基于单片机的家用电器定时开关控制系统设计

摘要：本次研究的家用电器定时控制装置的主要目的是达到准确控制时间，以节省能源和增加家用电器的使用寿命。完成自动控制，减少不必要的人为操作，节约用人成本。使许多需要人控制时间的工作变得更加简单、快捷。研究的主要内容是基于单片机的家用电器定时开关的控制，其主要由电源电路、实时时钟电路、显示电路、键盘输入电路、继电器输出控制电路等构成。时钟芯片电路具有一路定时设置，通过电子时钟的定时来控制继电器的闭合与断开，可以控制家用电器的电源通断。
关键字：单片机；家用电器；定时控制；

Based on the design of the control system of SCM household appliances timer switch

Abstract: The purpose of the research of household appliances timing equipment is to conserve energy and increase its working time through controlling time accurately.Completing automatic control can lead to reduce unnecessary mannual operation and save the employing cost.Besides,it makes jobs which need human beings'controlling time easy and convenient.The main subject of the research is based on the control of SCM household appliances timer switch,which is made up of power supply circuit, real-time clock circuit, display circuit, keyboard input circuit, relay output control circuit, etc.Clock chip circuit possesses of timing Settings,so it can control power hige of household appliances through electrical closed and disconnect achieved by electronic timing settings.
Key words:； microcontroller；household appliances;timer controlled；

目录

绪论

我们在日常生活中，经常碰到一些需要定时的事情，例如：空调可以定时开启与关闭，可以定在任何时间，洗衣机洗涤衣物需要定在几分钟到几十分钟的时间，电风扇需要定在数十分钟的时间。完成这种定时的定时器有多种多样，在家用电器中采用机械定时器就是根据一般上弦钟表原理设计的，这种定时器虽然结构简单，成本低，维修也比较方便，但是它的触头频繁接触和断开，大大的缩减了它的使用寿命，也不利于进一步全自动化。在电子技术突飞猛进的今天，电子定时器一定会逐步取而代之，这是不言而喻的。本课题是通过导师筛选提出的，电子设计课题不一定很大，只要通过亲手做一遍全过程，完成一个产品制作，收获是很大的。

家用电器的定时控制在国内外的研究已经具有了相对成熟的技术支持。功能更加完善，界面也更具人性化。定时器既可作倒计时秒表，又可进行定时，还可通过扩展完成其他功能，并且功能的相对转换也更加简单。现在的定时器主要是以由单片机作为智能芯片来控制，其基本组成主要由一般的控制电路，复位电路，键盘显示电路，实时时钟等。而随着科技的发展，定时器的应用也越来越广泛，各种智能化性能也开始逐步实现，现在的定时器可以实现遥控定时，语音定时，也可以延时定时，循环定时等。本次研究的是以单片机为控制核心的具有一定智能控制功能的定时器，具有操作简单，功能实用的特点。

本课题的意义在于：通过设计和制作本课题，把在学校学习到的知识融会贯通并应用到实际当中。做到学有所成，学有所用。并且希望通过本设计为节能减排做出贡献。电子定时器在家用电器中经常用于延时自动关机、定时。延时自动关机可用于：收音机、电视机、录音机、催眠器、门灯、路灯、汽车头灯、转弯灯以及其他电器的延时断电及延时自停电源等。定时可用于：照相定时曝光、定时闪光、定时放大、定时调速、定时烘箱、冰箱门开定时报警、水位定时报警、延时催眠器、延时电铃、延时电子锁、触摸定时开关等。例如：空调中的定时器，在工作一段时间之后便能自动切断电源停止工作。夏季夜间使用，入睡前先顶好时间，等睡熟后到了预定时间，空调自动关机。方便节能。定时器除了应用于家用电器外，还广泛地用于工业农业生产和服务设施，甚至军事等。

第一章　基于单片机的家用电器开关控制系统总体设计

1.1 系统总体设计

根据设计要求与设计思路，本设计主要由 8 个部分共同组成，其中主要包括一个主控制芯片 STC89C52，主要作用为对其他模块进行控制和管理，还有实时时钟芯片 DS1302，为系统提供精确的实时时钟。按键电路提供输入信号，可进行时间调整和定时器调整。LED 显示电路显示时间等。总体结构框图如下图所示。

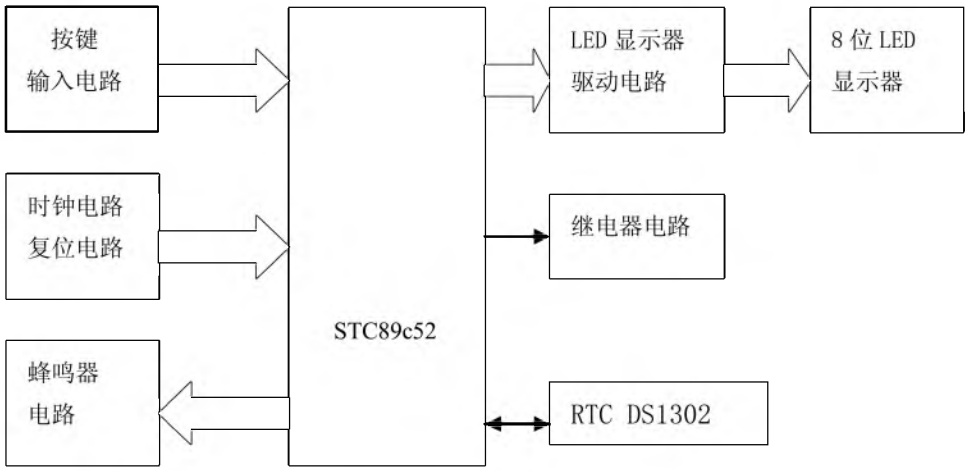


图 **1-1** 系统结构图

1.2 系统设计要求

当系统上电复位时，数码管显示当前的时间，此时间可以由按键进行调整。本系统共配置了六个按键，分别为 SET 键、CHANGE 键、INC 键、DEC 键、START 键和定时器复位键 RESET

系统时间调整：当系统上电复位时，显示系统当前时间。若系统当前时间与实际时间有误，可进行调整。首先，按一次 SET 键，系统进入时间调整状态，时钟停止，可调整分钟数，使分钟数加则按 INC 键，使分钟数减则按 DEC 键。再按一次 SET 键，进入小时调整状态，加则按 INC，减按 DEC。再按一次 SET 键，进入星期调整状态，同理，加则按 INC，减按 DEC。再按一次则退出调整。

系统日期调整：在系统显示时间的情况下，按一次 CHANGE 键，则系统显示当前的日期。若要调整当前的日期，首先，按一次 SET 键，系统时间停止，按 INC 和 DEC 可进行日的调整。再按一次 SET 可进行月的调整。再按一次可进行年的调整。再按一次则退出调整。

定时器功能设置：在系统显示时间的情况下，按两次 CHANGE 键，则进入定时器设置状态。按一次 SET 键，可进行秒钟数的调整，再按一次 SET 键，可进行分钟数的调整。再按一次 SET 键，可进行小时数的调整。调整完后，按 START 键，则进入定时器倒计时状态。当时间为零时，则停止倒计时，此时红灯亮，绿灯灭，继电器吸合/断开。接通/断开外部电路。若要重新启动定时器，可按定时器复位 RESET 键。

第二章 系统硬件设计

2.1 单片机概述

2.1.1 单片机的概念

单片机的全称是单片微型计算机（Single Chip Microcomputer），它是在微型计算机层次内与通用型微机并行发展的一个分支，并以其自生所具有的相对优势而常胜不衰。与突出通用性能的通用型微机不同，单片机的生存之道在于面向过程控制。由于它集成度高，体积小，因此其常被称为微控制器（Microcomputer）。

单片机在结构上最大的特点是将中央处理器（CPU）、程序存储器（ROM、EPROM、EEPROM及 Flash-ROM）、数据存储器（RAM）、输入/输出接口、可编程定时/计数器等功能单元集成在一块芯片上，有的甚至包含 A/D 转换器和 LCD 显示接口等，因而可以构成一个相对完备的计算机系统。设计者根据自己的实际需要进行设计和开发，所以其应用十分方便、灵活且成本也低。

2.1.2 单片机的内部结构

单片机的内部结构是使用单片机硬件资源的基础，也是有效提高编程代码质量的基础。通用单片机在内部由 CPU、片内振荡器、程序存储器、数据存储器、定时/计数器、可编程 I/O 口、串行口及中断系统组成。

CPU

CPU 又称中央处理器，它由运算器、控制器组成。程序中所执行的算术运算、逻辑运算、位运算及移位运算都在运算器内实现，它是一个负责运算处理的器件。控制器则是单片机的调度指挥核心，控制着程序的有效运行。

