



LJD-SY-5100S 学习实验开发板

用户手册

北京蓝海微芯科技发展有限公司

<http://www.ljd-2008.com>

第一章 学习板简介

LJD-SY-5100S 学习试验板，适合于零基础或者初中级单片机爱好者使用。本实验板包括键盘扫描、数码管显示、字符型 LCD 显示、I²C 接口 E²PROM 24C02 的读写、单片机基本 I/O 口的输入输出控制、计数器、外中断、蜂鸣器报警、唱歌、RS232 串口通讯操作，可使使用者熟悉单片机的基本用法。

试验板接口说明：

SK2: 1、2 短接，USB 供电

2、3 短接，外接 5V 供电

SK1 : 1 - 2 OFF

2 - 3 ON

JP4 : 电机扩展口

JP2 : 单片机扩展口

W1 : 调整字符液晶对比度

D0 - D7 : 发光二极管，对应单片机的 P0 口

S0 - S7 : 拨码开关，拨上去是低电平，拨下来是高电平，对应单片机的 P2 口

K1 : 单脉冲开关，接法如下

SW3 1-2 短接产生正脉冲：SW1 1-2 短接 T0 计数

SW1 2-3 短接 T1 计数

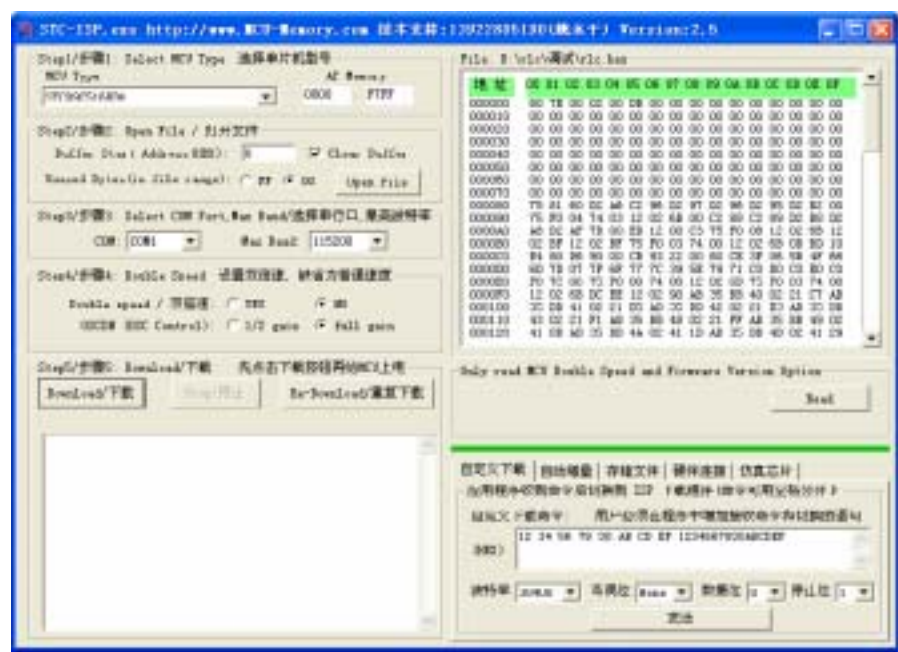
SW3 2-3 短接产生负脉冲：SW2 1-2 短接 INT0 边沿触发

SW2 2-3 短接 INT1 边沿触发

LJD-SY-5100S 采用最新的 STC89 系列单片机作为用户终端 CPU，该 CPU 除具备一般 51 功能以外，同时可以具备工业级，高速度等优点，同时利用我们提供的下载软件可以直接通过串行口进行下载烧录，不需另外配套专用下载线。

STC89 系列单片机大部分具有在系统可编程 (ISP) 的特性，ISP 的好处是省去购买通用编程器，单片机在用户系统上即可下载/烧录用户程序。大部分 STC89 系列单片机在销售给用户之前已经在单片机内部固化有 ISP 系统引导程序，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器 (速度比通用编程器快)。不要用通用编程器编程，否则有可能将单片机内部已固化的 ISP 系统引导程序擦除，造成无法使用 STC 提供的 ISP 软件下载用户的程序代码。

操作步骤：首先解压、安装、运行 STC-ISP 软件，将试验板的串口与计算机的串口连接，注意不要带电插拔，以防损坏串口！



步骤 1：选择你所使用的单片机型号，STC89C52RC

步骤 2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin, *.hex），可直接调光盘里可下载烧写的目标代码文件夹下的 hex 代码，进行测试

步骤 3：选择串口号，你所用的电脑串口是 COM1 还是 COM2

步骤 4：设置是否双倍速，双倍速选中 Double Speed 即可，STC89C52RC 系列出厂时为单倍速，用户可指定设为双倍速，如想从双倍速恢复成单倍速，则需用通用编程器擦除整个晶片方可，这会将单片机内部已烧录的 ISP 引导程序擦除。一般使用缺省设置即可，无须设置。

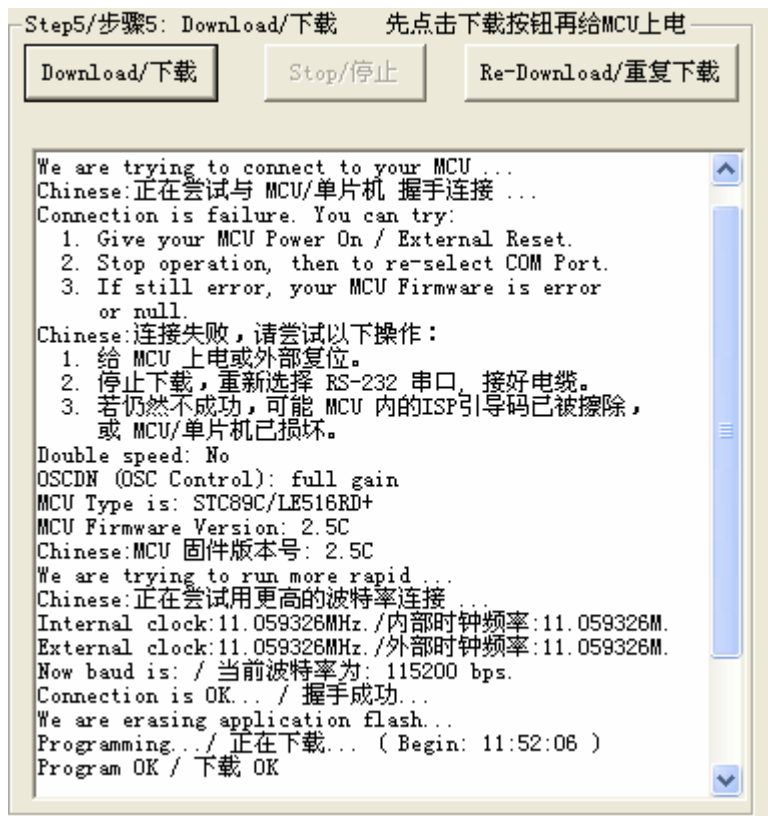
OSCDN：单片机时钟振荡器增益

选 1/2 gain 为降一半，降低 EMI；选 full gain（全增益）为正常状态

步骤 5：选择“Download/下载”按钮下载用户的程序进单片机内部，可重复执行步骤 5，也可选择“Re-Download/重复下载”按钮。

下载时注意看提示，主要看是否要给单片机上电或复位，下载速度比一般通用编程器快。一定要先选择“Download/下载”按钮，然后再给单片机上电复位（先彻底断电），而不要先上电。

下载编程成功如图所示：



这样即可上电运行用户的程序了。

怎样生成可烧写的 hex 文件？

在这里，我们使用的是 keilc 软件，其具有强大的编辑，编译和调试功能。首先安装光盘中提供的 keil7.09DEMO 版，不需要序列号，程序大小限制在 2k 以内。使用方法请观看多媒体演示文件。编译当前文件，在当前工程路径下就会生成 hex 文件。

