第 2-10 讲：DS1302 时钟

1. 学习目的

1. 了解 SPI 总线的特点。
2. 了解 DS1302 时钟芯片 3 线接口的特点及芯片使用注意事项。
3. 掌握 IAP15F2K61S2/IAP15W4K61S4 单片机读写 DS1302 时钟芯片的程序设计。

2. SPI 总线概述

SPI 是串行外设接口(Serial Peripheral Interface)的缩写，是一种高速、全双工、同步的通信总线。SPI 是 Motorola 公司推出的一种同步串行接口技术，SPI 由一个主设备和一个或多个从设备组成，在一次数据传输过程中，接口上只能有一个主机和一个从机能够通信。

SPI 总线的优点是操作简单、数据传输速率较高、全双工，缺点是只支持单个主机、没有指定的流控制，没有应答机制确认是否接收到数据。

2.1. 接口信号定义

SPI 总线接口包括以下四种信号：

1) MOSI（Master Output，Slave Input）：主器件数据输出，从器件数据输入。
2) MISO（Master Input， Slave Output）：主器件数据输入，从器件数据输出。
3) SCK（Serial Clock）：有时也称为 SCLK，时钟信号，由主器件产生。
4) CS（Chip select）：有时也称为 SS，从器件使能信号，由主器件控制，实际使用时，经常用 GPIO 来代替。

SPI 总线支持连接多个从机，如下图所示，SPI 主机通过连接到从机的片选信号使能/禁止从机，并且同时只能使能一个从机，因此总线里面有多少个从机，就需要多少个片选信号。当 SPI 主机需要和总线中某个从机进行通信时，主机会拉低对应的 CS 信号使能该从机，之后发起通信，通信完成后，拉高 CS 信号，解除总线的占用。

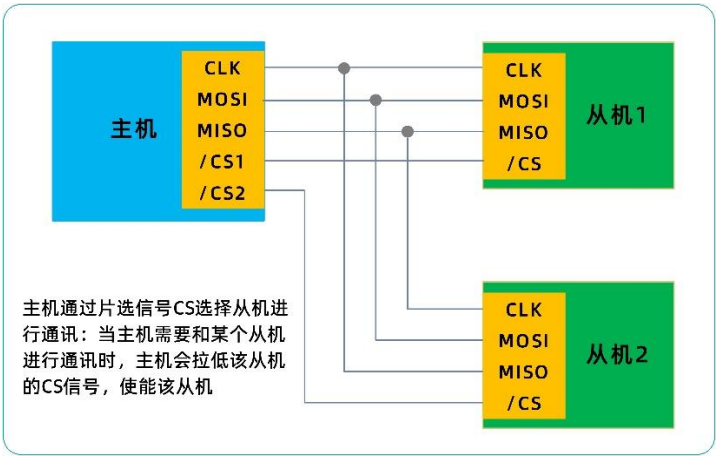


图 1：SPI 总线结构

对于 SPI 总线，我们还需要能深刻理解下面几个知识点。

1. 硬件片选和软件片选的区别

所谓硬件片选指的是 SPI 本身具有片选信号，当我们通过 SPI 发送数据时，SPI 外设自动拉低 CS 信号使能从机，发送完成后自动拉高 CS 信号释放从机，这个过程是不需要软件操作的。而软件片选则是需要使用 GPIO 作为片选信号，SPI 在发送数据之前，需要先通过软件设置作为片选信号的 GPIO 输出低电平，发送完成之后再设置该 GPIO 输出高电平。

2. SPI 总线是回环结构

SPI 是一个环形总线结构，如下图所示，主设备和从设备构成一个环形。在时钟 SCK 的作用下，主设备发送一个位到从设备，因为是环形结构，所以从设备必定会同时传送一个位到主设备。同样，主设备向从设备发送一个字节，从设备也必定会同时传送一个字节到主设备。理解了环形结构，就很容易理解下面几点：

- SPI 是全双工，同步的通信总线。
- 主设备向从设备发送数据时，无论我们需不需要从设备返回数据，从设备都会返回数据。
- 主设备从从设备读取一个字节数据时，为什么需要写一个字节数据到从设备：因为是环形结构，不写数据过去，对方的数据就不会被移位过来。

