传感器，特别适合于构成多点温度测控系统。所以 DS18B20 多应用与 HVAC 环境控制，建筑物、设备或机械内的温度检测，以及过程监视和控制中的温度检测。

管脚功能描述参见表 3.5。

表 3.5 DS18B20 详细引脚功能描述

序号	名称	引脚功能描述
1	GND	地信号
2	DQ	数据输入/输出引脚；开漏单总线接口引脚；当被用在寄生电源下，也可以

		向器件提供电源。
3	VDD	可选择的 VDD 引脚；当工作于寄生电源时，此引脚必须接地。

3.1.3.2 DS18B20 内部结构

DS18B20 的内部结构如图 3.5 所示。主要由 4 部分组成：64 位 ROM、温度传感器、非挥发的温度报警触发器 TH 和 TL、配置寄存器。

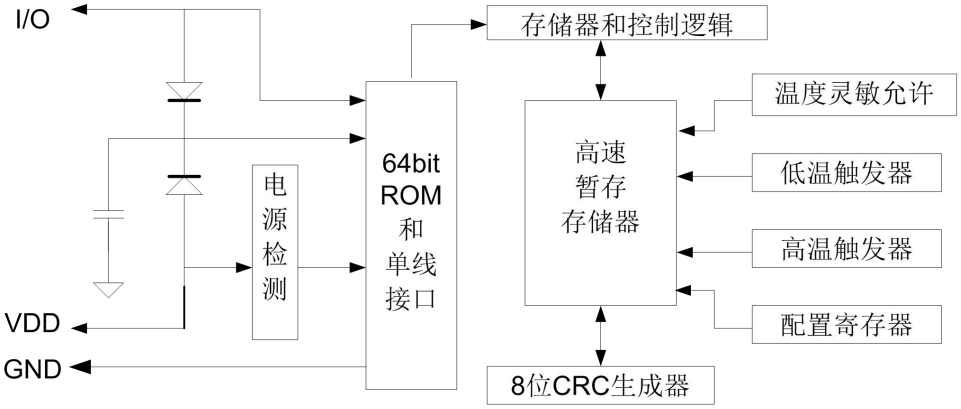


图 3.5 DS18B20 内部结构图

配置寄存器为高速暂存存储器中的第 5 个字节。DS18B20 在工作时按此寄存器中的分辨率将温度转换成相应精度的数值，其各位定义如表 3.6 所示。其中，TM 为测试模式标志位，出厂时被写入“0”，不能改变；R0、R1 是温度计分辨率设置位。

表 3.6 DS18B20 配置寄存器结构表

BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
TM	R1	R0	1	1	1	1	1

MSB LSB

其对应四种分辨率如表 3.7 所示，出厂时 R0、R1 被置为“1”，默认设置是 12 位分辨率，用户可根据需要给写配置寄存器以获得合适的分辨率。

表 3.7 配置寄存器与分辨率关系表

R0	R1	温度计分辨率/bit	最大转换时间/ms

0	0	9	93.75
0	1	10	187.50
1	0	11	375
1	1	12	750

温度信息的低位、高位字节内容还包括了符号位 S（是正温度还是负温度）和二进制小数部分，其具体形式如图 3.6。

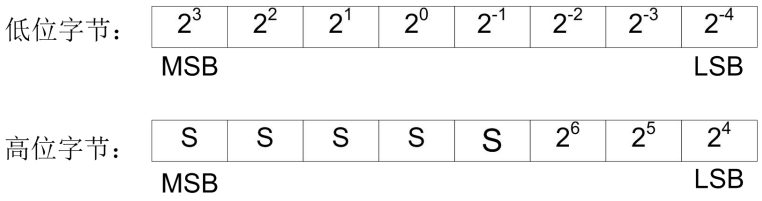


图 3.6 DS18B20 温度值格式表

图 3.6 所示是 12 位分辨率的情况，如果配置为低分辨率，则其中无意义位为“0”。

在 DS18B20 完成温度变换之后，温度值与存储在 TH 和 TL 内的告警触发值相比较。由于这些是 8 位寄存器，所以 9～12 位在比较时忽略。TH 或 TL 的高位直接对应于 16 位温度寄存器的符号位。如果温度测量的结果高于 TH 或低于 TL，那么器件内告警标志将置位，每次温度测量都会更新此标志位。只要告警标志置位，DS18B20 就将响应告警搜索命令，这也就允许单线上多个 DS18B20 同时进行温度测量，即使某处温度越限，也可以识别出正在告警的器件。

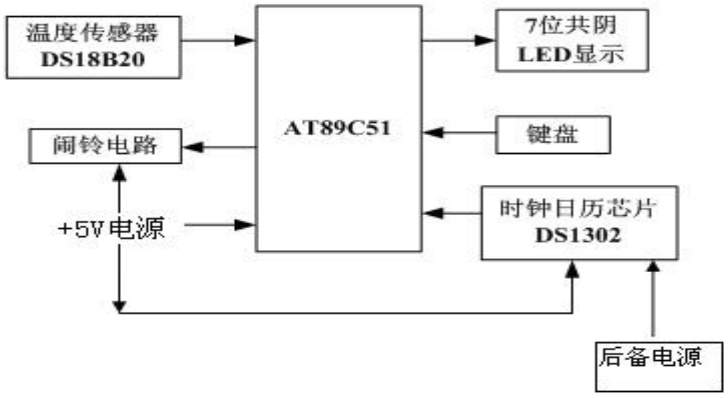
特别需要注意的是，与 DS18B20 配套使用的是频率为 11.0592MHz 单片机晶振，这决定了指令的运行时间，在软件设计中将根据此指令运行时间编写各种延时程序。

3.2 电子时钟硬件电路设计

电子闹钟至少要包括秒信号发生器、时间显示电路、按键电路、供电电源、闹铃指示电路等几部分。另外，本设计要求该电子钟能够采集环境温度，所以还需要温度采集

芯片。硬件电路框图参照图 3.7。

该系统使用 AT89C51 单片机作为核心，通过读取时钟日历芯片 DS1302 和温度传感器 DS18B20 的数据，完成此电子时钟的主要功能——时钟/日历和环境温度采集。使用比较通用的 8 段共阴数码管，做 7 位显示，分别显示时/年，分/月，秒/日，以及环境温度值。



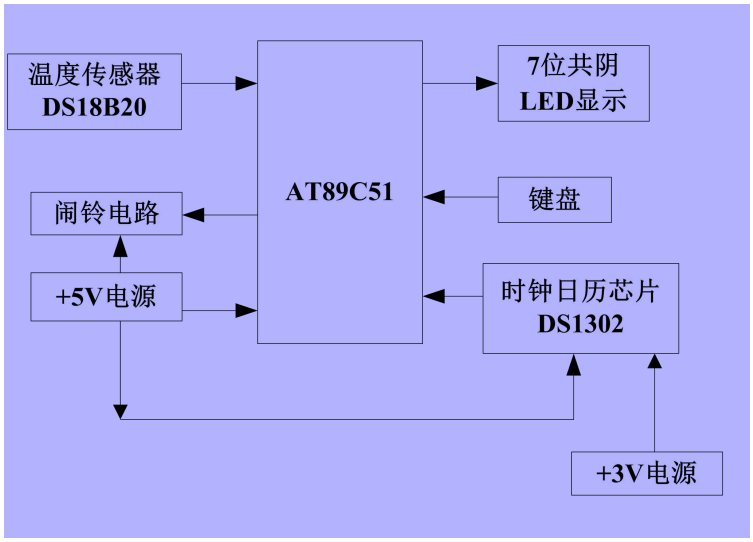


图 3.7 多功能电子时钟硬件系统框图

键盘是为了完成时钟/日历的校对和日历/温度的显示功能。由于此电子时钟要求具有闹铃功能，所以设计有闹铃电路，进行声音响铃。

整个电路使用了两种电源，+5V 电源将为整个电路供电。而+3V 电源仅作为 DS1302

的备用电源。当+5V 电源被切断后，DS1302 启用+3V 电源，可以保持 DS1302 继续工作。当+5V 电源恢复供电，LED 依旧显示当前时间，而不会因为断电使系统复位到初始化时间，避免了重新校时的麻烦。

具体电路图请参见附录。

3.2.1 时钟电路设计

系统时钟应用了实时时钟日历芯片 DS1302，其连接如图 3.8。该硬件电路设计简单，抗干扰能力强。

如图，AT89C51 单片机 P1.7 直接接 DS1302 的 RST 端，上电后，AT89C51 的 P1.7 脚自动输出高电平。P1.5 作为串行时钟接口，P1.6 作为时钟数据的 I/O。DS1302 采用双电源供电，平时由+5V 电源供电，当+5V 掉电之后，由图中 BT1（+3V 备用电池）供电。

特别需要注意 X1 和 X2 两端连接的晶振 Y1，该晶振频率为 32.768KHz。

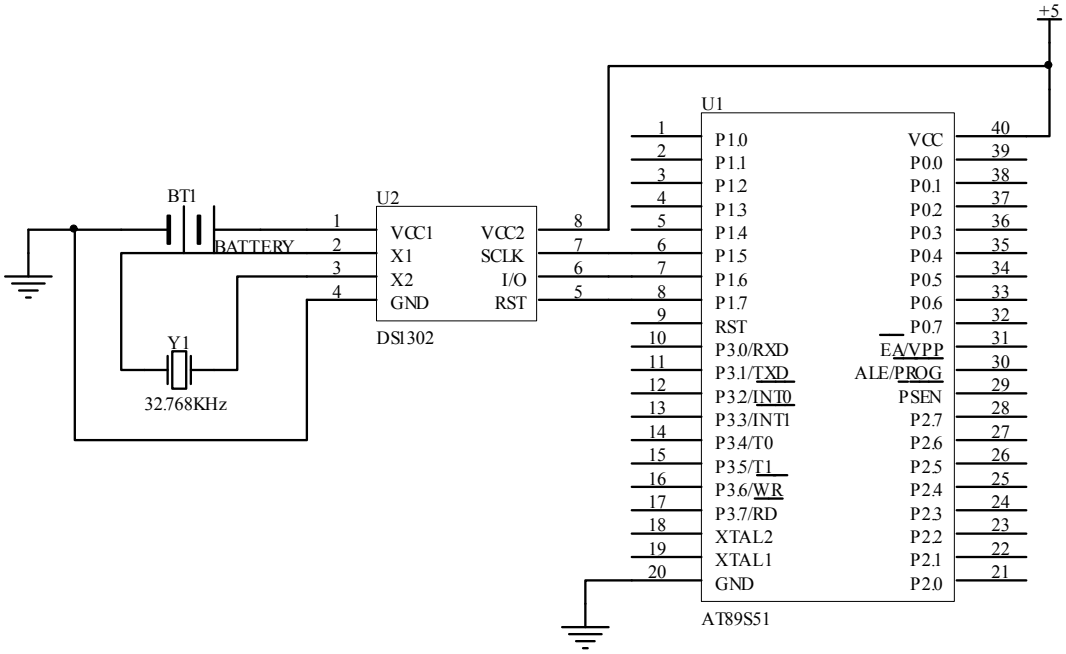


图 3.8 系统时钟电路

3.2.2 环境温度采集电路设计

本设计中使用 DS18B20 温度传感器进行环境温度采集和转化。如图 3.9 所示，

AT89C51 单片机的 P3.3 脚接 DS18B20 的 I/O 脚，作为数据的读入和写出口。电阻 R11 作为 DS18B20 的 I/O 口的上拉电阻，在读时隙结束时，I/O 引脚将通过此上拉电阻拉回至高电平。