第二章 软件实验部分

实验 1 CPU 片内清零程序

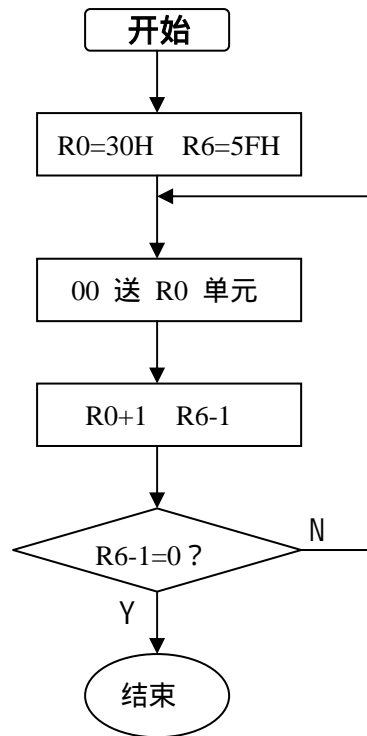
一、实验目的

掌握 MCS-51 汇编语言的设计，了解单片机的寻址方式以及调试方法。

二、实验内容

把单片机片内的 30H ~ 7FH 单元清零。

三、实验框图



四、实验步骤

用连续或者单步的方式运行程序，检查 30H-7FH 执行前后的内容变化。

五、实验思考

如果把 30H-7FH 的内容改为 55H，任何修改程序。

六、程序清单

文件名称：RAMCLR_1.ASM

```

ORG 0000H
CLEAR: MOV R0, #30H ;30H 送 R0 寄存器
      MOV R6, #4FH ;4FH 送 R6 寄存器 (计数)
CLR1:  MOV A, #00H ;00 送累加器 A
      MOV @R0, A ;00 送到 30H-7FH 单元
      INC R0 ;R0 加 1
      DJNZ R6, CLR1 ;不到 4F 个字节再清
WAIT:  LJMP WAIT
      END
  
```

实验 2 数据排序实验程序

一、实验目的

掌握外部 RAM 中数据的操作方法以及调试方法

二、实验内容

用冒泡法将 RAM 中几个单字节无符号整数，按照从小到大的次序重新排列。

三、实验框图

见下图

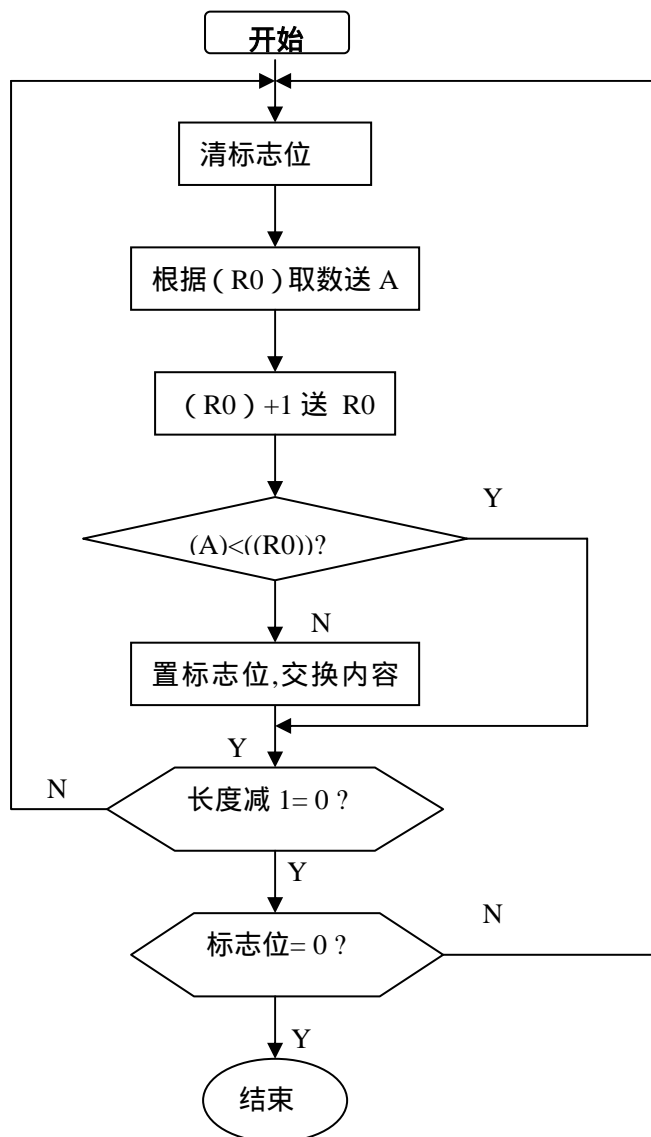
四、实验步骤

把 CPU 中 RAM 的 50H-5AH 中放入不等的数字，运行该程序，然后检查 50H-5AH 的数据是否按照从小到大排列。

五、思考

修改程序，把 50H-5AH 中的数据从大到小排列，任何设计程序

六、程序清单



; 文件名称: SJPX.ASM

```
ORG 0000H
PXCX: MOV R3, #50H
QL4:  MOV A, R3      ; 指针送 R0
      MOV R0, A
      MOV R7, #0AH   ; 长度送 R7
      CLR 00H        ; 标志位=0
QL2:  MOV A, @R0
      INC R0
      MOV R2, A
      CLR C
```

```

MOV    22H, @R0
CJNE  A, 22H, QL3 ;相等吗?
SETB  C
QL3:  MOV    A, R2
      JC    QL1      ;大于交换位置
      SETB  00H
      XCH  A, @R0
      DEC  R0
      XCH  A, @R0      ;大于交换位置
      INC  R0
QL1:  MOV    A, @R0
      DJNZ R7, QL2
      JB   00H, QL4      ;一次循环中有继续交换
      SJMP $           ;没有交换退出
      END

```

实验 3 无符号双字节乘法实验程序

一、实验目的

掌握 MCS-51 的计算指令的使用以及调试方法

二、实验内容

将 (R2R3) 和 (R6R7) 中双字节无符号整数相乘，积送 R4、R5、R6、R7。本程序是利用单字节乘法指令运算的。

三、实验框图

见下图

四、实验步骤

在 R2R3 和 R6R7 中输入无符号整数，连续或者单步运行程序，然后检查 R4，R5，R6，R7 中的内容。

五、程序清单

```

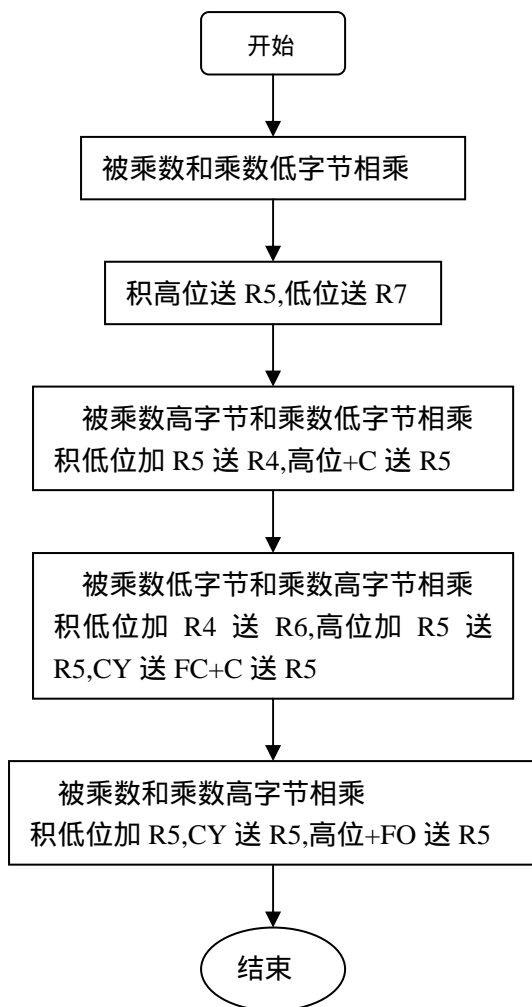
;文件名称: WMUL.ASM
      ORG  0000H
QKUL: MOV  A, R3
      MOV  B, R7
      MUL  AB          ; R3*R7
      XCH  A, R7      ; R7=(R3*R7)的低字节
      MOV  R5, B      ; R5=(R3*R7)的高字节
      MOV  B, R2
      MUL  AB          ; R2*R7
      ADD  A, R5
      MOV  R4, A
      CLR  A
      ADDC A, B
      MOV  R5, A      ; R5=(R2*R7)的高字节
      MOV  A, R6
      MOV  B, R3
      MUL  AB          ; R3*R6

```

```

ADD A, R4
XCH A, R6
XCH A, B
ADDC A, R5
MOV R5, A
MOV PSW. 5, C      ; 存 CY
MOV A, R2
MUL AB            ; R2*R6
ADD A, R5
MOV R5, A
CLR A
MOV ACC. 0, C
MOV C, PSW. 5     ; 加上一次加法的进位.
ADDC A, B
MOV R4, A
SJMP $
END

```



第三章 硬件实验部分

实验 1 广告灯的左移右移

功能说明：学习输出端口的控制

- (1) 做单一灯的左移右移，由硬件电路可知，输出“1”才能使 LED 亮。开始时 P00 亮 - P01 亮 - P02 亮 - ...P07 亮 - P06 亮 - ...P00 亮，重复循环。
- (2) 本练习可了解控制输出端口指令 MOV P0, A(MOV P0, #DATA)、MOV P1, A(MOV P0, #DATA)、MOV P2, A(MOV P2, #DATA)、MOV P3, A(MOV P3, #DATA)，只要给累加器值或常数值，然后执行上述的指令，即可达到控制输出的动作。