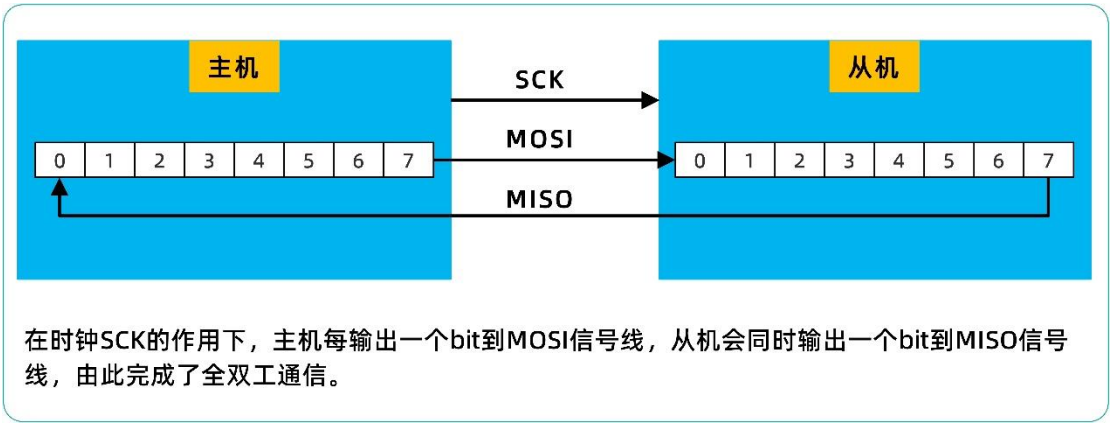


图 2：SPI 数据传输示意图

3. SPI 主机和从机之间连接时信号不需要交叉

SPI 主机和从机连接时，MOSI 和 MISO 信号是不需要交叉连接的，因为 MOSI 本身就表示了主机输出、从机输入，MISO 表示主机输入、从机输出，因此不能交叉连接。

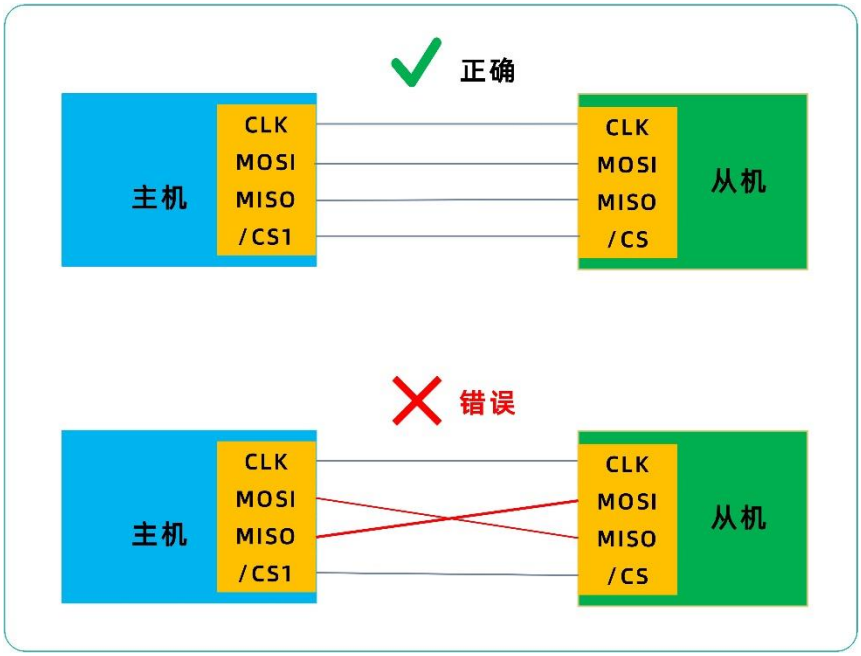


图 3：MOSI 和 MISO 不能交叉连接

2.2. SPI 的 4 种通信模式

SPI 总线共有 4 种通信模式：模式 0~模式 3，这 4 种通信模式是由时钟相位和时钟极性确定的。

1. 时钟相位 CPOL（Clock polarity）：SPI 总线空闲时，时钟信号 SCLK 的电平称为时钟极性，有以下两种模式：
   - CPOL=0：SPI 总线空闲时，时钟信号为低电平。
   - CPOL=1：SPI 总线空闲时，时钟信号为高电平。
2. 时钟极性 CPHA（Clock phase）：SPI 在时钟信号 SCLK 第几个边沿开始采样数据，有以下两种模式：
   - CPHA=0：在第 1 个时钟边沿进行数据采样。
   - CPHA=1：在第 2 个时钟边沿进行数据采样。

