摘 要

单片机是一种控制类型的微处理器，现在的应用领域非常的多，比如：工业上实现自动化、计算机与通信领域内、家用电子类的产品、现代化的机械武器设备、并且在医学中也有很大的应用。

而本文需要用到的单片机是 STC89C52 单片机，此单片机是由 STC 公司设计的，它是一种功耗非常低但性能却很高的 8 位数字芯片。内部嵌入了可用于编程的 Flash 存储器，加上高性能的 CPU 使得 STC89C52 可配合外部器件发挥其强大的自身功能。本文基于 STC89C52 单片机，通过编程实现了数字钟的设计，它不但能够显示 24 小时制的时间，而且能够人为的调整时间，同时它还可以显示具体的年、月、日。

本文实现的数字时钟主要是由四个部分构成，第一是提供振荡信号的 DS1302 芯片，第二为转换振荡信号、实现多种功能的 STC89C52 单片机，第三部分是可根据需要将结果呈现出来的 LCD1602 显示器，以及最后一部分独立按键，通过它可实现对时间的调节。本设计采用 C 语言和汇编混合编程，最终完成的程序能完成对 DS1320 芯片进行操作，使其产生秒振荡信号，控制 STC89C52 单片机实现相应功能。

本文设计的数字时钟和其他的数字时钟相比操作更加的方便简洁，使用者可以对时间进行自己的想要的调节，并且时间的显示也很精确。由于是使用 STC89C52 单片机，还降低了它的功耗，用户体验极佳。

关键词： DS1302 数字时钟芯片； STC89C52 单片机；LCD1602 液晶显示器；独立按键。

1 绪论

1.1 论文研究背景和目的

科学技术的快速发展带动 了单片机的发展，现已经走进人们的生活，并且在各个领域都有了自己的发展。比如说在工业的自动化上，现在已经有了机械化的生产流水线，还可以对数据进行检测和收集；在计算机与通信领域上，现在也有了手机、调制解调器和通信时使用的交换机等；在家用电子类产品上，还有洗衣机、微波炉、空调、冰箱、集成灶、电视机、智能汽车等；在现代化的机械武器上有战斗机、军舰、航母、原子弹、导弹等；在医学领域也有心脏起搏器、心电图机、脑电图机、肌电图机等。在单片机如此盛行生活节奏如此快的情况下，人们对时间的概念的需求也越来越高。而在市面上却很少有那种调节方便，功耗很低的电子时钟的出现，所以本文的目的就是利用 STC89C52 单片机功耗很低的特点来设计出一款便于用户操作、时间准确并且功耗低的电子时钟出来方便用户使用与操作。

1.2 论文的主要内容本论文分六章对单片机的时钟设计进行论述。

首先第一章介绍了本论文主要写的内容；第二章介绍了单片机的发展和分类，并介绍了单片机的内部结构；第三章深入研究 LCD1602 液晶显示器的各种设置工作原理及如何通过 C 语言编写代码实现对液晶显示器的控制；第四章主要研究了 DS1302 的内部各种寄存器和 RAM，并介绍如何通过 C 语言编写代码实现对时钟模块的控制，使其产生振荡

信号；第五章主要研究了 STC89C52 对 DS1302 产生的振荡信号进行控制并结合他们一起实现日历与时间的显示，并且可以利用按键来对显示的时间进行调节；最后第六章是对本论文做的内容做一个总结与展望。

论文的具体内容结构如下：

1)绪论：主要介绍论文内容。

2)单片机：介绍单片机的发展、分类与结构。

3)LCD1602 液晶显示器：介绍如何实现 LCD1602 的显示功能。

4)DS1302 时钟模块：主要介绍时钟模块的功能及其如何实现控制。

5)时钟设计的实现：各部件与 CPU 一起实现时间与日历的显示，并可以对时间与日历进行修改。

6)总结：对本论文进行总结。

2 单片机概述

2.1 什么是单片机

单片机就是集成了微处理器，存储寄存器和输入输出接口于一块硅片上的微型计算机。尽管它的绝大部分功能集成在一块小小的芯片上，但它还是有一个完整计算机所需要的大部分部件：CPU、内存、内部和外部总线系统，目前大部分还会具有外存。同时集成诸如通讯接口、定时器，实时时钟等