运算器中有算术逻辑单元 ALU、状态寄存器 PSW 等，在控制器中则有程序计数器 PC、指令寄存器等。

1. 程序计数器

程序计数器是一个 16 位的二进制计数器。其寻址范围为 64KB，执行指令时，其内容的低 8 位由 P0 口输出，高 8 位由 P2 口输出，当上电复位后，其计数值为零，因此大多数程序的起始地址多由 0000H 开始。在执行指令时，先按照程序计数器的地址从存储器中取出所需执行的指令，然后交与控制器处理执行。但完成指令操作后，程序计数器地址自动加 1，一便为下一次取指令做好准备，从而保证程序的有序运行。

2. 指令寄存器

指令寄存器的作用是存放指令代码，并又译码器和定时控制电路一道加以处理来生成相应的控制信号，进而完成指令操作。

3. 状态寄存器

状态寄存器是一个 8 位标志寄存器，用来存放指令执行后的有关状态。七个标志位的含义如表 2-1。

表 **2-1** 标志寄存器

D7	D6	D5	D4	D3	D2	D1	D0
C	AC	F0	RS1	RS0	OV	X	P

进位标志位 C：用于表示加减运算中，累加器 A（也记作 ACC） 的最高位有无进位或借位，执行加法运算时，若最高位有进位，则 C=1，否则 C=0。执行减法运算是，若最高位有借位，则 C=1，否则 C=0。

辅助进位位 AC：用于表示加减运算中有无低 4 位向高 4 位进位或借位。在执行加减运算时。若存在进位或借位，则 C=1，否则 C=0。

用户标志位 F0：该标志位是由用户自行以传递指令设置的，可以此来设置程序的流向。

寄存器选择位 RS1 和 RS2：用于进行工作工作寄存器的选择，通常使用的寄存器为 R0~R7，共 8 个。通过对 RS1 和 RS2 的设置，可将寄存器 R0~R7 分为 4 组，从而达到 32 个寄存器可以存放数据。当然，在同一组态下，只有 8 个寄存器，但可以根据需要在不同的时间段下，通过组态切换来达到扩展的目的。

溢出标志位 OV：用于表示在运算中是否发生溢出，若在运算中累加器 A 的值超出了 8 位数据范围，则 OV=1；否则 OV=0。因此，它是判别运算结果有效性的重要标志。

奇偶运算位 P:用于表示运算的结果中 1 的数量奇偶性，若有奇数个，则 P=1,否则 P=0。

4. 堆栈指针

堆栈指针 SP 是一个 8 位寄存器，它是用来表示堆栈栈顶的指针，能在堆栈操作后，自动实现堆栈指针的加 1 或减 1 处理。

堆栈与队列不同，它遵循的是“先入后出”原则，而队列是“先入先出”。在单片机内堆栈区域的大小是可变的，可在 00H~7FH 内设置。堆栈有栈底和栈顶之分，栈底是相对固定的，当栈底与栈顶地址相同时表示堆栈空，无数据，而两者的差值越大，表示堆栈内的数据越多。

当堆栈入栈操作时，堆栈指针加大；而执行出栈操作时，堆栈指针减小。堆栈的操作只能由入栈指令 PUSH 和出栈指令 POP 来执行，禁止使用由传送指令来存放数据。

5. 数据指针

数据指针 DPTR 是一个 16 位寄存器，是由 DPH 和 DPL 两个 8 位寄存器拼接而成的，因此在使用时，可以单独操作，其中，DPH 为 DPTR 的高 8 位，DPL 位它的低 8 位。

2.1.3 通用寄存器

通用寄存器共有 32 个，被分成 4 组，每组有 8 个被命为 R0~R7，在一个具体的时刻，CPU 只能使用其中的一组，具体的选择方法是使用程序状态寄存器中的 RS0~RS1，通用寄存器的地址映射关见表 2-2。

表 **2-2** 通用寄存器地址映射关系表

RS1	RS0	组别	R0~R7 所占单元地址
0	0	0 组	00H~07H
0	1	1 组	00H~07H
1	0	2 组	00H~07H
1	1	3 组	00H~07H

2.1.4 特殊功能寄存器

特殊功能寄存器用于作为与外围器件的接口，如并行接口、串行接口、定时/计数器及中断系统等。通过特殊功能寄存器可以向片内的外围器件写入数据进行控制，也可从片内的外围器件中读取相应的出来结果。特殊功能寄存器的地址映射关系如表 2-3。

表 **2-3** 特殊功能寄存器地址映射表

F8H									FFH
F0H	B								F7H
E8H									EFH
E0H	ACC								E7H
D8H									DFH
D0H	PSW								D7H
C8H									CFH
C0H									C7H
B8H	IP								BFH
B0H	P3								B7H
A8H	IE								AFH
A0H	P2		AUXR1				WDERST		A7H
98H	SCON	SBUF							9FH
90H	P1								97H
88H	TCON	TMOD	TL0	TL1	TH0	TH1	ZUXR		8FH
80H	P0	SP	DP0L	DP0H	DP1L	DP1H		PCON	87H

通常对于没有定义的存储单元不要使用，否则会产生一些不确定的效果，对此应有足够的重视。

2.1.5 内部存储器

在单片机内部，程序存储空间和数据空间是独立设置的，而这些存储空间又被映射到 4 个区域，即片内程序存储器、片外程序存储器、片内数据存储器及片外数据存储器。

1. 程序存储空间

程序存储空间可映射为片内程序存储器和片外程序存储器，具体被映射到哪一存储器，由单片机的 EA 引脚决定。当 EA 引脚为高电平时，被映射到内程序存储器；当为低电平时，被映射到外程序存储器。具体的内部程序存储空间的大小受单片机型号的影响。

2. 数据存储空间

数据存储空间也可映射为内数据存储器和外数据存储器。但在访问方式上，与程序存储器有一定的差异，它是通过不同的指令来访问数据存储器的。

2.1.6 并行 I/O 端口

并行 I/O 口是单片机的一个重要部分，他承担与外部交换信息的作用。该 I/O 口集输入缓冲、输出锁存及驱动为一体。对于 51 系列的单片机共有 4 个 8 位的并行 I/O 口，分别是 P0、P1、P2 和 P3 口。这些 I/O 口不但有输入输出的作用，有些还具有第二功能，以便于系统的扩展。

4 个并行 I/O 口均为双向的输入输出口，但又有着不同的特点。

（1） P0 口的输出为漏极开路式驱动，需要外接上拉电阻，阻值 5~10KB 间：而 P1、P2、P3 口则有内部的上拉电阻，无需再外接。当使用 P0 口做地址/数据线时，由于其可以直接驱动起 COMS 输入，故也无需再外加提升电阻。

（2） P1 口被称为准双向接口，它在执行输入操作时，若锁存器的数据为 0，则引脚电位始终被钳制在低电平，不能被输入高电平。为此必须在输入操作前，使用输出指令将其置 1，所以在 P1 口进行输入前，必须用指令对端口的电位进行上拉。

（3） P2 口和 P0 口可以联合进行存储器的扩展，此时 P0 口复用地址的低 8 位和数据线，地址信号用 ALE 引脚信号加以锁存，P2 口则作为地址的高 8 位。

（4） P3 口除了用做准双向 I/O 口之外，其还具有第二功能，见表 2-4。

表 **2-4 P3** 口第二功能

P3.0	P3.1	P3.2	P3.3	P3.4	P3.5	P3.6	P3.7
RXD	TXD	INT0	INT1	T0	T1	WR	RD

2.1.7 单片机最小系统

以某一种型号的单片机芯片为核心，配备支持单片机正常工作所需的最少的外部电路，就构成单片机最小应用系统。STC89c52 内部集成有程序存储器，构成最小应用系统时只需配备时钟电路和复位电路即可工作。该系统不需要扩展存储器，芯片的所有 I/O 接口都可以作为输入输出口使用。由于系统只需使用片内程序存储器，所以其 EA 引脚接高电平。如下图 2-5 所示。

