作 者
邹士洪

存储器24C08的串口读取与写入程序

24C08是一种采用I²C总线结构的小容量集成存储电路，多见于大屏幕彩电、手机电脑等电子电路之中。新型的电子产品集成度越来越高，这在实际使用中大大降低了硬件电路故障产生的几率。质量较好的集成电子产品，无故障使用时间都可以达到20000小时以上。所以此类电子产品的故障多集中发生于非关键性元件或者软件及系统设置参数之中。24C08就是一种主要用于存储系统设置参数的集成存储电路。由于现在的电子产品大多使用了计算机技术，通电初期主控电路需要获得一些系统初始数据（开机时间、整机电路配置等一些系统参数），而这些参数的设置起来较为专业，也很繁琐，不是每个人都能做好的，所以就有了这样一类存储器电路。在电子产品的调试过程中可以将系统所需参量集中保存写入存储器件之中，以后每次开机时都可以直接调用。这样不但避免了因为设置出现偏差可能导致的产品无法使用，也可免去每次开机重新进行设置的麻烦。这种类型的存储器容量一般都不大，只是用来储存系统正常工作所需的一些较为重要的参量。一件产品使用年头久了，储存数据容易出现问题，这时整机电路就会出现功能异常，或者完全不能工作。这样就需要我们对检查储存器数据进行修正，或者重新写入了。下面就以24C08为例介绍一下此类I²C总线存储器数据读取和写入的一般方法。图1即是这种读写工具的

