串行实时时钟芯片 DS1302 程序设计中的问题与对策

作者：河南师范大学 樊贵卿李庆武靳建华 清华大学电子工程系 刘润生

来源：《电子技术应用》

摘要：指出了串行实时时钟芯片 DS1302 程序设计中几个易被疏忽而导致错误的问题，分析了问题的原因，并给出了解决问题的方法。

关键词：串行时钟程序设计问题原因解决方法

美国 Dallas 公司推出的串行接口实时时钟芯片 DS1302 可对时钟芯片备份电池进行涓流充电。由于该芯片具有体积小、功耗低、接口容易、占用 CPUI / 0 口线少等主要特点，故该芯片可作为实时时钟 广泛应用于智能化仪器仪表中。

笔者在调试中发现 在对 DS1302 编程中有几个问题易被疏忽而导致错误，现提供给读者参考。

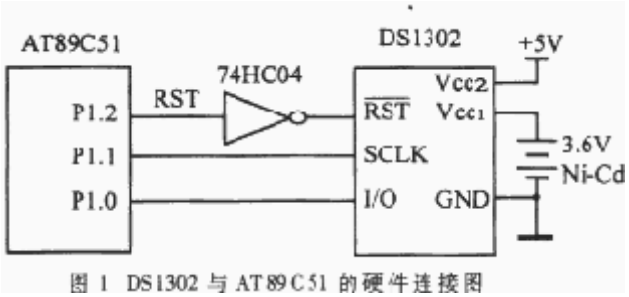

图 1 DS1302 与 AT89C51 的硬件连接图

1 读操作出现的错误

按照参考文献[2]的读操作程序框图和参考文献[1]、[2]所叙述的可知：单字节读操作每次需 16 个时钟，地址字节在前 8 个时钟周期的上升沿输入，而数据字节在后 8 个时钟周期的下降沿输出。据此结合图 1 的硬件连接图编制出了如下的单字节读程序：

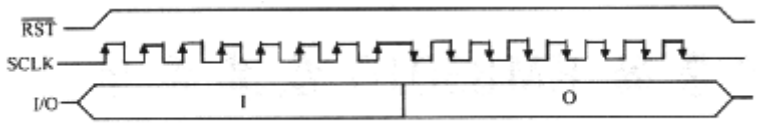

图 2 单字节读操作时序图

```
DS_READ  SETBP1.2；令=0。

CLRP1.1；令SCLK＝0。

 CLRP1.2；令=1，启动芯片。

LCALLDS_WSUB；写 8 位地址。

LCALLDS_RSUB；读出 8 位数据。

RET

DS_WSUB  MOVR7，#08H
```

ＷＬ００Ｐ　ＲＲＣＡ；Ａ为地址字节。

ＭＯＶＰ１．０，Ｃ

ＳＥＴＢＰ１．１；在时钟上升沿

ＮＯＰ；输入地址字节。

ＣＬＲＰ１．１

ＤＪＮＺＲ７　ＷＬ００Ｐ

ＲＥＴ

ＤＳ＿ＲＳＵＢ　ＳＥＴＢＰ１．０；为读数据作准备。

ＭＯＶＲ７　＃０８Ｈ

ＲＬ００Ｐ：ＳＥＴＢＰ１．１

ＮＯＰ

ＣＬＲＰ１．１；在第９个正脉冲的下

ＭＯＶＣ，Ｐ１．０；降沿开始输出数据。

ＲＲＣＡ；Ａ中为读出的数据。

ＤＪＮＺＲ７，ＲＬ００Ｐ

ＲＥＴ

若使用如下程序对 DS1302 的 RAM1　其内容为 5AH　进行读操作

ＲＥＡＤ：ＭＯＶＡ　＃１１０００１０１Ｂ；RAM1 单元的读地址。

ＬＣＡｌｌＤＳ＿ＲＥＡＤ；调用读子程序。

则程序执行后 A 中的数据为 2DH，显然读出的数据不正确。若再使用一条 RLA 指令调整后，则 A 中为 5AH，结果才正确。由此说明：使用上述程序读出的 RAM1 单元中的第 0 位数据实为第 1 位数据，读出的第 7 位数据实为第 0 位数据。

经笔者仔细研究时序图和多次试验得知，问题的原因在于：对于读操作时序，在 SCLK 出现第 8 个正脉冲时，上升沿输入地址字节的最后一位数据，而在此正脉冲的下降沿就要输出数据字节的第 0 位数据。然而笔者的程序中是在第 9 个正脉冲的下降沿才误认为输出了数据