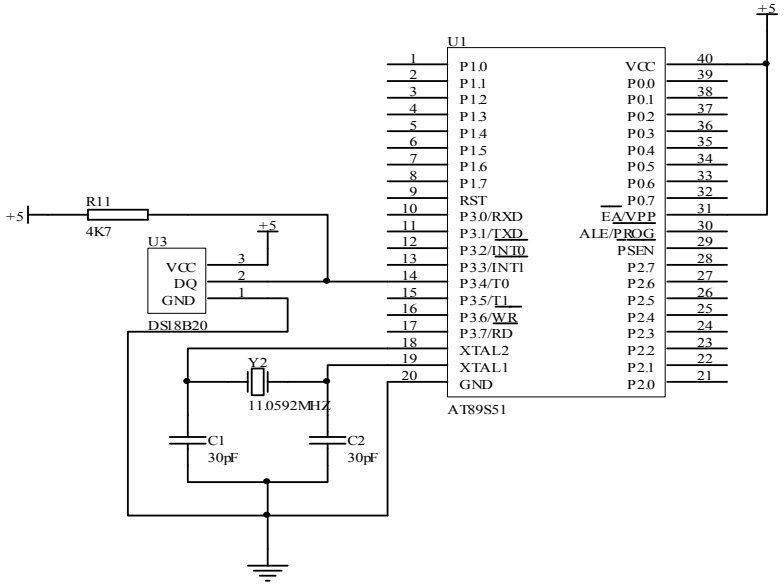


图 3.9 系统环境温度采集电路

3.2.3 显示电路

就时钟而言，通常可采用 LCD 显示或 LED 显示。对于一般的段式 LCD，需要专门的驱动电路，而且 LCD 显示的可视性较差；对于具有驱动电路和微处理器接口的液晶显示模块（字符或点阵），一般采用并行接口，对微处理器的接口要求较高，占用资源多。另外，AT89C51 本身没有专门的液晶驱动接口。LED 结构简单，体积小，功耗低，响应速度快，易于匹配，寿命长，可靠性高，而且显示亮度高，价格便宜，市场上也有专门的时钟显示组合 LED。故本设计中应用 7 位 8 段共阴 LED 实现显示部分，显示面板分布如图 3.10。

LED 显示分动态显示和静态显示：动态显示方式的硬件电路简单。但设计上如果处理不当，易造成亮度低，闪烁问题。因此合理的设计既应保证驱动电路易实现，又要保

证图像稳定，无闪烁。动态显示采用多路复用技术的动态扫描显示方式，复用的程度不是无限增加的，因为利用动态扫描显示使我们看到一幅稳定画面的实质是利用了人眼的暂留效应和发光二极管发光时间的长短，发光的亮度等因素。

静态显示，是由微型计算机一次输出显示模型后，就能保持该显示结果，直到下次发送新的显示模型为止。静态显示驱动程序简单，且 CPU 占用率低，但每个 LED 数码管需要一个锁存器来锁存每一个显示位的笔段代码，硬件开销大，仅适合显示位数较少的场合。为了在显示部分节省单片机 I/O 口，故采用静态显示方式。电路图参见图 3.10。

74LS164 是 8 位移位寄存器，应用该芯片驱动 LED 做显示部分，其优点在于连线简单，节省单片机 I/O 口，软件编程容易。

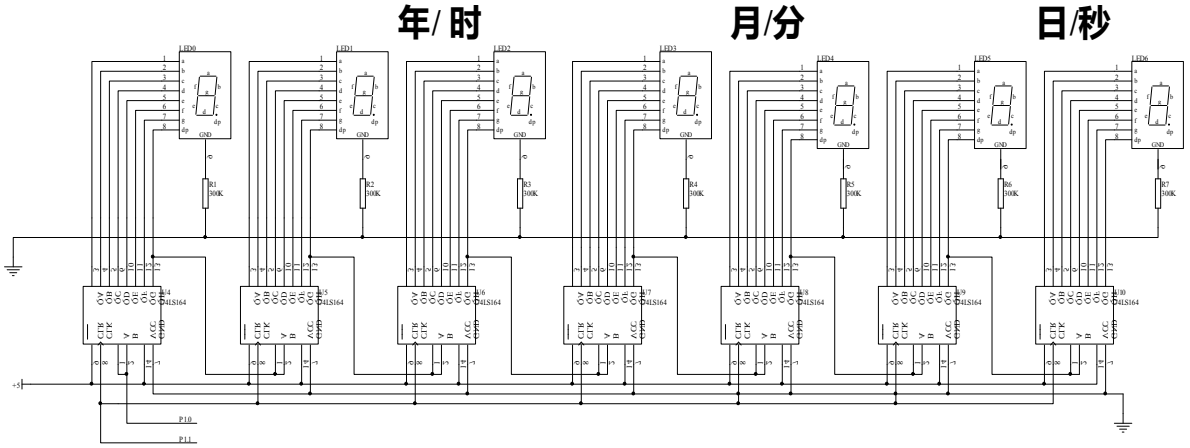


图 3.10 显示面板 LED 分布图

3.2.4 按键电路设计

根据功能需要，本时钟需要设置以下功能键：校对选择键，加 1 操作键，减 1 操作键，显示日期键，显示温度键，闹铃开关键。

按照键盘与 CPU 的连接方式可分为独立式键盘和矩阵式键盘。独立式键盘是各个按键相互独立，每个按键占用一个 I/O 口线，每根 I/O 口线上的按键不会影响其他 I/O 口上按键工作状态。独立式键盘电路配置灵活，软件结构简单，但每个按键必须占用一根 I/O 口，在按键数量较多时，I/O 口线浪费较大，且电路结构复杂。矩阵式键盘适合按键较

多时使用。由于本设计的电子钟最多需要 7 个按键，若采用矩阵式键盘时会有按键浪费，故采用的是独立式键盘。键盘电路如图 3.11。对于内置了上拉电阻的 I/O 引脚来说，外接上拉电阻没有意义。如图 3.11。

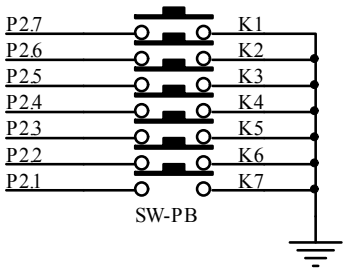


图 3.11 键盘电路

按键功能参见表 3.8。

表 3.8 按键功能表

按键	键名	功能	属性
K1	Calendar	显示日历	自锁
K2	T	显示温度	自锁
K3	FUN	功能选择	自动复位
K4	UP	数值加一操作	自动复位
K5	DOWN	数值减一操作	自动复位
K6	Enter/Snooze	确认键/贪睡	自动复位
K7	Alarm	闹铃开关	自锁

按键操作说明如下：

K1 键：该键为带自锁按键，在正常显示时间状态下，每次将按键按下， LED 数码管将显示日期；再次按下，按键弹出，重新显示时间。

K2 键：该键为带自锁按键，在正常显示时间状态下，每次将按键按下，LED 数码管将显示环境温度；再次按下，按键弹出，重新显示时间。

K3 键：该键为自动复位键，在正常显示时间状态下，第一次按下后，开始校对小时，以后每次按下都会分别进入对分、秒、闹铃时、闹铃分、年、月、日的校对状态。

K4 键：该键为自动复位键，在校对状态下，每次按动该键，都会使相应校对位进行加 1 操作。例如：校对小时状态，每按一下，小时位加 1，当加至小时最高值 23 时，再按 K4 键，小时位回 0。调分、秒、年、月、日与皆之相同，只是各位最高值不同。

K5 键：该键为自动复位键，与 K4 键类似，不同之处是该键每次按下将使相应校对位进行减 1 操作。

K6 键：该键为自动复位键，在校对状态下，按下该键，从校对状态返回时间显示状态；在响铃状态下，按下该键，闹铃进入贪睡状态。

K7 键：该键为带自锁按键，按下后闹铃开启，弹出后闹铃关闭。

3.2.5 闹铃电路设计

闹铃音乐可以直接采用蜂鸣器闹铃，如当前时刻与闹铃时间相同，单片机向蜂鸣器送出高低电平，蜂鸣器发声。采用蜂鸣器闹铃结构简单，控制方便，但是发出的闹铃声音单一。也可以在编程的时候编写一段音乐程序，待闹铃时间到时，调用该音乐程序给扬声器，便响起音乐。不过该方法只能做一些简单音乐，并且音乐程序会占用很多单片机存储资源。

闹铃的音乐不是本设计中的重点，故采用最简单的方法，占用单片机一根 I/O 口 P2.0， 中间用 PNP 型三极管 S9012 连接 P2.0 和蜂鸣器。当 P2.0 引脚为低电平时，S9012 的发射极和集电极导通，使蜂鸣器发声。当响铃标志位为“1”时，P2.0 送一定频率脉冲，使蜂鸣器 U11 发出声音。如图 3.12。

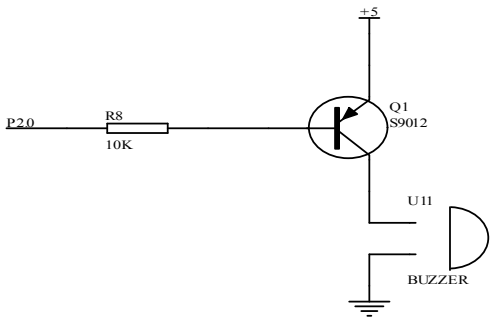


图 3.12 闹铃电路

第四章 电子时钟软件设计

本设计使用 C 语言编程。

4.1 主程序设计

第一次上电，系统先进行初始化， LED 显示初始时间“14：28：00”，并开始走时。初始日期为 2008 年 5 月 12 日，此刻若按 K1 键，LED 显示“080512”。