程序清单：

```
ORG 00H
START: MOV A, #0FFH ;左移初值
      CLR C ;c = 0
      MOV R2, #08H ;设左移 8 次
LOOP:  RLC A ;左移一位
      MOV P0, A ;输出至 P0
      LCALL DELAY ;延时 0.2 秒
      DJNZ R2, LOOP ;左移 7 次？
      MOV R2, #07H
LOOP1: RRC A ;右移一位
      MOV P0, A ;输出至 P0
      LCALL DELAY
      DJNZ R2, LOOP1 ;右移 7 次？
      JMP START
DELAY: MOV R3, #20 ;延时 0.2 秒
D1:    MOV R4, #40
D2:    MOV R5, #248
      DJNZ R5, $
      DJNZ R4, D2
      DJNZ R3, D1
      RET
      END
```

实验 2 输入端口的应用

功能说明：开关控制 P2 口的状态，将 P2 口的高低电平状态反映到 P0 口灯的亮灭，可以拨动 S0-S7，观察 D0-D7 灯的变化。

程序清单：

```
ORG 0000H
START: MOV P0, P2
      JMP START
      END
```

实验 3 计数器

功能说明：

- (1) T0 每输入脉冲 3 次则 P0 的 led 会做 BCD 码加 1 的变化，P03-P00 为个位，

P07-P04 为十位

(2) TIME0 工作在计数模式 0 下

(3) 将 SW1 1 - 2 短接, SW3 1 - 2 短接

程序清单:

```
ORG 0000H
START:  MOV R2, #00H           ;计数指针
        MOV TMOD, #00000100B  ;设定计数工作在方式 0
LOOP1:  MOV TH0, #(8192-3)/32  ;设定计数 3 次
        MOV TL0, #(8192-3) MOD 32
        SETB TRO              ;启动计数器
LOOP2:  JBC TF0, LOOP3        ;溢出吗? 是则跳到 LOOP3
        JMP LOOP2             ;不是则等待溢出
LOOP3:  MOV A, R2              ;计数指针加 1
        ADD A, #01H
        DA A                  ;做 BCD 码调整
        MOV R2, A
        CPL A
        MOV PO, A             ;输出至 P0
        JMP LOOP1
END
```

实验 4 中断的应用

功能说明: 将 SW2 1 - 2 短接, SW3 2 - 3 短接, 没有中断时, 4 个灯以闪烁为主, 响应中断后则以广告灯的形式显示, 由于采用下降沿触发, 所以对于中断有一次记忆功能。

程序清单: 中断 INT0.asm

```
ORG 0000H
JMP START
ORG 0003H
JMP INTERRUPT_0
START:  MOV SP, #60H          ;设定堆栈区
        MOV IE, #10000001B   ;中断致能, EA=EX0=1
        MOV TCON, #00000001B ;设定外部中断 0 为负缘触发

LOOP:  MOV A, #11110000B
        MOV PO, A            ;从 P1 输出到 LED
        CALL DELAY500        ;延迟约 0.5 秒
        MOV A, #00001111B
        MOV PO, A            ;从 P1 输出
        CALL DELAY500        ;延迟约 0.5 秒
        JMP LOOP

;=====
INTERRUPT_0:
        PUSH ACC
```

```

        PUSH    B
        MOV     B, #8           ;移动 8 次
        MOV     A, #11111110B  ;单一灯向右移
LOOP1:  MOV     P0, A           ;从 P1 输出到 LED
        CALL    DELAY500      ;延迟约 0.5 秒
        RR     A               ;右旋一次
        DJNZ   B, LOOP1       ;移动 8 次了吗?
        POP     B
        POP     ACC
        RETI

```

```

;-----
;延时子程序

```

```

DELAY:  PUSH    TIMER1;           延时 TIMER1*1 ms for 12MHz
        PUSH    TIMERO
DELAY1:  MOV     TIMERO, #250      ;循环一次需要 4 个机器周期, 时间为
4*1 μS
DELAY2:  NOP     ;                1 个周期
        NOP     ;                1 个周期
        DJNZ   TIMERO, DELAY2    ;2 个周期
        DJNZ   TIMER1, DELAY1
        POP     TIMERO
        POP     TIMER1
        RET
        NOP; -----
DELAY01: NOP;                    延时 1 ms
        PUSH   TIMER1
        MOV    TIMER1, #01H
        LCALL DELAY
        POP   TIMER1
        RET
        NOP; -----
DELAY10: NOP;                    延时 10 ms
        PUSH  TIMER1
        MOV   TIMER1, #10
        LCALL DELAY
        POP  TIMER1
        RET
        NOP; -----
DELAY100: NOP;                   延时 100 ms
        PUSH TIMER1
        MOV  TIMER1, #100
        LCALL DELAY
        POP  TIMER1
        RET

```

```

                                NOP; -----
DELAY250:                       NOP;           延时 250 ms
                                PUSH  TIMER1
                                MOV  TIMER1, #250
                                LCALL DELAY
                                POP   TIMER1
                                RET
                                NOP; -----
DELAY500:                       NOP;           延时 500 ms
                                PUSH  TIMER1
                                MOV  TIMER1, #250
                                LCALL DELAY
                                LCALL DELAY
                                POP   TIMER1
                                RET
                                NOP; -----
DELAY1S:                        NOP;           延时 1 sec
                                PUSH  TIMER1
                                MOV  TIMER1, #250
                                LCALL DELAY
                                LCALL DELAY
                                LCALL DELAY
                                LCALL DELAY
                                POP   TIMER1
                                RET
                                NOP; -----
END

```

实验 5 键盘扫描与数码管显示

硬件设计与基本概念

1 CH451 的功能与引脚介绍

CH451 是一个整合了数码管显示驱动和键盘扫描控制以及 μP 监控的多功能外围芯片。CH451 内置 RC 振荡电路，可以直接动态驱动 8 位数码管或者 6 4 位 LED，具有 BCD 译码或不译码功能，可实现数据的左移、右移、左循环、右循环、各数字独立闪烁等控制功能。CH451 内置大电流驱动级，段电流不小于 30 mA，字电流不小于 160 mA，并有 16 级亮度控制功能；在键盘控制方面，该器件内置 6 4 键键盘控制器，可实现 8×8 矩阵键盘扫描，并内置去抖动电路，可提供按键中断与按键释放标志位等功能；在外部接口方面，CH451 可选择简洁的 1 线串行接口或高速 4 线串行接口，且内置上电复位，可提供高电平有效复位和低电平有效复位两种输出，同时内置看门狗电路 Watch-Dog。CH451 提供有 2 8 引脚的 DIP 2 8 与 SOP 2 8 封装以及 DIP 2 4 S 封装形式，2 8 脚与 2 4 脚在功能上稍有差别，它们的引脚定义见表 1 所列。

表 1 CH451 的引脚说明

28 脚引脚号	24 脚引脚号	引脚名称	类型	引脚说明
---------	---------	------	----	------

23	2	VCC	电源	正电源，持续电流不小于 200mA
9	15	GND	电源	接地，持续电流不小于 200mA
25	4	LOAD	输入	4 线串行接口的数据加协，带上拉
26	5	DIN	输入	4 线串行接口的数据输入，带上拉
27	6	DCLK	输入	串行接口的数据时钟，带上拉，可同时用于看门狗的清除输入
24	3	DOUT	输出	串行接口的数据输出键盘中断
22 ~ 15	1、24 ~ 18	SEG7 ~ SEG0	三态输出及输入	数码管的段驱动，高电平有效，键盘扫描输入，高电平有效，带下拉
1 ~ 8	7 ~ 14	DIG7 ~ DIG0	输出	数码管的字驱动，低电平有效，键盘扫描输入，高电平有效，带下拉
12	不支持	RST	输出	上电复位和看门狗复位，高电平有效
13	不支持	RST	输出	上电复位和看门狗复位，低电平有效
28	不支持	RSTI	输入	上电复位门限调整或手工复位输入
14	不支持	ADJ	输入	段电流上限调整，带强下拉
10	不支持	CLK	输入	外接阻容振荡
11	不支持	CLKO	输出	CLK 引脚时钟信号的二分频输出
	17	NC		不连接，禁止使用

2 CH451 的操作命令

CH451 的操作命令均为 12 位，其中高 4 位为标识码，低 8 位为参数，各操作命令如下：

空操作：0000xxxxxxB (x 可为任意值，下同)