字节的第 0 位数据，此位数据事实上是第二个下降沿输出的，故实为数据字节的第 1 位数据。经笔者实验：只要 RST 保持为高电平，如果超过 8 个下降沿，它们将重新从第 0 位输出数据位，因程序中输出的最后一位数据位，是 9 个下降沿输出的数据位，故实为数据字节的第 0 位数据位。

由此可见，单字节读操作的时序图如改为图 2 所示时序图，则读者较容易理解可避免发生上述编程错误。

只要将上述的 DS__RSUB 子程序改为如下的子程序即可解决上述问题：

```
ＤＳ＿ＲＳＵＢｌ：ＳＥＴＢＰ１．０；为读数据作准备
ＭＯＶＲ７，＃０８Ｈ
ＲＬ００Ｐ：ＣＬＲＰ１．１；ＳＣＬＫ第８个正脉冲的
ＭＯＶＣ，Ｐ１．０；下降沿开始输出数据。
ＲＡＣ
ＳＥＴＢＰ１．１
ＤＪＮＺＲ７，ＲＬ００Ｐ
ＲＥＴ
```

2 禁止涓流充电出现的错误

涓流充电寄存器(TCR)控制着 DS1302 的涓流充电特性。据参考文献[1]、[2]介绍，寄存器的位(TCS)4～7 决定着是否具备充电性能。仅在 1010 编码的条件下才具备充电性能，其它编码组合不允许充电。位 2 和 3(DS)则在和之间选择是一个还是两个二极管串入其中。如果编码是 01，选择一个二极管；如果编码是 10，选择两个；其它编码将禁止充电。该寄存器的 0 和 1 位(RS)用于选择与二极管相串联的电阻值，其中编码 01 为 2kΩ；10 为 4kΩ；11 为 8kΩ；而 00 将不允许充电。笔者编制了如下的允许涓流充电的控制程序（选择一个二极管，充电限流电阻为 4kΩ）：

```
ＳＥＴＢＰ１．２；令=0
ＣＬＲＰ１．２；令ＳＣＬＫ＝０
ＣＬＲＰ１．２；令=1
ＭＯＶＡ　＃９０Ｈ；ＴＣＲ的写地址
ＬＣＡＬＬＤＳ＿ＷＳＵＢ
```

ＭＯＶＡ　＃１０１００１１０Ｂ；ＴＣＲ的命令

ＬＣＡＬＬＤＳ＿ＷＳＵＢ

用万用表串入与可充电池之间，执行程序后，则有电流流过万用表，表示充电正常。笔者通过将上述程序的第 6 句改为：MOVA，＃10100010B，即置 DS 为 00 来禁止涓流充电器工作。执行程序后，在与电池之间串入万用表，则仍有电流流过，表示尚未禁止充电。若将第 6 语句改为：MOV A，＃10101110B，即置 DS 为 11，执行上述程序后情况仍如此。若将第 6 语句改为：

MOV A，＃01010110B 即 TCS≠1010

或：MOV A，＃10100100B 即 RS=00 则充电被禁止。

笔者误认为芯片损坏，换上另一新购置的芯片，结果仍如此。随即笔者取下图 1 所示电路中的可充电池，换上一标称为 10kΩ的电阻对芯片进行了测试，测试结果如表 1 所示=5V。

表 1

命令（TRC）	10100101B 10100001B 10101101B	10100110B 10100010B 10101110B	10100111B 10100011B 10101111B
V_{ca}(V)	3.68	3.25	2.63
命令（TRC）	10101001B	10101010B	10101011B
V_{ca}(V)	3.14	2.77	2.44

由此可见，当涓流充电控制寄存器中的 DS 位为 00 和 11 时并不能禁止充电，而是选择了一个二极管充电，这说明参考文献中介绍的有误。若要想禁止充电器充电，应将第 6 句改为：MOVA，＃0101XX00B 即 TCS≠1010，RS=00，这样，就能双保险地禁止充电。

3 受干扰时钟／日历信息出现的错误

笔者将 DS1302 应用于某产品中，发现系统受到干扰时，有时其时钟停振不能正常工作，此时的时钟／日历信息也被修改。

经分析得知：系统受到干扰程序飞跑，在看门狗复位前，CPU 正好执行写程序将写保护寄存器的最高位置 0 为允许写（实际上，在系统校时程序之后已将其置为 1 禁止写），修改了时钟／日历信息且使秒寄存器的最高位置 1，致使时钟停振出现错误。

为避免此类错误的产生，笔者采用的方法是：在写程序中增加了某一检测条件，此条件为系统中某一口线上的电平，低电平条件满足。只有在实时校时过程中，才通过手动使此口线为低电平，实时校时过程完成后，又通过手动使此口线为高电平。这样只有实时校时过程中，才允许修改时钟／日历信息，因此起到了时钟／日历信息的写保护作用。