时钟极性 CPOL 时钟相位 CPHA 各有 2 种模式，他们两两组合就形成了 SPI 的 4 种通信模式，如下表所示。

表 1：SPI 总线的 4 种工作模式

模式	描述
模式 0	CPOL=0，CPHA=0
模式 1	CPOL=0，CPHA=1
模式 2	CPOL=1，CPHA=0
模式 3	CPOL=1，CPHA=1

SPI 的 4 种模式中，最常用的是模式 0 和模式 3。正是由于 SPI 有 4 种通信模式，因此当我们使用 SPI 总线时，需要去查询 SPI 总线中主机设备（如 IAP15W4K61S4）和从机设

备（如 SPI Flash）的数据手册，确定他们支持什么模式，从而选择适合的通信模式。

SPI 的 4 种模式的时序图如下。

1) 时钟相位 CPHA=0 时的时序

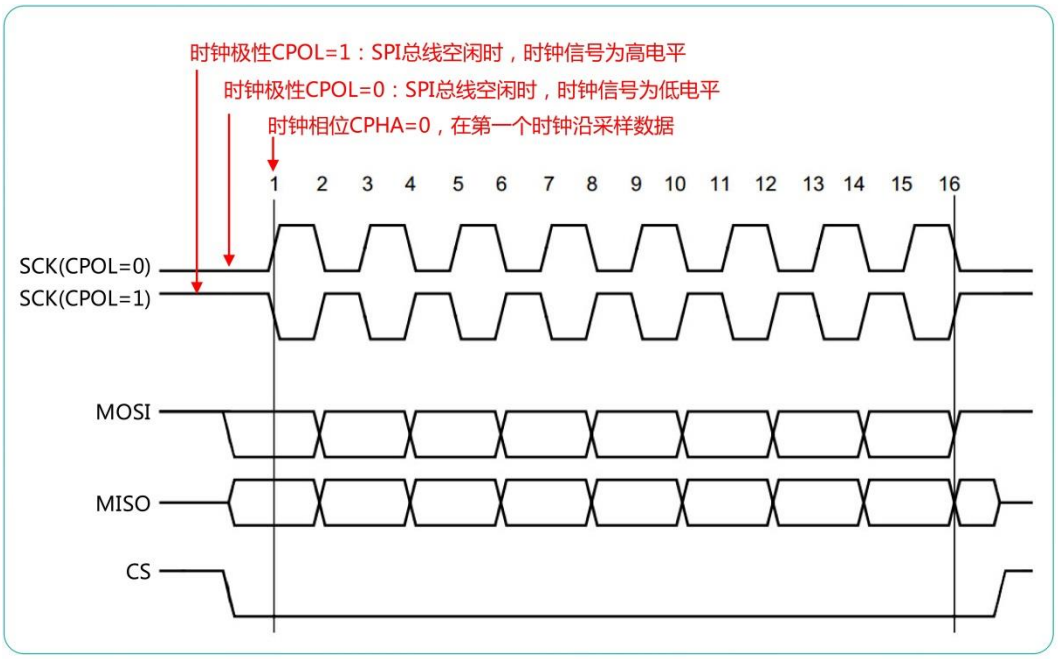


图 4：CPHA=0 时 SPI 时序

2) 时钟相位 CPHA=1 时的时序

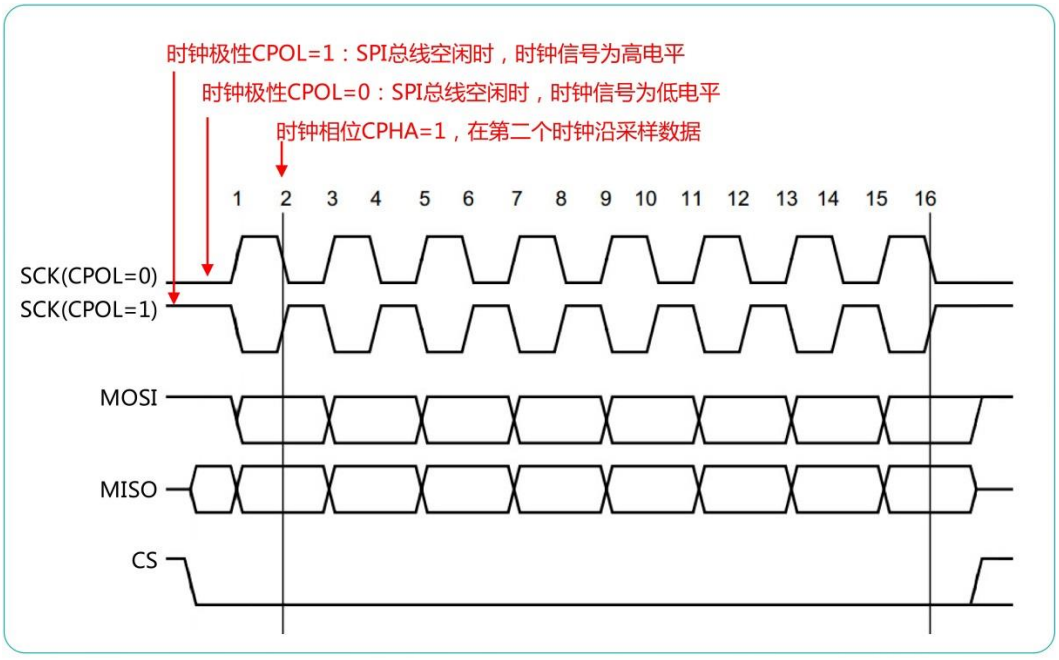


图 5：CPHA=0 时 SPI 时序

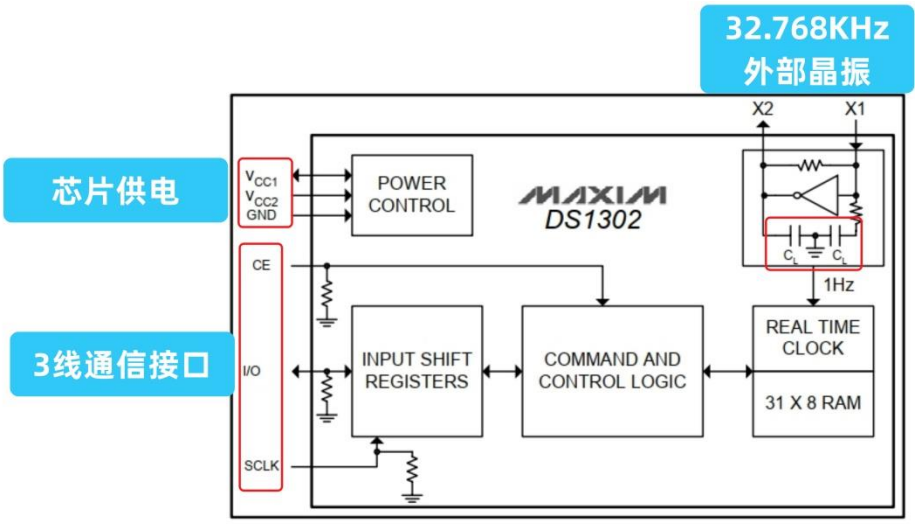
