好好学习天天向上

I²C总线的结构、工作时序与模拟编程

I²C总线（Inter Integrated Circuit）是飞利浦公司于上个世纪 80 年代开发的一种“电路板级”的总线结构。与其它串行接口相比，无论从硬件结构、组网方式、软件编程都有很大的不同。在AT89C51 系统上使用汇编语言模拟I²C总线的各种信号及编程原理，为自主开发、设计具有I²C总线接口的系统打下一个良好的基础，也为其它串口的模拟编程创造一个好的思路和可行的方法。

I²C总线的主要特点

1. 二线制结构。即双向的串行数据线 SDA、串行同步时钟线 SCL。总线上的所有器件其同名端都分别挂在 SDA、SCL 线上（见图 7.1）；
2. I²C总线所有器件的SDA、SCL引脚的输出驱动都为漏极开路结构（见图 7.2），通过外接上拉电阻将总线上所有节点的SDA、SCL信号电平实现“线与”的逻辑关系。这不仅可以将多个节点器件按同名端引脚直接挂在SDA、SCL线上，还使I²C总线具备了“时钟同步”、确保不同工作速度的器件同步工作；
3. 系统中的所有外围器件都具有一个 7 位的“从器件专用地址码”,其中高 4 位为器件类型地址（由生产厂家制定），低 3 位为器件引脚定义地址（由使用者定义），主控器件通过地址码建立多机通信的机制。因此I²C总线省去了外围器件的片选线，这样无论总线上挂接多少器件，其系统仍然为简约的二线结构；
4. I²C总线上的所有器件都具有“自动应答”功能，保证了数据交换的正确性；
5. I²C总线系统具有“时钟同步”功能。利用SCL线的“线与”逻辑协调不同器件之间的速度问题；
6. 在I²C总线系统中可以实现“多主机（主控器）”结构。依靠“总线仲裁”机制确保系统中任何一个主控器都可以掌握总线的控制权。任何一个主控器之间没有优先级，没有中心主机的特权。当多主机竞争总线时，依靠主控器对其SDA信号的“线与”逻辑，自动实现“总线仲裁”功能；
7. I²C总线系统中的主控器必须是带CPU的逻辑模块；而被控器可以是无CPU的普通外围器件，也可以是具有CPU的逻辑模块。主控器与被控器的区别在于SCL的发送权，即对总线的控制权；
8. I²C总线不仅广泛应用于电路板级的“内部通信”场合，还可以通过I²C总线驱动器进行不

同系统间的通信；

9. I²C总线的工作速度分为 3 种版本：S（标准模式），速率为 100kb/s。主要用于简单的检测与控制场合；F（快速模式），速率为 400kb/s；Hs（高速模式），速率为 3.4Mb/s。

I²C总线的系统和接口内部结构

2.1 I²C总线的系统结构

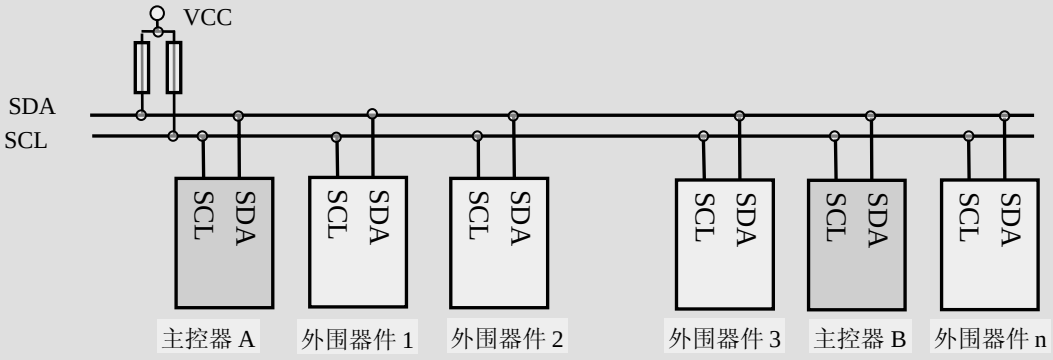

图 7.1　具有多主机的I²C总线的系统结构

2.2　I²C总线接口的内部结构

每一个I²C总线器件内部的SDA、SCL引脚电路结构都是一样的，引脚的输出驱动与输入缓冲连在一起。其中输出为漏极开路的场效应管、输入缓冲为一只高输入阻抗的同相器。这种电路具有两个特点：

①由于 SDA、SCL 为漏极开路结构，借助于外部的上拉电阻实现了信号的“线与”逻辑；

②引脚在输出信号的同时还将引脚上的电平进行检测，检测是否与刚才输出一致。为 “时钟同步”和“总线仲裁”提供硬件基础。

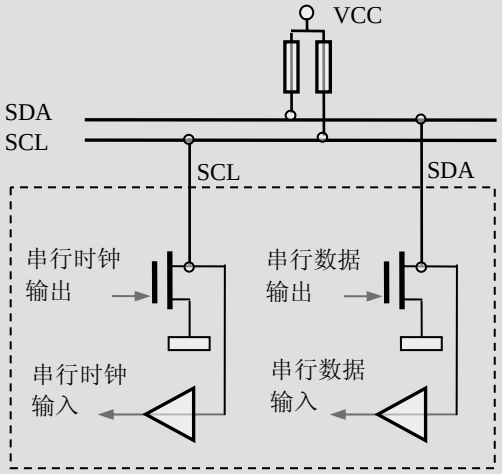

图 7.2　I²C总线接口内部结构

I²C总线的工作过程与原理

总线上的所有通信都是由主控器引发的。在一次通信中，主控器与被控器总是在扮演着两种不同的角色。

3.1 主控制器向被控器发送数据

操作过程如下：

（1）主控器在检测到总线为“空闲状态”（即 SDA、SCL 线均为高电平）时，发送一个启动信号“S”，开始一次通信的开始；

（2）主控器接着发送一个命令字节。该字节由 7 位的外围器件地址和 1 位读写控制位 R/W 组成（此时 R/W=0）；

（3）相对应的被控器收到命令字节后向主控器回馈应答信号 ACK（ACK=0）；

（4）主控器收到被控器的应答信号后开始发送第一个字节的数据；

（5）被控器收到数据后返回一个应答信号 ACK；

（6）主控器收到应答信号后再发送下一个数据字节 … …

（7）当主控器发送最后一个数据字节并收到被控器的 ACK 后，通过向被控器发送一个停止信号 P 结束本次通信并释放总线。被控器收到 P 信号后也退出与主控器之间的通信。

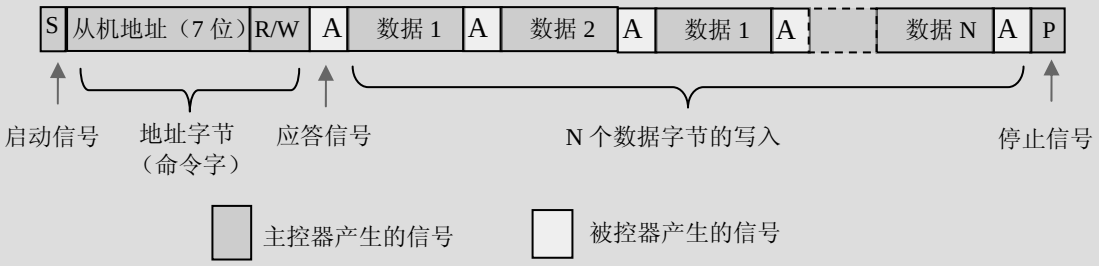

图 7.3 主控器向被控器写 N 个数据的过程

需要说明的是：①主控器通过发送地址码与对应的被控器建立了通信关系，而挂接在总线上的其它被控器虽然同时也收到了地址码，但因为与其自身的地址不相符合，因此提前退出与主控器的通信；②主控器的一次发送通信，其发送的数据数量不受限制。主控器是通过 P 信号通知发送的结束，被控器收到 P 信号后退出本次通信；③主机的每一次发送后都是通过被控器的 ACK 信号了解被控器的接收状况，如果应答错误则重发。

