

## 模拟 I<sup>2</sup>C 与 80C51 单片机的接口

南京办事处 李奇刚

I<sup>2</sup>C 总线是一双线串行总线，它提供一小型网络系统为总线上的电路共享公共的总线。总线上的器件有单片机 LCD 驱动器以及上 E<sup>2</sup>PROM 器等等。

两根双向线中，一根是串行数据线 (SDA)，另一根是串行时钟线 (SCL)。总线和器件间的数据传送均由这根线完成。每一个器件都有一个唯一的地址，以区别总线上的其它器件。当执行数据传送时，谁是主器件，谁是从器件详见表 1。主器件是启动数据发送并产生时钟信号的器件。被寻址的任何器件都可看作从器件。I<sup>2</sup>C 总线是多主机总线，意思是可以两个或更多的能够控制总线的器件与总线连接。

表 1 I<sup>2</sup>C 总线名词解释

术语	说 明
发送器	发送数据到总线上的器件
接收器	从总线上接收数据的器件
主器件	启动数据传送，并产生时钟信号的器件
从器件	被主器件寻址的器件
多主器件	一个以上的主器件能同时企图控制总线而不破坏信息
仲裁	一个以上的主器件同时试图控制总线时，只允许一个有效从而保证数据不被破坏的过程
同步	使两个或更多的器件的时钟信号同步的过程

总线上，每一次数据传送，都是由主器件发送起始信号开始，发送停止信号结束（见图 1）。主器件然后送从器件的特征地址。对 E<sup>2</sup>PROM 而言，从器件地址的前四位是固定的“1010”，接下来的三位标定器件的组合地址，以便知哪一个 2K 存贮器被寻址，最后一位是读写位，“1”表示读命令，“0”表示写命令（见图 2）。

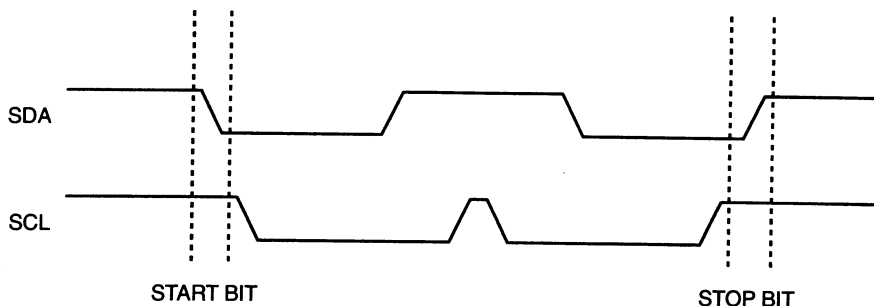


图 1 开始/停止时序

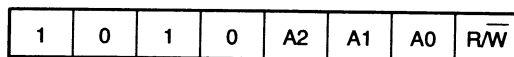


图 2 从地址位

当主器件送完起始控制命令后, 地址与自己相符的从器件会产生一个应答位 (见图 3)。进行读还是写操作, 将由 R/W 位决定。

### CAT24WCxx 与 80C51 单片机的接口

CATALYST 的 I<sup>2</sup>C 系列器件可直接和象英特尔的 MCS-51 系列单片机接口, MCS-51 系列包括 80C31/80C51、P87LPC764/P87LPC762、AT89C2051/89C1051 (GMS97C2051/97C1051) 及 P89C51/P89C52/89C54/89C58 (PHILIPS)、AT89C51/AT89C52 (GMS97C51/97C52) 等类型。

CATALYST 的 I<sup>2</sup>C E<sup>2</sup>PROM 是串行、非易失的存储器, 它包括从 1K 位 (CAT24WC01) 到 64K 位 (CAT24WC64) 各种容量的型号, 它们遵循 I<sup>2</sup>C 总线的协议, 使用 SDA, SCL 双线传输数据, 如前所述。

其中:

CAT24WC02 有一 8 字节二页面写缓冲器和一写保护引脚防止意外擦写。

CAT24WC04/08/16 等器件有 16 字节的页面写缓冲器。最多允许 8 个 CAT24WC02, 4 个 CAT24WC04, 两个 CAT24WC08 或 1 个 CAT24WC16 与 I<sup>2</sup>C 总线相连而地址不重复, CAT24WC32/64 有一 32 字节的页面写缓冲器, 最多允许 8 个这样的器件与 I<sup>2</sup>C 总线连接, 而有不同地址, 但每个器件必须通过 A0, A1 和 A2 三引脚接不同的上、下拉组合来区分。下面将介绍 80C51 微控制器对 E<sup>2</sup>PROM 进行字节写/任意地址读, 页面写/连续地址读等模式的标准程序, 图 4 是硬件接口图。