外围设备。而现在最强大的单片机系统甚至可以将声音、图像、网络、复杂的输入输出系统集成在一块芯片上。

单片机也被称为微控制器（Microcontroller），是因为它最早被用在工业控制领域。单片机由芯片内仅有 CPU 的专用处理器发展而来。最早的设计理念是通过将大量外围设备和 CPU 集成在一个芯片中，使计算机系统更小，更容易集成进复杂的而对体积要求严格的控制设备当中。INTEL 的 Z80 是最早按照这种思想设计出的处理器，从此以后，单片机和专用处理器的发展便分道扬镳。

2.2 单片机的分类

单片机按系列分可以分成两个系列，51 子系列和 52 子系列。在这两个子系列下单片机还有不同的不同的型号，每个子系列都分别对应了 4 个型号，不同的型号它们的片内存储器、片外存储器、I/O 口、中断源和定时/计数器都有所不同，具体情况如表 2.1：

系列	型号	片内存储器		片外存储器		I/O口		中断源	定时/计数器
		ROM	RAM	ROM	RAM	并行	串行		
51子系列	8031、80C31	无	256B	64KB	64KB	4*8	1	5	2*16
	8051、80C51	4KB ROM							
	8751、87C51	4KB EPROM							
	8951、89C51	4KB E^2PROM							
52子系列	8032、80C32	无	512B					6	3*16
	8052、80C52	8KB ROM							
	8752、87C52	8KB EPROM							
	8952、89C52	8KB E^2PROM							