✧ DS1302 的 3 线通信接口（比标准的 SPI 接口少了一根线）包含 RST/CE 线、SCLK 线、IO 线（双向传输数据用，标准的 SPI 则将此线分为 MISO 和 MOSI）3 条线。

3. 硬件设计

PK107D 开发板上板载了 DS1302 时钟电路，该 DS1302 时钟芯片与单片机之间是简易的 3 线通信接口，下文针对 DS1302 的特点及使用进行介绍 。

3.1. DS1302 时钟芯片

DS1302 时钟芯片是一款可慢速充电实时时钟芯片，该芯片包含实时时钟/日历和 31 字节的非易失性静态 RAM。他经过一个简单的串行 3 线接口与微处理器通信， 实时时钟/日历可对秒， 分， 时， 日， 周， 月， 和年进行计数， 对于小于 31 天的月， 月末的日期自动进行调整， 还具有闰年校正的功能。 时钟可以采用 24 小时格式或带 AM（上午） /PM（下午） 的 12 小时格式。 31 字节的 RAM 可以用来临时保存一些重要数据。该芯片的内部结构及管脚图如下。



管脚序号	符号	功能	管脚序号	符号	功能
1	V_{CC2}	主用电源	5	CE	复位
2	X1	32.768kHz 晶体	6	I/O	数据输入/输出
3	X2	32.768kHz 晶体	7	SCLK	串行时钟输入
4	GND	地	8	V_{CC1}	备用电源