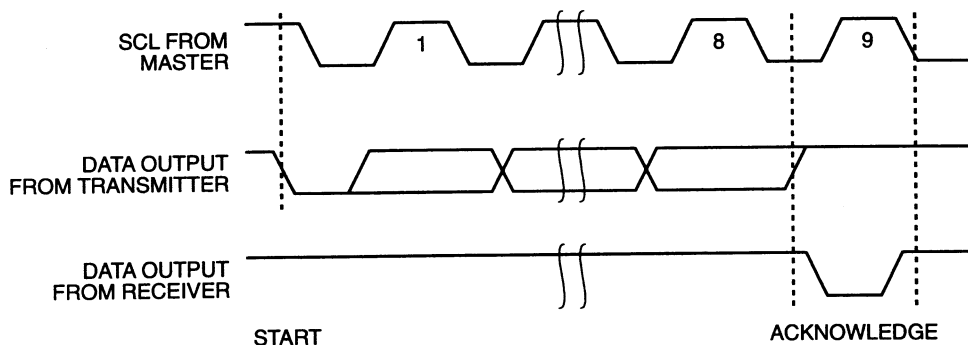


图 3 应答信号时序

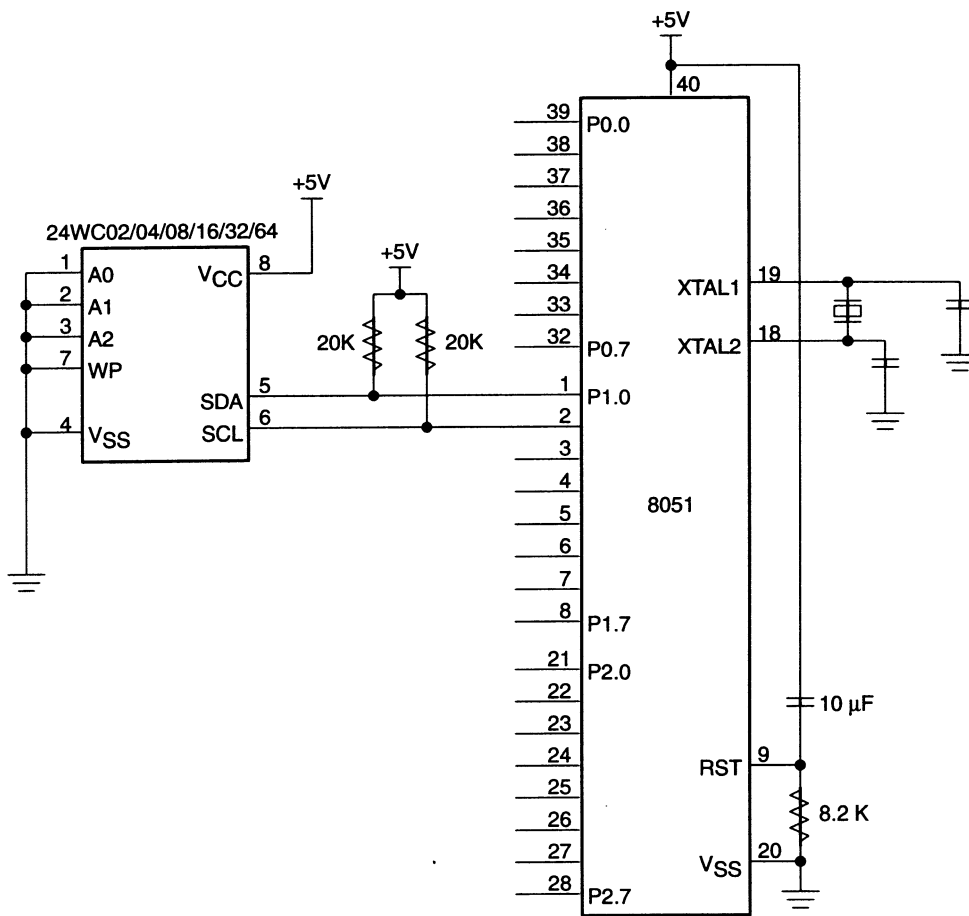


图 4 80C51 单片机的 I<sup>2</sup>C 接口

程序范例:

```

; *****
; 说明: 下面的程序代码是 80C51 对 CATLYST 的 I2C EEPROM 系列的操作。
; 有字节写/字节读和页写/顺序读等示范, 使用 80C51 的 P1.0, P1.1 两根
; 口线与 CAT24Cxx 连接。程序适用 CAT24WC02/04/08/16 等器件, 注意:
; 24WC02 是 8 字节的页写缓冲区, 24WC04/08/16 是 16 字节的页写缓冲区。
; *****