3.2 主控器接收数据的过程

过程简述如下：

（1）主机发送启动信号后，接着发送命令字节（其中 R/W=1）；

（2）对应的被控器收到地址字节后，返回一个应答信号并向主控器发送数据；

（3）主控器收到数据后向被控器反馈一个应答信号；

（4）被控器收到应答信号后再向主控器发送下一个数据 … …；

（5）当主机完成接收数据后，向被控器发送一个“非应答信号（ACK=1）”，被控器收到ASK=1 的非应答信号后便停止发送；

（6）主机发送非应答信号后，再发送一个停止信号，释放总线结束通信。

主控器所接收数据的数量是由主控器自身决定，当发送“非应答信号/A”时被控器便结束传送并释放总线（非应答信号的两个作用：前一个数据接收成功，停止从机的再次发送）。

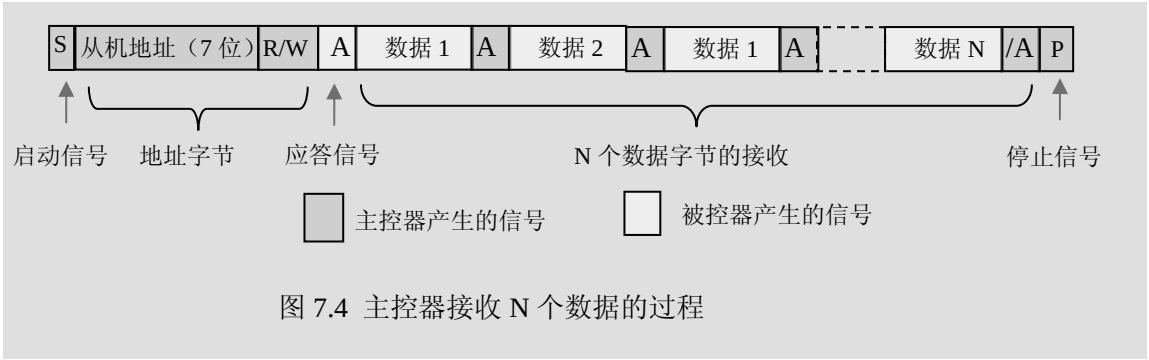

图 7.4 主控器接收 N 个数据的过程

I²C总线的信号时序

以主控器向被控器发送一个字节的数据（写操作 R/W=0）为例。整个过程由主控器发送起始信号 S 开始，紧跟着发送一个字节的命令字（7 位地址和一个方向位 R/W=0），得到被控器的应答信号（ACK=0）后就开始按位发送一个字节的数据。得到应答后发送 P 信号，一个字节的数据传送完毕。其数据传送的时序如图 7.5 所示。

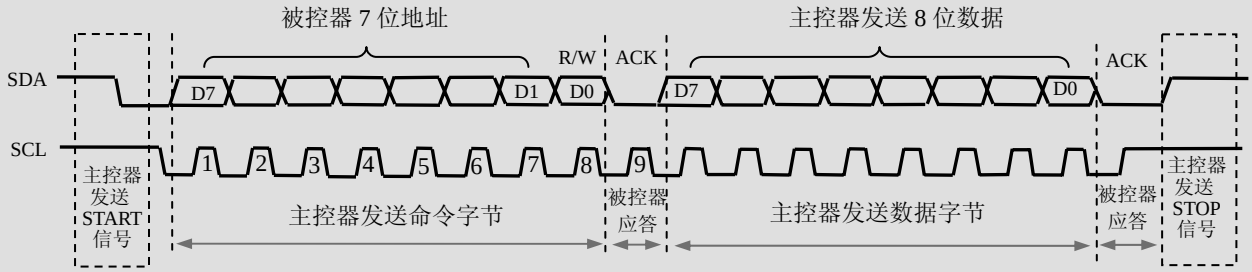

图 7.5 主控器发送一个字节数据的时序

在数据传送中数据高位(D7)在先， SDA 线上的数据在时钟脉冲 SCL 为低电平时允许变化（如图所示），在数据稳定后时钟脉冲为高电平期间传送数据有效。

主控器接收数据(R/W=1) 的时序类似于发送，主要区别有两点：①主机接收到数据后要向被控器发送应答信号（ACK=0）；②当主机接收完最后一个数据时向被控器返回一个“非应答信号/ACK=1”以通知被控器结束发送操作，最后主控器发送一位停止信号 P 并释放总线(参

见图 7.4)。这里具体的时序可以在后面的“接收子程序”中进行描述。

I²C总线的时钟同步与总线仲裁

I²C总线的SCL同步时钟脉冲一般都是由主控器发出作为串行数据的移位脉冲。每当SDA上出现一位稳定的数据后，在SCL上发送一个高电平的移位脉冲。

5.1 SCL 信号的同步

如果被控器希望主控器降低传送速度可以通过将SCL主动拉低延长其低电平时间的方法来通知主控器，当主控器在准备下一次传送发现SCL的电平被拉低时就进行等待，直至被控器完成操作并释放SCL线的控制控制权。这样以来，主控器实际上受到被控器的时钟同步控制。可见SCL线上的低电平是由时钟低电平最长的器件决定；高电平的时间由高电平时间最短的器件决定。这就是时钟同步，它解决了I²C总线的速度同步。

5.2 I²C总线上的总线仲裁

如果在同一个I²C总线系统中存有两个主控器，其时钟信号分别为SCK1、SCK2，它们都具有控制总线的能力。假设两者都开始要控制总线进行通信，由于“线与”的作用，实际的SCL的波形如图 7.6 所示。在总线做出仲裁之前，两个主控器都会以“线与”的形式共同参与SCL线的使用，速度快的主控器 1 等待落后的主控器 2（如图 7.6）。

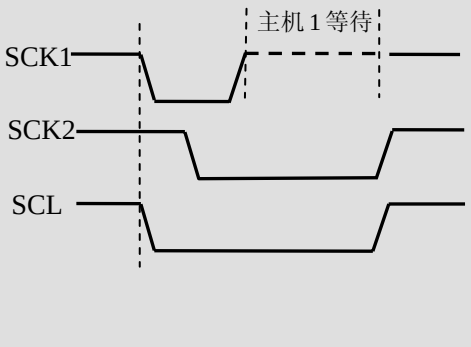

图 7.6 SCL 信号的同步

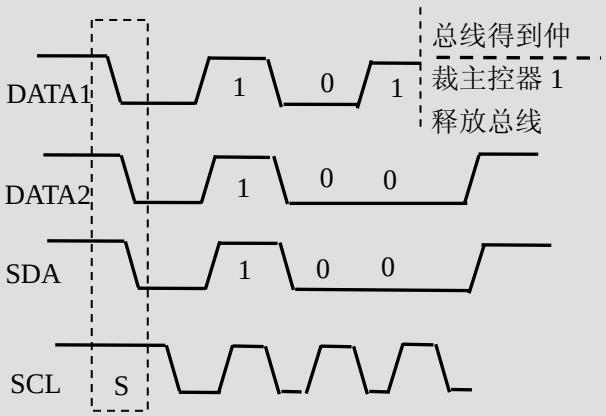

图 7.7 I²C总线上的总线仲裁时序图

对于 SDA 线上的信号的使用，两个主控器同样也是按照“线与”的逻辑来影响 SDA 上的电平变化。假设主控器 1 要发送的数据 DATA1 为“101 ……”；主控器 2 要发送的数据 DATA2 为“1001 ……”。总线被启动后两个主控器在每发送一个数据位时都要对自己的输出电平进行检测，只要检测的电平与自己发出的电平一致，他们就会继续占用总线。在这种

情况下总线还是得不到仲裁。当主控器 1 发送第 3 位数据“1”时（主控器 2 发送“0” ），由于“线与”的结果 SDA 上的电平为“0”，这样当主控器 1 检测自己的输出电平时，就会测到一个与自身不相符的“0”电平。这时主控器 1 只好放弃对总线的控制权；因此主控器 2 就成为总线的唯一主宰者。仲裁过程如图 5.2 所示。

不难看出：

① 对于整个仲裁过程主控器 1 和主控器 2 都不会丢失数据；

② 各个主控器没有对总线实施控制的优先级别；③总线控制随即而定，他们遵循“低电平优先”的原则，即谁先发送低电平谁就会掌握对总线的控制权。

根据上面的描述，“时钟同步”与“总线仲裁”可以总结如下规律：

①主控器通过检测 SCL 上的电平来调节与从器件的速度同步问题——时钟同步；

②主控器通过检测SDA上自身发送的电平来判断是否发生总线“冲突”——总线仲裁。

因此，I^2C总线的“时钟同步”与“总线仲裁”是靠器件自身接口的特殊结构得以实现的。

I^2C总线的工作时序与AT89C51 单片机的模拟编程

对于具有I^2C总线接口的高档单片机来说，整个通信的控制过程和时序都是由单片机内部的I^2C总线控制器来实现的。编程者只要将数据送到相应的缓冲器、设定好对应的控制寄存器即可实现通信的过程。对于不具备这种硬件条件的AT89C51 单片机来说只能借助软件模拟的方法实现通信的目的。软件模拟的关键是要准确把握I^2C总线的时序及各部分定时的要求。

单片机与I^2C器件的连接及引脚定义由图 7.8 所示，使用伪指令定义对I/O端口进行定义（设单片机的系统时钟fosc为 6M，即单周期指令的运行时间为 2μS）。

```
SDA     BIT     P1.0
SCL     BIT     P1.1
```