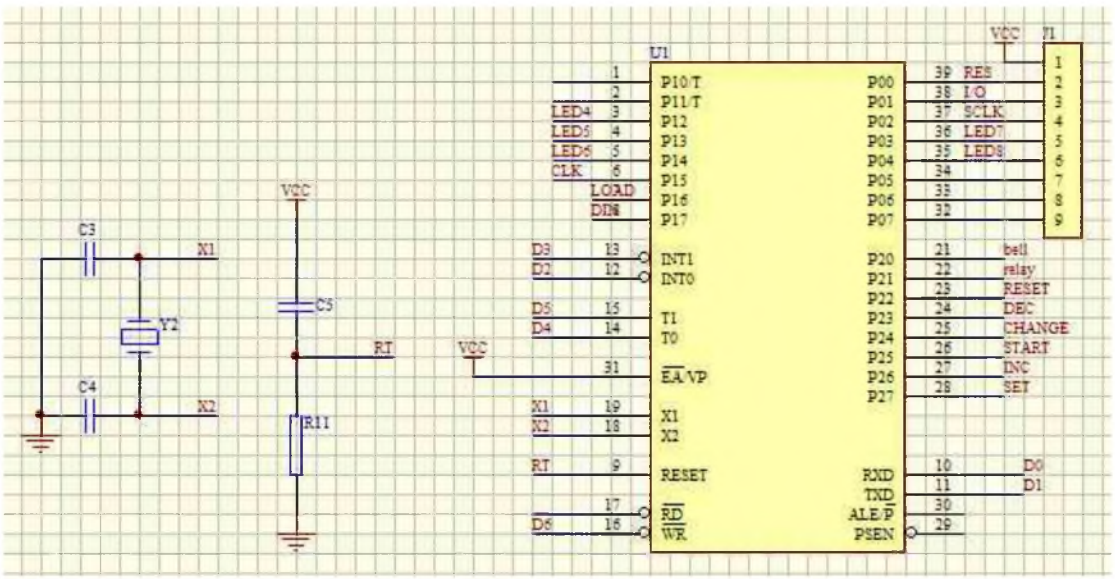


图 **2-1** 单片机最小系统

1. 复位电路

MCS-51 单片机的 RST 引脚上出现持续 24 个震荡周期的高电平信号时，单片机进入复位。本系统采用上电自动复位方式。如图 2-6。

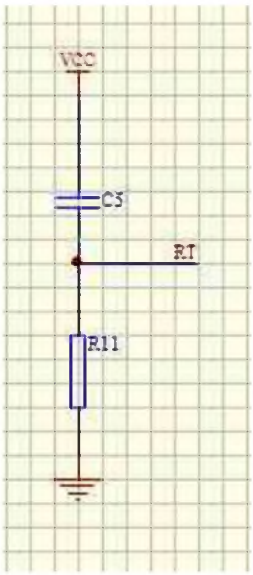


图 **2-2** 复位电路

2. 震荡电路

MCS-51 单片机内部集成了一个用于构成振荡器的高增益反向放大器，引脚 XTAL1 和 XTAL2 分别是这个放大器的输入端和输出端。XTAL1 和 XTAL2 引脚上外接晶体振荡器和陶瓷谐振器及微调电容片内的反向放大器作为反馈元件共同构成一个自激振荡器，其产生的脉冲直接送入内部时钟电路。连接方式如图 2-7 所示。

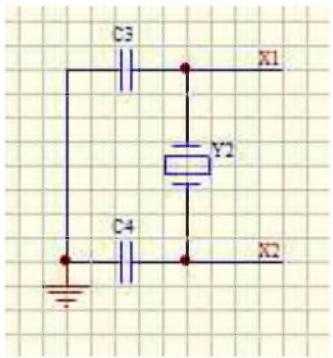


图 **2-3** 振荡电路

2.2 时钟日历芯片 DS1302 介绍

2.2.1 DS1302 概述

DS1302 是 Dallas 公司推出的涓流充电时钟芯片，内容有一个实时时钟/日历和 31 字节静态 RAM，实时时钟/日历电路提供秒、分、时、日、星期、月、年的信息，每月的天数和闰年的天数可自动调整，时钟操作可通过 AM/PM 选择采用 24 或 12 小时格式。DS1302 与单片机之间能简单的采用 SPI 同步串行的方式进行通信，仅需用到 3 跟信号线，分别为 RST（复位）、I/O（数据线）和 SCLK（同步串行时钟）。时钟/RAM 的读/写数据以一个字节或多达 31 个字节的字符组方式通信。DS1302 工作时功耗很低，保持数据和时钟信息时功率小于 1mW。以下是 DS1302 的一些主要特性。

- 实时时钟具有能计算 2100 年之前的秒、分、时、日、星期、月、年的能力，还有闰年调整

的能力。

- 31 字节的静态数据存储器。
- 串行 I/O 接口方式使得管脚数量最少。
- 宽范围工作电压为 2.0~5.5V。
- 低工作电流，在 2.0V 供电时，电流小于 300nA。
- 读/写时钟或 RAM 数据时有两种传递方式，即单字节传送和多字节传送字符组方式。
- 8 引脚 DIP 封装 8 引脚 SOIC 封装。
- 简单 3 线 S P I 同步串行总线接口。
- 与 TTL 兼容（VCC=5V）。
- 可选工业级温度范围为-40C~+85C。

2.2.2 DS1302 引脚功能

DS1302 有 3 种不同的引脚封装形式，分别为双列直插的 DIP8 和表面贴装的 SOIC8（150mil）、SOIC8（200mil），与这 3 种封装相对应的芯片型号分别是 DS1302、DS1302Z 和 DS1302S。这 3 种封装形式的芯片引脚如图 2-8 所示，引脚含义如下。

- X1、X2：32.768kHz 的晶振引脚。
- GND：信号地。
- RST：复位引脚。
- I/O：数据输入/输出引脚。
- SCLK：同步串行时钟输入引脚。
- VCC2：主电源电压输入引脚。
- VCC1：电池备份电源电压输入引脚。

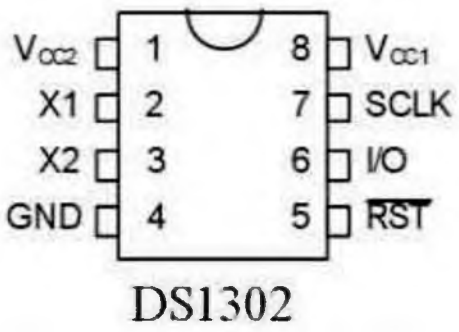


图 2-4 DS1302

2.2.3 DS1302 内部结构及工作原理

DS1302 有电源、移位寄存器、命令控制逻辑、振荡器、实时时钟及 RAM 组成，其内部结构如图 2-9 所示

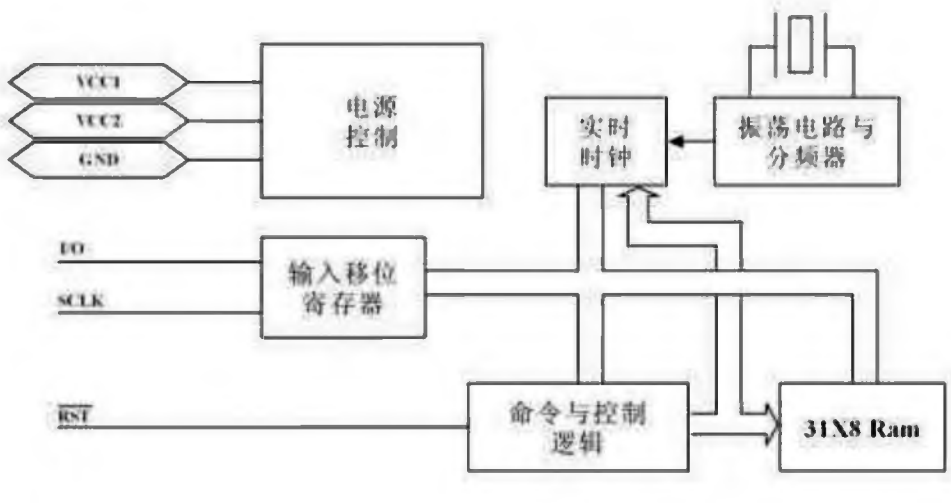


图 **2-5　DS1302** 内部结构框图

DS1302 的电源包括主电源和备用电源两部分。VCC1 在单电源与电池供电的系统中提供备份电源；VCC2 在双电源系统中提供主电源，在这种运用方式中，VCC1 连接到备份电源，以便在没有主电源的情况下能保存时间信息及数据。当 VCC2>VCC1+0.2V 时，VCC2 给 DS1302 供电；当 VCC2<VCC1 时，DS1302 由 VCC1 供电。

DS1302 的振荡器/分频器部分使用 32.768kHz 的石英晶体，32.768kHz 的晶振可以直接通过 X1、X2 两个引脚连接到 DS1302。

输入移位寄存器模块用于完成串行数据的输入/输出，数据在同步串行时钟 SCLK 的上升沿串行输入，无论是读周期还是写周期，也无论是单字节传送还是多字节传送方式，数据传送的开始的 8 位数据位用于指定 DS1302 中哪个字节将被访问。在开始的 8 个时钟周期将命令字节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。

输入移位寄存器模块的所有串行数据输入输出都必须通过将 RST 驱动至高电平状态来启动。复位引脚 RST 输入与控制逻辑相连，用于允许或禁止地址/命令序列送入移位寄存器。

同时，RST 还提供了终止单字节和多字节数据传送的手段，如果 RST 输入的为低电平，那么所有的数据传送将被终止并且 I/O 引脚变为高阻态，在 DS1302 的上电过程中，当 VCC>2.5V 之前，RST 必须为逻辑 0。此外，当把 RST 驱动至逻辑 1 状态时，SCLK 必须为逻辑 0，