单片机依次开始调用键盘扫描子程序、DS1302 子程序、DS18B20 子程序、闹铃子程序，经过延时，返回程序开头循环运行。

主程序流程图如图 4.1。

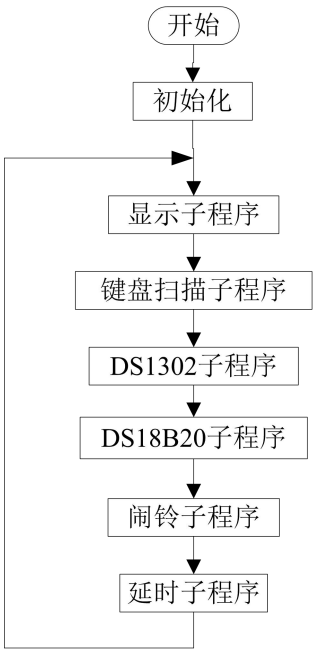


图 4.1 多功能电子钟主程序流程图

4.2 子程序设计

4.2.1 实时时钟日历子程序设计

该程序主要实现对 DS1302 写保护、充电，对年、月、日、时、分、秒等寄存器的读写操作。在读写操作子程序中都执行了关中断指令，因为在串行通信时对时序要求比

较高，而且在此是用 I/O 口软件模拟串行时钟脉冲，所以在通信过程中最好保证传输的连续性，不要允许中断。

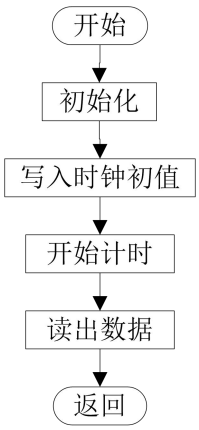


图 4.2 实时时钟日历子程序流程图

DS1302 每次上电时自动处于暂停状态，必须把秒寄存器的位 7 置位 0，时钟才开始计时。如果 DS1302 一直没有掉电，则不存在此问题。

在进行写操作时，需要先解除写保护寄存器的“禁止”状态。当用多字节模式进行操作时，必须写够 8 字节。

4.2.2 环境温度采集子程序设计

DS18B20 是 1—wire 单线器件，它在一根数据线上实现数据的双向传输，这就需要一定的协议来对读写数据提出严格的时序要求，而 AT89C51 单片机并不支持单线传输。因此，必须采用软件的方法来模拟单线的协议时序。

主机操作单线器件 DS18B20 必须遵循下面的顺序。

1. 初始化

单线总线上的所有操作均从初始化开始。初始化过程如下：主机通过拉低单线 480μs 以上，产生复位脉冲，然后释放该线，进入 RX 接收模式。主机释放总线时，会产生一个上升沿。单线器件 DS18B20 检测到该上升沿后，延时 15～60μs，通过拉低总线 60～240μs 来产生应答脉冲。主机接收到从机的应答脉冲后，说明有单线器件在线。

2. ROM 操作命令

一旦总线主机检测到应答脉冲，便可以发起 ROM 操作命令。共有 5 位 ROM 操作命令。如表 4.1。

表 4.1 DS18B20 的 ROM 操作命令

命令类型	命令字节	功能说明
Raed Rom	33H	此命令读取激光 ROM 中的 64 位，只能用于总线上单个 DS18B20 器件的情况，多挂则会发生数据冲突
Match Rom（匹配 ROM）	55H	此命令后跟 64 位 ROM 序列号，寻址多挂接总线上的 DS18B20。只有序列号完全匹配的 DS18B20 才能响应后面的内存操作命令，其他不匹配的将等待复位脉冲。此命令可用于单挂接或者多挂接总线。
Skip Rom（跳过 ROM）	CCH	此命令用于单挂接总线系统时，可以无需提供 64 位 ROM 序列号皆可运行内存操作命令。如果总线上接多个 DS18B20，并且在此命令后执行读命令，将会发生数据冲突。
Search Rom（搜索 ROM）	F0H	主机调用此命令，通过一个排除法过程，可以识别出总线上所有器件的 ROM 序列号。
Alarm Search（告警搜索）	ECH	此命令流程图和 Search Rom 命令相同，但是 DS18B20 只有在最近的一次温度测量时满足了告警触发条件，才会响应此命令。

3. 内存操作命令

在成功执行了 ROM 操作命令之后，才可以使用内存操作命令。主机可以提供 6 种内存操作命令，如表 4.2。

表 4.2 DS18B20 内存操作命令

命令类型	命令字节	功能说明
Write Scratchpad （写暂存器）	4EH	此命令写暂存器中地址 2～4 的 3 个字节（TH、TL 和配置寄存器）在发起复位脉冲之前，3 个字节都必须要写。
Rrad Scratchpad （读暂存器）	BEH	此命令读取暂存器内容，从字节 0 一直读取到字节 8。主机可以随时发起复位脉冲以停止此操作。
Copy Scratchpad （复制暂存器）	48H	此命令将暂存器中的内容复制进 E^2RAM，以便将温度告警触发字节存入非易失内存。如果在此命令后产生读时隙，那么只要器件在进行复制就会输出 0，复制完成后，再输出 1。
Convert T （温度转换）	44H	此命令开始温度转换操作。如果在此命令后主机产生读时隙，那么只要器件在进行温度转换就会输出 0，转换完成后再输出 1。
Recall E2 （重调 E2 存储器）	B8H	将存储在 E^2RAM 中的温度告警触发值和配置寄存器值重新拷贝到暂存器中。此重调操作在 DS18B20 加电时自动产生。
Read Power Supply （读供电方式）	B4H	主机发起此命令后的每个读数据时隙内，DS18B20 发信号通知它的供电方式：0 为寄生电源方式，1 为外部供电方式。

4. 数据处理

DS18B20 要求有严格的时序来保证数据的完整。在单线 DQ 上，存在复位脉冲、应答脉冲、写“0”、写“1”、读“0”和读“1”几种信号类型。其中，除了应答脉冲之外，均由主机产生。而数据位的读和写则是通过使用读、写时隙实现的。

首先了解写时隙。当主机将数据线从高电平拉至低电平时，产生写时隙。有 2 种类型的写时隙：写“1”和写“0”。所有写时隙必须在 60μs 以上（即由高拉低后持续 60μs 以上），各个写时隙之间必须保证最短 1μs 的恢复时间。DS18B20 在 DQ 线变低后的 15～60μs 的窗口对 DQ 进行采样，如果为高电平，就为写“1”；如果为低电平，就

为写“0”。对于主机产生写“1”时隙的情况，数据线必须先被拉低，然后释放，在写时隙开始后的 15μs，允许 DQ 线拉至高电平。对于主机写“0”时隙的情况，DQ 线必须被拉至低电平且至少保持低电平 60μs。

再来了解读时隙。当主机从 DS18B20 读数据时，把数据线从高电平拉至低电平，产生读时隙。数据线 DQ 必须保持低电平至少 1μs，来自 DS18B20 的输出数据在读时隙下降沿之后 15μs 内有效。因此，在此 15μs 内，主机必须停止将 DQ 引脚置低。在读时隙结束时，DQ 引脚将通过外部上拉电阻拉回至高电平。所有的读时隙最短必须持续 60μs，各个读时隙之间必须保证~~最短~~ 1μs 的恢复时间。环境温度采集子程序流程见图 4.3

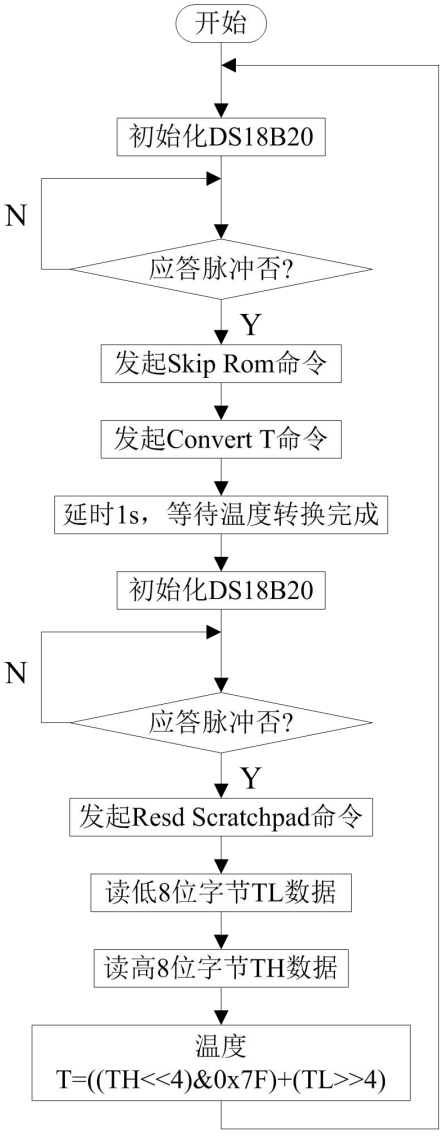


图 4.3 环境温度采集子程序流程图

所有的读写时隙至少需要 60μs，且每两个独立的时隙之间至少需要 1μs 的恢复时间。在写时序中，主机将在拉低总线 15μs 内释放总线，并向 DS18B20 写“1”。若主机拉低总线后能保持 60μs 的低电平，则向单总线器件写“0”。DS18B20 仅在主机发出读时隙时才向主机传输数据，所以，当主机向 DS18B20 发出读数据命令后，必须马上产生读时隙，以便 DS18B20 能传输数据。 ————程序见附录 C。

4.2.3 显示子程序设计

用 74LS164 驱动 LED 数码管静态显示电路，编程也很容易。只要将需要显示的数字编辑成对应的 BCD 码，逐位送入 74LS164 的 A、B 串行输入端，数码管将正常显示。关键之处是要实现根据键值显示不同的数字。显示子程序流程见图 4.4