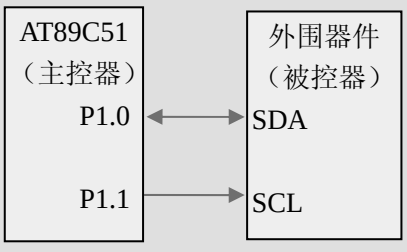

图 7.8　单片机与I^2C器件的连接

6.1 发送启动信号 S

在同步时钟线 SCL 为高电平时，数据线出现的由高到低的下降沿。

启动信号子程序 STA

```
STA：   SETB    SDA
        SETB    SCL
        NOP
        NOP             ;完成 4.7μS 定时
        CLR     SDA     ;产生启动信号
        NOP
        NOP             ;完成t HD,STA定时
        CLR     SCL
        RET
```

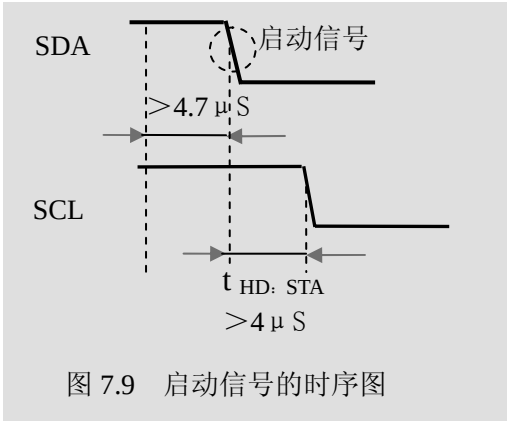

图 7.9　启动信号的时序图

【注】t HD,STA：起始信号保持时间，最小值为 4μS。在这个信号过后才可以产生第一个同步信号；

6.2 发送停止信号 P

在 SCL 为高电平期间 SDA 发生正跳变。

停止信号子程序 STOP

```
STOP：  CLR     SDA
        SETB    SCL
        NOP
        NOP             ;t SU: SOP定时
        SETB    SDA
        NOP
        NOP             ;t BUF定时
        CLR     SCL
        CLR     SDA
        RET
```

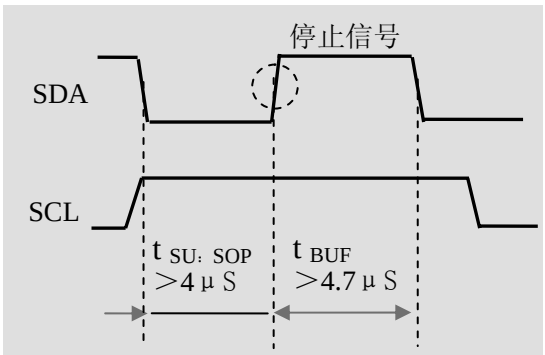

图 7.10 停止信号的时序图

【注】t SU: SOP停止信号建立时间应大于 4.0μS。t BUF P信号和S信号之间的空闲时间应大于 4.7μS。

6.3 发送应答信号 ACK

在 SDA 为低电平期间，SCL 发送一个正脉冲。

应答信号子程序 MACK

```
MACK：  CLR     SDA
        SETB    SCL
        NOP
        NOP             ;产生t HIGH定时
```

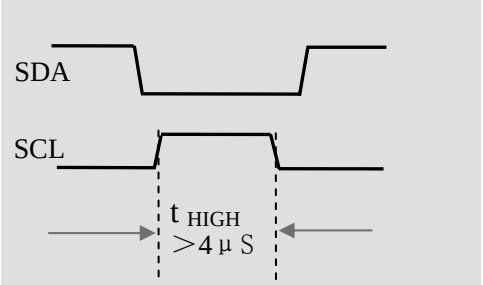

图 7.11　应答信号 MACK 时序图

```
        CLR     SCL
        SETB    SDA
        RET
```

【注】t_{HIGH} 同步时钟SCL高电平最小时间，应大于 4.0μS。

6.4 发送非应答信号 NACK

在 SDA 为高电平期间，SCL 发送一个正脉冲。

发送非应答信号子程序 MNACK

```
MNACK:  SETB    SDA
        SETB    SCL
        NOP
        NOP
        CLR     SCK
        CLR     SDA
        RET
```

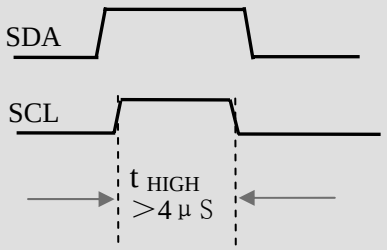

图 7.12　非应答信号 ACK 时序图

6.5 应答位检测子程序 CACK

与上面发送 ACK 和 NACK 信号不同，这是主控器对接收被控器反馈的应答信号进行的检测处理。在正常情况下被控器返回的应答信号 ACK=0。如果 ACK=1 则表明通信失败。在这个子程序中使用了一个位标志 F0 作为出口参数，当反馈给主控器的应答信号 ACK 正确时 F0=0；反之 F0=1。

```
CACK:   SETB    SDA         ;I/O 端口“写一”为输入做准备
        SETB    SCL
        CLR     F0
        MOV     C，SDA      ;对数据线 SDA 采样
        JNB     CEND        ;应答正确时转 CEND
        SETB    F0          ;应答错误时标志 F0 置一
CEND:   CLR     SCL
        RET
```

6.6 发送一个字节子程序 WRBYT（过程参见图 7.5）

模拟I^2C总线的时钟信号SCL，通过数据线SDA进行一个字节的数据发送。入口参数为累加器A，A中存有待发送的 8 位数据。按照I^2C的规范，先从最高位开始发送。

```
WRBYT:  MOV     R6，#08H    ;计数器 R6 赋初值 8
WLP:    RLCA                ;将 A 中的数据高位左移进入 Cy 中
        MOV     SDA，C      ;将数据位送入 SDA 线上
```

```
        SETB    SCL             ;产生 SCL 时钟信号
        NOP
        NOP                     ;产生t HIGH定时（大于 4μS）
        CLR     SCL             ;时钟信号变低
        DJNZ    R6，WLP         ;判断 8 次位传送是否结束
        RET
```

6.7 接收一个字节数据的子程序 RDBYT

模拟I^2C总线信号，从SDA线上读入一个字节的数据，并存于R2 或A中。

```
RDBYT:  MOV     R6，#08H
RLP:    SETB    SDA
        SETB    SCL
        MOV     C，SDA          ;采样 SDA 上的数据传到 Cy
        MOV     A，R2           ;R2 为接收数据的缓冲寄存器
        RLC     A               ;将 Cy 中的数据移入 A 中
        MOV     R2，A           ;数据送回缓冲寄存器
        CLR     SCL             ;时钟信号 SCL 拉低
        DJNZ    R6，RLP         ;8 位接收是否完成，未完成转 RLP
        RET
```

【说明】

①将I^2C总线的各种信号细划分为对应的子程序。当选择具有I^2C总线接口的外围器件进行编程时，就可根据具体的器件的特性和要求，合理的组合、调用这些子程序完成相应的功能；

②为了简化问题，上述的子程序对局部变量（如计数器、数据指针等）没有进行数据保护。为了使这些子程序具有很好的可移植性和通用性，编程者应当对它们进行进栈保护；

③上面的编程是设 AT89C51 的硬件系统采用 6MH 的系统时钟，这样指令 NOP 的执行时间是 2μS，如果采用 12MH 的系统时钟，NOP 指令的周期为 1μS，这样程序要作相应的改动以满足定时要求；

④时序中的定时时间按I^2C总线的标准模式（S模式-100KHZ）制定。

上面介绍了在AT89C51 单片机系统中，利用软件模拟的方式完成I^2C总线的各种基本时序和操作的编程。作为一个单片机系统的设计、开发者应当根据系统设计的需求，选择所需要的外围芯片构成硬件系统，再根据这些芯片的工作原理、控制方式及对应的编程命令来设计、编程，最终完成整个系统的设计工作。如果需要对I^2C总线作更详细、深入的了解，读者可以查询相关的资料。

需要说明的是：不同的I^2C总线接口芯片由于功能的不同，其工作过程和编程步骤是有区别的。但是不论如何编程其具体实施都是运用上述的各个子程序来编写出应用程序来。

芯片内部的单元寻址

作为I^2C总线的外围器件，大多器件还具有芯片内部的地址（如各个控制、状态寄存器，EEPROM的存储单元地址等），因此对大多数I^2C外围器件的访问实际上要分别处理“外围器件地址”和“器件内部的单元地址”这两部分内容。

7.1 内部单元的单字节访问

例如对 EEPROM 24C02 芯片的 00H 单元访问，操作时序如图 7.13 和图 7.14 所示。

图 7.13　AT24C02 的一个字节的写入帧格式(器件地址为 A0H)