表2.1 MCS-51单片机的分类

2.3 单片机的内部结构虽然单片机的类型很多，但它们的内部结构基本一致。如图 2.1 所示：

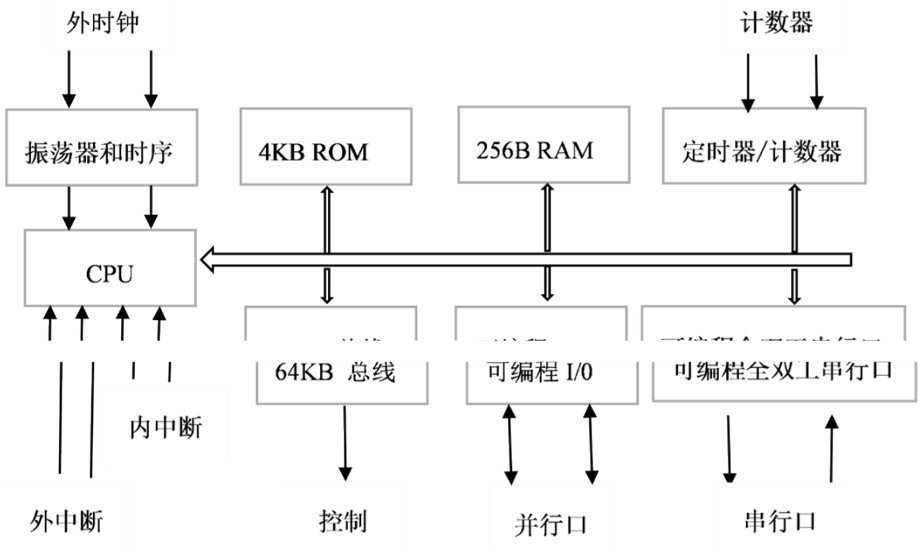


图 2.1 内部结构图 2.1 单片机系统结构框图从图 2.1 可以看出单片机主要集成了以下结构：

1)处理器（CPU)：是单片机结构中最重要的部件，负责控制整个单片机的工作。

2)数据存储器(RAM)：它内部有 128 字节数据存储器（RAM）和 21 个专用寄存器单元，它们是统一编址的，通常用于存放控制指令数据。

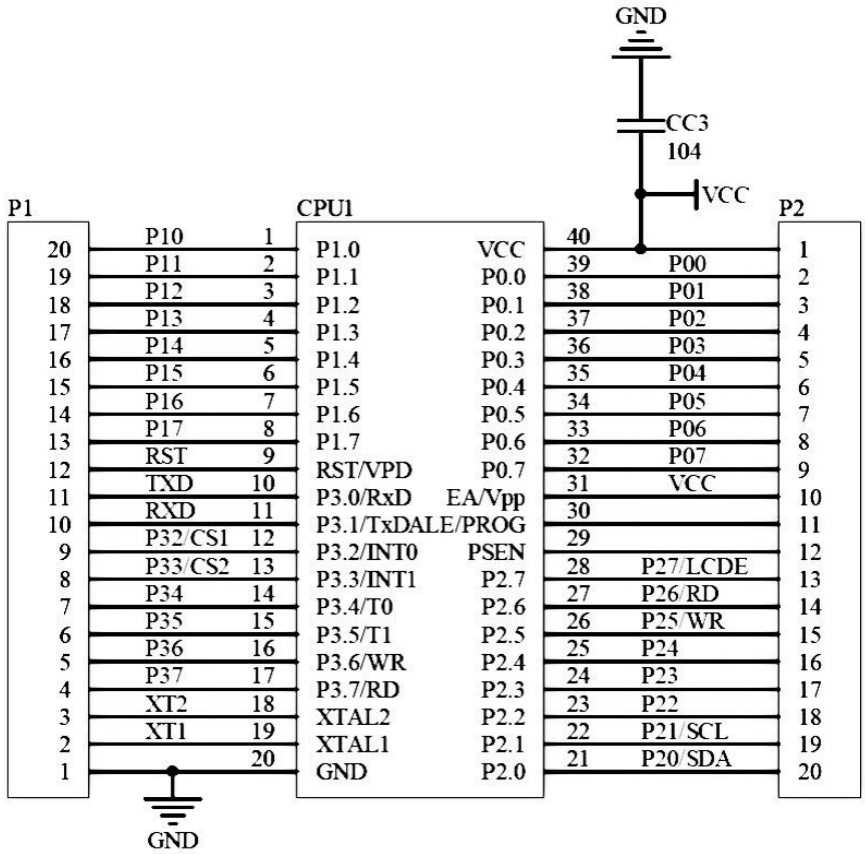
3)程序存储器(ROM)：它共有 4K 字节程序存储器（ROM)，用于存放用户程序和数据表格。

4)并行输入输出(I/O)口：它共有 4 个 8 位的并行 I/O 口(P0、P1、P2、P3)，用于输入输出数据。

5)全双工串行口：用于与其它设备间的串行数据传送。

6)中断系统：有六个中断源，可基本满足不同的控制要求。

7)时钟电路：它内部有最频率高达 12MHz 的时钟电路，用于产生时钟信号，但需外接晶体振荡器和振荡电容。



2.4 图 2.2 引脚连接图 STC89C52 各引脚连接图本单片机芯片有 40 个引脚，可被分成 3 中类型：

1)电源引脚和时钟引脚：VCC、GND、XTAL1、XTAL2。电源引脚一个接电源一个接地，时钟引脚外接时钟信号。

2)可编程控制引脚：RST、PSEN、ALE、Vpp。

3)I/O 口：有 4 组，每组 8 口，共 32 个 I/O 口，P0、P1、P2、P3。

在本程序设计中主要介绍一下与 LCD1602 液晶显示器连接的 P0 组的 8 个口和 P2.5、P2.6、P2.7；与 DS1302 时钟模块连接的 P3.4、P3.5、P3.6 口；与按键模块相关的外部中断源 0 与 P3.2 接口连接。

3LCD1602

3.1 LCD1602 工作原理

3.1.1 技术参数

本论文中使用的 LCD1602 液晶显示器由 5V 的电压来驱动，并且可以显示两行字符，每行一共有 16 个字符，但是却不可以对汉字进行显示。LCD1602 共有 16 个引脚，定义如下表 1 所示：

编号	符号	作用	编号	符号	作用
1	Vss	电源地	9	D2	数据总线
2	VDD	电源	10	D3	数据总线
3	VO	对比电压	11	D4	数据总线
4	RS	数据/命令选择端	12	D5	数据总线
5	R/W	读/写选择端	13	D6	数据总线
6	E	使能端	14	D7	数据总线
7	D0	数据总线	15	BLA	背光电源正极
8	D1	数据总线	16	BLK	背光电源负极