为了方便实现按键显示，程序中调用的都是各个标志位，通过判断标志位的“真”、“假”来决定显示的内容。程序见附录 C。

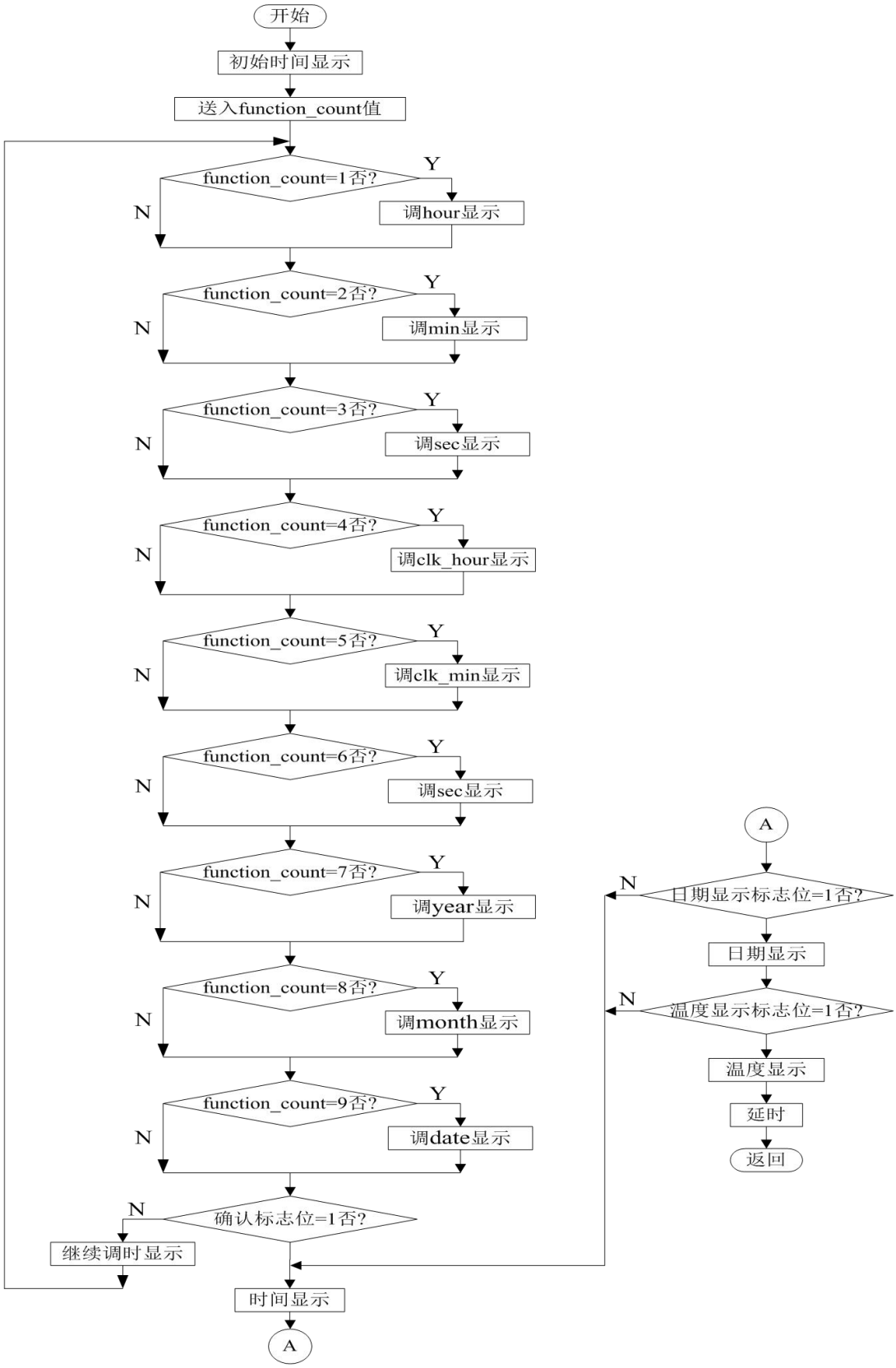


图 4.4 显示子程序流程图

~~为了方便实现按键显示，程序中调用的都是各个标志位，通过判断标志位的“真”、“假”来决定显示的内容。程序见附录 C。~~

键盘扫描子程序

单片机对键盘扫描的方法有随机扫描方式、定时扫描方式和中断扫描方式。

由于本设计中 AT89C51 单片机在系统中的主要任务是接受 DS1302 和 DS18B20 的数据并送出显示，完成时钟/日历校对和日期/温度显示控制。89C51 单片机完全有能力完成以上工作，所以采用随机扫描键盘方式，系统也能够正常运行。键盘扫描子程序流程见图 4.5。

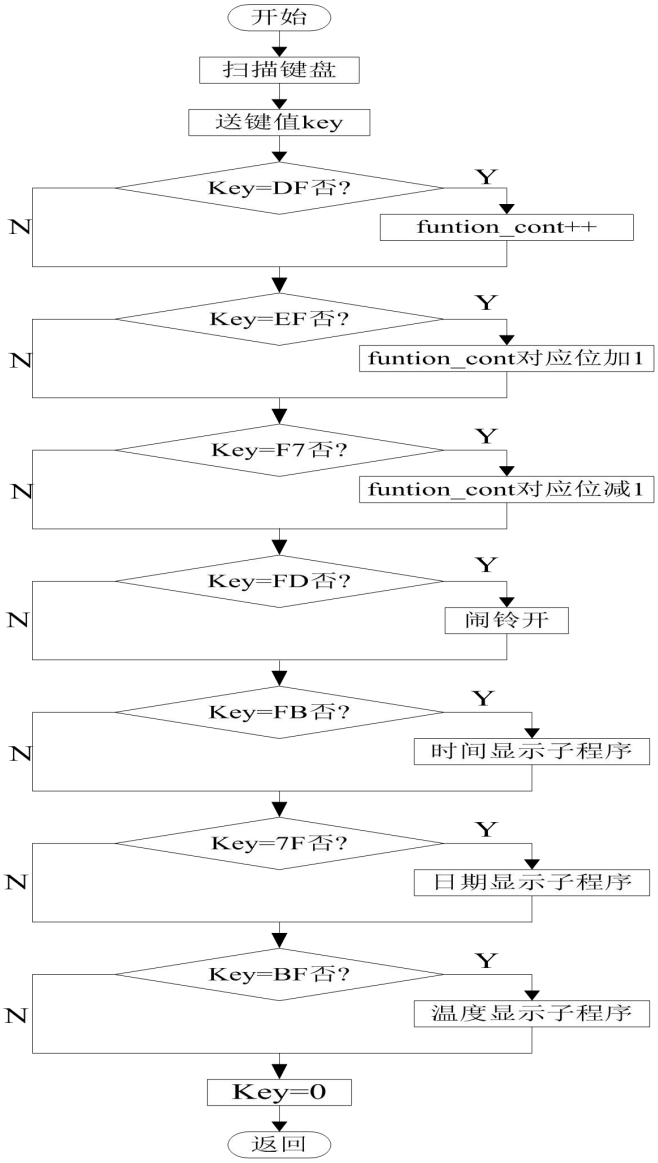


图 4.5 键盘扫描子程序

单片机扫描完键盘，得到键值，并根据键值转入执行对应任务，以实现按键功能。如果没有按键按下，则程序扫描到 Key=FFH，将键值 Key 清零，返回主程序。

程序见附录 C。

4.2.4 闹铃子程序设计

~~闹铃子程序最主要的任务是不断用时钟分~~闹铃子程序最主要的任务是不断用时钟分(min)与时(hour)同设定的闹铃分(clk_min)与闹铃时(clk_hour)比较，只要满足 min 等于 clk_min、hour 等于 clk_hour，响铃启动 5 分钟，并根据外部按键执行相应贪睡任务。程序见附录 C。

第五章 系统调试

调试工作分硬件调试和软件调试两部分，调试方法介绍如下：

首先，硬件调试主要是先搭建硬件平台，然后利用万用表等工具对电路检查，最后应用程序进行功能调试。硬件调试比较费时，需要细心和耐心，也需要熟练掌握电路原理。

然后，可以直接应用一些编辑或仿真软件进行软件调试，比如单片机 C51 编辑软件 Keil。该软件提供了一个集成开发环境 uVision，它包括 C 编辑器、宏编辑器、连接器、库管理和一个功能强大的仿真调试器。通过编译、运行，可以检查程序错误。但应用此方法，仍需要十分了解所使用元器件的工作方式和管脚连接方式。在软件调试过程中要仔细耐心，即便是多写或少些一两个字符，都无法编译成功。而有时往往在 Keil 中编译、运行无错，但烧录到单片机中运行起来就会出错，很可能是编程时管脚或时序编辑得不对。

调试过程是一个软硬件相结合调试的过程，硬件电路是基础，软件是检测硬件电路和实现其功能的关键。

在调试过程中，首先必须明确调试顺序。例如：本设计是在单片机系统基础上建立起来的，所以必须先确定单片机基础电路能否正常工作。为了正确显示时间，接下来还要确定显示电路能否正常工作。硬件调试的过程，也是软件调试的过程。

然后，要准备好调试的工具。硬件调试需要万用表、示波器等，软件调试一般需要诸如 Keil 等仿真编辑器。本人根据自己实际制作该电子时钟的经验，将调试过程介绍如下：

5.1 硬件调试

5.1.1 单片机基础电路调试

单片机基础电路包括电源、单片机、外部时钟震荡电路、复位电路和外部接口电路。调试过程需要注意以下几点：

1. 检查电源是否完好。

2. 单片机电源要连接正确，并且保证 AT89C51 的 31 号引脚接高电平。AT89C51 的 31 号引脚是外部程序存储器选择信号端，当该引脚为高电平时，单片机会一直从片内程序存储器内取指令。

3. 如果使用 P0 口做 I/O 口，要接上拉电阻。

4. 使用万用表排查电路中是否存在断路或者短路情况。焊接时容易出现管脚之间短路，所以在上电以前必须先排查电路。

5. 编辑一个使一组发光二极管循环点亮的程序并烧录到单片机内，上电运行，检查单片机是否正常工作，复位电路是否正确。

本人编辑了使一组 P1 口点亮 8 个发光二极管循环点亮的程序，程序代码如下：

```
#include <reg51.h>
void delay(void)                              //延时子程序
{
  Unsigned char i，  j，k;                    //延时时间根据变量 i，j，k 不同而改变
 for(i=50;i>0;i--)
for(j=50;j>0;j--)
for(k=250;k>0;i--);
}
void main()
```

```
{ unsigned int n;
  unsigned char code  ledp[8]={0xfe，0xfd，0xfb，0xf7，0xef，0xdf，0xbf，0x7f};
  while(1)
   { P1=0xFFH;                          //初始化 P1 口
     for(n=0;n<8;n++)
    {P1=ledp[n];
     delay();
    }}}
```

5.1.2 显示电路调试

本设计的显示电路使用了共阴 LED 和 74LS164。在连接显示电路之前要明确共阴型 8 段 LED 的 10 个管脚与各段发光二极管的对应关系，熟悉 74LS164 管脚位置，然后才能开始进行连接。在连接过程中，需要注意以下几点：