```

```

SCL          BIT    P1.0
SDA          BIT    P1.1

SLV_ADDR    EQU    0101B          ; 固定从地址 (控制码)

DATAOUT     EQU    R5            ; 读数据寄存器

```

```

ACK_READ      EQU      10000101B      ; 应答检测读命令

                DSEG
                ORG      0030H
PAGE_DATA:    DS       1
BLK_ADDR:     DS       1
BYTE_ADDR:    DS       1
BYTE_DATA:    DS       1

                ORG      0040H

STACK:        DS       31

                SEG
                ORG      0040H
                LJMP     BEGIN
                ORG      0100H

BEGIN:         MOV      SP, #STACK      ; 初始化

                MOV      BLK_ADDR, #000B ; 初始化 2K

                MOV      BYTE_DATA, #55H ; 字节数据
                MOV      BYTE_ADDR ; 字节地址
                MOV      PAGE_DATA, #0AAH ; 页面数据

                ACALL    PAGE_WR        ; 调页面写

                ACALL    SEQ_RD         ; 调顺序地址读

                ACALL    BYTE_WR        ; 调字节写

                ACALL    SELECT_RD      ; 调任意地址读
DONE:         LJMP     DONE            ; 循环等待，程序运行到

; *****
;      此时，可观察各寄存器
;      内容是否正确。
;      写一字节
; *****

BYTE_WR:      ACALL    START_BIT       ; 送起始位
                MOV     A, #SLC_ADDR    ; 4 位从地址
                MOV     R7, #4          ; 位数
                ACALL    SHFT0
                MOV     A, BLK_ADDR     ; 2K 地址空间
                MOV     R7, #3
                ACALL    SHFT0
                MOV     A, #00          ; 读/写位为 0，表示写
                MOV     R7, #1
                ACALL    SHFT0
    
```

```

ACALL  SLAVE_ACK          ; 送从应答

MOV    A, BYTE_ADDR      ; 字节地址
MOV    R7, #8
ACALL  SHFT0
ACALL  SLAVE_ACK
MOV    A, BYTE_DAT;      ; 字节数据
MOV    R7, #8
ACALL  SHFT0
ACALL  SLAVE_ACK
ACALL  STOP_BIT          ; 停止位
ACALL  ACK_POL           ; 调应答检测，等写周期结束

```

RET

```

;*****
;          页面写
;*****

```

```

PAGE_WR:    ACALL  START_BIT
            MOV    A, #SLV_ADDR

```

```

MOV    R7, #4
ACALL  SHFT0
MOV    A, BLK_ADDR
MOV    R7, #3
ACALL  SHFT0
MOV    A, #00
MOV    R7, #1
ACALL  SHFT0
ACALL  SLAVE_ACK
MOV    A, BYTE_ADDR
MOV    R7, #8
ACALL  SHFT0
ACALL  SLAVE_SCK
MOV    R4, #0FH

```

```

NEXT_DAT:
MOV    A, PAGE_DATA      ; 写 16 个字节到 EEPROM
MOV    R7, #8
ACALL  SHFT0
ACALL  SLAVE_ACK
DJNZ  R4, NEXT_DATA
ACALL  STOP_BIT
ACALL  SCK_POL

```

RET

```

;*****
;  应答检测
;*****

```

```

ACK_POL:      MOV     R3, #40H                ; 限制最大时间
ACK_LOOP:     DJNZ   R3, DONE_YET
              SJMP   DN_ACKPOL
DONE_YET:     ACALL  START_BIT
              MOV    A, #ACK_READ
              MOV    R7, #8
              ACALL  SHFT0
              ACALL  SLAVE_ACK
              JC     ACK_LOOP                ; 无应答循环
DN_ACKPOL:    ACALL  STOP_BIT
              RET
    
```

```

; *****
; 送数子程
; 入口条件: A 中为待送数据, R7 为需送位数 (1<= R7 <=8 )
; *****
    
```

```

SHFT0:       CLR     SCL                    ;
NXTSHF:      CLR     SCL
              RRC    A
              MOV    SDA, C
              SETB   SCL
              DJNZ   R7, NXTSHF
              RET
    
```

```

; *****
; 起始位
; *****
    
```

```

START_BIT:   SETB   SCL
              NOP
              SETB   SDA
              NOP
              CLR    SDA
              NOP
              CLR    SCL
              RET
    
```

```

; *****
; 停止位
; *****
    
```

```

STOP_BIT:    CLR     SDA
              NOP
              SETB   SCL
              NOP
              SETB   SDA
              RET
    
```

```

; *****
    
```