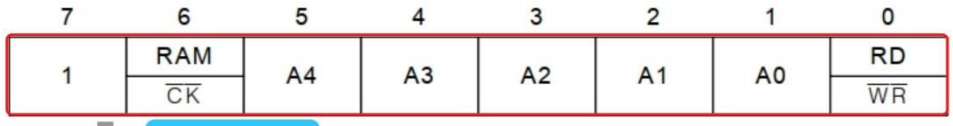
图 6：DS1302 内部结构及管脚图

相比 1 条线的单总线、2 条线的 I2C，DS1302 的 3 线通信相对比较简单。每次传送时，需要先发送 8bit 命令字节，再发送/接收 8bit 数据。

CE/RST 管脚先拉高，再在 IO 线上准备好数据，然后 SCLK 拉高，一个上升沿，发完 1bit。注意，微处理器向 DS302 发送数据的话，每次都是 SCLK 上升沿发送 1bit。如果是微处理器从 DS302 接收数据的话，注意，在传完最后命令字节的高电平之后的第一个下降沿 DS1302 发送数据。时序如下图所示。

图 7：DS1302 读写字节时序图

DS1302 命令字节：每一数据传送由命令字节初始化，最高有效位 MSB（位 7）必须为逻辑 1。 如果他是零， 禁止写 DS1302。 位 6 为逻辑 0 指定时钟/日历数据。位 6 为逻辑 1 指定 RAM 数据。 位 1 至 5 指定进行输入或输出的特定寄存器。 最低有效位 LSB（位 0）为逻辑 0 指定进行写操作（输入）； 逻辑 1 指定进行读操作（输出）。 命令字节总是从最低有效 LSB 位 0 开始输入。



READ	WRITE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RANGE
81h	80h	CH	10 Seconds			Seconds				00–59
83h	82h		10 Minutes			Minutes				00–59
85h	84h	12/$\overline{24}$	0	10 $\overline{\text{AM/PM}}$	Hour	Hour				1–12/0–23
87h	86h	0	0	10 Date		Date				1–31
89h	88h	0	0	0	10 Month	Month				1–12
8Bh	8Ah	0	0	0	0	0	Day			1–7
8Dh	8Ch	10 Year				Year				00–99
8Fh	8Eh	WP	0	0	0	0	0	0	0	—
91h	90h	TCS	TCS	TCS	TCS	DS	DS	RS	RS	—

WP：写保护位，该位为1时， 禁止对任何其他寄存器进行写操作。

图 8：DS1302 命令字节示意图

3.2. DS1302 时钟电路

开发板上的 DS1302 硬件电路如下图所示。为保证开发板断电情况下时钟芯片仍可工作，该电路加了后备电池座（可用 CR1220 电池），需注意该电池不支持充电，仅供演示开发板断电时时钟信息一直不间断功能。