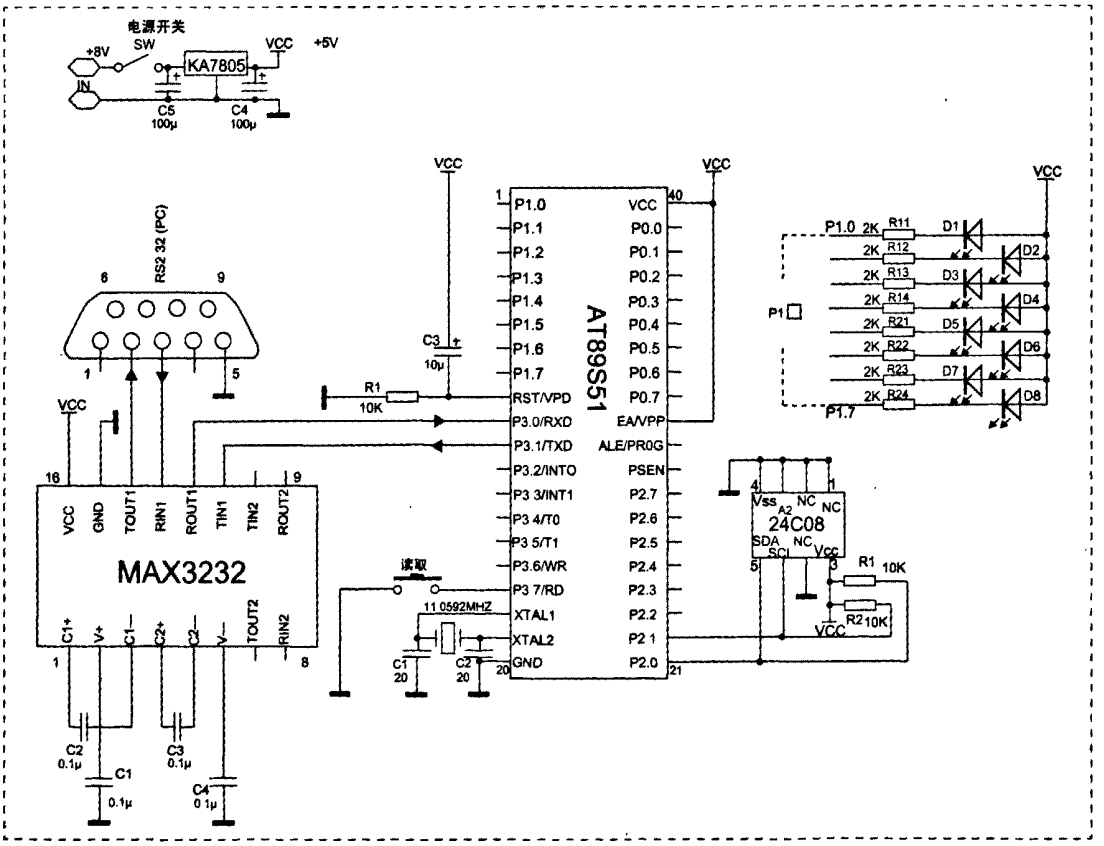

图1

实际电路构成 。图 2 是 24C08A 的引脚图和外观图。

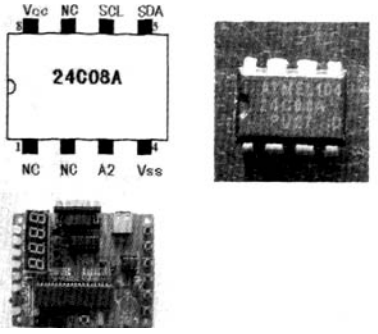

图 2

这种 24C08 的数据读写工具由单片机主控电路与电脑两个相对独立的部分组合而成。做为一个功能齐全的存储器读写电路，不能没有一个可视化的介面，所以电脑在这里最大的功能就是起到直观显示编辑 24C08 数据的作用。单片机依然使用比较常见的 51 系列中的 89S51。将编制好的读写程序写入内部便好了。在电脑一方，需要一种功能全面的串口调试软件配合，这里使用 PORTTEST 这种串口调试工具。单片机与电脑之间的联机通讯通过 RS232 串口完成。RS232 接口芯片采用 MAX3232,MAX3232 的最大数据传输率低于 1Mbps，而 24C08 的总线最高时钟频率 400KHZ，所以双方的传输速率还是很相称的。这里采用 RS232 通讯方式中最简单的发送、接收、地端三线传输的方式。

单片机 89S51 内部具有一个全双工异步通讯控制器。所谓的全双工就是说通讯双方的发送、接收两种工作可以同时进行，由系统内部电路自动进行协调，可以做到互不干扰。

由于 RS232 通讯多数情况下不采取同步传输的方式，而是由通讯双方事先对数据格式以及传输速率这两种异步传输必要的两种参量进行约定，在实际通讯过程中双方依约定各自产生相应的时钟信号进行同步，来实现数据的识别。应用中，双方的时钟将不可能做到绝对的统一，总是会有一定差异，主要包括中心频率的差异与时钟频率的漂移。电子振荡器的频率虽然非常非常稳定，但是随环境气温等变化会产生一定波动。这种频率波动极为微小，而且仅是围绕中心频率的窄幅波动，所以双方时钟自身的频率漂移对异步传输的影响可谓微乎其微，完全可以忽略不计的。但是这两种时钟中心频率的差异是固定不变的。虽然在异步通讯的初始化期间两种时钟被同步，但是随时间推移它们的相位差又会逐步拉大，达到一定程度，必然会导致双方的数据传输出现混乱。那时，传递的信息将不能正确识别。所以 RS232 异步通讯通常采用一种 10 位异步传输的方式。这是一种单字节数据传输方式，即由数据发送方先送出一位起始信号、随后发送 8 位 1 个字节的数据、完成后立即发送一位停止信号，以这样一个过程做为一次完整的异步数据传输。当发送下一位数据时，重新进行启动同步。这样的做法，将把双方时钟差异限定在十个时钟周期之内，只要双方时钟误差保持在 5% 以内，数据传输基本上不会出错。付出的代价是传输速率降低。但是用在对传输速率要求不高，且数据量不大的场合还是值得的。

由于标准的 RS232 电平是将 (- 3) - (- 15v) 的电压表示为高电平，3 - 15v 的电压作为低电平的，所以电脑的串口都采用 12V 的供电电压，而单片机都是使用 5V 乃至更低的工作电压，只能发出 TTL 或 CMOS 逻辑电平。这样它们之间就还需要一种接口电路来承担逻辑电压转换工作，同时要保证不会影响到异步传输的速度与质量。集成电路 MAX3232 是专用的 RS232 电平转换电路，可以很好地完成这一任务。它与 RS232 在引脚排列与功能方面完全兼容，只是外围元件参数稍有差异。