在 DS1302 内部共包括 40 个数据字节，其中有 7 个时钟/日历字节、1 个时钟写保护控制字节、1 个涓流充电控制字节和 31 个 RAM 字节。在对这些地址字节进行读写访问时，由命令字节中的数据位控制。

2.2.4　DS1302 接口读写操作

1.DS1302 数据读写时序

在对 DS1302 进行数据读写操作时，所有的读写操作必须由命令字节来初始化。命令字节的数据格式如下图所示：

表 **2-5 DS1302** 命令字节的数据格式

D7	D6	D5	D4	D3	D2	D1	D0
1	RAM/CK	A4	A3	A2	A1	A0	RD/W

命令字节的 D7 必须为逻辑 1，如果它为 0，将禁止 DS1302；D6 为逻辑 0 时指定访问时钟/日历数据，为逻辑 1 时指定访问 RAM 数据；D5~D1 指定进行输入或输出的特定寄存器单元地址；D0 为逻辑 0 时指定进行写操作，为逻辑 1 时指定进行读操作。

外部处理器向 DS1302 写数据时，在写命令字节的 8 个 SCLK 周期的上升沿输入数据字节，如果有更多的 SCLK 周期，它们将会被忽略。外部处理器从 DS1302 读数据时，跟随在读命令字节 8 个 SCLK 周期之后，DS1302 会在下 8 个 SCLK 周期的下降沿输出数据。需要注意的是，DS1302 输出的第一个数据位发生在命令字节最后一位后的第一个下降沿处，而且在读操作过程中要保持 RST 为高电平状态，如果有额外的 SCLK 时钟周期，DS1302 将重新发送数据字节，这一操作特性使得 DS1302 具有多字节连续读取能力。如下图 2-10 对 DS1302 单字节数据读写读写时序进行了描述。

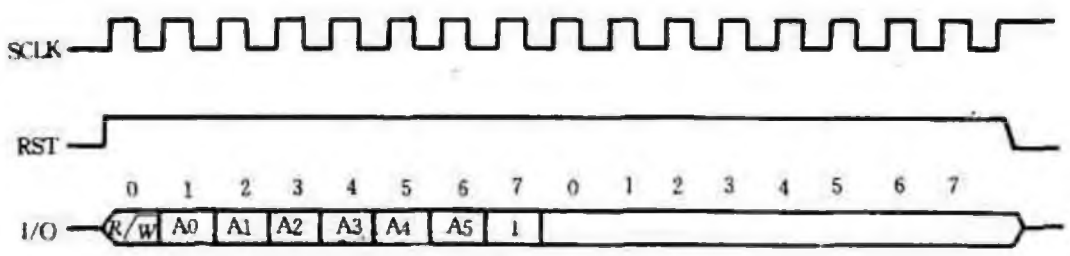


图 **2-6** 单字节数据传输时序

除了采用单字节方式进行数据处理外，也可以采用突发方式进行多字节连续读写。通过对地址 31 寻址（命令位的 D1~D5 均为逻辑 1），可以把时钟/日历或 RAM 寄存器规定为多字节方式。在读字节方式写时钟寄存器时，读或写从地址 0 的位 0 开始，与使用 DS1302 时一样，当以多字节方式写时钟寄存器时，必须按照数据传送的次序写最先的 8 个寄存器。但是以多字节方式写 RAM 时，不必写所有的 31 个字节，不敢是否写了全部的 31 个字节，所写的每一个字节都会被传送到 RAM。

3. DS1302 内部寄存器读写

通过单字节读写方式或突发字节连续读写方式，可以对 DS1302 内部的时钟/日历、制寄存器和 RAM 存储器进行访问。DS1302 的时钟/日历、控制寄存器共 10 个单元（包括一个时钟突发访问寄存器），如下表 2-11 所示的命令字节的数据格式，可以得到外部处理器在访问这些寄存器单元时应采用的命令格式。

表 **2-6** 时钟/日历、控制寄存器读写命令字节的数据格式

寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
秒	1	0	0	0	0	0	0	RD/W
分	1	0	0	0	0	0	1	RD/W
时	1	0	0	0	0	1	0	RD/W
日	1	0	0	0	0	1	1	RD/W
月	1	0	0	0	1	0	0	RD/W
星期	1	0	0	0	1	0	1	RD/W
年	1	0	0	0	1	1	0	RD/W
写保护控制	1	0	0	0	1	1	1	RD/W

涓流充电控制	1	0	0	1	0	0	0	RD/W
时间突发访问	1	0	1	1	1	1	1	RD/W

时钟/日历和控制寄存器单元的数据以 BCD 码格式存放，如下表 2-12 所示

表 **2-7 DS1302** 内部时钟/日历和控制寄存器内容

寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
秒	CH	10SEC			SEC			
分	0	10MIN			MIN			
时	12/24	0	10	HR	HR			
			A/P					
日	0	0	0	10DATE	DATE			
月	0	0	0	10M	MONTH			
星期	0	0	0	0	0	DAY		
年	10YEAR				YEAR			
写保护控制	WP	0	0	0	0	0	0	0
涓流充电控制	TCS	TCS	TCS	TCS	DS	DS	RS	RS

秒寄存器的 D7 定义为时钟暂停位，当此位设置为逻辑 1 时，时钟振荡停止，DS1302 进入低功耗待机模式，其消耗电流小于 100nA；当此为设置为逻辑 0 时，时钟将被启动。

小时寄存器的 D7 定义为 12 或 24 小时方式选择位，当它为高电平时，选择 12 小时方式；当它设置为低电平时表示选择 24 小时方式。在 12 小时方式下，小时寄存器的 D5 用于 AM/PM 指示，D5 为逻辑高电平时表示 PM；在 24 小时方式下，D5 则是第二个小时位。

写保护寄存器的 D7 是写保护位，其余低 7 位置为 0。在对时钟或内部 RAM 单元进行写操作前 D7 必须为 0.。当 D7 处于高电平状态时，写保护位防止对任何其他寄存器进行写操作。

涓流充电寄存器用于控制 DS1302 的涓流充电特性。其中，涓流充电特性选择（TCS）位 D7~D4 处于 1010 模式时，才能时涓流充电器工作，其他所有模式都将禁止涓流充电器，在 DS1302 上电时涓流充电器被禁止。涓流充电二极管（DS）位 D3~D2 用于选择连接在 VCC2 与 VCC1 之间的二极管数目，当 DS 为 01 时选择一个二极管，当 DS 为 10 时选择两个二极管，DS 为 00 和 11 时涓流充电器被禁止，与 TCS 无关。涓流充电电阻（RS）为 D1~D0 用于选择连接在 VCC2 和 VCC1 之间的电阻，RS 为 00 时无电阻连接，RS 为 01 时电阻值为 2K，RS 为 10 时电阻值为 4K，RS 为 11 时电阻值为 8K，如果 RS 设定为无电阻连接的 00，那么涓流充电器将被禁止，与 TCS 无关。

时钟/日历和控制寄存器的最后一个单元是时钟/日历突发多字节控制方式控制字。当时钟/日历命令字节在多字节方式下工作时，最先的 8 个时钟/日历寄存器可以从地址为 0 的第 0 位开始连续的读或写。在制定突发方式连续写时钟/日历寄存器时，如果写保护设置为高电平，那么没有数据会传送到 8 个时钟/日历寄存器中的任何一个。此外，在突发多字节连续读写方式下，涓流充电控制器是不可访问的。

DS1302 除了包含 9 个字节的时钟/日历和控制寄存器单元外，还包括 31 个字节的 RAM 存储单元。对 RAM 存储单元的访问，既可以采用单字节读写的方式，也可以采用突发多字节连续写。在多字节工作方式下，可以从地址 0 的第 0 位开始顺序读或写 31 个 RAM 寄存器。如下表 2-13 对访问内部 RAM 单元的命令格式进行了描述。

表 **2-8 DS1302** 内部 **RAM** 单元读写命令地址

RAM 单	D7	D6	D5	D4	D3	D2	D1	D0

元名称								
RAM0	1	1	0	0	0	0	0	RD/W
RAM1	1	1	0	0	0	0	0	RD/W
…	…	…	…	…	…	…	…	…
RAM30	1	1	1	1	1	1	0	RD/W
RAM 突发访问	1	1	1	1	1	1	1	RD/W

4. DS1302 的应用

DS1302 与 MCS—51 单片机的接口电路

DS1302 实时时钟/日历芯片与 MCS-51 系列单片机的连接十分简单，只需要利用单片机的 3 个 I/O 引脚来对 DS1302 的 SCLK、I/O 和认识他进行控制即可。如图 2-14 所示本设计中的接口电路，DS1302 的主电源输入端接系统电源 VCC，其设备电源输入端接 3V 可充电电池用于掉电保护。