;从应答

;\*\*\*\*\*

```
SLAVE_ACK:      NOP
                 NOP
                 CLR      SCL
                 NOP
                 SETB    SDA
                 NOP
                 NOP
                 SETB    SCL
                 NOP
                 NOP
                 MOV     C, SDA      ; 保存应答位
                 CLR     SCL
                 RET
```

; \*\*\*\*\*

; 主应答

; 注：该子程序在“顺序地址读”时需用

; \*\*\*\*\*

```
MSTR_ACK:       CLR      SCL
                 NOP
                 CLR     SDA
                 NOP
                 NOP
                 SETB    SCL
                 NOP
                 CLR     SCL
                 NOP
                 SETB    SDA
                 RET
```

;\*\*\*\*\*

; 无需应答

;\*\*\*\*\*

```
NO_ACK:         SETB    SDA
                 NOP
                 SETB    SCL
                 NOP
                 CLR     SCL
                 RET
```

;\*\*\*\*\*

; 读任意地址

;\*\*\*\*\*

```
SELECT_RD:
```

```

ACALL  START_BIT
MOV    A, #SLV_ADDR
MOV    R7, #4
ACALL  SHFT0
MOV    A, #BLK_ADDR
MOV    R7, #3
ACALL  SHFT0
MOV    A, #0
MOV    R7, #1
    ACALL  SHFT0
    ACALL  SLAVE_ACK

    MOV    A, BYTE_ADDR
    MOV    R7, #8
    ACALL  SHFT0
    ACALL  SLAVE_ACK

    ACALL  START_BIT                ; 新起始位
    MOV    A, #SLV_ADDR
    MOV    R7, #4
    ACALL  SHFT0
    MOV    A, BLK_ADDR
    MOV    R7, #3
    ACALL  SHFT0
    MOV    A, #1
    MOV    R7, #1
    ACALL  SHFT0
    ACALL  SLAVE_ACK

```

```

CLOCK8:    MOV    R7, #8
            SETB   SCL
            NOP
            MOV    C, SDA
            CLR    SCL
            MOV    A, DATAOUT
            RLC    A
            MOV    DATAOUT, A
            DJNZ   R7, CLOCK8      ; 读八位数据存 DATAOUT 中
            ACALL  NO_ACK
            ACALL  STOP_BIT
            RET

```

; \*\*\*\*\*  
; 顺序地址读  
; 注：顺序地址读，可以连续顺序地读出整个存储器的内容。  
; \*\*\*\*\*

```

SEQ_RD:    ACALL  START_BIT

            MOV    A, #SLV_ADDR
            MOV    R7, #4

```

```

ACALL  SHFT0
MOV    A, BLK_ADDR
MOV    R7, #3
ACALL  SHFT0
MOV    A, #0
MOV    R7, #1
ACALL  SHFT0
ACALL  SLAVE_ACK

MOV    A, BYTE_ADDR
MOV    R7, #8
ACALL  SHFT0
ACALL  SLAVE_ACK

ACALL  START_BIT
MOV    A, #SLV_ADDR
MOV    R7, #4
ACALL  SHFT0
MOV    A, BLK_ADDR
MOV    R7, #3
ACALL  SHFT0
MOV    A, #1
MOV    R7, #1
ACALL  SHFT0
ACALL  SLAVE_ACK

NEXT_BYTE: MOV    R6, #0FH
ONE_BYTE:  MOV    R7, #8           ; 读 16 个字节
           SETB   SCL
           NOP
           CLR    SCL
           NOP
           DJNZ   R7, ONE_BYTE
           ACALL  MSTR_ACK
           DJNZ   R6, NEXT_BYTE

LST_BYTE: MOV    R7, #8
           SETB   SCL
           NOP
           CLR    SCL
           NOP
           DJNZ   R7, LST_BYTE   ; 读最后一字节
           ACALL  NO_ACK         ; "无需应答"位
           ACALL  STOP_BIT
           RET
           ;
           END

```