I²C 总线存储器的基本读写程式可参考本刊七期中《I²C 总线存储器的数据复制》一文。24C08 的存储容量为 8Kbit,1024 字节，这里对它的数据读取采用了一种连续的方式。24C08 这种数据读写模式可以简要概括如图 3 所示。

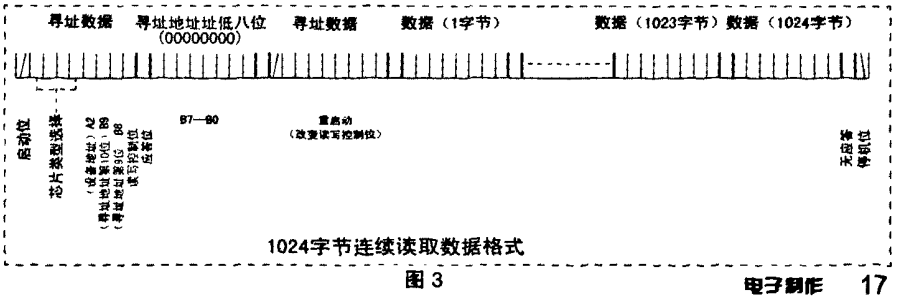

图 3

24C08 可以储存 1024 个字节的数据，1024 个字节的寻址编码要用 10 位的二进制数据来表示，所以对 24C08 的读取与 24C02 有些不同，它的寻址地址超过 8 位，所以须分两次发出，10 位二进制地址的最高两位 B9、B8 紧随芯片选择位 A2（A2 与芯片第 3 脚相对应）发出，剩余的八位地址数据做为一个完整的字节，在 24C08 对单片机指令做出响应后，随后发出。24C08 读取到两字节数据后，内部相应地址储存的数据将在串行时钟配合下，移位串行从 SDA 引脚输出。此类存储器的读取都比较简单，采用连续的方式可以提高程序执行的效率。在连续读取模式下，单片机完整读到每一字节数据后必须做出应答，由 24C08 识别后，控制内部地址存储器自动加 1，下一地址所储存的数据就会自动移位输出。不断重复这一过程，直至单片机停止回应，并发出停机信号，24C08 进入待机，单片机也就完成了对 24C08 内部数据的读取工作。在对 24C08 进行数据读取时，如果没有发出寻址地址数据，24C08 将始终从上一次读写结束后内部地址寄存器保留的地址数据开始进行读取。

由于数据写入需要的时间要比读取所用的时间长得多，所以大多存储电路的写入方式相对读取也都要复杂得多，24C08 也不例外。它的单字节读写原则与 24C02 一致，只是由于容量不同，24C08 寻址位要多出两位，这里就不再重述了。为了提高程序效率，这里对 24C08 的数据写入也是采用的连续写入方式。连续写入的方式与连续读取的方式截然不同。在这种连续写入方式下，24C08 只能一次性写入长度不超过 16 字节的数据；而 24C08 的连续读取却是没有任何限制的，可以无限地循环下去。

图 4 为 24C08 的这种数据页写入格式的详图。在对 24C08 进行数据写入时，寻址数据是必须的，24C08 的数据写入是一个相对独立的过程。24C08 内部具有独立的数据寄存器电路，在连续写入过程中，24C08 将把接收的字节数据依次序先存在这些寄存器中，如果发送方发出的数据长度超过了 16 字节，16 位寄存器将产生循环移位，先期发出的数据将被复写，直至收到发送方传来的停机信号，接收完成，最后接收的 16 字节数据被保留，之后 24C08 正式进入真正的写入过程。此时外部总线上任何的电压变化都不会得到响应，包括单片机发出的标准总线起动信号。具体实现方法就是使 SDA 线维持高电平，使单片机得不到应答进入等待状态，直至 24C08 完成数据的写入工作。所以单片机起动 I2C 总线通讯过程时，总要以 24C08 是否做出应答做为进入下一步骤的依据，而进入了正式读写过程后便不需要如此了。16 字节数据的写入周期，大约要占用 10ms 的时间。