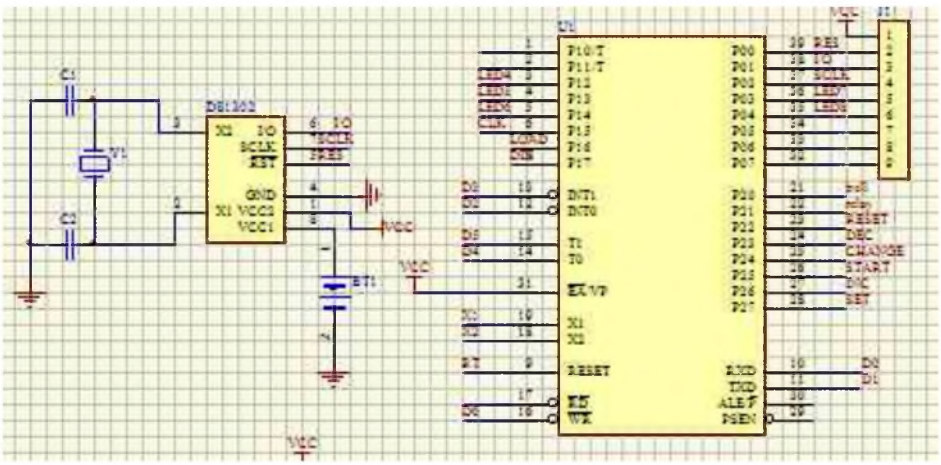


图 **2-7 DS1302** 与单片机的连接

2.3 MAX7219 串行芯片介绍

2.3.1 MAX7219 概述

在单片机应用系统中，通常需要用 LED 数码管对系统的状态进行观察。一般情况下，LED 数码管的显示方式有静态显示和动态显示两种。不管是静态显示还是动态显示，单片机都工作在并口 I/O 口状态或存储器方式下，需要占用比较多的 I/O 口线。如果采用 MAX7219 作为 LED 数码管的接口电路，则只需要占用单片机的三根线就可串行实现 8 位 LED 的显示驱动和控制。

MAX7219 是美国 MAXIN 公司生产的串行输入/输出共阴极显示驱动器。采用三线制串行接口技术进行数据的传送，可直接与单片机连接，用户能方便的修改内部参数实现多位 LED 数码管的显示。MAX7219 片内含有硬件动态扫描显示控制，每块芯片可驱动 8 个 LED 数码管。

2.3.2 MAX7219 的外形封装及引脚功能

MAX7219 是 7 段共阴极 LED 数码管的驱动器，采用 24 引脚的 DIP 和 SO 两种封装形式，其外形封装如下图所示

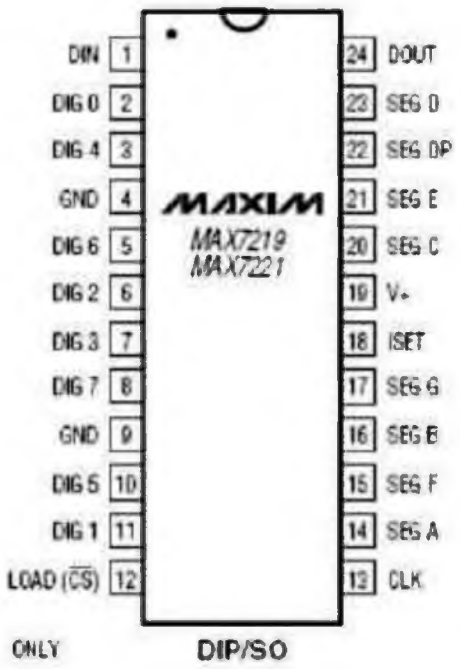


图 **2-8 MAX7219** 的外形封装

MAX7219 各引脚的功能如下。

- Din：串行数据输入端。在 CLK 的上升沿，数据被锁入 16 位内部移位寄存器中。
- DIG0~DIG7：8 位数码管驱动线，输出位选择信号，从数码管的共阴极吸收电流。
- GND：地线。
- LOAD：装载数据控制端。在 LOAD 的上升沿，最后送入的 16 位串行数据被锁存到移位寄存器中。
- CLK：串行数据输入端。在 LOAD 的上升沿，数据被送入内部移位寄存器；在 CLK 的下降沿，数据由 Dout 端输出。
- SEGa~SEGg：LED7 段数码管段位驱动端，用于驱动当前 LED 段码。
- SEG dp：小数点驱动端。
- ISET：LED 段峰值电流设置端，ＩＳＥＴ端通过一只电阻与电源Ｖ＋相连，调节电阻值，可改变ＬＥＤ段提供的峰值电流。
- Ｖ＋：＋５Ｖ电源。
- Dout：串行数据输出端。进入 Din 的数据在 16.5 个时钟脉冲后送到 Dout 端，Dout 在级联时传送到下一片 MAX7219d Din 端。

2.3.3 MAX7219 的内部结构

MAX7219 的内部结构如下图 2-16 所示，MAX7219 主要由段驱动器、段电流基准、二进制 ROM、数位驱动器、5 个控制寄存器、16 位移位寄存器、8×8 双端口 SRAM、地址寄存器和译码器、亮度脉宽调制器、多路扫描电路等部分组成。

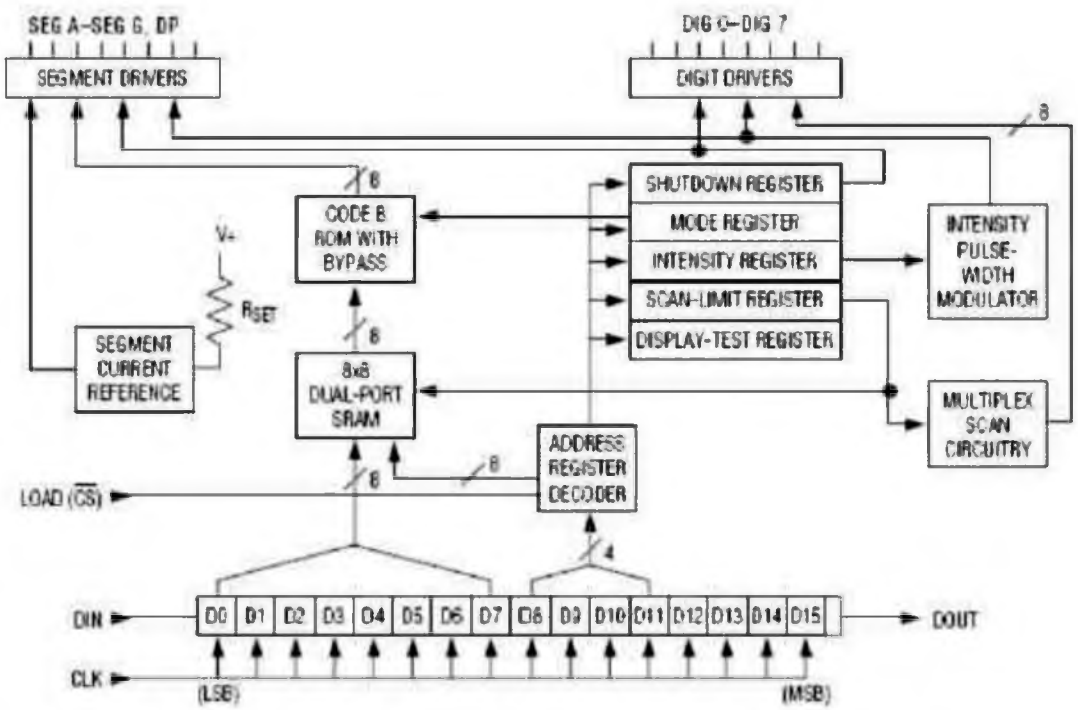


图 **2-9 MAX7219** 的内部结构

数位驱动器用于选择某位 LED 显示。串行数据以 16 位数据包的形式从 Din 引脚输入，在 CLK 的每个上升沿时，不管 LOAD 引脚的工作状态如何，数据一位一位的串行送入片内 16 位移位寄存器中。在第 16 个 CLK 上升沿出现的同时或之后，在下一个 CLK 上升沿出现之前，LOAD 必须变成高电平，否则移入移位寄存器的数据将会被丢失。16 位数据包格式如下。

表 **2-9 16** 位数据包格式

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
×	×	×	×	地址				MSB			数	据			LSB

从表中可以看出，D15~D12 为无关位，取任意值，通常全为“1”，D11~D8 为四位地址，D7~D0 为 5 个控制寄存的命令字或 8 位 LED 待显示的数据位。在 8 位数据中，D7 为最高位，D0 为最低位。一般情况下，程序先送控制命令，再将数据送到显示寄存器，但必须每 16 位为一组，从最高位开始送入数据，一直送到最低位为止。

通过对 D11~D8 中 4 位地址译码，可寻址 14 个内部寄存器，即 8 个数位寄存器、5 个控制寄存器及 1 个空操作寄存器。14 个内部寄存器地址如下。空操作寄存器主要用于多个 MAX7219 级联，允许数据通过而不对当前 MAX7219 产生影响。