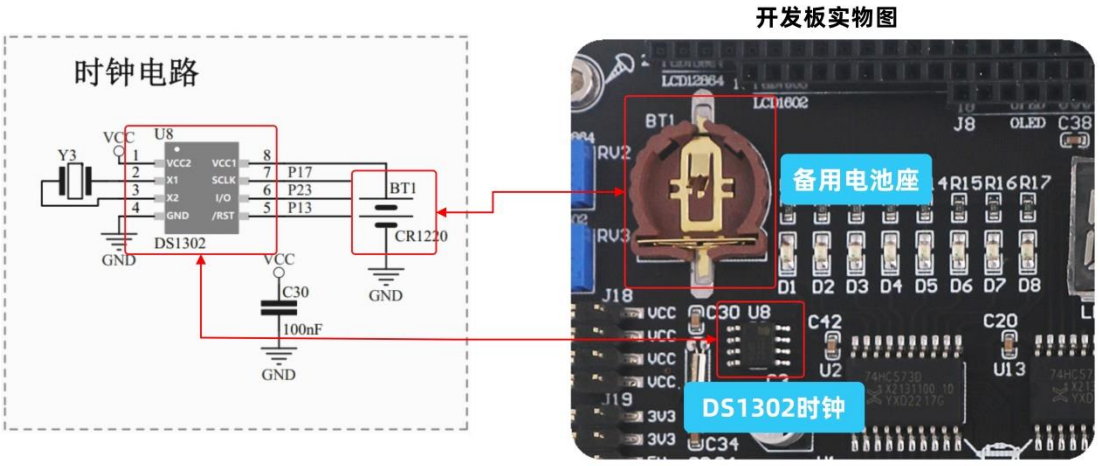


图 9：DS1302 电路

DS1302 时钟电路占用的 IAP15F2K61S2/IAP15W4K61S4 的引脚如下表：

表 2：连接 DS1302 引脚分配

名称	引脚	说明
SCLK	P1.7	独立 IO 口
I/O	P2.3	独立 IO 口
RST	P1.3	独立 IO 口

✧ 注：为了方便读者理解单片机访问 DS1302 时钟的编程，DS1302 时钟的 3 线通信时序以及读、写时钟信息操作将在软件设计部分讲解。

4. 软件设计

4.1. DS1302 时钟显示实验

✧ 注：本节的实验是在“实验 2-9-1：DS18B20 温度传感器 - 数码管显示”的基础上修改，本节对应的实验源码是：“实验 2-10-1：DS1302 时钟 - 数码管显示”。

4.1.1. 实验内容

配置 IAP15F2K61S2/IAP15W4K61S4 单片机与 DS1302 时钟通信的引脚，按照时序设计 DS1302 相关函数，完成以下操作。

- 单个字节写入：向 DS1302 写入 1 个字节数据。
- 向指定地址写数据：按照时序要求向 DS1302 发送数据。
- 从指定地址读取数据：按照时序要求从 DS1302 读取数据。
- 读写 RTC 时钟信息。

4.1.2. 代码编写

1. 新建一个名称为“ds1302.c”的文件及其头文件“ds1302.h”保存到工程的“Source”文件夹，并将“ds1302.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放

ds1302 读写相关的函数。

2. 引用头文件

因为在“main.c”文件中使用了“ds1302.c”文件中的函数，所以需要引用下面的头文件“ds1302.h”。

代码清单：引用头文件

```
1.  //引用 ds1302 的头文件
2.  #include    " ds1302.h"
```

3. 在 ds1302.c 文件中编写对 DS1302 的基本操作函数，如下表所示。

表 3：DS1302 相关函数汇集

序号	函数名	功能描述
1	Write_Ds1302	向 DS1302 写单字节数据。
2	Write_Ds1302_Byte	向 DS1302 发送数据。
3	Read_Ds1302_Byte	从 DS1302 读取数据。
4	Read_RTC	读 RTC 时钟信息。
5	Write_RTC	写 RTC 时钟信息。
6	DS1302_Init	RTC 初始化。

关于每个操作 DS1302 时钟相关函数，下面给出详细代码。

代码清单：向 DS1302 写单字节数据

```
1. /*******************************************************************************
2. 功能描述：写一个字节数据
3. 入口参数：uint8 addr, uint8 *p, uint8 number
4. 返回值：无
5.  *******************************************************************************/
6.  void Write_Ds1302(unsigned  char temp)
7.  {
8.      unsigned char i;
9.      for (i=0;i<8;i++)
10.     {
11.         SCK=0;
12.         SDA=temp&0x01;
13.         temp>>=1;
14.         SCK=1;
15.     }
16. }
```

代码清单：向 DS1302 发送数据

```
/*********************************************************************************************
功能描述：向指定地址写数据
入口参数：unsigned char address, unsigned char dat
返回值：无
*********************************************************************************************/
void Write_Ds1302_Byte( unsigned char address,unsigned char dat )
{
    RST=0;  _nop_();
    SCK=0;  _nop_();
    RST=1;  _nop_();
    Write_Ds1302(address);
    Write_Ds1302(dat);
    RST=0;
}
```

代码清单：从 DS1302 读取数据

```
/*********************************************************************************************
功能描述：从指定地址读取数据
入口参数：unsigned char address
返回值：无
*********************************************************************************************/
unsigned char Read_Ds1302_Byte ( unsigned char address )
{
    unsigned char i,temp=0x00;
    RST=0;  _nop_();
    SCK=0;  _nop_();
    RST=1;  _nop_();
    Write_Ds1302(address);
    for (i=0;i<8;i++)
    {
        SCK=0;
        temp>>=1;
        if(SDA)
        temp|=0x80;
        SCK=1;
    }
    RST=0;  _nop_();
    SCK=0;  _nop_();
    SCK=1;  _nop_();
    SDA=0;  _nop_();
    SDA=1;  _nop_();
```

```
    return (temp);
}
```