空操作命令对 CH451 不产生任何影响。该命令可以在多个 CH451 级联的应用中透过前级 CH451 向后级 CH451 发送操作命令而不影响前级 CH451 的状态。例如，要将操作命令 001000000001B 发送给两级级联电路中的后级 CH451 (后级 CH451 的 DIN 引脚连接到前级 CH451 的 DOUT 引脚)，只要在该操作命令后添加空操作命令 000000000000B 再发送，那么，该操作命令将经过前级 CH451 到达后级 CH451，而空操作命令留给了前级 CH451。另外，为了在不影响 CH451 的前提下变化 DCLK 以清除看门狗计时器，也可以发送空操作命令。在非级联的应用中，空操作命令可只发送高 4 位。

芯片内部复位：001000000001B

内部复位命令可将 CH451 的各个寄存器和各种参数复位到默认的状态。芯片上电时，CH451 均被复位，此时各个寄存器均复位为 0，各种参数均恢复为默认值。

字数据移位：0011000000[D1][D0]B

字数据移位命令共有 4 个：开环左移、右移，闭环左移、右移。D0 为 0 时为开环，为 1 时为闭环，D1 为 0 时左移，为 1 时为右移。开环左移时 DIG0 引脚对应的单元补 00H，此时不译码方式显示为空格，BCD 译码方式时显示为 0；开环右移时，DIG7 引脚对应的单元补 00H；而在闭环时 DIG0 与 DIG7 头尾相接，闭环移位。

设定系统参数：01000000[WDOG][KEYB][DISP]B

该命令用于设定 CH451 的系统级参数 如看门狗使能 WDOG、键盘扫描使能 KEYB、显示驱动使能 DISP 等。各个参数均可通过 1 位数据来进行控制，将相应的数据位置为 1 可启用该功能，否则关闭该功能 (默认值)。

设定显示参数：0 1 0 1[MODE][LIMIT][INTENSITY]B

此命令用于设定CH451的显示参数，如译码方式MODE（1位）、扫描极限LIMIT（3位）、显示亮度INTENSITY（4位）等。译码方式MODE为1时选择BCD译码方式，为0时选择不译码方式。CH451默认工作于不译码方式，此时8个数据寄存器中字节数据的位7~位0分别对应8个数码管的小数点和段G~段A，当数据位为1时，对应的数据段（或发光管）点亮；数据位为0时熄灭。CH451工作于BCD译码方式主要应用于数码管驱动，单片机只要给出二进制数的BCD码，便可由CH451将其译码并直接驱动数码管以显示对应的字符。BCD译码方式是对数据寄存器中字节数据的位4~位0进行兼容BCD的译码，可用于控制段驱动引脚SEG6~SEG0的输出，它们对应于数码管的段G~段A，同时可用字节数据的位7控制段来驱动引脚SEG7的输出以对应数码管的小数点，字节数据的位6和位5不影响BCD译码的输出，它们可以是任意值。将位4~位0进行BCD译码可显示以下28个字符，其中0000B~0111B分别对应于“0~F”、1000B~1101B分别对应于“ ” 空格、“+” + 或加号、“-” 负号或减号、“=” 等于号、“(” 左方括号、“)” 右方括号、“_” 下划线、“H”、“L”、“P”、“.” 小数点、其余值为空格。

扫描极限LIMIT控制位001B~111B和000B（默认值）可分别设定扫描极限1~7和8。显示亮度INTENSITY控制位的0001B~1111B和0000B（默认值）则用于分别设定显示驱动占空比1/16~15/16和16/16，以实现16级显示亮度控制。

设定闪烁控制：0 1 1 0[D7S][D6S][D5S][D4S][D3S][D2S][D1S][D0S]B

设定闪烁控制命令用于设定CH451的闪烁显示属性，其中D7S~D0S分别对应于8个字驱动DIG7~DIG0。闪烁属性D7S~D0S分别通过1位数据控制，将相应的数据位置为1可使能闪烁显示，否则为正常显示，不闪烁（默认值）。

加载字数据：1[DIG__ADDR][DIG__DATA]B

加载字数据命令用于将字节数据DIG__DATA（8位）写入DIG__ADDR（3位）指定的数据寄存器中。DIG__ADDR的000B~111B分别用于指定数据寄存器的地址0~7，并分别对应于DIG0~DIG7引脚驱动的8个数码管。DIG__DATA为待写入的字节数据。

读取按键代码：0 1 1 1xxxxxxxB

读取按键代码命令用于获得CH451最近检测到的有效按键的按键代码。该命令是唯一的具有数据返回的命令，CH451通常从DOUT引脚输出按键代码，按键代码总是7位数据，最高位是状态码，位5~位0是扫描码。读取按键代码命令的位数据B7~B0可以是任意值，所以控制器可以将该操作命令缩短为4位数据B11~B8。例如，CH451检测到有效按键并中断时，如按键代码是5EH，则先向CH451发出读取按键代码命令0111B，然后再从DOUT获得按键代码5EH。

CH451所提供的按键代码为7位，位2~位0是列扫描码，位5~位3是行扫描码，位6是状态码（键按下为1，键释放为0）。例如，连接DIG3与SEG4的键被按下时，按键代码为63H，键被释放后，按键代码是23H。单片机可以在任何时候读取按键代码，但一般在CH451检测到有效按键而产生键盘中断时读取按键代码，此时按键代码的位6总是1。另外，如果需要了解按键何时释放，单片机可以通过查询方式定期读取按键代码，直到按键代码的位6为0。表2是连接在DIG7~DIG0与SEG7~SEG0之间的键被按下时，CH451所提供的按键代码。这些按键代码具有一定的规律，如果需要键被

释放时的按键代码，可将表 2 中的按键代码的位 6 置 0，也可将表中的按键代码减去 40H。应注意的是：CH451 不支持组合键，也就是说，同一时刻，不能有两个或者更多的键被按下。

表 2 CH451 的键盘编码表

按键代码	DIG7	DIG6	DIG5	DIG4	DIG3	DIG2	DIG1	DIG0
SEG0	47H	46H	45H	44H	43H	42H	41H	40H
SEG1	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
SEG2	57H	56H	55H	54H	53H	52H	51H	50H
SEG3	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
SEG4	67H	66H	65H	64H	63H	62H	61H	60H
SEG5	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
SEG6	77H	76H	75H	74H	73H	72H	71H	70H
SEG7	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H

3 串行接口应用电路

CH451 与 MCS-51 单片机的连接参考原理图，其中 DOUT 引脚最好连接到单片机的中断输入引脚，这样可用中断方式响应按键。如果连接到非中断输入引脚，则应该使用查询方式确定 CH451 是否检测到有效按键，同时还可向单片机提供复位信号 RESET 并带 Watch-Dog 功能。CH451 的段驱动引脚串接的电阻 R1 (200 Ω) 用于限制和均衡段驱动电流。在 5V 电源电压下，串接 200 Ω 电阻通常对应 13mA 段电流。CH451 具有 64 键的键盘扫描功能，为了防止键被按下后在 SEG 信号线与 DIG 信号线之间形成短路而影响数码管显示，一般应在 CH451 的 DIG0~DIG7 引脚与键盘矩阵之间串接限流电阻 R2，其阻值可以从 1k Ω 至 10k Ω 。

将 P1.6 与 DIN 连接可用于输入串行数据，串行数据输入的顺序是低位在前，高位在后。另外，在上电复位后，CH451 默认选择 1 线串行接口，如需选择 4 线串行接口，则应在 DCLK 输出串行时钟之前，先在 DIN 上输出一个低电平脉冲，以通知 CH451 为 4 线串行接口。将 P1.7 与 DCLK 连接可提供串行时钟，以使 CH451 在其上升沿从 DIN 输入数据，并在其下降沿从 DOUT 输出数据。LOAD 用于加载串行数据，CH451 一般在其上升沿加载移位寄存器中的 12 位数据以作为操作命令进行分析并处理。也就是说，LOAD 的上升沿是串行数据帧的帧完成标志，此时无论移位寄存器中的 12 位数据是否有效，CH451 都会将其当作操作命令来处理。应注意的是，在级联电路中，单片机每次输出的串行数据必须是单个 CH451 的串行数据的位数乘以级联的级数。

```

    MOV P1,#60H           ;禁止其它芯片
    CLR    DIN            ;初始化 CH451
    SETB   DCLK
    SETB   DIN
    SETB   LOAD
    SETB   DOUT
    NOP
    MOV    B,#04H        ;设置 CH451
    MOV    A,#03H        ;关看门狗开显示键盘
    nop
    LCALL  WRITE
    NOP
START1:
    CLR    IT1           ;置外部信号为低电平触发
    CLR    IE1           ;清中断标志
    SETB   PX1
    SETB   EX1           ;允许键盘中断
    SETB   EA           ;开总中断
    MOV    R5,#00H
TT1:
    MOV    A,R5
    LCALL  TT
    MOV    B,#08H        ;加载字数据 1
    LCALL  WRITE
    LCALL  DELAY_1S
    LCALL  DELAY_1S
    MOV    B,#03H        ;字数据左移
    MOV    A,#00H
    LCALL  WRITE
    INC    R5
    CJNE   R5,#010H,TT1
    JMP    START1
TT:
    MOV    DPTR,#TAB
    MOVC   A,@A+DPTR
    RET
    NOP
    JMP    START1
TAB:
    DB    03FH           ;0
    DB    006H           ;1
    DB    05BH           ;2
    DB    04FH           ;3
```

```

    DB    066H    ;4
    DB    06DH    ;5
    DB    07DH    ;6
    DB    07H     ;7
    DB    07FH    ;8
    DB    06FH    ;9
    DB    77H     ;A
    DB    07CH    ;B
    DB    039H    ;C
    DB    5EH     ;D
    DB    079H    ;E
    DB    071H    ;F