表 2-10　14 个内部寄存器地址

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

5 个控制寄存器是译码模式寄存器、亮度调节寄存器、管段模式寄存器、显示测试寄存器。在使用 MAX7219 时，首先必须对 5 个控制寄存器进行初始化。5 个控制寄存器的设置含义如下。

译码模式寄存器（地址：×9）：决定数位驱动器的译码方式，共有 4 种译码模式。每一位对应一个数位。其中，“1”代表 B 码方式，“0”表示不译码方式。驱动 LED 数码管时，应将数位驱动器设置为 B 码方式。一般情况下，应将数据位设置为全“0”，即选择“全非议码方式”。在此方式下，8 位数据位分别对应 7 个段和小数点。

亮度调节寄存器（地址：×A）：用于 LED 数码管显示亮度强弱的设置。利用其 D3~D0 位控制内部亮度脉宽调制器 DAC 的占空比来控制 LED 段电流的平均值，实现 LED 的亮度控制。D3~D0 的取值范围为 0000~1111，对应电流的占空比则从 1/32、3/32 变化到 31/32，共 16 级，D3~D0 的值越大，LED 显示越亮。亮度控制寄存器中的其他各位未使用，可置任意值。

扫描限制寄存器（地址：×B）：用于设置显示数码管的个数（1~8）。该寄存器的 D2~ D0（第三位）指定要扫描的位数，D7~D3 无关，支持 0~7 位，个数位均已 1.3KHz 的扫描频率被分路驱动。当 D2~ D0=111 时，可接 8 个数码管。

关断模式寄存器（地址：×C）：用于关断所有显示器，有两种显示模式，D0=0 时，关断所有显示器，但不会消除各寄存器中保持的数据；D0=1 时，正常工作状态。剩下各位未使用，可取任意值。通常情况下，选择正常操作状态。

显示测试寄存器（地址：×F）：用于检测外界 LED 数码管是工作在测试状态还是正常操作状态。D0=0，LED 处于正常工作状态；D0=1 时，LED 处于显示测试状态，所有 8 个 LED 全亮，电流占空比为 31/32，D7~D1 位未使用，，可取任意值。一般情况下，选择正常工作状态。

2.3.4　MAX7219 工作时序

MAX7219 的工作时序如下，在 CLK 的每个上升沿，都有一位数据从 Din 端输入，加载到 16 位移位寄存器中，在 LOAD 的上升沿，输入的 16 位串行数据被锁存到数位或控制寄存器中，LOAD

必须在第 16 个 CLK 上升沿出现时或在下一个 CLK 上升沿出现之前，变为高电平；否则，移入移位寄存器的数据将丢失。

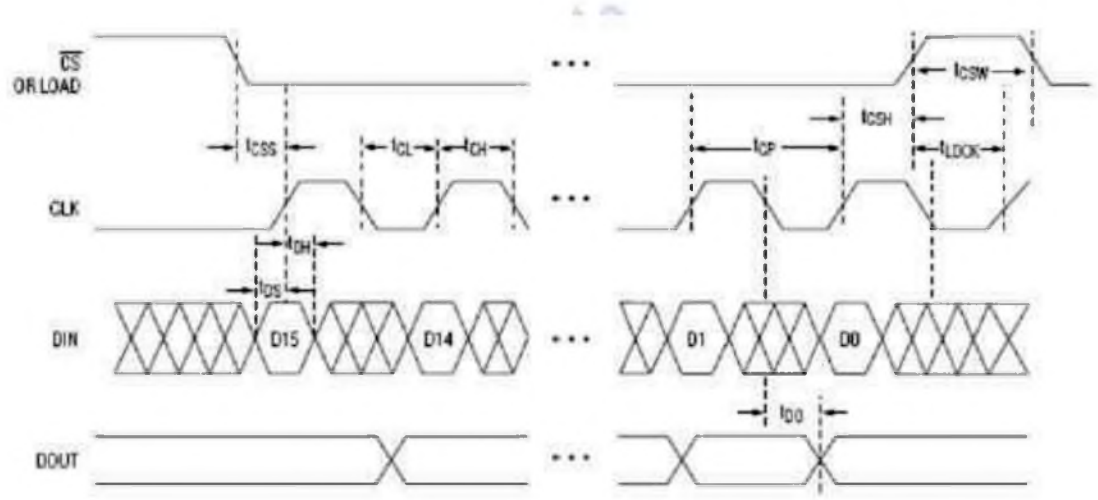


图 **2-10 MAX7219** 的工作时序

2.4 数码管显示接口设计

在众多的显示器件中，LED 数码管以其价格低廉而被广泛使用。LED 数码管是由发光二极管构成的，按照制造材料的不同，可以发出红、绿、黄等颜色的光。按照封装形式的不同，LED 数码管可分为七段式、米字型数码管。发光二极管的极性连接方式可分为共阴极数码管和共阳极数码管。而按封装数码管位数的不同，又有一位式数码管和多位集联式数码管之分。当然也有一些特制的，如以汉字形式出现的 LED 显示器件。

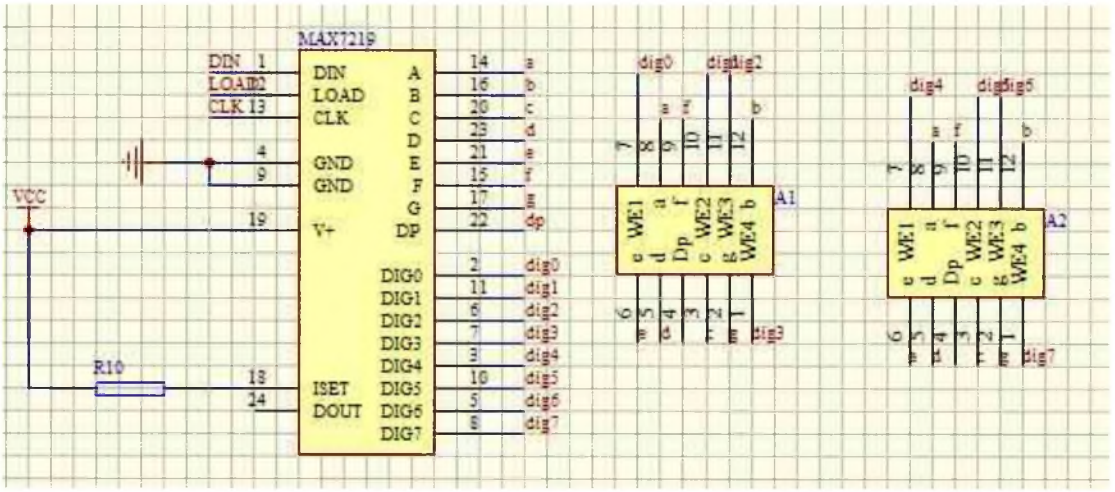
LED 数码管的共同特性是，它属于自身发光性器件，适用于光线较暗的场所，在光线较强的场所使用它，往往会造成难于识读的问题，需要适当添加遮挡层，以加强反差对比。

能耗大、显示信息量小是 LED 数码管的不足之处，故它不适宜作为便携式仪器或装置的显示器件。

2.4.1 共阴极数码管设计

共阴极数码管是指在数码管内部，个发光二极管的阴极是连接在一起的，这样有利于使用，也便于减少封装的引脚数。

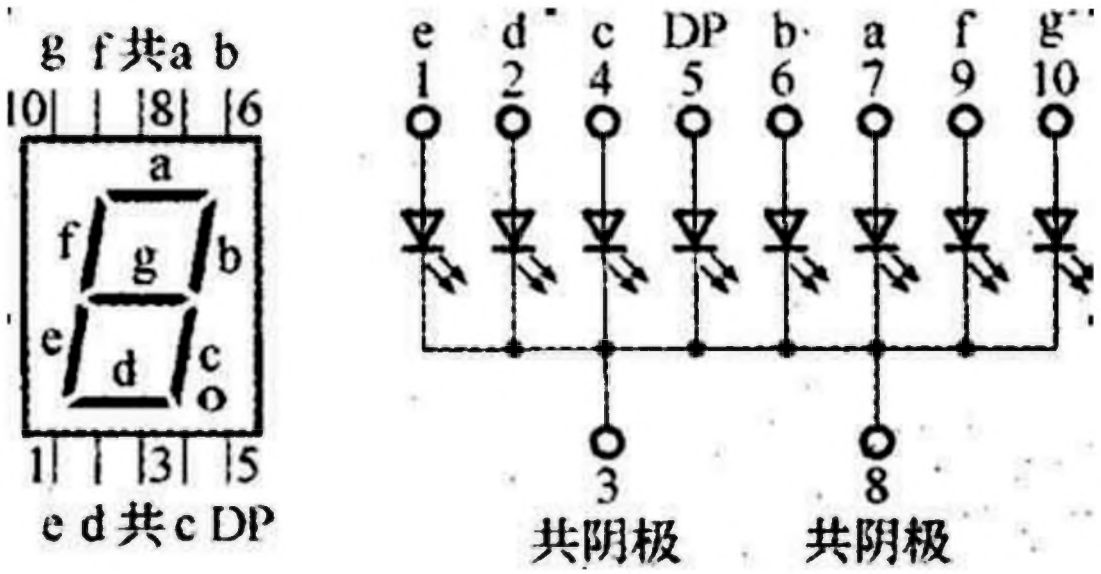
由于点亮发光二极管一般需要几毫安的电流，故在使用 LED 数码管时要配相应的驱动器。在本设计系统中，使用 MAX7219 作为驱动电路。如下图所示