代码清单：读 RTC 信息

```
/*****************************************************************************************
功能描述：读 RTC 函数
入口参数：无
返回值：无
*****************************************************************************************/
void Read_RTC(void)
{
    static u8 dat;

    //读取秒值
    dat = Read_Ds1302_Byte(0x81);
    second =  dat/16*10+dat%16;
    //读取分钟值
    dat = Read_Ds1302_Byte(0x83);
    minute =  dat/16*10+dat%16;
    //读取小时值
    dat = Read_Ds1302_Byte(0x85);
    hour =  dat/16*10+dat%16;

//  //读取日
//  dat = Read_Ds1302_Byte(0x87);
//  //读取月
//  dat = Read_Ds1302_Byte(0x89);
//  //读取星期
//  dat = Read_Ds1302_Byte(0x8B);
//  //读取年
//  dat = Read_Ds1302_Byte(0x8D);
//  //读取
//  dat = Read_Ds1302_Byte(0x8F);
}
```

代码清单：写 RTC 信息

```
/*****************************************************************************************
功能描述：写 RTC 函数
入口参数：无
返回值：无
*****************************************************************************************/
```

```
void  Write_RTC(void)
{
    Write_Ds1302_Byte(0x8E,0x00);

    //写秒值
    Write_Ds1302_Byte(0x80,(second/10<<4)|(second%10));
    //写分钟值
    Write_Ds1302_Byte(0x82,(minute/10<<4)|(minute%10));
    //写小时值
    Write_Ds1302_Byte(0x84,(hour/10<<4)|(hour%10));

    Write_Ds1302_Byte(0x8E,0x80);
}
```

代码清单：RTC 初始化

```
/*********************************************************************************************
功能描述：RTC 初始化
入口参数：无
返回值：无
*********************************************************************************************/
void DS1302_Init(void)
{
    static u8  ErrFlag;               //错误标志位
    ErrFlag = 0;
    Read_RTC();                      //读 DS1302 时钟值，即读出时、分、秒

    if(second >= 60) ErrFlag = 1;    //读出的秒超出范围，判断出错
    if(minute >= 60) ErrFlag = 1;    //读出的分超出范围，判断出错
    if(hour   >= 24) ErrFlag = 1;    //读出的时超出范围，判断出错
    if(ErrFlag)                      //一旦读出数据出错，重新赋时钟值，默认 11:59:30
    {
        second = 30;
        minute = 59;
        hour  = 11;
        Write_RTC();             //把新的时钟值写进 DS1302 模块
    }
}
```

编写好 DS1302 相关函数后，我们就可以实时读取 RTC 信息并在数码管显示，具体代码如下。

代码清单：主函数

```
/*******************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*******************************************************************************/
int main(void)
{
    static u8 temp[8];

    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
    P1M1 &= 0x77;   P1M0 &= 0x77;     //设置 P1.3，P1.7 为准双向口
    P2M1 &= 0xF7;   P2M0 &= 0xF7;     //设置 P2.3 为准双向口

    SEG_off();                  //控制 8 位数码管/点阵不显示
    leds_off();                 //熄灭 D1~D8 指示灯
    ULN2003_off();              //控制步进电机、蜂鸣器、继电器等不工作
    delay_ms(10);               //延时

    timer2_init();              //定时器 2 初始化
    timer2_start();             //启动定时器 2
    EA = 1;                     //使能总中断
    delay_ms(10);
    DS1302_Init();              //DS1302 时钟初始化

    while(1)
    {
        memset(temp, 0, 8);         //将 temp 数组初始化（清零）
        Read_RTC();                 //读 DS1302 时钟值，即读出时、分、秒

        if(hour >= 10)   temp[0] = hour / 10;
        else            temp[0] = 0;
        temp[1] = hour % 10;
        temp[2] = 16;
        temp[3] = minute / 10;
        temp[4] = minute % 10;
        temp[5] = 16;
        temp[6] = second / 10;
        temp[7] = second % 10;

        //数码管显示时钟信息
        LEDseg_DispUpdata(LEDSEG_1,temp[0],LEDSEG_DP_OFF);//更新第 1 个数码管显示内容
```

```
43.         LEDseg_DispUpdata(LEDSEG_2,temp[1],LEDSEG_DP_OFF);//更新第 2 个数码管显示内容
44.         LEDseg_DispUpdata(LEDSEG_3,temp[2],LEDSEG_DP_OFF);//更新第 3 个数码管显示内容
45.         LEDseg_DispUpdata(LEDSEG_4,temp[3],LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
46.         LEDseg_DispUpdata(LEDSEG_5,temp[4],LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
47.         LEDseg_DispUpdata(LEDSEG_6,temp[5],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
48.         LEDseg_DispUpdata(LEDSEG_7,temp[6],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
49.         LEDseg_DispUpdata(LEDSEG_8,temp[7],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
50.
51.         delay_ms(500);              //延时 500ms 读取一次时钟信息
52.     }
53.}
```