1. LED 数码管各管脚与 74LS164 各管脚的对应关系要十分清楚，所有 LED 数码管与 74LS164 的连接方式要统一。

2. 因为是移位显示，所以需要注意前一位 74LS164 的 QH 脚要与下一位 74LS164 的 A、B 脚连接。

3. 明确单片机管脚功能。本设计定义了 P1.0 连接 74LS164 的 A、B 脚，P1.1 连接 74LS164 的 CLK 脚。

4. 74LS164 的 CLR 脚接高电平。

5.检查好电源正负端和 P1.0、P1.1 连接是否正确。检查无误后上电，检查显示电路是否正确。

5.1.3 DS1302 电路调试

该电路包含 DS1302 芯片、主电源、备用电源、晶振等部分。在与单片机连接的过程中需要注意以下几点：

1. 清楚 DS1302 与单片机连接的管脚。本设计定义为：DS1302 的 SCLK 连接 P1.5，I/O 连接 P1.6，RST 连接 P1.7。
2. 注意电源正负极连接。
3. DS1302 接 32.768KHz 的晶振。该晶振体型比较小，在焊接时要小心，注意不要将晶振引脚弄断。同时也要尽量使晶振离 DS1302 的 X1、X2 引脚近距离焊接。
4. 编写 DS1302 的时钟/日历程序，只要求能够正确显示时间。烧录进单片机，检查电路电源正负极连接是否正确，检查 P1.0 和 P1.1 引脚接线是否正确。检查无误后可以上电检查。

本人编写了一段汶川特大地震发生的时间显示程序，设置初始时间为 14:28:00，初始日期为 2008 年 5 月 12 日。上电后 LED 数码管显示“142800”，之后开始走时。观察 32 分钟之后，数码管显示“150000”，证明 DS1302 电路正确。源程序见附录 C

5.1.4 按键电路调试

按键电路比较简单，故调试起来也很容易。如果确保按键焊接正确，只需在 DS1302 的调试程序上加上一段日历显示子程序，并在主程序中写入：

```
If(P_7==0)
{
  dis_calendar;
}
```

日历显示子程序原理与时钟显示子程序原理相同，源程序见附录 C。该程序的功能是：当按下 K7 时，第 1～6 位 LED 数码管马上由时间显示日期。当 K7 弹出后，数码管 1～6 位有显示日期转为显示实时时间。

5.2 软件调试

在硬件调试完毕的基础上，需要进一步完善程序，也就是进入软件调试阶段。在本

设计中，软件调试主要分三大部分：实时时钟日历子程序调试、环境温度采集子程序调试、按键子程序调试。将这三部分调试成功，那么整个设计的软件部分也就基本完成了。

在硬件调试部分，已经将实时时钟日历子程序调试完毕了，只需在主程序中调用按键子程序即可，源程序见附录 C，这里不再赘述。

5.2.1 环境温度采集子程序调试

DS18B20 温度传感器使用起来非常方便，不但接线少，而且编程容易。该温度传感器在读写数据时需要严格的时序，为了方便编写对应的延时程序，此时单片机一般都选用 11.0592MHz 的晶振。

温度显示子程序与时间显示子程序原理相同，源程序见附录 C

5.2.2 键盘子程序调试

依据设计要求，键盘子程序需要完成对时间/日历的校对、日期/温度的显示和闹铃的开关。为了便于显示子程序和闹铃子程序的调用，除了 K1、K2 键以外，其余按键都定义功能标志位。例如：

```
If(K7==0)
{
alarm_flag= true;
}
```

在调用闹铃子程序时，闹铃标志位为“1”，则开启闹铃，否则关闭闹铃。源程序见附录 C。

结　论

随着社会的发展和进步，生产生活对时钟的需求越来越大，对时钟的体型、功能的要求也各有不同。例如，本设计中的温度功能就能够给我们带来很多的方便。所以多功能的电子时钟在今后的应用也会越来越广泛。

基于单片机实现带温度显示的电子时钟，仅仅是众多方法之一。并且市场上的实时时钟日历芯片品类繁多，IC 化的传感器各种各样，显示方式也愈趋于人性化。所以这类时钟有多种实现方案，能够实现的功能也很多。本文采用 51 单片机 C 语言进行编程，当然也可以应用汇编语言编程。由于自己能力有限，汇编语言学的不是很好，故未采用汇编语言编写程序。其实我们还可以根据需求为电子时钟增设新功能，在增设新功能的同时，我们可以进一步学习单片机的使用，各种外围器件，特别是芯片的应用，为自己以后从事工程技术工作打下良好的基础。

致　谢

在论文完成之际，我首先要感谢陈卫峰老师的热情关怀和悉心指导。在我撰写论文的过程中，陈卫峰老师给与了很大的帮助与支持，无论是在论文的选题、构思和资料的收集方面，还是在论文的研究方法以及成文定稿方面，我都得到了他悉心细致的教诲和无私的帮助，特别是他广博的学识、深厚的专业技术、严谨的治学精神和一丝不苟的工作作风使我终生受益，在此表示真诚地感谢和深深的谢意。

在论文的写作过程中，也得到了许多同学的支持和帮助，特别是我们宿舍的陶星伟同学和杨兴峰同学的大力帮助与支持。在此一并致以诚挚的谢意。 感谢所有关心、支持、帮助过我的良师益友。

最后，向在百忙中抽出时间对本文进行评审并提出宝贵意见的各位老师表示衷心地感谢！

参考文献

1. 马忠梅，张凯等. 单片机的 C 语言应用程序设计（第 4 版） 北京：北京航空航天大学出版社 2007

2. 侯宝玉，陈忠平等. 基于 Proteus 的 51 系列单片设计与仿真 北京：电子工业出版社 2008

3. 电子制作杂志社 电子制作 2008 合订本 上下册 北京：《电子制作》杂志社 2008

4. 李朝青 单片机原理及接口技术（第 3 版） 北京： 北京航空航天大学出版社 2005

5. 窦振中 基于单片机的嵌入式系统工程设计 北京：中国电力出版社 2008

6. 谭浩强 C 程序设计 北京：清华大学出版社 2005

7. 田淑华 电路基础 北京：机械工业出版社 2005

8. 杨志忠 数字电子技术（第 2 版） 北京：高等教育出版社 2003

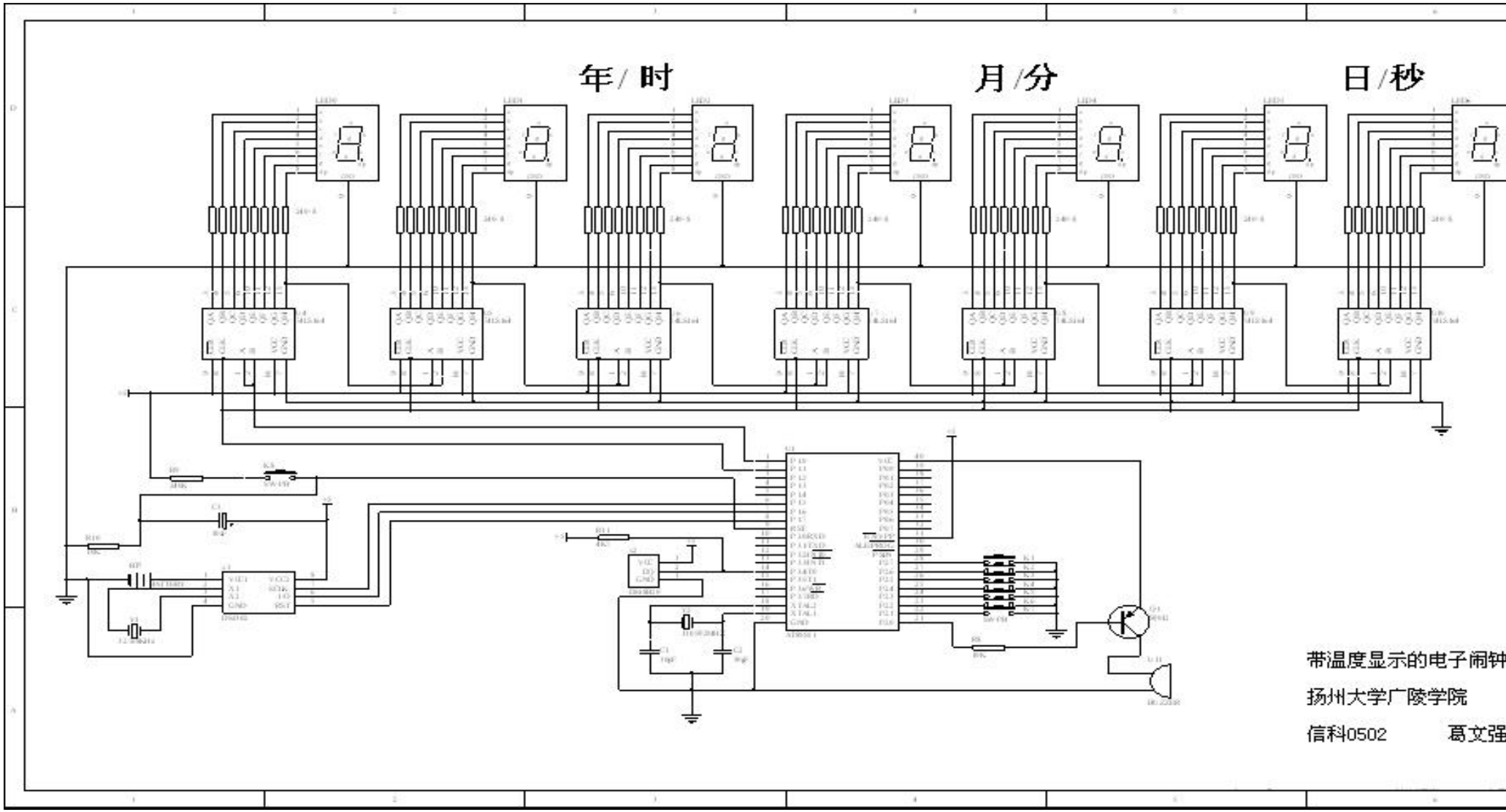
9. 胡宴如 模拟电子技术（第 2 版） 北京：高等教育出版社 2004

10. 柳淳 电子爱好者入门 北京：中国电力出版社 2008

附录 A　带温度显示的电子闹钟 元器件一览表

类别	序号	型号	数量（单位）	用途
电源	1	+5V 电池	1 个	系统电源
	2	+3V 电源	1 个	时钟备用电源
IC 芯片	3	AT89C51	1 片	CPU
	4	DS1302	1 片	实时时钟日历芯片
	5	DS18B20	1 片	环境温度采集传感器
	6	74LS164	7 片	8 位移位寄存器
LED	7	共阴 8 段数码显示	7 只	时间和温度显示
按键	12	自锁/自动复位	8 个	功能键盘
电容	13	30pF	2 个	单片机时钟震荡电路
	14	10μF	1 个	复位电路
晶振	15	11.0592MHz	1 支	单片机时钟震荡电路
	16	32.768KHz	1 支	DS1302 时钟震荡电路
电阻	17	240Ω	7 支	限制 LED 亮度
	18	100Ω	1 支	复位电路
	19	10KΩ	2 支	限压保护
三极管	20	S9012	1 个	闹铃电路
蜂鸣器	21	74F378	1 个	闹铃电路
面薄板	22	焊接性	2 块	基础电路和显示电路