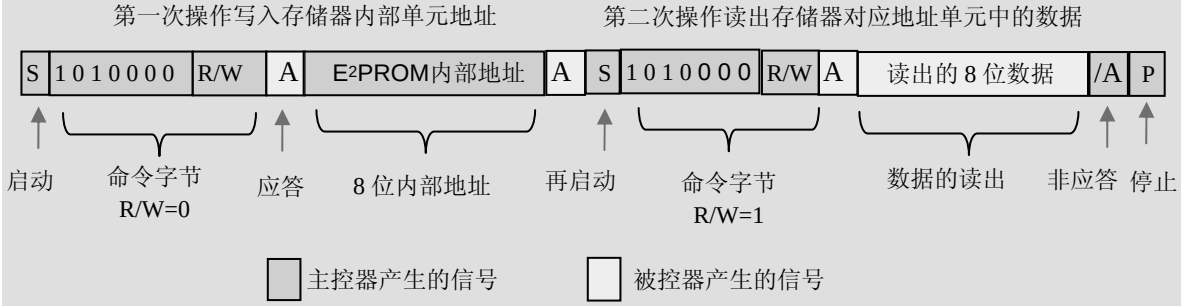

图 7.14 读指定地址存储单元中的一个数据帧格式（器件地址为 A0H）

从图 7.14 中可以见到：一个带芯片内部单元地址的“读操作”是要发送两次命令字节的：

（1）首先发送一个“写”操作的控制字（外围芯片地址 A0H 即 R/W=0）；

（2）紧接着将内部单元地址发送出去（如：00H)。这也是为什么前面是一个“写”命令的原因；

（3）当主控器收到外围器件的应答信号后，重新发送一个“启动信号”和一个“读”操作的命令字（A1H 即 R/W=1)；

（4）外围器件收到命令并返回应答信号后，将内部单元（如 00H）的数据发到 SDA 线上；

（5）主控器收到信号后向外围器件返回一个“非应答信号”后，发送一个停止信号并释放总线；

（6）外围器件收到主控器发出的“非应答信号”/A 后，停止数据的传送，释放总线。

7.2 内部单元的多字节访问

在很多情况下，对内部单元的访问往往是多字节的。如对 EEPROM 几个连续单元数据的读操作或者写操作，又比如：对外围器件中相关几个控制、状态寄存器的访问等。

对于具有内部单元地址的I^2C接口的外围器件，其内部都设计有一个“内部地址计数器”，每访问一次内部单元（无论是控制、状态寄存器还是EEPROM存储单元）其地址指针就会“自动加一”。这种设计简化了对内部数据的访问操作。因此如果要访问一个数块，只要在发送控制命令时指定一个首地址即可。也就是因为这个原因，在访问内部一些相关的控制、状态寄存器数据时，应当利用这一特点，连续访问这些单元（尽管某些单元的内容无用），这样可以节省对外围器件的访问操作。

对于连续访问的数据数量是由主控器来控制的，具体地说是通过向外围器件发送“非应答信号”来结束这个数据的操作。对于数据块的“读、写操作”要注意两点：

（1） 在读操作中要发送两次命令字：第一个是带有外围器件地址的“写”命令(R/W=0)，作用是将后续发出的内部地址写入到外围器件中的“地址计数器”中，第二个是发送带有外围器件的“读”命令（R/W=1），开始真正的“读操作”。两个命令字之间是一个“启动信号 S”来分割的;

（2） 在写操作中，某些外围器件（如 EEPROM）其连续写入的数据是受到限制的，如 24C02 每次连续写入的数据不能超过 8 个字节（这与其内部输入缓冲器的数量有关）。操作时续如下：

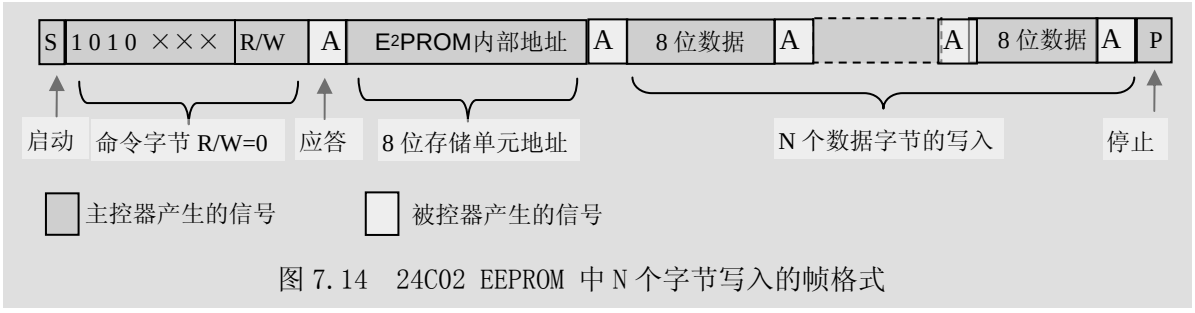

图 7.14　24C02 EEPROM 中 N 个字节写入的帧格式

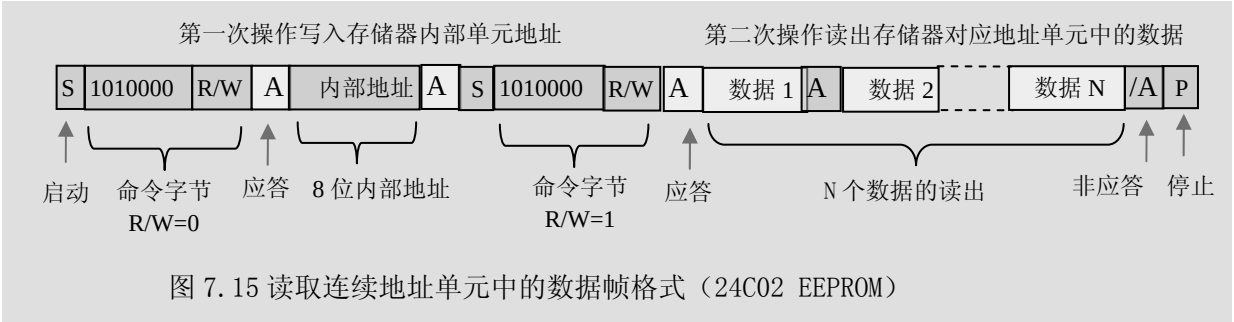

图 7.15 读取连续地址单元中的数据帧格式（24C02 EEPROM）

7.7.3 具有内部单元地址的多字节读/写子程序

在下列的两个子程序中包含了前面所描述的各种子程序，对单片机的引脚定义参加见图7.16。

在程序的前面还要使用伪指令定义以配合单片机引脚与外围器件的连接

```
SDA     BIT     P1.0
SCL     BIT     P1.1
```

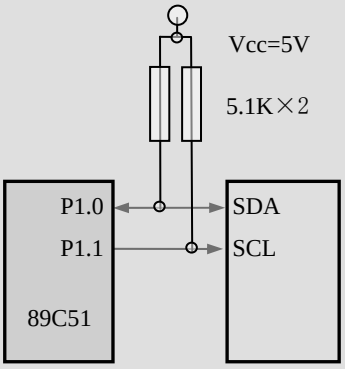

图 7.16 单片机与I^2C芯片的连接

(1) 具有内部单元地址的多字节读子程序 RDADD.ASM （参见图 7.15）

```
;*****************************************************;
通用的 I2C 通讯子程序（多字节读操作）
;入口参数 ：R7 字节数;
;R0 目标数据块首地址; R2 从器件内部子地址;
;R3 器件地址（写）; R4 器件地址（读）
;相关子程序   WRBYT、STOP、CACK、STA、MNACK、RDBYT
;*****************************************************
RDADD:  PUSH    PSW
        PUSH    ACC
        LCALL   STA
        MOV     A,R3        ;取器件地址（写）
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB   F0,RDADD       ;如果应答不正确返回重来
        MOV A,R2            ;取内部地址
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB   F0,RDADD       ;如果应答不正确返回重来
        LCALL   STA
        MOV     A,R4        ;取器件地址（读）
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB   F0,RDADD       ;如果应答不正确返回重来
RDN:    LCALL   RDBYT       ;读入数据（出口参数：A）
        MOV     @R0,A       ;存入缓冲区
        DJNZ    R7,ACK
        LCALL   MNACK
        LCALL   STOP
        POP ACC
        POP PSW
        RET
ACK:    LCALL   MACK
        INC R0
```