表3.1 LCD1602接口信号说明

各引脚图片如图 3.1 所示：

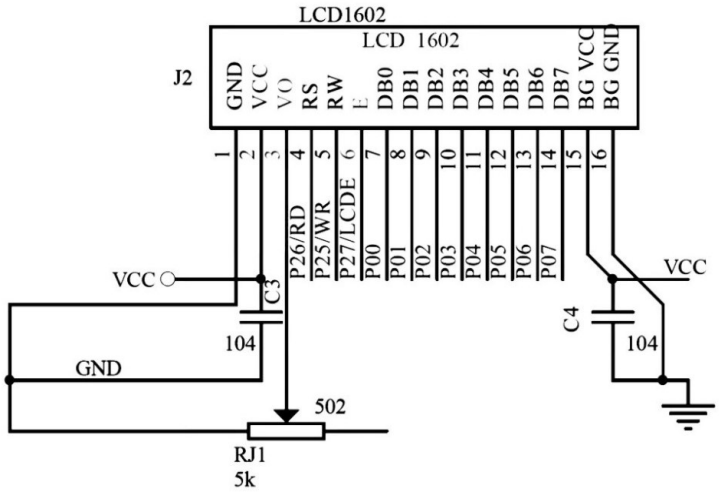


图 3.1 LCD1602 各引脚连接图

RS 端与单片机 P2.6 端连接，RW 端与单片机 P2.5 端连接，E 端与单片机 P2.7 端连接，DB0~DB7 分别与单片机的 P0.

1~P0.7 端相连。所以要利用以下函数对 PIN 口进行定义：

```
#define LCD1602_DATAPINS P0 sbit LCD1602_E=P2^7;  sbit LCD1602_RW=P2^5;  sbit
LCD1602_RS=P2^6;
```

定义好之后即可通过对单片机各端口的控制从而实现对 LCD1602 各端口的控制。在写入命令与数据时由于都是写

入所以 RW 端始终为 0 不需要修改。首先写入命令“LCD1602_RS = 0；”；然后写放入命令“LCD1602_DATAPINS = com；”；经过一段延时“Lcd1602_Delay1ms(1)；”等命令稳定之后给使能端一个高电平“LCD1602_E = 1；”，再次延时一段时间以待命令写入，最后将使能端置为低电平。写入数据与写入命令操作由两个函数独立完成，函数内部唯一的区别在于写入数据时 RS 端为高电平“LCD1602_RS = 1；”。操作如下：写命令：

```
void LcdWriteCom(uchar com) {
LCD1602_RS = 0;
LCD1602_DATAPINS = com;
Lcd1602_Delay1ms(1);
LCD1602_E = 1;
Lcd1602_Delay1ms(5);
LCD1602_E = 0;
}
```

写数据：

```
void LcdWriteData(uchar dat)
{
LCD1602_RS = 1;
LCD1602_DATAPINS = dat;
Lcd1602_Delay1ms(1);
LCD1602_E = 1;
Lcd1602_Delay1ms(5);
LCD1602_E = 0;
}
```

3.1.2DDRAM

LCD1602 内部有显示数据存储器 DDRAM，共 80 字节字符显示位与 DDRAM 的地址关系如表 3.2 所示：

显示位置	DDRAM地址	
	第一行	第二行
1	00	40
2	01	41
3	02	42
4	03	43
5	04	44
6	05	45
7	06	46
:	:	:
40	27	67

表 3.2 显示位置与 DDRAM 地址关系

DDRAM 第一行的地址从 00H 到 27H 结束，第二行从 40H 开始到 67H 结束，每行一共有 40 个地址，但是液晶显示器每行却只有 16 个字符可显示。所以我们一般只选用前 16 个地址即可。当我们在液晶显示屏上的第一行第一列上显示一个字符时将该字符的 ASCII 码写入第一行第一列对应的地址时发现在屏幕上并不能正常显示出这个字符，此时必须在第一行第一列地址 00H 的基础上加上 80H 才能正常显示。

3.1.3 数据指针设置

控制器内部设计有数据地址指针，用户可以通过它们访问内部全部的 RAM，具体如表 3.3：

指令码	功能
80H+地址码	设置数据地址指针

表 3.3 数据指针设置

在此主函数程序中首先要在 LCD1602 液晶显示器的第一行中显示年、月、日和星期。前面已经讲过显示时要在原来地址的基础上加上 80H，所以在主函数中执行完 LCD1602 的初始化函数之后需要用“LcdWriteCom(0x80)；”命令先将数据指针定位在第一行的第一个字处，然后开始写入后面的数据。而时、分、秒则要写在第二行，初始位置在第二行第一个字，所以此时需要将第二行第一个字的地址码加上 80H。所以第二行重新定位数据指针“LcdWriteCom
(0x80+0x40)；”，同样开始写入后面的数据。