```

,*******键盘处理*******

CH451_INT1:

```

    PUSH   PSW           ;现场保护
    PUSH   ACC
    PUSH   B
    MOV    R4,#06H

```

YY:

```

    MOV    B,#08H
    MOV    A,#00H
    LCALL  WRITE
    MOV    B,#03H       ;字数据左移
    MOV    A,#00H
    LCALL  WRITE
    DJNZ   R4,YY
    LCALL  INTER

```

K1:

```

    MOV    R3,DATA_KEY
    CJNE   R3,#40H,K2
    JMP    LED_0

```

K2:

```

    MOV    R3,DATA_KEY
    CJNE   R3,#41H,K3
    JMP    LED_1

```

K3:

```

    MOV    R3,DATA_KEY
    CJNE   R3,#42H,K4
    JMP    LED_2

```

K4:

```

    MOV    R3,DATA_KEY
    CJNE   R3,#43H,K5
    JMP    LED_3

```

K5:

```
      MOV     R3,DATA_KEY
      CJNE   R3,#48H,K6
      JMP    LED_4
K6:
      MOV     R3,DATA_KEY
      CJNE   R3,#49H,K7
      JMP    LED_5
K7:
      MOV     R3,DATA_KEY
      CJNE   R3,#4AH,K8
      JMP    LED_6
K8:
      MOV     R3,DATA_KEY
      CJNE   R3,#4BH,K9
      JMP    LED_7
K9:
      MOV     R3,DATA_KEY
      CJNE   R3,#50H,K10
      JMP    LED_8
K10:
      MOV     R3,DATA_KEY
      CJNE   R3,#51H,K11
      JMP    LED_9
K11:
      MOV     R3,DATA_KEY
      CJNE   R3,#52H,K12
      JMP    LED_A
K12:
      MOV     R3,DATA_KEY
      CJNE   R3,#53H,K13
      JMP    LED_B
K13:
      MOV     R3,DATA_KEY
      CJNE   R3,#58H,K14
      JMP    LED_C
K14:
      MOV     R3,DATA_KEY
      CJNE   R3,#59H,K15
      JMP    LED_D
K15:
      MOV     R3,DATA_KEY
      CJNE   R3,#5AH,K16
      JMP    LED_E
K16:
```

```

        MOV     R3,DATA_KEY
        CJNE   R3,#5BH,K17
        JMP    LED_F
K17:
        POP    ACC
        POP    PSW
        CLR    IE1
        RETI
        NOP
        LJMP   START
LED_A:
        MOV    B,#08H
        MOV    A,#077H
        LCALL  WRITE
        JMP    DELAY1
        NOP
        LJMP   START
LED_B:
        MOV    B,#08H
        MOV    A,#07CH
        LCALL  WRITE
        JMP    DELAY1
        NOP
        LJMP   START
LED_C:
        MOV    B,#08H
        MOV    A,#039H
        LCALL  WRITE
        JMP    DELAY1
        NOP
        LJMP   START
LED_D:
        MOV    B,#08H
        MOV    A,#05EH
        LCALL  WRITE
        JMP    DELAY1
        NOP
        LJMP   START
LED_E:
        MOV    B,#08H
        MOV    A,#079H
        LCALL  WRITE
        JMP    DELAY1
        NOP

```

```

        LJMP     START
LED_F:
        MOV     B,#08H
        MOV     A,#071H
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_0:
        MOV     B,#08H
        MOV     A,#03FH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_1:
        MOV     B,#08H
        MOV     A,#06H
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_2:
        MOV     B,#08H
        MOV     A,#05BH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_3:
        MOV     B,#08H
        MOV     A,#04FH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_4:
        MOV     B,#08H
        MOV     A,#066H
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_5:

```

```

        MOV     B,#08H
MOV     A,#06DH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP   START
LED_6:
        MOV     B,#08H
MOV     A,#07DH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP   START
LED_7:
        MOV     B,#08H
MOV     A,#007H
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP   START
LED_8:
        MOV     B,#08H
MOV     A,#07FH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP   START
LED_9:
        MOV     B,#08H
MOV     A,#06FH
        LCALL  WRITE
        JMP     DELAY1
        NOP
        LJMP   START
DELAY1:
        CLR     IT1           ;置外部信号为低电平触发
        CLR     IE1           ;清中断标志
        SETB    PX1
        SETB    EX1           ;允许键盘中断
        SETB    EA
        LCALL  DELAY_1S
;LCALL  DELAY_1S
;LCALL  DELAY_1S
;LCALL  DELAY_1S

```

```

        POP     B
        POP     ACC
        POP     PSW
        RETI
        NOP
        LJMP    START
;*****
WRITE:
        PUSH   ACC
        CLR    EX0
        CLR    LOAD
        MOV    R7,#08H
WRITE_1:
        RRC    A
        CLR    DCLK
        MOV    DIN,C
        SETB   DCLK
        DJNZ   R7,WRITE_1
        MOV    A,B
        MOV    R7,#004H
WRITE_2:
        RRC    A
        CLR    DCLK
        MOV    DIN,C
        SETB   DCLK
        DJNZ   R7,WRITE_2
        SETB   LOAD
        SETB   EX1
        POP    ACC
        RET
;*****
INTER:
        PUSH   PSW           ;现场保护
        PUSH   ACC
        CLR    EX1
        CLR    LOAD         ;命令开始
        MOV    A,#0F7H      ;读键值命令,忽略 12 位命令的低 8 位,高 4 位用作结束
标志
INTER_4:
        SETB   C           ;在高位添 0 以检测位数据结束
        RRC    A           ;低位在前,高位在后
        CLR    DCLK
        MOV    DIN,C       ;送出一位数据
        SETB   DCLK       ;产生时钟上升沿锁通知 CH451 输入位数据

```

```

        CJNE  A,#0FFH,INTER_4      ;位数据未完继续,共 4 位,完成后为 0FFH
        SETB  LOAD                  ;产生加载上升沿通知 CH451 处理命令数据
        MOV   A,#0FCH              ;该数据用以检测位数据结束
INTER_7:
        MOV   C,DOUT                ;读入一位数据
        CLR   DCLK                  ;产生时钟下降沿通知 CH451 输出下一位
        RLC   A                    ;数据移入 ACC,高位在前,低位在后
        SETB  DCLK
        JC    INTER_7              ;位数据未完继续,共 7 位,完成后才移出 0
        MOV   DATA_KEY,A         ;保存键值
INC    DATA_F
        POP   ACC
        POP   PSW
        SETB  EX1
        CLR   IE1                  ;清中断标志,该指令需根据实际情况作修改
        RET
;*****
DELAY_1S:
        MOV   TIMER1,#1
TEST_DYA:  MOV   TIMER2,#255
TEST_DYA1: MOV   TIMER3,#255
TEST_DYA2: NOP
        NOP
        DJNZ  TIMER3,TEST_DYA2
        DJNZ  TIMER2,TEST_DYA1
        DJNZ  TIMER1,TEST_DYA
        RET
END

```

实验 6 I2C (24C02) 读、写

1、串行 EEPROM (24C02) 接口方法

在新一代单片机中,无论总线型还是非总线型单片机,为了简化系统结构,提高系统的可靠性,都推出了芯片间的串行数据传输技术,设置了芯片间的串行传输接口或串行总线。串行总线扩展接线灵活,极易形成用户的模块化结构,同时将大大简化其系统结构。串行器件不仅占用很少的资源和 I/O 线,而且体积大大缩小,同时还具有工作电压宽,抗干扰能力强,功耗低,数据不宜丢失和支持在线编程等特点。目前,各式各样的串行接口器件层出不穷,如:串行 EEPROM,串行 ADC/DAC,串行时钟芯片,串行数字电位器,串行微处理器监控芯片,串行温度传感器等等。