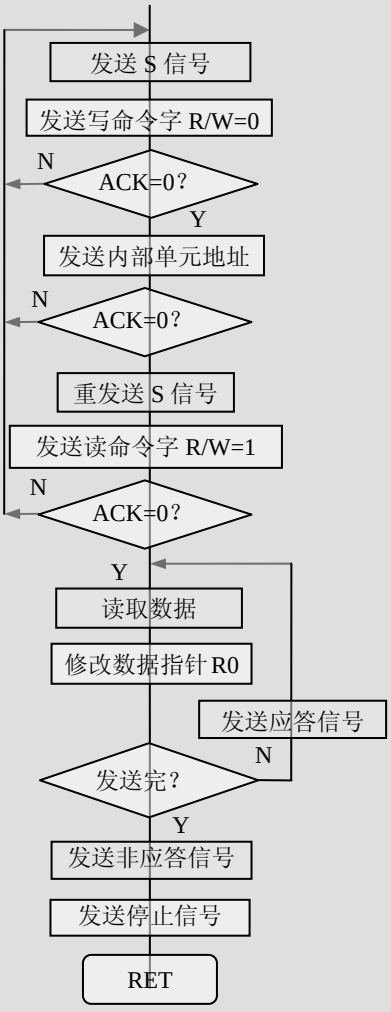

图 7.17 读 N 个数据的流程图

```
        SJMP    RDN
;*********************************************************
```

(2) 具有内部单元地址的多字节写子程序（参见图 7.14）

```
;*********************************************************
;通用的 I2C 通讯子程序（多字节写操作）
;入口参数
;R7 字节数；
;R0 源数据块首地址；R2 从器件内部子地址；
;R3 器件地址（写）；R4 器件地址（读）
;相关子程序   WRBYT、STOP、CACK、STA、MNACK
;*********************************************************
WRNBYT: PUSH    PSW
        PUSH    ACC
WRADD:  MOV     A,R3            ;取外围器件地地址（包含 r/w=0）
        LCALL   STA             ;发送起始信号 S
        LCALL   WRBYT           ;发送外围地址
        LCALL   CACK            ;检测外围器件的应答信号
        JB      F0,WRADD        ;如果应答不正确返回重来
        MOV     A,R2            ;区内部单元地址
        LCALL   WRBYT           ;发送内部寄存器首地址
        LCALL   CACK            ;检测外围器件的应答信号
        JB  F0,WRADD            ;如果应答不正确返回重来
WRDA:   MOV A,@R0
        LCALL   WRBYT           ;写数据代外围器件
        LCALL   CACK            ;检测外围器件的应答信号
        JB  F0,WRADD            ;如果应答不正确返回重来
        INC R0
        DJNZ    R7,WRDA
        LCALL   STOP
        POP ACC
        POP PSW
        RET
;*********************************************************
```

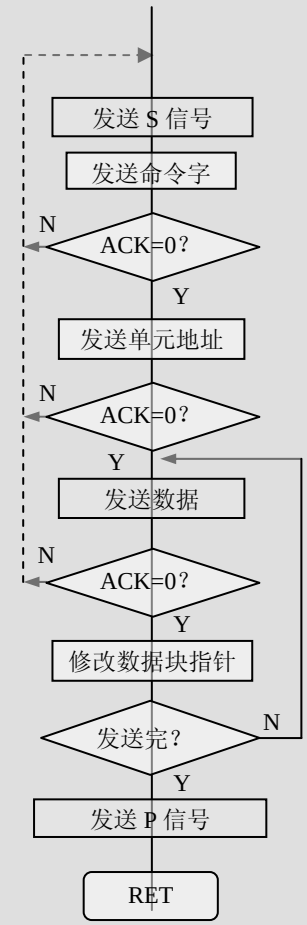

图 7.18 写 N 个数据的流程

I²C总线外围器件编程实验

在DP-51PROC综合实验台的D5区，分别设计了三种具有I²C总线接口的外围器件。它们分别是：

1. PCF8563T 实时时钟RCT芯片（芯片外围地址A2H/A3H）；
2. CAT24WC02 EEPROM芯片（芯片外围地址A0H/A1H）；
3. ZLG7290 LED动态显示、键盘扫描芯片（芯片外围地址70H/71H）。

运用第七章所描述的模拟编程方法可以很方便的实现对上述器件的读写控制及各种实验练习。在完成每一个独立的芯片实验基础上，还可以将它们有机的结合起来构成一个具有一定使用价值的综合设计题目。

通过这一章的实践，不仅可以帮助我们掌握对I²C外围器件的编程方法，而且还可以进一步感受到I²C系统的简洁、方便的硬件结构，为将来实际工程应用打下一个良好的基础。

8.1 24系列EEPROM 芯片CAT24WC02 结构原理及实验

24 系列E²PROM是目前单片机系统中应用比较广泛的存储芯片。采用I²C总线接口，占用单片机的资源少、使用方便、功耗低、容量大，被广泛应用于智能化产品设计中。

8.1.1 24系列EEPROM器件简介

24系列E²PROM为串行接口的用电来擦除的可编程COMS只读存储器。擦除次数高达10万次以上，典型的擦除时间为5ms，片内数据存储时间可达40年以上。采用单+5V供电，工作电流1mA，备用状态10μA。

（1） 24系列E²PROM芯片的引脚定义　　引脚说明如下：

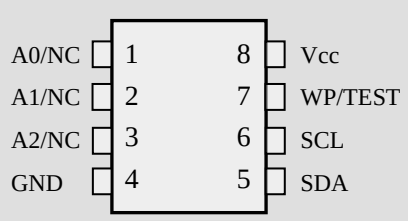

图 8.1　24系列E²PROM芯片引脚图

- SDA：串行数据输入/输出端，漏极开路结构，使用时必须外接一个5.1k的上拉电阻。通信时高位在先；
- SCL：串行时钟输入端，用于对输入数据的同步；
- WP：写保护。用于对写入数据的保护。WP=0不保护；WP=1保护，即所有的写操作失效，此时的E²PROM实际上就是一个只读存储器；
- A0～A2：器件地址编码输入。I²C总线外围器件的地址由7位组成：其中

高 4 位为生产厂家为每一型号芯片固定设置的地址也称“特征码”；低 3 位以“器件地址编码输入”的形式留给用户自行定义地址。理论上在同一个I²C总线系统中最多可以使用 8 个同一型号的外围器件；

- TEST：测试端。生产厂家用于对产品的检验，用户可以忽略；
- Vcc：+5V 电源输入端；
- NC：空脚。

（2） 24 系列E²PROM芯片特性及分类

在 24 系列产品中芯片可以划分 4 种类型。由于设计的年代不同，其性能、容量、器件地址编码的方式等各不相同。其中：第一类的芯片属于早期产品，不支持用户引脚自定义地址功能，所以在一个系统中只能使用一个该型号的芯片。同时还不具备数据保护功能；

第二类的芯片是目前常用的类型。不仅具备数据保护，还有用户引脚地址定义功能，所以在一个系统中可以同时使用 1～8 个该信号的芯片；第三类芯片基本上类似于第二类，区别在于器件地址的控制比较特殊；第四类芯片的主要特点是大容量，并支持全部的器件定义地址，因此在一个系统中可同时使用 8 个该型号的芯片。

类别	型 号	容量	页数	连续写入数据个数	器件地址编码	系统可用数量	硬件保护区域	命令字节格式		
								型号特征地址	引脚页地址	R/W
								D7 D6 D5 D4	D3 D2 D1	D0
一	AT24C01	128	×	8	不支持	1	不支持	1 0 1 0	× × ×	1/0
	AT24C01A	128	×	8	A2 A1 A0	8	全 部	1 0 1 0	A2 A1 A0	1/0
	AT24C02	256	×	8	A2 A1 A0	8	全 部	1 0 1 0	A2 A1 A0	1/0
二	AT24C04	512	2	16	A2 A1 NC	4	高 256	1 0 1 0	A2 A1 P0	1/0
	AT24C08	1K	4	16	A2 NC NC	2	不支持	1 0 1 0	A2 P1 P0	1/0
	AT24C16	2K	8	16	NC NC NC	1	高 1K	1 0 1 0	P2 P1 P0	1/0
三	AT24C164	2K	8	16	A2 A2 A0	8	高 1K	1 A2 A1 A0	P2 P1 P0	1/0
四	AT24C32	4K	×	32	A2 A2 A0	8	高 1K	1 0 1 0	A2 A1 A0	1/0
	AT24C64	8K	×	32	A2 A2 A0	8	高 2K	1 0 1 0	A2 A1 A0	1/0

表 8.1　24 系列E²PROM芯片特性、分类表

表 8.1 列出了 24 系列 E2PROM 芯片的特性与分类。对于表中内容说明如下：

① “容量”是指字节数，如 128 是指 128×8，即 128 个字节、每个字节为 8bit ；

② “页数”是指将存储器中每 256 个字节为一页。当芯片的存储容量小于等于 256 个字节时其容量实际上局限于一页的范围之内；

③ “连续写入块字节数”是指主控器向E²PROM存储器一次连续写入的字节数的数量。与普

通的SRAM存储器不同,在写数据过程中E²PROM要占用大量的时间来完成存储器单元的擦除、写入操作。为了提高整个的系统运行速度，在芯片的设计中采用了“写入数据缓冲器”结构，即主控器通过总线高速将待写入的数据先送入到E²PROM内部的数据缓冲器中，然后留给E²PROM自己逐一写入。这种设计方法可以极大地提高主控器的工作效率，当E²PROM在烧写数据时主控器可以进行其他的工作。在 24 系列E²PROM中，不同的芯片其内部的缓冲单元的数量是不同的,在编程中一次连续写入E²PROM的数据字节数不能超过缓冲器的单元数,否则会出现错误。因此所谓的“写入块字节数”实际上就是指E²PROM“写入数据缓冲器”的数量；