3.1.4 显示模式设置

在使用 LCD1602 显示之前要对其进行显示的初始化设置，因为需要用它来显示，所以要先将显示打开，其指令码

如表 3.4 所示：

指令码								功能
0	0	1	1	1	0	0	0	设置显示开

表 3.4 显示模式设置

在显示过程当中按是否需要显示光标闪烁以及读写一个地址后指针是加还是减、光标是加还是减对对应了不同的指令码，如表 3.5 所示：

指令码								功能
0	0	0	0	1	D	C	B	当 D=1 时开显示；当 D=0 时关显示； 当 C=1 时有光标；当 C=0 时无光标； 当 B=1 光标闪烁；当 B=0 光标不闪烁。
0	0	0	0	0	1	N	S	N=1 当读或写一个字符后地址指针加 1，且光标加 1； N=0 当读或写一个字符后地址指针减 1，且光标减 1； S=1 当写一个字符时，整屏显示左移或右移。 S=0 当写一个字符时，整屏显示不移动。3

表 3.5 显示开关及光标设置

由指令码可知在此程序中首先通过“LcdWriteCom(0x38);”打开显示；由于此程序中不需要光标的显示所以要设置开显示但不显示光标“LcdWriteCom(0x0c);”；在写一个字符之后要接着写下一个字符，所以要让地址指针在写入一个字符之后会自动加一位“LcdWriteCom(0x06);”；初始化的过程中要对显示屏进行清零显示并且在第一行第一列开始写入数据与命令“LcdWriteCom(0x01);
LcdWriteCom(0x80);”。

LCD 初始化：

```
void LcdInit()
{
LcdWriteCom(0x38); //开显示
LcdWriteCom(0x0c); //开显示不显示光标
LcdWriteCom(0x06); //写一个指针加 1
LcdWriteCom(0x01); //清屏
LcdWriteCom(0x80); //设置数据指针起点
}
```

3.1.5LCD1602 写操作时序图

在 LCD1602 开始传输命令之前，首先要将使能端设定为低电平。在此程序当中由于只对 LCD1602 进行写操作所以 R
/W 端可以直接设置为低电平，开始时选择发送命令所以将 RS 端也设置为低电平，当电平都设置好之后就可以将使能端置为高电平开始传入命令，保持一段时间之后传输结束就可以将使能端降为低电平。当写入数据时就要将 RS 端在使能清零后置为高电平。

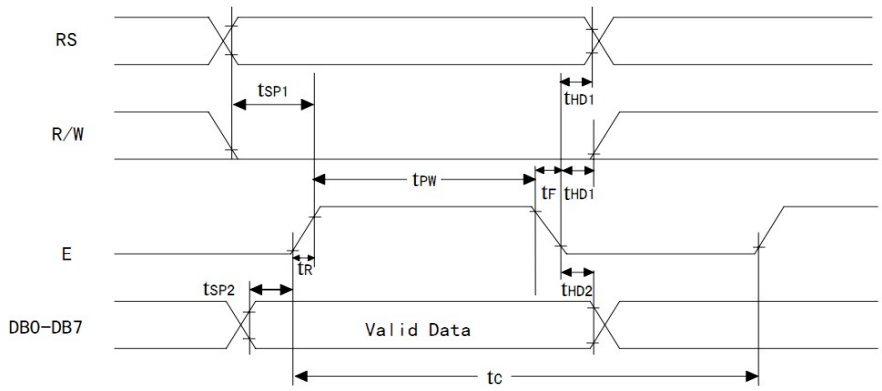


图 3.2 LCD1602 写操作时序图

3.2LCD1602 程序设计总结

每个程序的设计都有一定的思路，所有程序的编写都要在第一行写上所包含的头文件，并定义好所需要的关键字总体设计总结如下：

1)在任何程序的开始都需要先包含所需要的头文件“#include<>”;

2)写好头文件之后要对自己需要的一些关键字重新定义“#define 新名字 原名字”;

3)在了解 LCD1602 与单片机连接的各个对应的管脚之后要对这个管脚进行定义以便后续对管脚的高低电平进行控制;

4)由于此函数需要用到延时函数，所以要写一个延时函数对某些操作进行延时处理“void Lcd1602_Delay1ms
(uint c);”;

5)在使用 LCD1602 液晶显示器时要对显示器进行清零处理“void LcdInit();”，以免之前的显示影响现在的显示;

6)LCD1602 的写入命令与写入数据操作分别由两个独立的函数来完成，函数内部唯一的区别就是写入命令时 RS 端选择低电平，写入数据时 RS 端选择高电平。写入命令“void LcdWriteCom(uchar com);”，写入数据“void
LcdWriteData(uchar dat); ”;