附录 B　带温度显示的电子闹钟　硬件电路原理图



附录 C　带温度显示的电子闹钟程序

程序 C.1 主程序

```
//主程序的功能是对子程序进行调用，并设定显示延时时间
#include "reg51.h"                          //头文件；
#include "typedef.h"
#include "lcd.h"
#include "key.h"
#include "alarm_clock.h"
#include "ds1302.h"
sbit DAT=P1^0;                              //74LS164 的 A、B 脚接单片机 P1.0；
sbit CLK=P1^1;                              //74LS164 的 CLOCK 脚接单片机 P1.1；
sbit Calendar=P2^7;                         //定义日历显示按键 K7 接单片机 P2.7；
sbit WDZ=P2^6;                              //定义温度显示按键 K6 接单片机 P2.6；
sbit FUN=P2^5;                              //定义功能选择键 K5 接单片机 P2.5;
sbit UP=P2^4;                               //定义加 1 键 K4 接单片机 P2.4；
sbit DOWN=P2^3;                             //定义减 1 键 K3 接单片机 P2.3；
sbit Ente_Snooze=P2^2;                      //定义确认/贪睡键 K2 接单片机 P2.2；
sbit Alarm=P2^1;                            //定义闹铃开关键 K1 接单片机 P2.1；
sbit beeper=P2^0;                           //定义闹铃接口 P2.0；
#define uint unsigned int
#define uchar unsigned char
#define true      1                         //定义 true=1；
#define false     0                         //定义 false=0；
#define FUNCTION 0xDF                       //定义 FUN 键值为 DFH；
#define UP            0xEF                  //定义 UP 键值为 EFH；
#define DOWN          0xF7                  //定义 DOWN 键值为 F7H；
#define ALARM         0xFB                  //定义 ALARM 键值为 FBH；
#define Ente_Snooze       0xFB                    //定义 E/S 键值为 BFH；
void key_task(void);
void process(uchar current_key);
extern bit flash_flag;                        //定义全局变量（标志位）；
extern uchar function_count;
```

```
extern bit alarm_flag;
extern bit key_enable;
void dis();
void sendbyte();
void    reset_3w();
void    wbyte_3w(uchar);
uchar   rbyte_3w();
void    write_byte(uchar Clock_Add，uchar Clock_Data);
uchar read_byte(uchar);
void write_clock_burst();
void   ds1302_init();
void   ds1302_task();
void   lcd_disp_time1();
void   dis_WD();
void   ds18b20();
void alarm_clock(void);
void delay(unsigned int time)                    //10ms 延时
{      unsigned char a，b，c;
       for(a=0;a<time;a++)
           for(b=0;b<10;b++)
                 for(c=0;c<120;c++);      }
void main()                                      //主程序；
{       ds1302_init();                           //初始化 DS1302；
         beeper=1;                               //初始化闹铃管脚；
while(1)                                         //循环；
      {    key_task();                           //扫描键盘子程序；
           ds1302_task();                        //DS1302 子程序；
            ds18b20();                           //DS18B20 子程序；
            alarm_clock();                       //闹铃子程序；
           disp_time();                          //时间显示子程序；
            delay(55);                           //延时；
           beeper=1;                             //闹铃管脚置 1；
   }}
```

程序 C.2 电子时钟程序

```
//主要是对时钟芯片 DS1302 初始化；
```

```
#include "reg51.h"
#include "typedef.h"
#include "lcd.h"
#include "ds1302.h"
#include "key.h"
void     reset_3w();
void     wbyte_3w(uchar);
uchar    rbyte_3w();
void     write_byte(uchar Clock_Add，uchar Clock_Data);
uchar read_byte(uchar);
void write_clock_burst();
void     ds1302_init();
void    ds1302_task();
#define uint unsigned int
#define uchar unsigned char
/*-------------定义初始化时间----------*/
uchar sec=00;
uchar min=25;
uchar hour=14;
uchar date=12;
uchar month=5;
uchar year=8;
uchar day_of_week;
uchar clk_hour=14;
uchar clk_min=28;         //-14:28-
   /*-----------------定义寄存器地址----------------------*/
#define READ_SEC_ADD       0x81                  //读秒寄存器
#define READ_MIN_ADD       0x83                  //读分寄存器
#define READ_HOUR_ADD      0x85                  //读时寄存器
#define READ_DATE_ADD      0x87                  //读日寄存器
#define READ_MONTH_ADD     0x89                  //读月寄存器
#define READ_DOW_ADD       0x8B                  //读周寄存器
#define READ_YEAR_ADD      0x8D                  //年寄存器
#define WRITE_SEC_ADD      0x80                  //写秒寄存器
#define WRITE_MIN_ADD      0x82                  //写分寄存器
```

```
#define WRITE_HOUR_ADD    0x84                    //写时寄存器
#define WRITE_DATE_ADD    0x86                    //写日寄存器
#define WRITE_MONTH_ADD 0x88                      //写月寄存器
#define WRITE_DOW_ADD     0x8A                    //写周寄存器
#define WRITE_YEAR_ADD    0x8C                    //写年寄存器
#define CLOCK_BURST_ADD 0xBE                      //时钟多字节传送模式
extern void write_clock_burst();
extern void  write_byte (uchar Clock_Add，uchar Clock_Data);
void sendbyte();
void      reset_3w();
void      wbyte_3w(uchar);
uchar     rbyte_3w();
void      write_byte(uchar Clock_Add，uchar Clock_Data);
uchar read_byte(uchar);
void write_clock_burst();
void      ds1302_init();
void    ds1302_task();
sbit SCLK = P3^5;                                 //定义管脚 SCLK
sbit IO      = P3^6;                              //定义管脚 I/O
sbit RST    = P3^7;                               //定义管脚 RST
uchar sec，min，hour，date，month，year;
uchar code hex2bcd[] = {
    0x00，  0x01，  0x02，  0x03，  0x04，  0x05，  0x06，  0x07，  0x08，  0x09，
/* 00-09 */
    0x10，  0x11，  0x12，  0x13，  0x14，  0x15，  0x16，  0x17，  0x18，  0x19，
/* 10-19 */
    0x20，  0x21，  0x22，  0x23，  0x24，  0x25，  0x26，  0x27，  0x28，  0x29，
/* 20-29 */
    0x30，  0x31，  0x32，  0x33，  0x34，  0x35，  0x36，  0x37，  0x38，  0x39，
/* 30-39 */
    0x40，  0x41，  0x42，  0x43，  0x44，  0x45，  0x46，  0x47，  0x48，  0x49，
/* 40-49 */
    0x50，  0x51，  0x52，  0x53，  0x54，  0x55，  0x56，  0x57，  0x58，  0x59，
/* 50-59 */
    0x60，  0x61，  0x62，  0x63，  0x64，  0x65，  0x66，  0x67，  0x68，  0x69，
```

```
    /* 60-69 */
        0x70，  0x71，  0x72，  0x73，  0x74，  0x75，  0x76，  0x77，  0x78，  0x79，
    /* 70-79 */
        0x80，  0x81，  0x82，  0x83，  0x84，  0x85，  0x86，  0x87，  0x88，  0x89，
    /* 80-89 */
        0x90，  0x91，  0x92，  0x93，  0x94，  0x95，  0x96，  0x97，  0x98，  0x99，
    /* 90-99 */
    };
    uchar code bcd2hex[] = {
         0，   1，   2，   3，   4，   5，   6，   7，   8，   9，  0，  0，  0，  0，  0，
0，  /* 00-09 */
        10，  11，  12，  13，  14，  15，  16，  17，  18，  19，  0，  0，  0，  0，  0，
0，  /* 10-19 */
        20，  21，  22，  23，  24，  25，  26，  27，  28，  29，  0，  0，  0，  0，  0，
0，  /* 20-29 */
        30，  31，  32，  33，  34，  35，  36，  37，  38，  39，  0，  0，  0，  0，  0，
0，  /* 30-39 */
        40，  41，  42，  43，  44，  45，  46，  47，  48，  49，  0，  0，  0，  0，  0，
0，  /* 40-49 */
        50，  51，  52，  53，  54，  55，  56，  57，  58，  59，  0，  0，  0，  0，  0，
0，  /* 50-59 */
        60，  61，  62，  63，  64，  65，  66，  67，  68，  69，  0，  0，  0，  0，  0，
0，  /* 60-69 */
        70，  71，  72，  73，  74，  75，  76，  77，  78，  79，  0，  0，  0，  0，  0，
0，  /* 70-79 */
        80，  81，  82，  83，  84，  85，  86，  87，  88，  89，  0，  0，  0，  0，  0，
0，  /* 80-89 */
        90，  91，  92，  93，  94，  95，  96，  97，  98，  99，  0，  0，  0，  0，  0，
0，  /* 90-99 */
    };
    void reset_3w()                          //复位子程序
    {   SCLK = 0;
        RST = 0;                             //复位 DS1302，中止数据传送
        RST = 1;                             //启动数据传送
    }
```

```
void wbyte_3w(uchar W_Byte)              //写字节子程序
{   uchar i;
    for(i = 0; i < 8; ++i)
    {   IO = 0;
        if(W_Byte & 0x01)
        { IO = 1; }
        SCLK = 0;
        SCLK = 1;
        W_Byte >>= 1;              }}
uchar   rbyte_3w()                   //读字节子程序
{   uchar i;
    uchar R_Byte;
    uchar TmpByte;
    R_Byte = 0x00;
    IO = 1;
    for(i = 0; i < 8; i++)
    {   SCLK = 1;
        SCLK = 0;
        TmpByte = (uchar)IO;
        TmpByte <<= 7;
        R_Byte >>= 1;
        R_Byte |= TmpByte;       }
    return R_Byte;       }
void     write_byte(uchar Clock_Add，uchar Clock_Data)
{   reset_3w();
    wbyte_3w(Clock_Add);
    wbyte_3w(Clock_Data);
    reset_3w();          }
void write_clock_burst()
{   reset_3w();                          //复位
    wbyte_3w(CLOCK_BURST_ADD);    //写多字节传送模式寄存器
    wbyte_3w(hex2bcd[sec]);              //写入初始化时间
    wbyte_3w(hex2bcd[min]);
    wbyte_3w(hex2bcd[hour]);
    wbyte_3w(hex2bcd[date]);
```

```
    wbyte_3w(hex2bcd[month]);
    wbyte_3w(hex2bcd[day_of_week]);
    wbyte_3w(hex2bcd[year]);
    wbyte_3w(0);          /* must write control register in burst mode */
    reset_3w();
}
uchar   read_byte(uchar Clock_Add)        //  从 DS1302 读一个字节
{   uchar Clock_Data;
    reset_3w();
    wbyte_3w(Clock_Add);
    Clock_Data = rbyte_3w();
    reset_3w();
    return(Clock_Data);          }
void    ds1302_init()    /* --- initialize time & date for default value --- */
{   reset_3w();
    wbyte_3w(0x8e);     //  写保护寄存器
    wbyte_3w(0);        //去保护
    reset_3w();
    wbyte_3w(0x90);     //  写涓流充电寄存器
wbyte_3w(0xab);    //开涓流充电，在 VCC1 和 VCC2 之间串入 2 个二极管，8KΩ电阻
    write_clock_burst();    //写入初始化时间
}
void ds1302_task()
{   if(!key_enable)
    {    sec = bcd2hex[read_byte(READ_SEC_ADD)];          //读 DS1302 sec
         min = bcd2hex[read_byte(READ_MIN_ADD)];        //读 DS1302 min
         hour = bcd2hex[read_byte(READ_HOUR_ADD)];    //读 DS1302 hour
        date = bcd2hex[read_byte(READ_DATE_ADD)];
      day_of_week = bcd2hex[read_byte(READ_DOW_ADD)];
         month = bcd2hex[read_byte(READ_MONTH_ADD)];    //读 DS1302 month
         year = bcd2hex[read_byte(READ_YEAR_ADD)];      //读 DS1302 year
   }}
```