④ “器件地址编码”指器件 7 位地址码中低 3 位引脚地址的定义功能。理论上I²C总线外围的低 3 位地址是由器件本身的 3 个引脚的电平来确定的，这种方法为在一个系统中使用多个同一型号的芯片带来了灵活性。但在实际设计中 7 位地址码中的低 3 位不全留给使用者使用和定义。这在I²C总线外围器件中也是常见的；

⑤ “系统可用数量”是指在同一个I²C总线系统中可同时使用某一型号芯片的数量。不难看出，这个数据实际上是由芯片本身的“器件地址编码”功能来决定的；

⑥ “硬件写保护区域”是指对E²PROM存储器中原先写入的数据进行保护。与普通的SRAM不同，E²PROM存储的数据往往是一些重要的参数（如表格、程序运行参数等），采用保护措施后可以防止误操作而破坏系统的软件系统。保护功能是通过芯片的WP引脚接高电平实现的。在实际应用中可由主控器（单片机）的一个I/O口线控制或直接与Vcc或接地处理；

⑦ “命令字节格式”是指芯片的地址码加方向控制 R/W 位。这实际上是主控器寻址外围器件的命令字。在这个字节中，除了最低位 D0 是由主控器发出的“读”或“写”控制码外，高 7 位中的高 4 位由厂家已经定义为 1010（AT24C164 除外），其余低 3 位根据芯片型号（容量）的不同而不同。这低 3 位（d3、d2、d1）的定义实际上与芯片的“器件地址编码”即引脚地址定义功能有关：

（Ⅰ）对于 A2～A0 引脚全部参与器件地址定义的情况，注意这也是存储单元不分页的芯片。因此，7 位地址码实际上是一种规范的“4+3” 格式，即 4 位特征码加上 3 位器件

地址码。只要使用者在硬件上将芯片的 A2～A0 引脚处理好，则该芯片的地址就被唯一的确定下来。以 AT24C01A 为例：将芯片的 A2～A0 全部接地，这样芯片的 7 位地址为 1010000，主控器要去读该芯片中的数据，其命令字节为 10100001（R/W=1）；

（Ⅱ）对于芯片引脚 A2～A0 部分参与器件地址定义的芯片（如 AT24C04/08），其没有参与地址定义的引脚（如 A0/A0、A1）实际上在命令字的对应位置上起到一个“页选 Pi”的功能，其页选数正好与不参与器件地址定义引脚的个数有关；

（Ⅲ）对于芯片引脚 A2～A0 全不介入器件引脚定义的芯片（如 AT24C16），虽然其硬件引脚 A2～A0 无用，但在命令字对应的位置上实际上成为页地址的选择位，所以主控器寻址该器件时，其命令字中的 7 位地址实际上是 4 位特征码加 3 位“页地址”。

（Ⅳ）对于第三类芯片 AT24C164 而言，其 A2～A0 全部参与器件地址定义，存储区域又分为 8 页。那么如何将这些“器件地址”和“页地址”信息通过命令字表达出来呢？ 只有占用原来特征码的三个位的位置了，这是一种较为特殊的寻址方式；

（Ⅴ）对于第四类芯片（AT24C32/64），虽然其存储容量大大超过了 256 字节，但采用了不分页的处理方法。这就意味着主控器必须使用双字节的地址信息来确定具体的存储单元（而其它型号的存储单元地址为单字节）。

⑧ “R/W”读写控制位，也称方向位。R/W=1 为读操作；R/W=0 为写操作；

（3）芯片寻址与存储单元寻址

E^2PROM作为I^2C总线的外围器件不仅需要芯片的地址（4 位特征码+3 位器件地址）供主控器寻址，还要有与读写操作相关的存储单元地址。这就决定了主控器对E^2PROM的访问不同于其它常规外围器件的操作过程。

对于绝大多数的 24 系列E^2PROM芯片对于容量超过 256 字节的芯片都具有页选功能，这样通过芯片地址来指向芯片和要访问的页，然后再使用一个字节的“页内地址”来指明存储单元。所以在这种情况下其存储单元地址是单字节结构；而对AT24C32/64 型号的芯片，因为其存储区域没有分页，而存储容量又大大的超出 256 个字节。所以对 4K/8K的访问只能采用 13 位地址，实际上就不得不采用二个字节的形式来指明访问的存储单元。

8.1.2 24 系列 EEPROM 芯片的读写操作

（一） 写操作

写操作分为字节写和数据块写两种模式。

（1） 字节写

在这种方式中，主控器首先发送一个命令字（特征码+器件地址+R/W），待得到外围器件的应答信号ACK后，再发送一个字节/二个字节的内部单元地址，这个内部单元地址被写入到E²PROM的地址指针中去。主控器收到E²PROM的应答信号后就向E²PROM发送一个字节的数据（高位在先），E²PROM将SDA线上的数据逐位接收存入输入缓冲器中，并向主控器反馈应答信号。当主控器收到应答信号后，向E²PROM发出停止信号P并结束操作、释放总线。而E²PROM收到P信号后，激活内部的数据编程周期，将缓冲器中的数据烧写到指定的存储单元中。在E²PROM的数据编程周期中为了保证数据烧写的正确性和完整性，对所有的输入都采取无效处理、不产生任何的应答信号，直到数据编程周期结束，数据被写入指定的单元后，E²PROM才恢复正

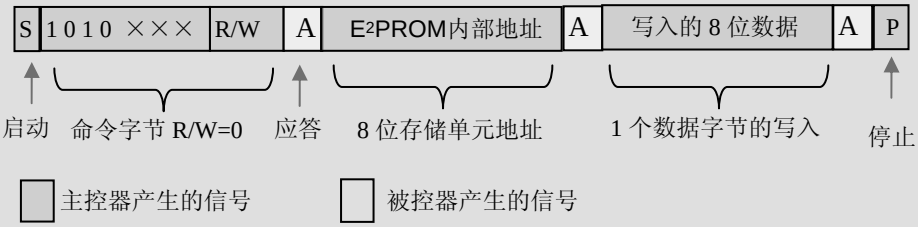

图 8.2 AT24C01/02/04/08/16 的字节写入帧格式

常的工作状态。

（2） 数据块写

基本操作类同字节写，但有几点应当注意：

① 连续写入的数据数量不能超过芯片本身“数据缓冲器”单元的数量（详见表 8.1）；

② 主控器通过发送停止信号 P 作为操作过程的结束，实际上起到控制写入数量的作用；

③ 当存储器收到主控器的停止信号后，激活“数据编程周期”，开始数据的烧写过程。

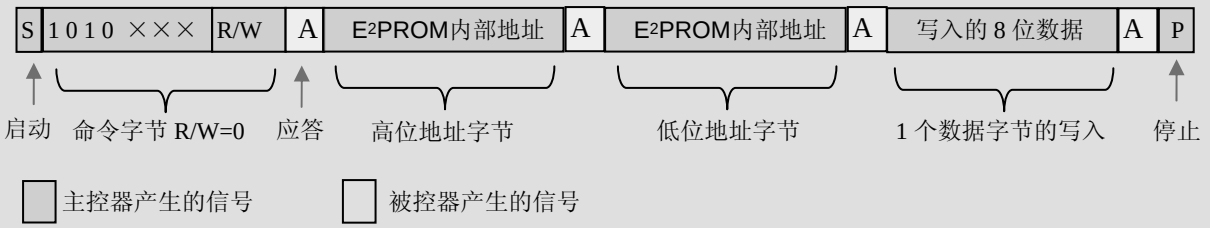

图 8.3 AT24C32/64 的字节写入帧格式

在这个过程结束前，存储器不接收外部的任何信号；

④ 烧写数据的时间取决于数据的数量，如数量 N=8，则时间约为 8ms；如果 N=32，则时

间为 32ms。

AT24C32/64 的数据块类同 AT24C01/02/04/08/16（见图 2.3）。

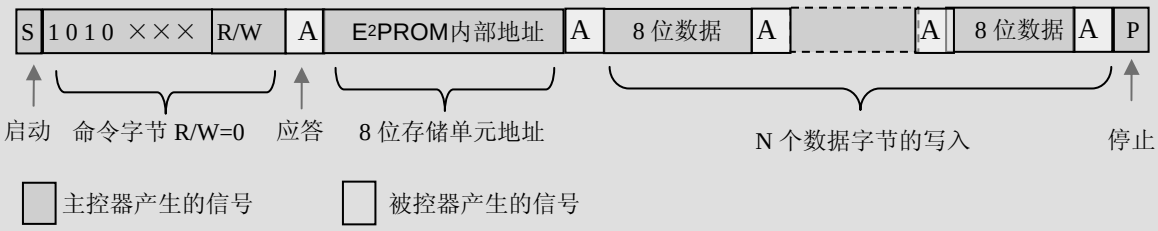

图 8.4 AT24C01/02/04/08/16N 个字节写入的帧格式

（二） 读操作

与写操作不同，读操作分为两个步骤完成：

① 利用一个写操作（R/W=0）发出寻址命令并将内部的存储单元地址写入E²PROM的地址指针中。在这个过程中E²PROM反馈应答信号，以保证主控器判断操作的正确性；