2-11 数码管显示电路

2.4.2 数码管显示原理

通常使用的 7 段数码管，它由 7 个发光二极管组成。这 7 个发光二极管 a~g 呈“日”字排列，其结构及连接如下图所示。



2-12 数码管内部结构

2.4.3 LED 数码管的接法

所有发光二极管连接在一起，这种接法称为是共阴极接法。共阴极的 LED 为低电平时，对应的段码被点亮。一般共阴极可以不外接电阻。

LED 数码管的发光二极管亮暗组合实质是不同电平组合，也就是为 LED 数码管提供不同的代

码，这些代码为字型码。字形代码与这 8 段 LED 的关系如下。

表 **2-11** 字形代码与这 **8** 段 **LED** 的关系

数据字	D7	D6	D5	D4	D3	D2	D1	D0
LED 段	Dp	g	f	e	d	c	b	a

字形代码与十六进制的对应关系如下表：

表 **2-12** 字形代码与十六进制数的对应关系

字符	dp	g	f	e	d	c	b	a	段码
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH
6	0	1	1	1	1	1	0	1	7DH
7	0	0	0	0	0	1	1	1	07H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	0	1	1	1	1	6FH
A	0	1	1	1	0	1	1	1	77H
B	0	1	1	1	1	1	0	0	7CH
C	0	0	1	1	1	0	0	1	39H
D	0	1	0	1	1	1	1	0	5EH
E	0	1	1	1	1	0	0	1	79H
F	0	1	1	1	0	0	0	1	71H

2.4.4 LED 数码管的显示方式

在单片机应用系统中一般需要使用多个 LED 数码管，多个 LED 数码管是由 n 根位选线和 8×n 根段选线连接在一起，根据显示方式不同，位选线与段选线的连接方式也不相同。段选线控制字符选择，位选线控制显示位的亮或暗。

LED 数码管有静态显示和动态显示两种方式。

静态显示就是当 LED 数码管要显示某一个字符时，相应的发光二极管恒定的导通或截止。例如，LED 数码管要显示“0”时，a、b 、c、d、e、f 导通，g、dp 截止。单片机只需将所要显示的数据送出去，直到下一次显示数据需要更新时再传送一次数据，显示数据稳定，占用 CPU 时间少。但是，采用这种显示方式，因为每一位都需要一个 8 位输出口控制，所以占用硬件多，如果单片机系统中有 n 个 LED 数码管，则需 8×n 根 I/O 口线，所占用的 I/O 资源多，需进行扩展。

动态显示就是一位一位的轮流点亮各位数码管，对于每一位 LED 数码管来说，每隔一段时间点亮一次，即 CPU 需要时刻对数码管进行刷新，显示数据有闪烁感，占用 CPU 的时间较长。并且，数码管的点亮既与点亮时的导通电流有关，也与点亮、间隔时间的比例有关。调整电流和时间的参数，

可实现亮度较高、较稳定的显示。但是，若数码管的位数不大于 8 位时，只需两个 8 位 I/O 口。

2.5 蜂鸣器设计

2.5.1 蜂鸣器概述

蜂鸣器属于发声类器件，由于其结构简单，功耗低而广为使用，按信号源来区分，可分为有源蜂鸣器和无源蜂鸣器两类。有源蜂鸣器的频率是固定的，一般用于提示或警告；无源蜂鸣器由外部信号来控制其频率，可以发出不同的音调，以模拟某种声响或演奏乐曲。

2.5.2 蜂鸣器设计

蜂鸣器发声原理是电流通过电磁线圈，使电磁线圈产生磁场来驱动振动膜发声的，因此需要一定的电流才能驱动，单片机 I/O 引脚输出的电流较小，单片机输出的 TTL 电平基本上驱动不了蜂鸣器，因此需要增加一个电流放大的电路。如下图所示。

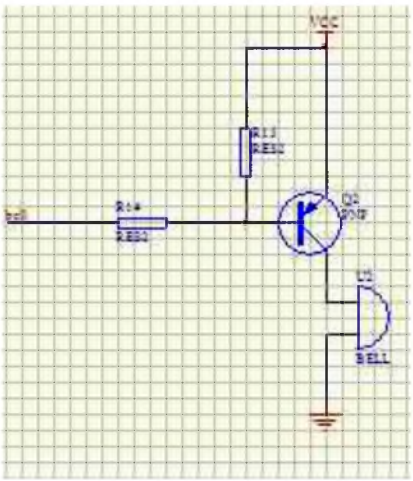


图 **2-13** 蜂鸣器电路

蜂鸣器的正极接到三极管集电极 C，蜂鸣器的负极接到地，三极管的基级 B 经过限流电阻后由单片机的 P2.0 引脚控制，当 P2.0 输出高电平时，三极管截止，没有电流流过线圈，蜂鸣器不发声；当 P2.0 输出低电平时，三极管导通，这样蜂鸣器的电流形成回路，发出声音。因此，我们可以通过程序控制 P2.0 脚的电平来使蜂鸣器发出声音和关闭。

2.6 继电器设计

2.6.1 继电器概述

继电器是一种电子控制器件，它具有控制系统（又称输入回路）和被控制系统（又称输出回路），通常应用于自动控制电路中，它实际上是用较小的电流去控制较大电流的一种“自动开关”。故在电路中起着自动调节、安全保护、转换电路等作用。

2.6.2 继电器设计

电磁式继电器一般由铁芯、线圈、衔铁、触点簧片等组成的。只要在线圈两端加上一定的电压，线圈中就会流过一定的电流，从而产生电磁效应，衔铁就会在电磁力吸引的作用下克服返回弹簧的拉力吸向铁芯，从而带动衔铁的动触点与静触点（常开触点）吸合。当线圈断电后，电磁的吸力也随之消失，衔铁就会在弹簧的反作用力返回原来的位置，使动触点与原来的静触点（常闭触点）释放。这样吸合、释放，从而达到了在电路中的导通、切断的目的。对于继电器的“常开、常闭”触点，可以这样来区分：继电器线圈未通电时处于断开状态的静触点，称为“常开触点”；处于接通状态的静触点称为“常闭触点”。本设计原理图如下。

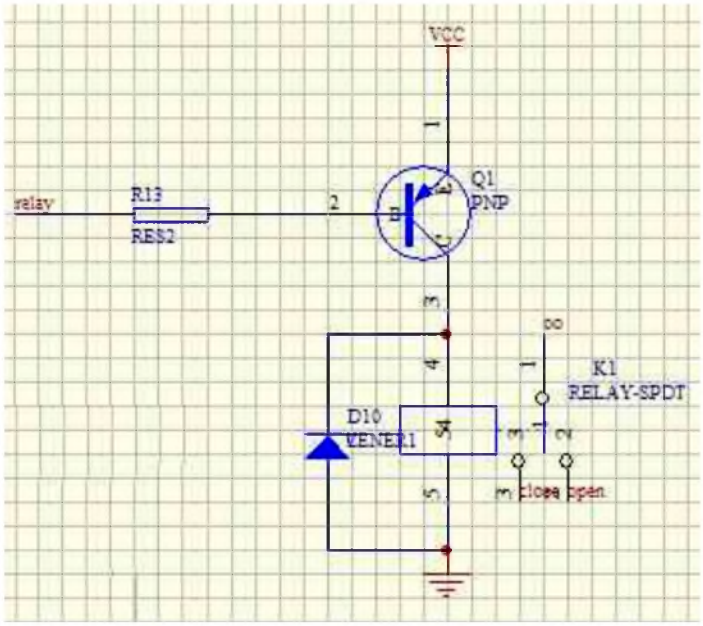


图 **2-14** 继电器电路

第三章　软件系统设计

3.1 主程序软件设计

单片机上电复位时，首先进行初始化，包括单片机的初始化，定时器初始化，DS1302 初始化，MAX7219 初始化，其次进入一个循环系统，在循环系统中，检测 flag3 是否等于 0，若等于 0，则调用时间显示程序。若不等于 0，则检测 flag3 是否等于 1，若等于 1，则调用日期显示程序。若 flag3 等于 2，则调用定时器显示程序。最后将要显示的数字通过 MAX7219 输送到 8 位数码管上显示。主程序的流程图如下所示