4.1.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。

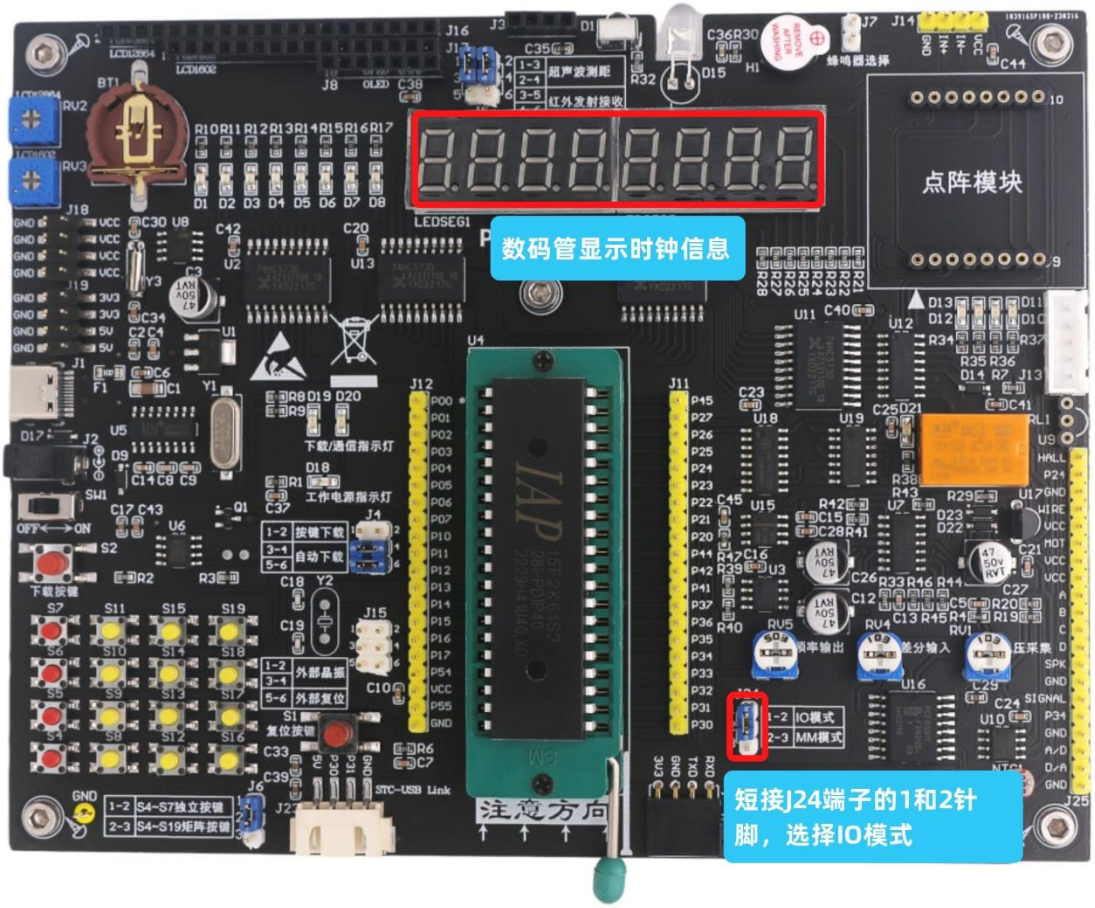


图 10：跳线帽短接

4.1.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-10-1：DS1302 时钟 -

数码管显示”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。

2) 双击“…\ds1302 \project”目录下的工程文件“ds1302.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“ds1302.hex”位于工程的“…\ds1302 \Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12 MHz。
5) 程序运行后，可在数码管上观察到实时变化的时钟信息。