② 主控器重新发出一个开始信号S、再发送一个读操作的命令字（R/W=1），当E²PROM收到命令字后，返回应答信号并从指定的存储单元中取出数据通过SDA线送出。；

另外，因为读操作没有“数据烧写”操作，因此不使用数据缓冲器。这样连续读数据的数量不受数据缓冲器数量的限制。

读操作有三种情况：

（1）读当前地址单元中的数据

在串行E^2PROM芯片内部有一个可以自动加一的地址指针。每当完成一次读/写操作时，其指针都会自动加一指向下一个单元。只要芯片不断电，指针中的内容就一直保留。当主控器没有指定某一存储单元地址时，则E²PROM就按当前地址指针中的地址内容寻址、操作。在这种情况下，因为不用对E²PROM中的地址指针重新赋值，所以省去对E²PROM的写操作（见图 8.5）。

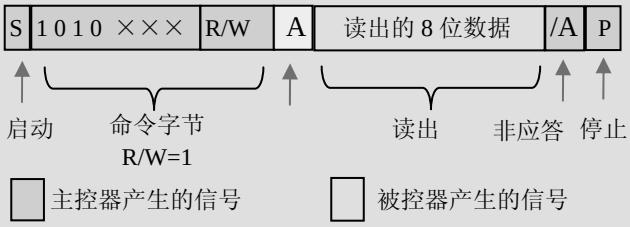

图 8.5 读当前地址单元数据的帧格式

（2）读指定地址存储单元中的数据

首先利用一个写操作（R/W=0）发出寻址命令以便将后续的内部地址写入E²PROM的地址指针中。然后主控器重新发出一个开始信号S、再发送一个读操作的命令字（R/W=1），当E²PROM收到命令字后，回应答信号并从指定的存储单元中取出数据通过SDA线送出（参

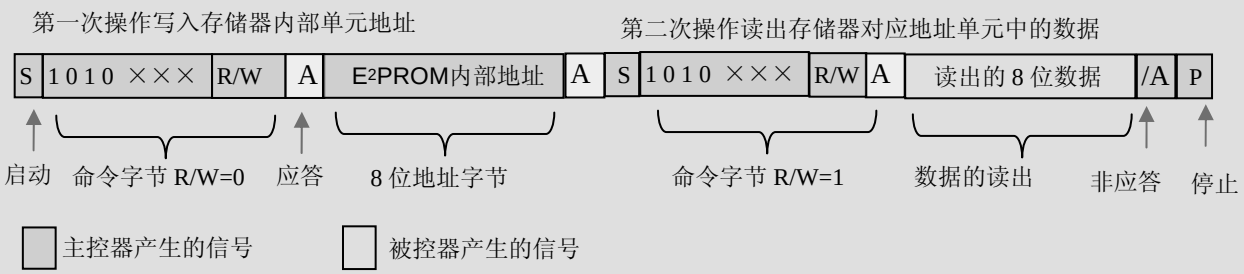

图 8.6 读指定地址存储单元中的数据帧格式（AT24C01/02/04/08/16）

见图 8.6）。

（3）读取连续地址单元中的数据

在进行连续数据读操作时应当注意：连续操作时地址不要超出该芯片所规定的页内地址的范围，否则将发生地址重叠错误。在图 2.7 中给出的是 AT24C01/02/04/08/16 芯片的操作帧格式，AT24C32/64 型号的区别在于第一次写操作时的存储单元地址为双字节字节（参见

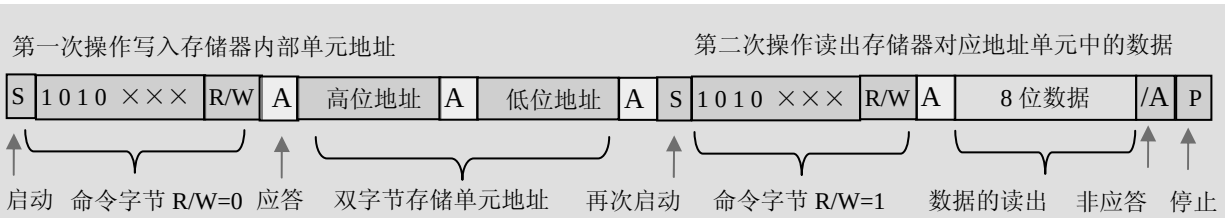

图 8.7 读指定地址存储单元中的数据帧格式（AT24C32/64）

图 8.3），其余部分是一样的。

8.1.3 24 系列 EEPROM 芯片的读写软件编程(参见 7.7.3 的内容）。

8.1.4 CAT24WC02 EEPROM 读写编程实验

与 24 系列E²PROM芯片相同，在DP-51PROC实验台上的**CAT24WC02 与**前面描述的AT24C02具有相同的参数和特征。利用DP-51PROC 上的硬件资源、编写对AT24C02 的数据写入、数据读出来验证其功能，掌握I2C总线外围器件的编程方法。

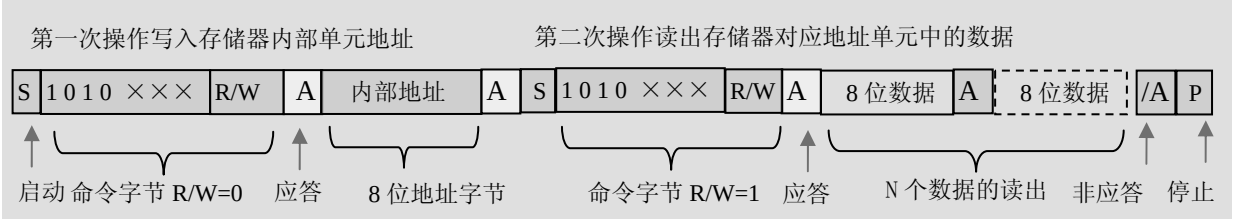

图 8.8 读取连续地址单元中的数据帧格式（AT24C01/02/04/08/16）

（1）实验内容及要求

整个实验分为两种运行模式：

- 烧写数据、读出数据。当烧写、读出操作正常后，关闭实验台的电源系统；
- 重新为实验台上电，直接读出 EEPROM 中前一次所烧写的数据，以验证 EEPROM 中数据的“非易失性”。

两种运行模式由 SW1 控制：当运行于第一种模式时，SW1 必须事先至于高电平（逻辑“1”）；第二种运行模式时，SW1 要事先置于低电平（逻辑“0”）。两种模式之间要有一次停电的过程，以验证 EEPROM 掉电时数据不丢失的特点。

（2）实验电路与连接

使用两条连接线实现 I2C 的组网联接，另使用一条连接线将 P1.7 与 SW1 连接作为程序的读写控制信号（如图 8.9 所示）。

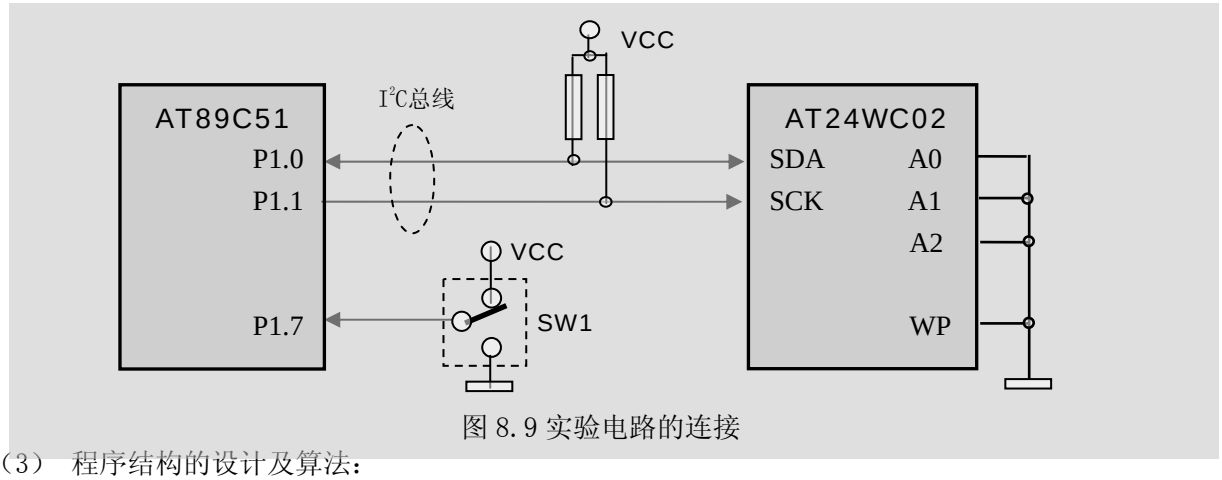

图 8.9 实验电路的连接

（3） 程序结构的设计及算法：

首先在单片机的 30H-37H 中建立一个内容为 00H-07H 的数据块，然后分别将其烧写到 EEPROM 的 00H-07H 单元中。再将 EEPROM 中所烧写进的 8 个数据读回到单片机内存 38H-3FH 中来。在调试程序时，采用“断点”的运行方式，在 EEPROM 所烧写的数据读回到单片机的存储器后，利用观察窗口对存储器中的 38H-3FH 数据进行观察、验证，看一下是否为烧写的数据。