7)最后在主函数中对 LCD1602 液晶显示器进行主控。

4DS1302 时钟模块

4.1 DS1302 特性

DS1302 是一个时钟模块，在本程序中利用它来产生振荡信号。在使用一个器件之前都先要了解此器件的有什么功能，可以工作在什么样的情况之下，其特性如下：

1)DS1302 时钟模块能够计算秒、分、时、日、月、星期、年，还可以对闰年进行调整;

2)在 DS1302 时钟模块的内部含有 31 个字节的静态 RAM，可给用户访问;

3)采用串行数据传输方式，使得管脚数量最少；

4)工作电压范围：2.0~5.5V；

5)工作电流：2.0V 时，小于 300nA；

6)采用 8 脚 DPI 封装或 SOIC 封装；

7)与 TTL 兼容，Vcc=5V；

8)它的温度范围为：-40~85 摄氏度；

9)具有涓流充电功能；

10)采用主电源和备份电源双电源供电；

4.2DS1302 内部结构

DS1302 内部结构如图 4.1 所示：

图 4.1 DS1302 内部结构

Vcc1Vcc2GND

I/OSCLK

RST

DS1302 主要由实时时钟、振荡电路与分频器、数据存储与 RAM、命令与控制逻辑、输入移位寄存器、和电源控制等组成。在接通电源传输控制字节与数据时从 I/O 口输入与输出，并在 SCLK 为低电平时将使能端置为高电平，此时再给一个时钟周期信号即可将数据输入到移位寄存器当中，或者输出数据，一旦使能端变为低电平就会停止输入输出数据。而振荡电路与分频器外接一个 32.768KHz 的晶振控制实时时钟计时。

4.3DS1302 硬件及引脚功能

DS1302 的引脚如图 4.2 所示：

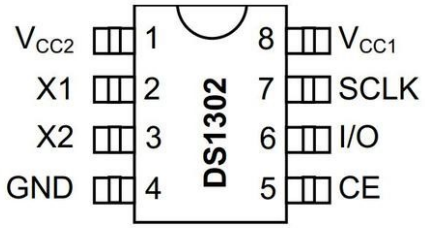


图 4.2 引脚图

引脚	名称	功能
1	VCC2	双供电设备中的主电源供应脚。
2	X1	接入 32.768KHz 晶振引脚
3	X2	
4	GND	电源地
5	CE	使能端，当 CE=1 时允许读写 DS1302 时钟模块数据；当 CE=0 时不允许读数据。
6	I/O	双向输入线。
7	SCLK	串行时钟输入端，控制输入与输出。
8	VCC1	双供电设备中的备用电源供应脚。

表 4.1 各引脚及功能

DS1302 各引脚与单片机连接如图 4.3 所示：

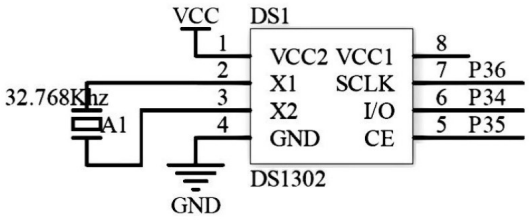
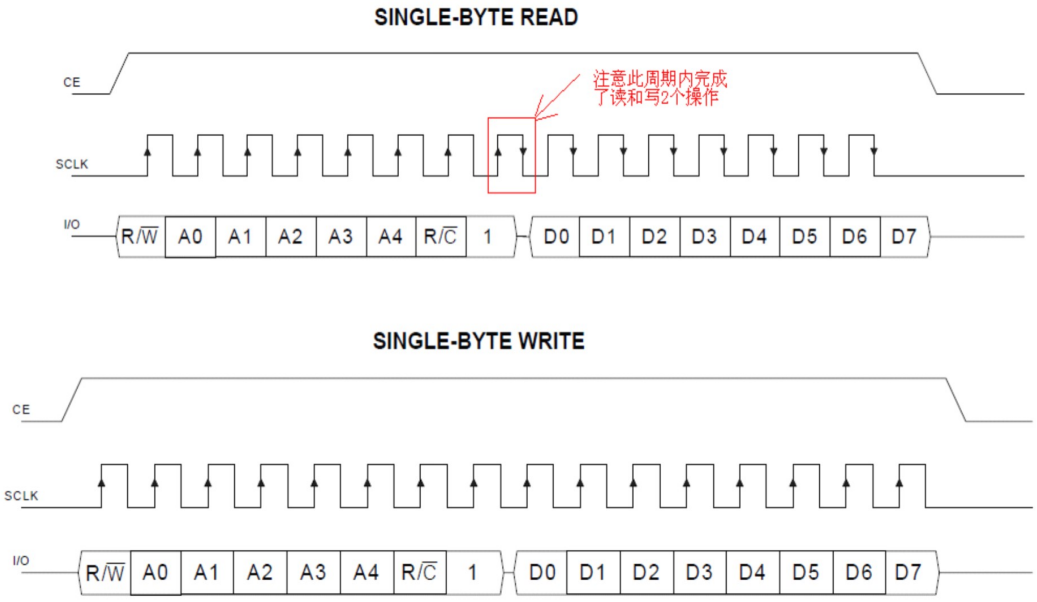