串行 EEPROM 是在各种串行器件应用中使用较频繁的器件,和并行 EEPROM 相比,串行 EEPROM 的数据传送的速度较低,但是其体积较小,容量小,所含的引脚也较少。所以,它特别适合于需要存放非挥发数据,要求速度不高,引脚少的单片机的应用。这里介绍串行 EEPROM 芯片,以及它们和单片机的接口技术。

2、串行 EEPROM 及其工作原理

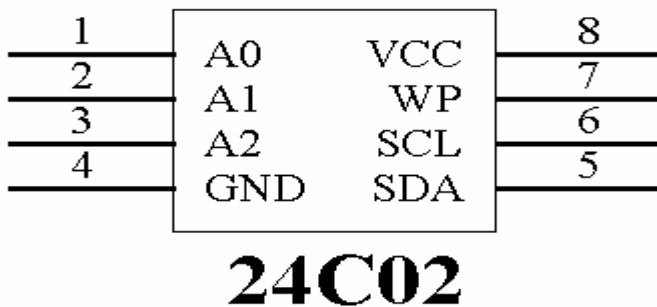
串行 EEPROM 中,较为典型的有 ATMEL 公司的 AT24CXX 系列以及该公司生产的

AT93CXX 系列 较为著名的半导体厂家 ,包括 Microchip 国家半导体厂家等 ,都有 AT93CXX 系列 EEPROM 产品。

AT24CXX 系列 EEPROM

AT24CXX 系列的串行电可改写及可编程只读存储器 EEPROM 有 10 种型号 ,其中典型的型号有 AT24C01A/02/04/08/16 等 5 种 ,它们的存储容量分别是 1024/2048/4096/8192/16384 位 ,也就是 128/256/512/1 024/2048 字节。这个系列一般用于低电压 ,低功耗的工业和商业用途 , 并且可以组成优化的系统。这个系统还有多种包括 5V(4.5~5.5V),2.7V(2.7~5.5V),2.5V(2.5~5.5V),1.8V(1.8~5.5V)等 4 种电压级别。它们的封装有 8 引脚 PDIP 方式 , 8 引脚和 16 引脚 SOIC 方式。信息存取采用 2 线串行接口。这里我们就 24C02 的结构特点 , 其他系列比较类似。

3、结构原理及引脚



AT24C02 有地址线 A0~A2 , 串行数据引脚 SDA , 串行时钟输入引脚 SCL , 写保护引脚 WP 等引脚。很明显 , 其引脚较少 , 对组成的应用系统可以减少布线 , 提高可靠性。

各引脚的功能和意义如下。

VCC 引脚 , 电源+5V。

GND 引脚 , 地线。

SCL 引脚 , 串行时钟输入端。在时钟的正跳沿即上升沿时把数据写入 EEPROM ; 在时钟的负跳沿即下降沿时把数据从 EEPROM 中读出来。

SDA 引脚 , 串行数据 I/O 端 , 用于输入和输出串行数据。这个引脚是漏极开路的端口 , 故可以组成 “ 线或 ” 结构。

A0,A1,A2 引脚 , 是芯片地址引脚。在型号不同时意义有些不同 , 但都要接固定电平。

WP 引脚 , 写保护端。这个端提供了硬件数据保护。当把 WP 接地时 , 允许芯片执行一般读写操作 ; 当把 WP 接 VCC 时 , 则对芯片实施写保护。

NC 引脚 , 无用引脚 , 不接任何电平。

4、存储器的组织及运行

存储器的组织 : 对于不同的型号 , 存储器的组织不一样 , 其关键原因在于存储器容量存在差异。对于 AT24CXX 系列的 EEPROM , 其典型型号的存储器组织如下。

AT24C01A:内部含有 128 个字节 , 故需要 7 位地址对其内部字节进行寻址

AT24C02:内部含有 256 个字节 , 故需要 8 位地址对其内部字节进行读写。

5、运行方式 :

对于时钟及数据传送 , 串行数据 I/O 端口 SDA 一般需要用外部上拉电阻将其电平拉高。加到 SDA 的数据只有在串行时钟 SCL 对于低电平的时间周期内可以改变。当串行时钟 SCL 处于高电平时 , SDA 的数据变化用于指明起始或停止状态。在 SCL 为高电平期间 , 如果 SDA

从低电平上升到高电平，则表示起始状态；如果 SDA 从高电平下降到低电平，则表示停止状态。

起始状态：当 SCL 为高电平时，SDA 由高电平变到低电平则处于起始状态。起始状态应处于任何其他命令之前。

停止状态：当 SCL 处于高电平时，SDA 从低电平变到高电平则处于停止状态。在执行完读序列信号之后，停止命令将把 EEPROM 置于低功耗的备用方式(Standby Mode)。

应答信号：应答信号是由接受数据的器件发出的。当 EEPROM 接受完一个写入数据之后，会在 SDA 上发一个 "0" 应答信号。反之，当单片机接受完来自 EEPROM 的数据后，单片机也应向 SDA 发 ACK 信号。ACK 信号在第 9 个时钟周期时出现。

备用方式(Standby Mode)：AT24C01A/02/04/08/16 都具有备用方式，以保证在没有读写操作时芯片处于低功耗状态。在下面两种情况中，EEPROM 都会进入备用方式：第一，芯片通电的时候；第二，在接到停止位和完成了任何内部操作之后。

AT24C02 等 5 种典型的 EEPROM 在进入起始状态之后，需要一个 8 位的“器件地址字”去启动存储器进行读或写操作。在写操作中，它们有“字节写”，“页面写”两种不同的写入方法。在读操作中，有“现行地址读”，随机读和“顺序读”种各具特点的读出方法。下面分别介绍器件寻址，写操作和读操作。

器件寻址：所谓器件寻址(Device Addressing)就是用一个 8 位的器件地址字(Device Address Word)去选择存储器芯片。在逻辑电路中的 AT24CXX 系列的 5 种芯片种，即 AT24C01A/02/04/08/16 中，如果和器件地址字相比较结果一致，则读芯片被选中。下面对器件寻址的过程和意义加以说明。

芯片的操作地址

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	A2	A1	A0	R/W

用于存储器 EEPROM 芯片寻址的器件地址字如图所示。它有 4 种方式，分别对应于 1K/2K,4K,8K 和 16K 位的 EEPROM 芯片。

从图中看出：器件地址字含有 3 个部分。第一部分是高 4 位，它们称为 EEPROM AT24C01A/02/04/08/16 的标识第二部分称为硬布线地址，它们是标识后的 3 位。第三部分是最低位，它是读/写操作选择位。

第一部分：器件标识，器件地址字的最高 4 位。这 4 位的内容恒为 "1010"，用于标识 EEPROM 器件 AT24C01A/02/04/08/16。

第二部分：硬布线地址，是与器件地址字的最高 4 位相接的低 3 位。硬布线地址的 3 位有 2 种符号： $A_i(i=0\sim 2), P_j(j=0\sim 2)$ 其中 A_i 表示外部硬布线地址位

对于 AT24C10A/02 这两种 1K/2K 位的 EEPROM 芯片，硬布线地址为 "A2,A1,A0"。在应用时，"A2,A1,A0" 的内容必须和 EEPROM 芯片的 A2,A1,A0 的硬布线情况，即逻辑连接情况相比较，如果一样，则芯片被选中；否则，不选中。

AT24C01A/02:真正地址=字地址

第三部分：读/写选择位，器件地址字的最低位，并用 R/W 表示。当 R/W=1 时，执行读操作；当 R/W=0 时，执行写操作。

当 EEPROM 芯片被选中时，则输出 "0"；如果 EEPROM 芯片没有被选中，则它回到备用方式。被选中的芯片。其以后的输入，输出情况视写入和读出的内容而定。

写操作：AT24C01A/02/04/08/16 这 5 种 EEPROM 芯片的写操作有 2 种：一种是字节写，另一种是页面写。

字节写

这种写方式只执行 1 个字节的写入。字节写的过程如图所示，其写入过程分外部写和内部写两部分，分别说明如下。

在起始状态中，首先写入 8 位的器件地址。则 EEPROM 芯片会产生一个 "0" 信号 ACK 输出作为应答；接着，写入 8 位的字地址，在接受了字地址之后，EEPROM 芯片又产生一个 "0" 应答信号 ACK；随后，写入 8 位数据，在接受了数据之后，芯片又产生一个 "0" 信号 ACK 作为应答。到此为止，完成了一个字节写过程，故应在 SDA 端产生一个停止状态，这是外部写过程。