（4） 实验的参考程序清单

```
;*********************************************************************
;这是一个 I2C 总线 EEPROM_24C02 的实验程序
;*********************************************************************
SDA     BIT     P1.0
SCL     BIT     P1.1
WSLA    EQU     0A0H
RSLA    EQU     0A1H
```

```
        ORG     8000H
        LJMP    8100H
;*********************************************************************
;          主程序
;*********************************************************************
        ORG 8100H
START:  SETB    P1.7            ;P1.7 设定为输入口
        JNB     P1.7,LOOP11     ;如果 P1.7=0 则读 EEPROM 数据
        MOV     R7,#08H         ;如果 P1.7=1 则先写入后读出
        MOV     R0,#30H
        CLR     A
LOOP:   MOV     @R0,A
        INC     R0
        INC     A
        DJNZ    R7,LOOP
AA:                             ;数据块的写操作开始
        MOV     R7,#08H         ;设定写入数据字节个数
        MOV     R0,#30H         ;设定源数据块的首地址
        MOV     R2,#00H         ;设定外围芯片的内部地址
        MOV     R3,#WSLA
        LCALL   WRNBYT
                                ;数据块读操作开始
LOOP11: MOV     R7,#08H         ;设定数据字节数
        MOV     R0,#38H         ;设定目标数据地址
        MOV     R2,#00H         ;设定外围器件内部地址
        MOV     R4,#RSLA        ;设定读命令
        MOV     R3,#WSLA        ;设定写命令
        LCALL   RDADD           ;调用读数据块子程序
        SJMP    LOOP11          ;在此处设定一个断点
                                ;之所以不返回到 START 是为了减少不必要的写
                                ; 延长 EEPROM 使用寿命
;*********************************************************************
;通用的 I2C 通讯子程序（多字节写操作）
;入口参数 R7 字节数,R0:源数据块首地址
;R0 原数据块首地址；R2 从器件内部子地址;R3:外围器件地址（写）
;相关子程序   WRBYT、STOP、CACK、STA
;*********************************************************************
WRNBYT: PUSH    PSW
        PUSH    ACC
WRADD:  MOV     A,R3            ;取外围器件地地址（包含 r/w=0）
        LCALL   STA             ;发送起始信号 S
        LCALL   WRBYT           ;发送外围地址
        LCALL   CACK            ;检测外围器件的应答信号
        JB      F0,WRADD        ;如果应
```

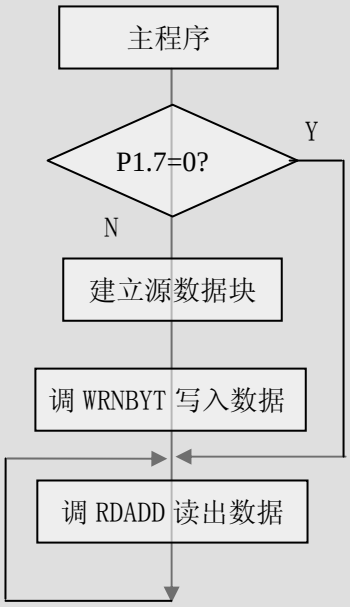

图 8.10 主程序流程图

```
        MOV     A,R2
        LCALL   WRBYT       ;发送内部寄存器首地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB      F0,WRADD    ;如果应答不正确返回重来
WRDA:   MOV     A,@R0
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB      F0,WRADD    ;如果应答不正确返回重来
        INC     R0
        DJNZ    R7,WRDA
        LCALL   STOP
        POP     ACC
        POP     PSW
        RET
;*********************************************************************
;通用的 I2C 通讯子程序（多字节读操作）
;入口参数 R7 字节数；
;R0 目标数据块首地址；R2 从器件内部子地址；
;R3 器件地址（写）；R4 器件地址（读）
;相关子程序   WRBYT、STOP、CACK、STA、MNACK
;*********************************************************************
RDADD:  PUSH    PSW
        PUSH    ACC
RDADD1: LCALL   STA
        MOV     A,R3        ;取器件地址（写）
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB      F0,RDADD1   ;如果应答不正确返回重来
        MOV A,R2            ;取内部地址

        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB      F0,RDADD1   ;如果应答不正确返回重来

        LCALL   STA
        MOV     A,R4        ;取器件地址（读）
        LCALL   WRBYT       ;发送外围地址
        LCALL   CACK        ;检测外围器件的应答信号
        JB      F0,RDADD1   ;如果应答不正确返回重来

RDN:    LCALL   RDBYT
        MOV     @R0,A
        DJNZ    R7,ACK
        LCALL   MNACK
```

```
        LCALL   STOP
        POP     ACC
        POP     PSW
        RET
ACK:    LCALL   MACK
        INC R0
        SJMP    RDN
;*****************************************************************
;                   模拟 I2C 信号子程序
;*****************************************************************
        ORG 8200H
STA:    SETB    SDA ;启动信号 S
        SETB    SCL
        NOP
        NOP
        NOP
        NOP
        NOP         ;产生 4.7US 延时
        CLR SDA
        NOP
        NOP
        NOP
        NOP         ;产生 4.7US 延时
        CLR SCL
        RET
;*****************************************************************
STOP:   CLR SDA     ;停止信号 P
        SETB    SCL
        NOP
        NOP
        NOP
        NOP
        NOP         ;产生〉4US 延时
        SETB    SDA
        NOP
        NOP
        NOP
        NOP
        NOP         ;产生 4.7US 延时
        CLR SCL
        CLR SDA
        RET
;*****************************************************************
MACK:   CLR SDA ;发送应答信号 ACK
```

```
        SETB    SCL
        NOP
        NOP
        NOP
        NOP     ;产生>4US 延时
        CLR SCL
        SETB    SDA
        RET
;*****************************************************************
MNACK:  SETB    SDA ;发送非应答信号 NACK
        SETB    SCL
        NOP
        NOP
        NOP
        NOP     ;产生> 4US 延时
        CLR SCL
        CLR SDA
        RET
;*****************************************************************
CACK:   SETB    SDA ;应答位检测子程序
        SETB    SCL
        CLR F0
        MOV C,SDA   ;采样 SDA
        JNC CEND    ;应答正确时转 CEND
        SETB    F0  ;应答错误时 F0 置一
CEND:   CLR SCL
        RET
;*****************************************************************
WRBYT:  MOV R6,#08H ;发送一个字节子程序
WLP:    RLC A   ;(入口参数 A)
        MOV SDA,C
        SETB    SCL
        NOP
        NOP
        NOP
        NOP
        NOP
        CLR SCL
        DJNZ    R6,WLP
        RET
;*****************************************************************
RDBYT:  MOV R6,#08H ;接收一个字节子程序
RLP:    SETB    SDA
        SETB    SCL
```

```
        MOV C,SDA
        MOV A,R2
        RLC A
        MOV R2,A
        CLR SCL
        DJNZ    R6,RLP   ;(出口参数 R2)
        RET
;*********************************************************************
DELAY:  PUSH    00H      ;延时子程序
        PUSH    01H
        MOV R0,#00H
DELAY1: MOV R1,#00H
        DJNZ    R1,$
        DJNZ    R0,DELAY1
        POP 01H
        POP 00H
        RET
;*********************************************************************
        END
;*********************************************************************
```

【提示】：可以将 I2C 所有信号的子程序、多字节数据写、多字节数据读子程序作为库函数保留起来，对于后续的 I2C 总线编程会带来极大的方便。

【思考题】将上述程序改为：将数据 00H-0FH 烧写到 EEPROM 的 10H-1FH，并读回到单片机的 40H-4FH 中，程序应如何修改？

【提示】：24WC02 EEPROM 每次连续写入数据不能超过 8 个字节，16 个字节应当分为两次完成。