程序 C.3 键盘子程序

```
//主要是用于对时间的设定与调整
#include "reg51.h"
```

```
#include "typedef.h"
#include "key.h"
#include "ds1302.h"
uint wait_time;
bit    key_enable;
bit   flash_flag;
bit   alarm_flag;
uchar flash_count;
uchar function_count=0;
uchar key;
uchar key_push;
uchar key_l;
void key_scan()                                    //扫描键盘；
{    uchar l，a，PUSH;
      PUSH=P2&0xFF;
     if( PUSH!=0xFF)
    {    for(l=500;l>0;l--)                        //延时；
        for(a=50;a >0;a--) ;
      if( PUSH!=0xFF)
      {    key_push=P2&0xFF;
           key=key_push;                           //key 等于键值；
      }}}
void process(uchar current_key)                    //按键功能子程序；
{    switch (current_key)
     {case FUNCTION:                               //功能选择键；
           {     key_enable=true;
                 function_count=function_count+1;
            if(function_count>=9)
                  function_count=1;        }
        break;
     case UP:                                      //加 1 操作键；
         {        if(function_count==1)             //function_count=1，则秒加 1 操作；
          {       sec++;
                  if(sec>=60)                      //秒加到 60，则被置 0；
                      sec=0;
```

```
            write_byte(WRITE_SEC_ADD，hex2bcd[sec]);  //写入秒寄存器；
    }
    if(function_count==2)                    //function_count=2，则分加 1 操作；
     {        min++;
            if(min>=60)                      //分加到 60，则被置 0；
                min=0;
            write_byte(WRITE_MIN_ADD，hex2bcd[min]);  //写入分寄存器；
    }
        else if(function_count==3)           //function_count=3，则小时加 1 操作；
        {       hour++;
            if(hour>=24)                     //小时加到 24，则被置 0；
                hour=0;
            write_byte(WRITE_HOUR_ADD，hex2bcd[hour]);    //写入小时寄存器；
        }
    else if(function_count==4)               //function_count=4，则闹铃分钟加 1 操作；
        {    clk_min++;
            if(clk_min>=60)                  //分钟加满 60 自动置 0；
                clk_min=0;        }
        else if(function_count==5)           //function_count=5，则闹铃小时加 1 操作；
        {      clk_hour++;
            if(clk_hour>=24)                 //小时加满 24 自动置 0；
                clk_hour=0;          }
        else if(function_count==6)           //function_count=6，则年加 1 操作；
        {      year++;
              if(year>99)
                year=0;
            write_byte(WRITE_YEAR_ADD，hex2bcd[year]);
        }  }
        else if(function_count==7)           //function_count=7，则月加 1 操作；
        {    month++;
            if(month>=13)
            month=1;
            write_byte(WRITE_MONTH_ADD，hex2bcd[month]);        }
        else if(function_count==8)           //function_count=8，则日加 1 操作；
                {   date++;
```

```
                    if(date>=31)
                      date=0;
            write_byte(WRITE_DATE_ADD，hex2bcd[date]);        }
        break;
    case DOWN:                                  //键盘减 1 操作功能
{   if(function_count==1)
        {       if(sec==0)
                  sec=60;
                 --sec;
            write_byte(WRITE_SEC_ADD，hex2bcd[sec]);          }
        if(function_count==2)
        {     if(min==0)
            min=60;
            --min;
            write_byte(WRITE_MIN_ADD，hex2bcd[min]);        }
        else if(function_count==3)
        {       if(hour==0)
                  hour=24;
            --hour;
            write_byte(WRITE_HOUR_ADD，hex2bcd[hour]);          }
else if(function_count==4)
        {     if(clk_min==0)
                  clk_min=60;
            --clk_min;}
        else if(function_count==5)
        {   if(clk_hour==0)
                  clk_hour=24;
            --clk_hour;   }
        else if(function_count==6)
          {       if(year==0)
                  year=100;
            --year;
            write_byte(WRITE_YEAR_ADD，hex2bcd[year]);     }
        else if(function_count==7)
        {       if(month==1)
```

```
                    month=13;
                    --month;
                write_byte(WRITE_MONTH_ADD，hex2bcd[month]);          }
            else if(function_count==8)
            {       if(date==0)
                    date=31;
                    --date;
                write_byte(WRITE_DATE_ADD，hex2bcd[date]);            }
        break;
    case ALARM:
        {       alarm_flag=(~alarm_flag);          }
        break;
case Ente_Snooze
    {          Time= true;                                          //确认键；
    if((alarm_flag==true)&(clk_hour==hour)&(clk_min==min))    //贪睡功能；
        {          clk_min = clk_min +5;          } }
   default:
        break;              }}
void key_task()
{       key_scan();
        if(key==0x29)
    {     wait_time=wait_time+1;
        if(wait_time==3)
        {     wait_time=0;
            key_enable=false;          }}
    else
        wait_time=0;
    if(function_count!=0)
    {       flash_count++;
                if(flash_count<20)
            flash_flag=true;
       else if((flash_count>20)&&(flash_count<40))
            flash_flag=false;
             if(flash_count>=40)
            flash_count=0;     }
```

```
 process(key);
 if(Calendar==0)                    //如果 K7 按下，显示日期；
      {dis_Calendar ()；}
 else if(WDZ==0)                    //如果 K6 按下，显示温度；
       {void   dis_WD();}
  if(Time=true)                     //如果按下确认键，直接显示时间;
       { disp_time();}
 key=0;          }
```

程序 A.4 闹铃程序　　主要是设定闹铃的条件

```
#include "reg51.h"
#include "typedef.h"
#include "lcd.h"
#include "alarm_clock.h"
#include "key.h"
uchar freq_count;
void alarm_clock()
{     unsigned char a，b，c;
   if((alarm_flag==true)&(clk_hour==hour)&(clk_min==min))
   {    freq_count++;
        if(freq_count<30)
        {          beeper=false;       }
   for(a=0;a<2;a++)
      for(b=0;b<10;b++)
        for(c=0;c<120;c++);
        if(freq_count>=30)
        {      beeper=true;       }
          if(freq_count==100)
          freq_count=0;        }}
```