在这个过程中，控制 EEPROM 的单片机应在 EEPROM 的 SCL，SDA 端送入恰当的信号。当然在一个字节写过程结束时，单片机应以停止状态结束写过程。在这时，EEPROM 进入内部定时的写周期，以便把接受的数据写入到存储单元中。在 EEPROM 的内部写周期中，其所有输入被屏蔽，同时不响应外部信号直到写周期完成。这是内部写过程。内部写过程大约需要 10ms 时间。内部写过程处于停止状态与下一次起始状态之间，页面写

这种写入方式执行含若干字节的 1 个页面的写入。对于 AT24C01A/02，它们的 1 个页面含 8 个字节；页面写的开头部分和字节写一样。在起始状态，首先写入 8 位器件地址；待 EEPROM 应答了 "0" 信号 ACK 之后，写入 8 位字地址；又待芯片应答了 "0" 信号 ACK 之后，写入 8 位数据。

随后页面写的过程则和字节写有区别。

当芯片接受了第一个 8 位数据并产生应答信号 ACK 之后，单片机可以连续向 EEPROM 芯片发送共 1 页面的数据。对于 AT24C01A/02，可发送共 1 个页面的 8 个字节（连第一个 8 位数据在内）。对于 AT24C04/08/16，则共可发送 1 个页面共 16 个字节（连第一个 8 位数据在内）。当然，每发一个字节都要等待芯片的应答信号 ACK。

之所以可以连续向芯片发送 1 个页面数据，是因为字地址的低 3~4 位在 EEPROM 芯片内部可实现加 1，字地址的高位不变，用于保持页面的行地址。页面写和字节写两者一样可，都分为外部写和内部写过程。

应答查询：应答查询是单片机对 EEPROM 各种状态的一种检测。单片机查询到 EEPROM 有应答 "0" 信号 ACK 输出，则说明其内部定时写的周期结束，可以写入新的内容。单片机是通过发送起始状态及器件地址进行应答查询的。由于器件地址可以选择芯片，则检测芯片送出到 SDA 的状态就可以知道其是否有应答了。

读操作：读操作的启动是和写操作类同的。它一样需要图所示的器件地址字。和写操纵不同的就是信号为时执行读操作。

读操纵有 3 种方式，即现行地址读，随机读和顺序读。下面分别说明它们的工作过程。

现行地址读

在上次读或写操纵完成之后，芯片内部字地址计数器会加 1，产生现行地址。只要没有再执行读或写操作，这个现行地址就会在 EEPROM 芯片保持接电的期间一直保存。一旦器件地址选中 EEPROM 芯片，并且有 R/W=1，则在芯片的应答信号 ACK 之后把读出的现行地址的数据送出。现行地址的数据输出时，就由单片机一位一位接受，接收后单片机不用向 EEPROM 发应答信号 ACK "0" 电平，但应保证发出停止状态的信号以结束现行地址读操作。现行地址读会产生地址循环覆盖现象，但和写操纵的循环覆盖不同。在写操纵中，地址的循环覆盖是现行页面的最后一个字节写入之后，再行写入则覆盖同一页面的第一个字节。而在现行地址读操纵中，地址的循环覆盖是在最后页面的最后一个字节读出之后，再行读出才覆盖第一个页面的第一个字节。

随机读

随机读和现行地址读的最大区别在于随机读会执行一个伪写入过程以把字地址装入 EEPROM 芯片中，然后执行读出，

显然，随机读有 2 个步骤。

执行伪写入——把字地址送入 EEPROM，以选择需读的字节。

执行读出——根据字地址读出对应内容。

当 EEPROM 芯片接收了器件地址及字地址时，在芯片产生应答信号 ACK 之后，单片机必须再产生一个起始状态，执行现行地址读，这时单片机再发出器件地址并且令 R/W=1，则 EEPROM 应答器件地址并行输出被读数据。在数据读出时由单片机执行一位一位接收，接收完毕后，单片机不用发“0”应答信号 ACK，但必须产生停止状态以结束随机读过程。

应该注意：在随机读的第二个步骤是执行现行地址读的，由于第一个步骤时芯片接收了字地址，故现行地址就是所送入的字地址。

顺序读

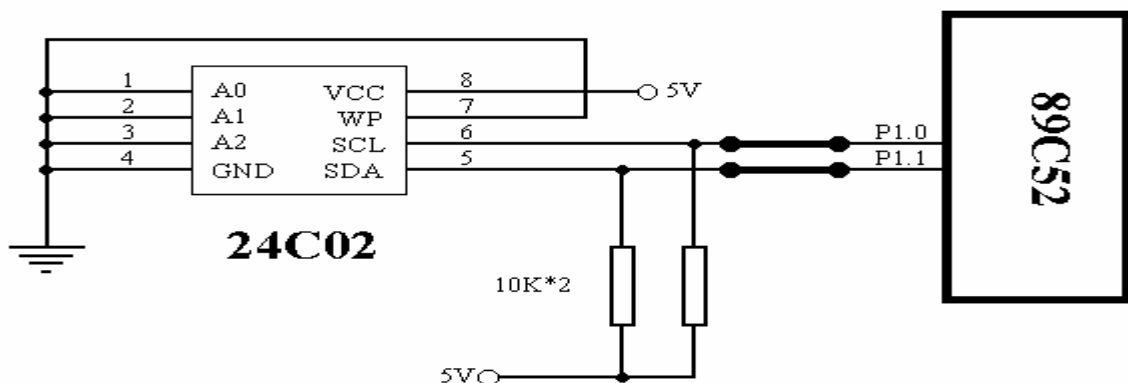
顺序读可以用现行地址读或随机读进行启动。它和现行地址读。随机读的最大区别在于：顺序读在读出一批数据之后才由单片机产生停止状态结束读操作；而现行地址读和随机读在读出一个数据之后就由单片机产生停止状态结束读操作。

执行顺序读时，首先执行现行读或随机读的有关过程，在读出第一个数据之后，单片机输出“0”应答信号 ACK。在芯片接收应答信号 ACK 后，就会对字地址进行计数加 1，随后串行输出对应的字节。当字地址计数达到存储器地址的极限时，则字地址会产生覆盖，顺序读将继续进行。只有在单片机不再产生“0”应答信号 ACK，而在接收数据之后马上产生停止状态，才会结束顺序读操作。

在对 AT24CXX 系列执行读写的 2 线串行总线工作中，其有关信号是由单片机的程序和 EEPROM 产生的。有两点特别要记住：串行时钟必须由单片机程序产生，而应答信号 ACK 则是由接收数据的器件产生，也就是写地址或数据时由 EEPROM 产生 ACK，而读数据时由单片机产生。

(4) AT24CXX 系列应用注意事项

AT24CXX 系列型号：AT24CXX 系列 EEPROM 有 13 种型号。它们的容量不同，执行页历写时的页历定义不同，进行读写时的地址位数也不同，器件地址不同。有关主要指标在应用中要加以区别和注意。



实验程序

1、写把 CPU 的 RAM 31H ~ 38H 的数据送到 24C02 的 00H ~ 07H 单元

CPU 写数据到 24C02 的数据格式如下：

S	从控制器的地址 +W	A	写入单元地 址	A	DATA 1	A	DATA2	A	...	A	DATA N	A	P
---	---------------	---	------------	---	-----------	---	-------	---	-----	---	-----------	---	---

其中：S 表起始条件，A 表示应答响应，P 表示停止条件

； 文件名称：W24C02.ASM

```

SCL EQU P1.0
SDA EQU P1.1
ORG 0000H
WR_EEROM: MOV R6,#40H
MOV 30H,#00H
MOV R0,#30H
W_LOOP:    LCALL WR_DATA
W_LOOP2:  INC @R0
DJNZ R6,W_LOOP1
SJMP STOP
W_LOOP1:  LCALL WR_DATA2
SJMP W_LOOP2
STOP:     LCALL STOP24
SJMP $

;-----
WR_DATA:  LCALL START
MOV A,#0A0H
LCALL WBYTE
WR_DATA1: MOV A,@R0
LCALL WBYTE
WR_DATA2: MOV A,#99H
LCALL WBYTE
RET

;-----
WBYTE:   NOP
MOV R3,#08H
WBY0:   CLR SCL
RLC A
MOV SDA,C
SETB SCL
DJNZ R3,WBY0
CLR SCL
NOP
SETB SCL
NOP
JB SDA,$
CLR SCL
NOP
RET

;-----
START:  CLR    SCL
NOP
SETB   SDA
NOP