24C08 各引脚功能描述如下，它的 1、2、7 脚都是空脚，使用中一般都直接接地。3 脚是 24C08 唯一的一位芯片选择位，所以在同一条 I²C 总线上只可以同时挂接两片 24C08 存储器。24C08 的 4、8 两脚为芯片的供电端。采用 5V 的工作电压时，串行时钟的频率最高可以达到 400KHZ，在实际使用中，为了保证 24C08 的稳定工作，通常要通过在汇编程序中插入空操作这种做法，来降低同步时钟频率，以保证与 24C08 的时钟频率相匹配。5、6 两脚分别为 24C08 的 SDA、SCL 两端，这是 I2C 总线中仅须的两条数据通道。一般使用中都是需要通过外接两只上拉电阻来维持两脚的静态输出为高电平。89S51 的异步通讯接口虽然是全双工的，由于使用简单的三线通讯方式，要注意做好双方传输速

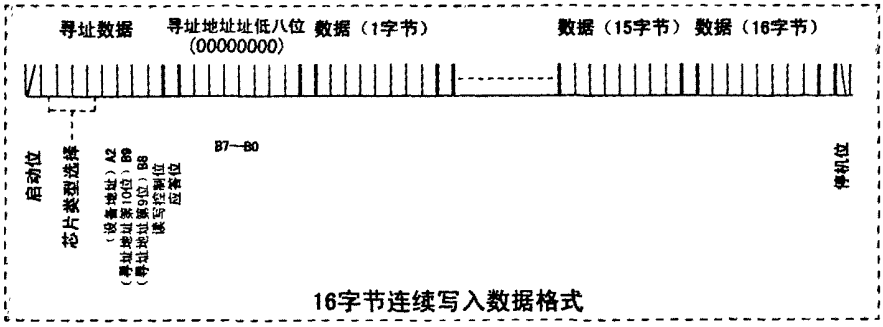

图 4

度的协调工作。单片机采用 11.0592MHz 的晶振，它的指令周期约为 1 微秒。然而单片机在异步接收的同时还要承担将电脑传输的数据写入 24C08 中这个任务。这种简单的三线异步通讯，电脑的数据发送将严格按照约定进行，而不理会单片机对 24C08 的数据写入是否已经完成。芯片内部 RS232 接口对数据的接收自动完成的，但不会将接收的数据主动传送给单片机，在实际读取过程中，程序必须预留适当的时间来完成对异步接收标志位 RI 的检测与控制。如果写入耗费的时间太多，则单片机 RS232 接口收到的数据将得不到及时处理，而接收控制位 RI 不能及时清零，电脑侧发送的数据也将会丢失。所以异步传输的速率设置就比较重要了，设置快了，必将导致双方 RS232 接口数据传送不能正常进行和 24C08 数据写入出错。那选择多少的传输速率才算合适 呢？单片机主要是通过串口控制寄存器 SCON 的接收中断位 RI 的指示来完成数据接收的。在单片机 RS232 接口接收完一位完整的 1 字节数据后，接口自动将 RI 置位，指示单片机对接收数据进行处理并要求单片机及时清除 RI 位，以便 RS232 接口可以进入下一次数据接收过程。其中的关键已经很明显了，就是要求单片机必须在两次单字节异步数据传输的中间时间及时完成对 24C08 的 16 字节数据写入工作，并转入对 RI 信号的监测工作。综合以上个因素和实际调试结果，将 RS232 的传输速率设为 9600bps 比较恰当。

在 24C08 的数据读取的过程中，由于单片机为数据发送一方，所以占据绝对的主导地位，而且电脑的速度又要远远超过单片机，串口调试软件对单片机的响应速度可以完全不用考虑。只要程序没有问题，24C08 的数据读取必定不会出错。

表 1 为 89S51 异步通讯接口常用几种通讯方式下的参量设置：

表 1

波特率	晶振频率(MHz)	串行模式(SMOD)	TH1预置初值
19200	11.0592	1	FDH
9600	11.0592	0	FDH
4800	11.0592	0	FAH
2400	11.0592	0	F4h
1200	11.0592	0	E8h