图 4.3 连接引脚

DS1302 的 SCLK 引脚与单片机的 P3.6 接口连接，I/O(DSIO)引脚与单片机 P3.4 接口连接，CE(RST)端与单片机的 P3.

5 接口连接。所以要利用以下命令对 PIN 口进行定义： sbit DSIO=P3^4; sbit RST=P3^5; sbit SCLK=P3^6;



4.4 图 4.4 通信时序

DS1302 通信时序

DS1302 时钟模块在进行写操作时首先传送一个控制字节，告诉它我要进行写操作，并在前 8 个时钟周期每个时钟周期的上升沿输入，然后在后 8 个时钟周期的上升沿写入数据。在进行读操作时传送一个不同的控制字节，告诉它我要进行读操作，并在前 8 个时钟周期每个时钟周期的上升沿输入，然后在第 8 个时钟周期的下降沿开始读出数据。所以每次都要传送两个字节。

只有在时钟信号 SCLK 端口为低电平时，才能将使能端 CE 置为高电平。所以在进行操作之前先将 SCLK 置为低电平，然后将 CE 置为高电平。

```
{
SCLK = 0;//置低
_nop_(); //延时一个机械周期
RST = 1; //置高
_nop_(); //延时一个机械周期
}
```

写操作：CE 置高后，I/O 总线上开始从低位到高位传输写操作控制字节，8 各字节传输结束之后在 SCLK 的下一个上升沿开始传输初始数据，传输结束将 CE 置低。利用 for 循环对 8 位命令和数据进行写入： for (n=0; n<8; n++) //传送八位地址命令

```
{
DSIO = addr & 0x01;//数据从低位开始传送 addr >>= 1; //将命令移动到下一位
SCLK = 1; //上升沿写入命令
_nop_();
```

```
SCLK = 0;
_nop_();
}
```

写入数据与写入命令基本一致，将 addr 改为 dat 即可。

读操作：CE 置高后，I/O 总线上开始从低位到高位传输读操作控制字节，8 各字节传输结束之后在 SCLK 的本周期的下降沿马上开始读取数据，读取结束后将 CE 置低。注意读操作时在第 8 个周期处完成了写与读两个操作。代码与写操作基本一致。不同之处有以下两点：

1)需要多定义一个变量 dat1 来存储从最低位开始读出的数据；

2)在写操作中地址与数据的移位都在传送之后，而读操作中虽然写地址的移位也在传送之后，但是读数据的操作要在传送之前写成“dat = (dat>>1) | (dat1<<7);”： for(n=0; n<8; n++) //读取 8 位数据

```
{
dat1 = DSIO; //从最低位开始接收 dat = (dat>>1) | (dat1<<7);
SCLK = 1;
_nop_();
SCLK = 0; //DS1302 下降沿时，放置数据
_nop_();
}
```

在读数据时是要读取出数据的，所以它有一个返回值，在程序的后面要加一个“return dat; ”。

4.5 寄存器及片内 RAM

4.5.1 控制寄存器

控制寄存器用于存放 DS1302 的控制命令字，DS1302 的 CE 引脚回到高电平后写入的第一个字就是控制命令。它用于对 DS1302 读写过程进行控制，格式如表 4.2 所示：

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

表 4.2 控制寄存器　　其中 D7 固定为 1；当 D6=1 时对片内 RAM 进行读写操作、D6=0 时对日历、时钟寄存器进行读写操作；D5~D1 为地址位，用于选择进行读写的日历、时钟寄存器或片内 RAM。对日历、时钟寄存器或片内 RAM 的选择见表 4.3 与表 4.4；D0 为读写选择，当 D0=0 时写入；当 D0=1 时读出。

寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
	1	RAM/CK	A4	A3	A2	A1	A0	R/W
秒寄存器	1	0	0	0	0	0	0	0或1
分寄存器	1	0	0	0	0	0	1	0或1
时寄存器	1	0	0	0	0	1	0	0或1
日寄存器	1	0	0	0	0	1	1	0或1
月寄存器	1	0	0	0	1	0	0	0或1
周寄存器	1	0	0	0	1	0	1	0或1
年寄存器	1	0	0	0	1	1	0	0或1

表4.3 时钟寄存器

寄存器名称	D7	D6	D5	D4	D3	D2	D1	D0
	1	RAM/CK	A4	A3	A2	A1	A0	R/W
写保护寄存器	1	0	0	0	1	1	1	0或1
慢充电寄存器	1	0	0	1	0	0	1	0或1
时钟突发模式	1	0	1	1	1	1	1	0或1
RAM0	1	1	0	0	0	0	0	0或1
:	1	1	:	:	:	:	:	0或1
RAM30	1	1	1	0	1	1	0	0或1
RAM突发模式	1	1	1	0	1	1	1	0或1

表 4.4 片内 RAM

无论在读还是写操作的过程当中都需要先写入地址之后才能继续进行写入数据和读取数据，所以在写程序开始时要将每个寄存器的控制字节写好。通过上表可以分别写出当进行读操作时的秒、分、时、日、月、周、年寄存器
的控制字节“uchar code READ_RTC_ADDR[7] = {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};”；写操作时的控制字节“uchar code WRITE_RTC_ADDR[7] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};”。需要传输的初始时间数据是 2013 年 1 月 1 日周二，12 点 0 分 0 秒：“uchar TIME[7] = {0x00, 0x00, 0x12, 0x13, 0x04, 0x02,
0x21};”。

4.5.2 日历、时钟寄存器

DS1302 共有 12 个寄存器，其中有 7 个与日历、时钟有关，存放的数据为 BCD 码形式。日历、时钟寄存器的格式表如图 4.5 所示：

寄存器名称	取值范围	D7	D6	D5	D4	D3	D2	D1	D0
秒寄存器	00~59	CH	秒的十位			秒的个位			
分寄存器	00~59	0	分的十位			分的个位			
时寄存器	01~12或00~23	12/24	0	A/P	HR	小时的个位			
日寄存器	01~31	0	0	日的十位		日的个位			
月寄存器	01~12	0	0	0	1或0	月的个位			
周寄存器	01~07	0	0	0	0	星期几			
年寄存器	01~99	年的十位				年的个位			

写保护寄存器		WP	0	0	0	0	0	0	0
时钟突发模式				S		S			S

表 4.5 格式说明：

1)数据都以 BCD 码形式；

2)小时寄存器的 D7 位用于选择是 12 小时制还是 24 小时制，当为 1 时选 12 小时制，当为 0 时选 24 小时制。

3)秒寄存器中的 CH 位为时钟暂停位，当为 1 时钟暂停，为 0 时钟开始启动。

在开始使用 DS1302 之前要对 DS1302 进行初始化，由于 DS1302 有写保护的功能，所以在对其写命令时要关闭它的

写保护功能才能进行读写命令与数据。由上表可知关闭写保护的十六进制数为“0x00”，所以通过命令

“Ds1302Write(0x8E,0X00);”关闭写保护功能。关闭写保护之后就可以开始写入命令和数据了：

```
for (n=0; n<7; n++)//写入 7 个字节的时钟信号：分秒时日月周年
{
Ds1302Write(WRITE_RTC_ADDR[n],TIME[n]);
}
```

写好数据之后还要把写保护打开，以免后续的操作对数据造成影响。打开写保护的十六进制为“0x80”，所以通过命令“Ds1302Write(0x8E,0x80);”打开写保护功能。

4.6 BCD 码转换

我们时钟日历寄存器使用的是 8421 码型的 BCD 码，BCD 码还有 5421 码、2421 码等，其中 8421 码型的 BCD 码最最常用。

BCD 码是用四位二进制数表示一位十进制数的 0-9 这十个数简称 BCD 码，其转换如表 4.6 所示：

0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

表 4.6 转换形式

当十进制数大于 9，比如十进制的 10 要转换为 BCD 码就让前四位表示十进制的十位，后四位表示十进制的个位。所以十进制的 10 转换为 BCD 码为 00010000。或者可将 10 转为二进制，然后让二进制数加上 6 的二进制数 0110 也可以得到 BCD 码。