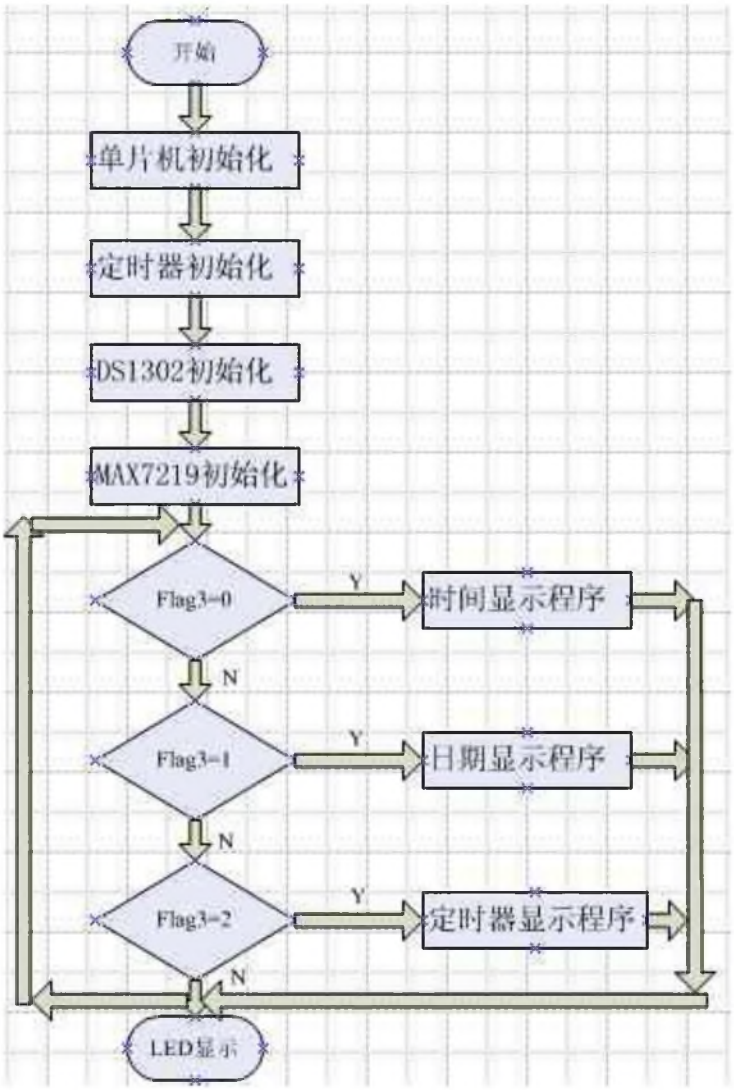


图 **3-1** 主程序流程图

3.1.1 主程序

```
void main(void)
{

    Init_t0();                    /*定时器初始化*/
    count1=1;
//  P3=0xff;
    DisableWP();
    WriteSec(0x00);                /*DS1302 初始化*/
    WriteMin(0x00);
    WriteHr(0x00);
    WriteDay(0x01);
    WriteMn(0x01);
    WriteYs(0x11);
    WriteWe(0x07);
    ready();
    while(1)
   {
    if(flag3==0)                   //时间显示
    {
      red=0;
      blue=1;
      green=1;
      readtime();
      scantime();
      displaytime();
    }
    else if(flag3==1)              //日期显示
    {
      red=1;
      blue=0;
      green=1;
      readdate();
      scandate();
      displaydate();
    }

    else if(flag3==2)              //定时器显示
    {
      red=1;
      blue=1;
      green=0;
      timerscan();
```

```
    timerread();
    timerdisplay();
}

if(CHANGE==0)                          //切换键
{
      flag3++;
      if(flag3>=3)
      flag3=0;
      while(CHANGE==0);
      flag2=0;
}

if(flag1==1)
      {
            if(t==20)
            {
              t=0;
              count1--;
              if(count1==-1)
              {
                    count1=59;
                    count2--;
                    if(count2==-1)
                    {
                          count2=59;
                          count3--;

                          if(count3==-1)
                          count3=23;
                    }
              }
            }
       }
      if(count1==0&count2==0&count3==0)
      {
            TR0=0;
            flag1=0;
            end=0;              //定时时间到服务程序
            bell=0;

      }
      else
      {
```

```
        end=1;
        bell=1;

    }


     if(flag1==1)              //绿灯亮、灭
        start=0;
     else
        start=1;

     if(reset==0)              //复位键
    {
        flag5=1;
        while(reset==1);
    }
    if(flag5==1)
    {
        end=1;
        bell=1;
        count1=1;
        flag5=0;
    }


  }
}
```

3.2 DS1302 读写程序设计

3.2.1 DS1302 读写控制程序设计

通过单片机对 DS1302 进行读写访问，只需要严格遵循 DS1302 数据读写时序即可。在程序编写过程中，值得注意的是，在写时钟/日历寄存器单元之前，应打开 DS1302 写保护，并需要在 DS1302 的初始化过程中正确设定涓流充电寄存器的内容。

在从 DS1302 中读取数据时，从 DS1302 输出的第一个数据位发生在 8 位命令字节最后一位之后的第一个下降沿处，也就是说在第 8 个 SCLK 时钟脉冲的下降沿处，第 1 个数据位即开始输出。所有，Get1302 子函数在读取数据的循环体中，仅循环了 7 次，但由于 SCLK 在循环体外是高电平，而进入循环后第一条 SCLK 置零指令会产生第 1 个下降沿，这个下降沿并不是由循环产生的，所以尽管只有 7 次循环，但实际上是读出了 8 位数据。

3.2.2 如下所示为本设计 **DS1302** 的读写程序。

```
      void InputByte(uchar dd)      //写一个字节到 1302
{
    uchar i;
      ACC=dd;
      for(i=8;i>0;i--)
        {
          dat=A0;
          clk=1;
          clk=0;
          ACC=ACC>>1;
        }
}


void OutputByte(void)       //从 1302 读一个字节的数据
{
    uchar i;
    dat=1;
    for(i=8;i>0;i--)
    {
        ACC=ACC>>1;
        A7=dat;
        clk=1;
        clk=0;
    }
    dd=ACC;
}


void Write(uchar addr,uchar num)   //向 1302 写地址、数据
{
    rst=0;
    clk=0;
    rst=1;
    InputByte(addr);
    InputByte(num);
    clk=1;
    rst=0;
}
void Read(uchar addr)   //从 1302 中读数据
{
    rst=0;
```

```
    clk=0;
    rst=1;
    InputByte(addr);
    OutputByte();
    clk=1;
    rst=0;
}
```

3.3 MAX7219 编程设计

3.3.1 设计分析

无论是 MAX7219 的初始化，还是 8 个 7 段数码管的显示，均必须对数据进行写入。16 位数据包分成 2 个 8 位的字节进行传送，第一个字节是地址，第二个字节是数据。在 16 位数据包中，D15~D12 可以任意写，在此均置为“1”；D11~D8 决定所选通的内部寄存器地址；D7~D0 为待显示数据，8 个 LED 数码管的显示内容在 TABLE 中。MAX7219 与单片机的连接显示程序流程图如下

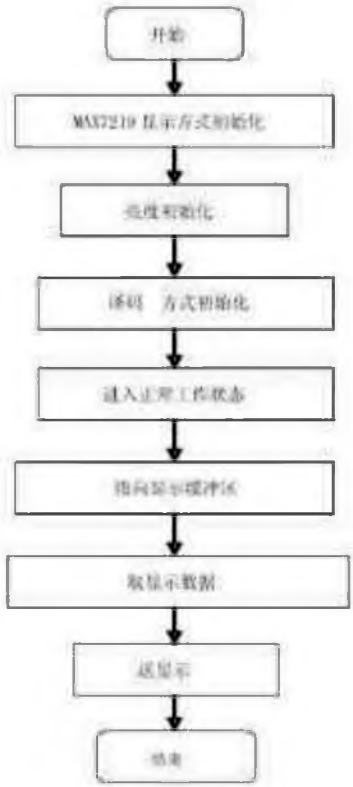


图 **3-2 MAX7219** 程序流程图

3.3.2 本设计中 **MAX7219** 控制程序如下。

```
void send(void)//  向 7219 送一个字节
{
    uchar co;
    bitmsb=x;
    for(co=0;co<8;co++)
    {
        if(m7)
        {
            clkk=0;
            date=1;
            _nop_();
            clkk=1;
        }
        else
        {
            clkk=0;
            date=0;
            _nop_();
            clkk=1;
        }
        bitmsb<<=1;
    }
}

void wr(void)          //向 7219 发控制字和数据
{
    load=0;
    send();
    x=y;
    send();
    load=1;
}

void ready(void)     //7219 初始化
{
    x=0xfb;
    y=0x07;
    wr();
    x=0xf9;
    y=0x00;
    wr();
    x=0xfa;
```

```
    y=0x0c;
    wr();
    x=0xfc;
    y=0x01;
    wr();
}
```