程序 C.5 显示程序

```
//当符合不同的条件时，该程序使数码管显示不同的数据；
#include "reg51.h"
#include "typedef.h"
#include "lcd.h"
#include "key.h"
uchar sec=45;
```

```
uchar min=0;
uchar hour=8;
uchar date=1;
uchar month=1;
uchar year=6;
uchar day_of_week;      //2006-01-01 Sun 00:00:00
uchar clk_hour=8;
uchar clk_min=1;        //08:00
uchar disp_buffer2[7];
unsigned char
dis[]={0x3f，0x06，0x5b，0x4f，0x66，0x6d，0x7d，0x07，0x7f，0x6f，0x08，0x40，
0x01};//低中高
void   disp_time(void)
{    unsigned char gsb，led，led1，jj;
    if(function_count<=1)
      {      if((flash_flag==true)&&(function_count==1))     //时显示
             {     disp_buffer[0]=dis[3];
                   disp_buffer[2]=dis[hour%10];    //正常显示
                   disp_buffer[1]=dis[hour/10];
        if(key_enable==false)
             {   function_count=0;
                 disp_buffer[0]=dis[11];           } }
         else
           {        if(alarm_flag==true)       //闹钟开关显示
                     disp_buffer[0]=dis[12];
          else
              if(function_count==0)
              disp_buffer[0]=0;
               disp_buffer[2]=dis[hour%10];
              disp_buffer[1]=dis[hour/10];      }
     if((flash_flag==true)&&(function_count==2))  //分显示
      {       disp_buffer[0]=dis[2];
             disp_buffer[4]=dis[min%10];   //正常显示
             disp_buffer[3]=dis[min/10];
              if(key_enable==false)
```

```
        {     function_count=0;
              disp_buffer[0]=dis[11];          } }
    else
      {     disp_buffer[4]=dis[min%10];
           disp_buffer[3]=dis[min/10];          }
if((flash_flag==true)&&(function_count==3))   //秒显示
         {     disp_buffer[0]=dis[3];
               disp_buffer[6]=dis[sec%10];    //正常显示
              disp_buffer[5]=dis[sec/10];
             if(key_enable==false)
        {          function_count=0;
              disp_buffer[0]=dis[11];
         }}
       else
       {        disp_buffer[6]=dis[sec%10];
          disp_buffer[5]=dis[sec/10];          }
       for(gsb=0;gsb<7;gsb++)
          {    led=disp_buffer[gsb];
           for(jj=0;jj<8;jj++)
              {     led1=led&0x1;
                   if (led1==0x1)
                  data_led=1;//DATA=1
                   else
                        data_led=0;//DATA=0
                   clk_led=0;clk_led=1;//clk=0~1
                        led=led>>1;          }}}
else   if(function_count>3)
       {    if((flash_flag==true)&&(function_count==4))     //约定时间设置
            {     disp_buffer2[1]=dis[11];
                  disp_buffer2[2]=dis[11];
                  disp_buffer2[3]=dis[11];
                  disp_buffer2[4]=dis[11];
                   disp_buffer2[0]=dis[4]；
                  disp_buffer2[6]=dis[clk_min%10];    //预约分正常显示
                   disp_buffer2[5]=dis[clk_min/10];
```

```
            if(key_enable==false)
             {        function_count=0;
                      disp_buffer[0]=dis[11];          } }
  if((flash_flag==true)&&(function_count==5))
    {     disp_buffer2[1]=dis[11];
          disp_buffer2[2]=dis[11];
          disp_buffer2[5]=dis[11];
          disp_buffer2[6]=dis[11];
           disp_buffer2[0]=dis[5];
          disp_buffer2[4]=dis[clk_hour%10];      //预约时正常显示
           disp_buffer2[3]=dis[clk_hour/10];
         if(key_enable==false)
    {     function_count=0;
        disp_buffer[0]=dis[11];           }}
if((flash_flag==true)&&(function_count==8))        //日设置
    {   disp_buffer2[1]=dis[11];
        disp_buffer2[2]=dis[11];
        disp_buffer2[3]=dis[11];
        disp_buffer2[4]=dis[11];
        disp_buffer[0]=dis[6];
        disp_buffer2[6]=dis[date%10];       //日正常显示
         disp_buffer2[5]=dis[date/10];
         if(key_enable==false)
    {        function_count=0;
           disp_buffer[0]=dis[11];          }}
  if((flash_flag==true)&&(function_count==7))        //月设置
    {        disp_buffer2[1]=dis[11];
             disp_buffer2[2]=dis[11];
             disp_buffer2[5]=dis[11];
             disp_buffer2[6]=dis[11];
              disp_buffer2[0]=dis[7];
             disp_buffer2[4]=dis[month%10];          //月正常显示
              disp_buffer2[3]=dis[month/10];
               if(key_enable==false)
             {        function_count=0;
```

```
                    disp_buffer[0]=dis[11];             } }
    if((flash_flag==true)&&(function_count==6))        //年设置
       {    disp_buffer2[5]=dis[11];
            disp_buffer2[6]=dis[11];
            disp_buffer2[3]=dis[11];
            disp_buffer2[4]=dis[11];
            disp_buffer2[0]=dis[8];
            disp_buffer2[2]=dis[year%10];        //年正常显示
             disp_buffer2[1]=dis[year/10];
              if(key_enable==false)
            {       function_count=0;
                    disp_buffer[0]=dis[11];
            } }
        for(gsb=0;gsb<7;gsb++)
             {   led=disp_buffer2[gsb];
                 for(jj=0;jj<8;jj++)
                    {       led1=led&0x1;
                            if (led1==0x1)
                          data_led=1;//DATA=1
                     else
                           data_led=0;//DATA=0
                        clk_led=0;clk_led=1;//clk=0~1
                        led=led>>1;
                     }}} }
void dis_Calendar ()                                  //显示日历子程序
   {
    unsigned char gsb,led,led1,jj;
    disp_buffer[2]=tab[hour%10];
    disp_buffer[1]=tab[hour/10];
    disp_buffer[4]=tab[min%10];   //正常显示
    disp_buffer[3]=tab[min/10];
    disp_buffer[6]=tab[sec%10];    //正常显示
    disp_buffer[5]=tab[sec/10];
     for(gsb=0;gsb<7;gsb++)
             {
```

```
                led=disp_buffer[gsb];
                for(jj=0;jj<8;jj++)
                  {
                       led1=led&0x1;
                      if (led1==0x1)
                      DAT=1;//DATA=1
                  else
                      DAT=0;//DATA=0
                  CLK=0;CLK=1;//clk=0~1
                       led=led>>1;
    } }}}
```

程序 C.6 环境温度采集显示子程序

```
//精度 0.1℃，四位数码管显示。
#include "reg51.h"
#include "intrins.h"                                    //_nop_();延时函数用
#define   Disdata P0                                    //段码输出口
#define   discan   P2                                   //扫描口
#define   uchar unsigned char
#define   uint   unsigned int
sbit       DQ=P3^4;                                     //温度输入口
sbit DAT=P1^0;                          //74LS164 的 A、B 脚接单片机 P1.0；
sbit CLK=P1^1;                          //74LS164 的 CLOCK 脚接单片机 P1.1；
sbit       music=P2^0;
sbit       hold=P2^6;
uint       h;
uint       temp;
unsigned char presence,flash=0;
uchar code ditab[16]=
{0x00,0x01,0x01,0x02,0x03,0x03,0x04,0x04,0x05,0x06,0x06,0x07,0x08,0x08,0x09,0x09};
    //温度小数部分用查表法
uchar code tab[]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,0xfe,0xf6,0x00,0x02,0x02};
//共阴 LED 段码表 "0"   "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9" "不亮"   "-"
uint data temp_data[2]={0x00,0x00};                     //读出温度暂放
uchar display[7];
uchar gsb,led,led1,jj;
```

```
/*****************延时函数************************/
void delay_2(uint t)
{       for (;t>0;t--);         }
/*  蜂鸣器驱动函数
 beep()
  {     unsigned char i;
     for (i=0;i<100;i++)
     {     music=1;
           delay_2(20);
          music=0;
           delay_2(20);        } }
/****************DS18B20 复位函数************************/
ow_reset(void)
{       char presence=1;
        while(presence)
    {     while(presence)
          {       DQ=1;_nop_();_nop_();//从高拉倒低
                  DQ=0;
                  Delay_2(50);                    //550 μ s
                  DQ=1;
                  delay(6);                     //66 μ s
                  presence=DQ;                    //presence=0  复位成功,继续下一步
       }
    Delay_2(45);                  //延时 500 us
    presence=~DQ;             }
DQ=1;                           //拉高电平
}
/****************DS18B20 写命令函数************************/
void write_byte(uchar val)                          //向 1-WIRE  总线上写 1 个字节
{     uchar i;
      for(i=8;i>0;i--)
       {       DQ=1;_nop_();_nop_();                          //从高拉倒低
               DQ=0;_nop_();_nop_();_nop_();_nop_();    //5 us
               DQ=val&0x01;                                    //最低位移出
               Delay_2(6);                                     //66 μ s
```

```
            val=val/2;                                  //右移 1 位
        }
    DQ=1;
    delay(1);          }
/****************DS18B20 读 1 字节函数***********************/
uchar read_byte(void)   //从总线上取 1 个字节
{       uchar i;
        uchar value=0;
    for(i=8;i>0;i--)
    {     DQ=1;_nop_();_nop_();
          value>>=1;
          DQ=0;_nop_();_nop_();_nop_();_nop_();          //4μs
          DQ=1;_nop_();_nop_();_nop_();_nop_();          //4  μs
          if(DQ)value|=0x80;
          delay_2(6);                                    //66μs
    }
DQ=1;
return(value);
}
/****************读出温度函数***********************/
read_temp()
{
   ow_reset();                                  //总线复位
     if(presence==1)
    { beep();flash=1;}                          //DS18B20 不正常，蜂鸣器报警
     Delay_2(200);
   write_byte(0xcc);                            //发命令
   write_byte(0x44);                            //发转换命令
   ow_reset();
   delay(1);
   write_byte(0xcc);                            //发命令
   write_byte(0xbe);
   temp_data[0]=read_byte();                    //读温度值的第字节
   temp_data[1]=read_byte();                    //读温度值的高字节
   temp=temp_data[1];
```

```
   temp<<=8;
   temp=temp|temp_data[0];                  // 两字节合成一个整型变量。
   return temp;                             //返回温度值
}
/****************温度数据处理函数**********************/
//二进制高字节的低半字节和低字节的高半字节组成一字节,这个
//字节的二进制转换为十进制后,就是温度值的百、十、个位值,而剩
//下的低字节的低半字节转化成十进制后,就是温度值的小数部分
/*********************************************************/
work_temp(uint tem)
{
    uchar n=0;
    if(tem>6348)                                // 温度值正负判断
       {tem=65536-tem;n=1;}                  // 负温度求补码,标志位置 1
        display[6]=tem&0x0f;                  // 取小数部分的值
   display[2]=ditab[display[6]];             // 存入小数部分显示值
   display[6]=tem>>4;                        // 取中间八位,即整数部分的值
     display[0]=13;
    display[1]=12;
   display[5]=(display[6])/100;              // 取百位数据暂存
   display[3]=(display[6])%100;              // 取后两位数据暂存
   display[4]=(display[3])/10;               // 取十位数据暂存
   display[3]=(display[3])%10;
/******************符号位显示判断**************************/
   if(!display[5])
   {       display[5]=10;                    //最高位为 0 时不显示
            if(!display[4])
   {        display[4]=10;                    //次高位为 0 时不显示
             }}
   if(n)
   {        display[5]=11;   //负温度时最高位显示"}
   if(display[4]==1)
  {       beep();                          //当温度低于 20℃时开始报警
   }}
disp()
```

```
{     for(gsb=0;gsb<4;gsb++)
            {     led=tab[display[2+gsb]];
                for(jj=0;jj<8;jj++)
                  {     led1=led&0x1;
                        if (led1==0x1)
                        DAT=1;//DATA=1
                  else
                       DAT=0;//DATA=0
                  CLK=0;CLK=1;//clk=0~1
                       led=led>>1;
 } }}
/****************温度采集显示子程序************************/
ds18b20 ()
{ int i;
   Disdata=0xff;                          //初始化端口
   discan=0xff;
   hold=1;
   ow_reset();                            //开机先转换一次
   write_byte(0xcc);                      //Skip ROM
   write_byte(0x44);                      //发转换命令
     while(1)
   {     if(flash)
         Disdata=0x00;                    //DS18B20 不正常关闭显示
        if(hold==0)
        for(i=0;;i++)
        {disp();        if(hold==1) break;       }
         else
         { work_temp(read_temp());              //处理温度数据
           disp();
        }                                 //显示温度值
        Delay_2(100);          disp();
   delay(1000);  delay(1000);  delay(1000);
        }}
```