24C08 的数据连续读取及 1024 字节写入的完整汇编程序如下：

```
;==============================
sda             EQU   p2.0
scl             EQU   p2.1
address1        EQU   08H
address2        EQU   09H
sbit                p37=p3^7
sbit            p36=p3^6
org           0000H
ljmp          start
;========总线启动
i2c_start:
    setb      scl
    setb      sda
    nop
    nop
    clr       sda
    nop
    nop
    clr       scl
ret
;========总线停止
i2c_stop:
    clr       sda
    nop
    nop
    setb      scl
    nop
    nop
    setb      sda
ret
;========存储器应答
i2c_ack:
    setb      sda
    nop
    nop
    setb      scl
    Jb        sda,i2c_ack0
    clr       c
    sjmp      i2c_ack_end
```

```
    i2c_ack0:
      setb    c
    i2c_ack_end:
      nop
      nop
      clr     scl
ret
;======== 单片机应答
mi2c_ack:
      nop
      nop
      clr     sda
      nop
      nop
      setb    scl
      nop
      nop
      clr     scl
ret
;========= 发送八位数据
i2c_send:
      mov     b,#08H
i2c_send1:
      rlc     a
      mov     sda,c
      setb    scl
      nop
      nop
      clr     scl
      djnz    b,i2c_send1
ret
;========= 接收八位数据
i2c_receive:
      mov     b,#08H
      clr     a
      setb    sda
i2c_receive_a:
      setb    scl
      nop
      nop
      mov     c,sda
      rlc     a
      clr     scl
      djnz    b,i2c_receive_a
ret
;============
delay:
      mov     r1,#255
      djnz    r1,$
ret
;=========16 字节页连续写入程序

write:
      MOV     A,address1
      lcall   i2c_start
      lcall   i2c_send
      lcall   i2c_ack
      Jc      write
      mov     A,address2
      lcall   i2c_send
      lcall   i2c_ack
      mov     r2,#10h
      write_b:
      jnb     rl,$
      clr     rl
      mov     a,sbuf
      mov     p1,a
      lcall   i2c_send
      lcall   i2c_ack
      djnz    r2, write_b
      lcall   i2c_stop
ret
;========== 主程序
start:
      mov     SP,#60H
      mov     p0,#255
      mov     p1,#255
      mov     p2,#255
      mov     p3,#255
    mwrite:
      mov     TMOD,#20h
;T1: 工作模式 2
      mov     PCON,#00h
      mov     TH1,#0FDH       ; 初始化
```

```
    9600kbps 波特率
        mov     SCON,#40h           ;Standard   settings
UART settings
        SETB    REN                 ; 允许接收
        SETB    TR1                 ;T1 开始工作
        mov     address1,#10100000b
        mov     address2,#00000000b
       wri:
        lcall   write
        mov     a,address2
        add     a,#10h
        mov     address2,a
        jc      s
        mov     a,address1
        cjnE    a,#10101000b,wri
        clr     TR1
        ljmp    wa
       s:
        mov     a,address1
        add     a,#10b
        mov     address1,a
        mov     a,address1
        cjnE    a,#10101000b,wri
        clr     TR1
       wa:
        jb      p37,$
        call    delay
        jnb     p37,$
       read:
        MOV     A,#10100000b
        lcall   i2c_start
        lcall   i2c_send
        lcall       i2c_ack
        Jc          read
        mov         A,#00000000b
        lcall       i2c_send
        lcall       i2c_ack
        mov     TMOD,#20h       ;T1: 工作模式 2
        mov     PCON,#00h
        mov     TH1,#0FDH       ; 初始化 9600kbps
波特率
        mov     SCON,#50h       ;Standard UART
        SETB    REN             ; 允许接收
        SETB    TR1             ;T1 开始工作
        setb    TJ
       read_a:
        lcall   i2c_start
        mov         a,#10100001b
        lcall   i2c_send
        lcall       i2c_ack
        Jc          read_a
        mov     r1,#04h
        mov     r2,#00h
       read_b:
        lcall       i2c_receive
        lcall       mi2c_ack
        jnb     TI,$
        clr     TI
        mov     sbuf,a
        mov     p1,a
        djnz    r2, read_b
        djnz    r1, read_b
        lcall       i2c_stop
        clr     TR1
        ljmp    mwrite
        END
```