```

```

SETB SCL
NOP
CLR SDA
NOP
CLR SCL
RET

```

```

;-----
STOP24: CLR SCL
NOP
CLR SDA
NOP
SETB SCL
NOP
SETB SDA
NOP
CLR SCL
RET

```

END

2、读 24C02 的 00H ~ 07H 到 CPU 的 RAM 31H ~ 38H 单元。
CPU 从 24C02 内读数据格式如下:

S	从控制器的地址	A	写入读单元地	S	从控制器的地址+R	A	Data	A	...	Data	A	P
	+W		址				1			n		

其中: S 表起始条件, A 表示应答响应, P 表示停止条件

;文件名称:R24C02.ASM

```

SCL EQU P1.0
SDA EQU P1.1
ORG 0000H
RD_EEROM: MOV R6,#08H
MOV 30H,#00H
MOV R0,#30H
MOV R1,#40H
R_LOOP: LCALL START
MOV A,#0A0H
LCALL WBYTE
MOV A,@R0
LCALL WBYTE
R_LOOP1: LCALL START
MOV A,#0A1H
LCALL WBYTE
LCALL RBYTE
MOV @R1,A
INC R1
DJNZ R6,R_LOOP1

```

```
LCALL STOP24
SJMP $
```

```
;-----
```

```
RBYTE: NOP
```

```
MOV R3,#08H
RBY0: CLR SCL
NOP
SETB SCL
NOP
MOV C,SDA
RLC A
DJNZ R3,RBY0
CLR SCL
NOP
SETB SDA
NOP
SETB SCL
RET
```

```
;-----
```

```
WBYTE: MOV R3,#08H
```

```
WBY0: CLR SCL
NOP
RLC A
MOV SDA,C
SETB SCL
DJNZ R3,WBY0
CLR SCL
NOP
SETB SCL
NOP
JB SDA,$
CLR SCL
NOP
RET
```

```
;-----
```

```
START: CLRSCL
```

```
NOP
SETB SDA
NOP
SETB SCL
NOP
CLRSDA
NOP
NOP
```

```

NOP
CLR SCL
RET
;-----
STOP24: CLR SCL
NOP
CLR SDA
NOP
SETB SCL
NOP
SETB SDA
NOP
CLR SCL
RET
END

```

实验 7 串行通讯

可以通过串行通信电缆与计算机进行通讯，实现数据信息的交换。其引脚说明如下：

引脚	名称	功能
2	RXD	接 PC 机的 RXD 信号线
3	TXD	接 PC 机的 TXD 信号线
5	GND	信号地
1、4、6 7、8、9	空	未用

串口接收程序如下：

```

;-----
ORG 0000H
JMP START
START: MOV SP,#60H ; 设定堆栈区
MOV SCON,#01010000B ; 设定串行方式 MODE1,接收时 REN=1
MOV TMOD,#20H ; 设定定时器 1 为 模式 2
ORL PCON,#10000000B ; SMOD=1
MOV TH1,#0F4H ; 设定波特率为 4800
SETB TR1 ; 定时器 1,开始计时
MOV R0,#30H
MOV R7,#05H
AGAIN: JNB RI,$
CLR RI
MOV A,SBUF
MOV @R0,A
INC R0

```

```

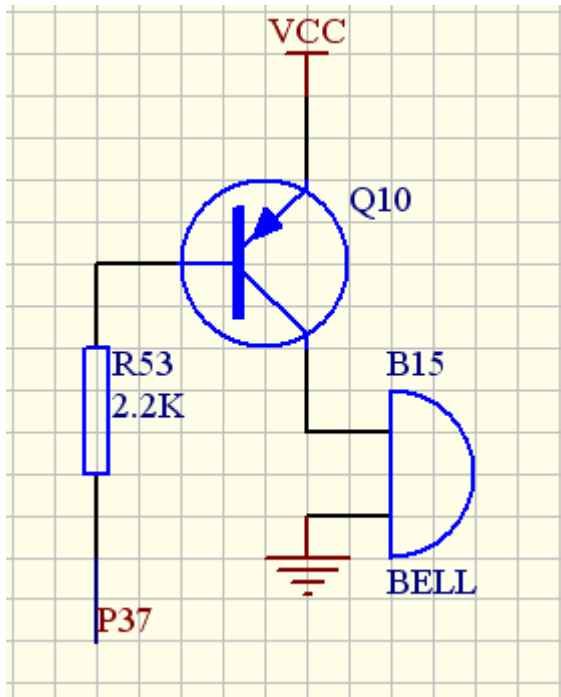
        DJNZ    R7,AGAIN
        RET
    END
;-----
串口发送程序如下：
;-----
        ORG    0000H
        JMP    START
START:   MOV    SP,#60H        ; 设定堆栈区
        MOV    SCON,#01010000B ; 设定串列方式 MODE1,接收时 REN=1
        MOV    TMOD,#20H      ; 设定定时器 1 为 模式 2
        ORL    PCON,#10000000B ; SMOD=1
        MOV    TH1,#0F4H      ; 设定波特率为 4800
        SETB   TR1            ; 定时器 1 ,开始计时
        MOV    A,#30H         ; 0..9 的 ASCII 码为 30H...39H
AGAIN:   MOV    SBUF,A         ; 送发送缓冲区
        JNB    TI,$           ; 等待发送完成
        CLR    TI
        INC    A              ; 发送下一个
        CJNE   A,#3AH, AGAIN
        RET
    END
;-----

```

实验 8 音乐的应用

硬件设计与基本概念

要产生音频脉冲，只要算出某一音频的周期（ $1/\text{频率}$ ），然后将此周期除以 2，即为半周期的时间。利用定时器计时这个半周期时间，每当计时到后就将输出脉冲的 I/O 反相，然后重复计时此半周期时间再对 I/O 反相，就可在 I/O 脚上得到此频率的脉冲。利用单片机内部定时器使其工作在计数器模式 MODE1 下，改变计数值 TH0 及 TL0 以产生不同频率的方法。



电路如图所示，蜂鸣器的动作由 P3.7 控制。当输出低电平时蜂鸣器发出响声，输出高电平时不响，我们只要控制蜂鸣器输出高低电平的时间和变化频率就可以让蜂鸣器发出悦耳的音乐。

实例程序如下：

```

*****
LJD-SY-5100\example\Buzz\happybirthday.asm
*****

ORG 0000H
JMP START
ORG 000BH
JMP TIM0
START:MOV TMOD,#01H
      MOV IE,#82H
START0:MOV 30H,#00H
NEXT:  MOV A,30H
      MOV DPTR,#TABLE
      MOVC A,@A+DPTR
      MOV R2,A
      JZ END0
      ANL A,#0FH
      MOV R5,A
      MOV A,R2
      SWAP A
      ANL A,#0FH
      JNZ SING
      CLR TR0
      JMP D1

```

```

SING:  DEC A
      MOV 22H,A
      RL  A
      MOV DPTR,#TABLE1
      MOVC A,@A+DPTR
      MOV TH0,A
      MOV 21H,A
      MOV A,22H
      RL  A
      INC A
      MOVC A,@A+DPTR
      MOV TL0,A
      MOV 20H,A
      SETB TR0
D1: CALL DELAY
      INC 30H
      JMP NEXT
END0:      CLR  TR0
      JMP START0
TIM0:      PUSH ACC
      PUSH PSW
      MOV  TH0,21H
      MOV  TL0,20H
      CPL  P3.7
      POP  PSW
      POP  ACC
      RETI
DELAY:      MOV  R7,#02
D2:  MOV  R4,#187
D3:  MOV  R3,#248
      DJNZ R3,$
      DJNZ R4,D3
      DJNZ R7,D2
      DJNZ R5,DELAY
      RET
TABLE1:
      DW 64260,64400,64524,64580
      DW 64684,64777,64820,64898
      DW 64968,65030,65058,65110
      DW 65157,65178,65217
TABLE:
      DB 82H,01H,81H,94H,84H
      DB 0B4H,0A4H,04H
      DB 82H,01H,81H,94H,84H

```

DB 0C4H,0B4H,04H
DB 82H,01H,81H,0F4H,0D4H
DB 0B4H,0A4H,94H
DB 0E2H,01H,0E1H,0D4H,0B4H
DB 0C4H,0B4H,04H
DB 82H,01H,81H,94H,84H
DB 0B4H,0A4H,04H
DB 82H,01H,81H,94H,84H
DB 0C4H,0B4H,04H
DB 82H,01H,81H,0F4H,0D4H
DB 0B4H,0A4H,94H
DB 0E2H,01H,0E1H,0D4H,0B4H
DB 0C4H,0B4H,04H
DB 00H
END