将程序编译写入单片机之后，就可以使用了。在启动电脑侧串口通讯软件之前，先要对通讯方式与速度进行正确地配置，如图 5 所示。这里使用的是 COM1 口，具体要根据实际使用情况进行配置，无论使用哪一串口，都不会到影响程序的执行效果。程序的执行流程如下：首先电脑要通过串口依次将 1024 字节数据发送到单片机之中。程序的启动是由电脑软件控制的，POSTTEST 软件首先执行串口发送程序。单片机串口接收到一字节数据后，立刻将接收中断位 RI 置位，单片机以此为据取出收到的数据并进入对 24C08 的写入过程。循环往复直至 1024 字节数据写入工作完成为止。然后进入等待，等到 P37 “读取” 按键被按动一次，便开始执行读取程序。将 24C08 内部数据读出后，及时通过串口发送到电脑软件之中，

（下转 58 页）

```
        J=0;
        PORTB=0xAA; // PORTB 口外
LED 全灭
 PORTC=0x00; // PORTC 口外接 LED 全灭
        delay(400); //调延时函数
        RB6=1;      //圆形红灯亮
        RB7=0;      //禁止机动车通行
        RB0=1;      //手掌形红灯亮
        RB1=0;      //禁止行人通行
        delay(3300000); //调延时函数
        }
        }
  void delay(unsigned long int K) //带形参数
        {                    //延时函数
        unsigned  long int d=K;     //说明语句
        while(--d)                  //执行语句
                {;
                }
                }
```

说明：上述交通路口红、绿灯 C 程序，是笔者观察到交通路口的红绿灯图案，编辑该程序的目的，是帮助大家学习 C 语言程序。对于程序中的延时函数值，读者可以根据需要进行调整，以将其用于其他 C 程序中，作为可任意改变时间的延时函数值。

需要说明的是，实际的交通路口红、绿灯管理系统，是一种基于多路口的红、绿、黄灯工程管理体系，考虑的问题比本文中的程序更多、更复杂。

（上接 21 页）

显示于 POSTTEST 的接收方框中。在读取、写入两种过程中的操作数据都会通过单片机 P1 接口的八只发光管动态地显示出来。一次写入、读取为一个完整的流程，完成后程序将返回起点，等待下一次写入、读取的过程，如图 7 所示。

图 5

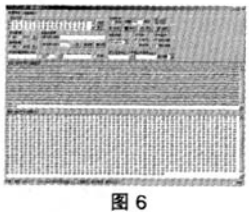

图 6

现在的串口软件种类较多，随便哪一种都.可以拿来使用。将 1024 字节 hex 数据粘贴到软件的数据发送窗口，按动发送键，程序即可启动。串口软件数据传递要选择 HEX、ASCII 中的 HEX 类型。

程序执行完毕后，要对写入结果作一下校验。所有的结果都将显示在软件主窗口中。效验必须通过以下几项：1、实际发送、接收的位数应该做到完全一致，软件窗口的右上方显示了实际接收、发送完成的次数。2、是对读写数据的校验。在这一过程中，不需要对所有的数据都进行比对，只要首尾各 16 字节的数据没有错误，中间数据便不会出错。两项检查都通过了，则 24C08 写入完成，写入数据可用。否则重写一次即可。

各种 I2C 总线存储器的连续读取的原则基本一致，差异仅在设备地址以及寻址地址的格式有些不同，所以以上程序中的读取部分仅须改动 1、2 处数据便可以做到通用。而各种不同容量的 I2C 总线存储器写入方法复杂多样，耗费时间也有差别，写入程序要应用到其它器件中需要作较大改动，仔细参考它们各自的技术文档。

电脑串口带电插拔容易损坏，所以编程器的通电必须严格按顺序进行。使用之前，先不要接通单片机的电源，首先做好通电使用中的电脑主机与编程器 RS232 接口的连接，然后才可以接通编程器一方的电源。

整个电路取材不多电路也算不上复杂，但是个人制作起来依然是有一定难度。现在市面上有不少编程器散装套件出售，所以可以直接购置套件自行焊接组装。直接使用支持 MCS—51 系列单片机语言的成品编程器更好，根据 I2C 与 RS232 接口的实际布设将程序稍加改动，就可以将程序移植到各种同类型编程器之中。

存储器24C08的串口读取与写入程序

作者： 邹士洪
作者单位：
刊名： 电子制作
英文刊名： ELECTRONICS DIY
年，卷(期)： 2010(10)