第6章

单片机基本扩展技术

6.1 51单片机最小系统

6.1.1 片内有ROM型单片机最小系统

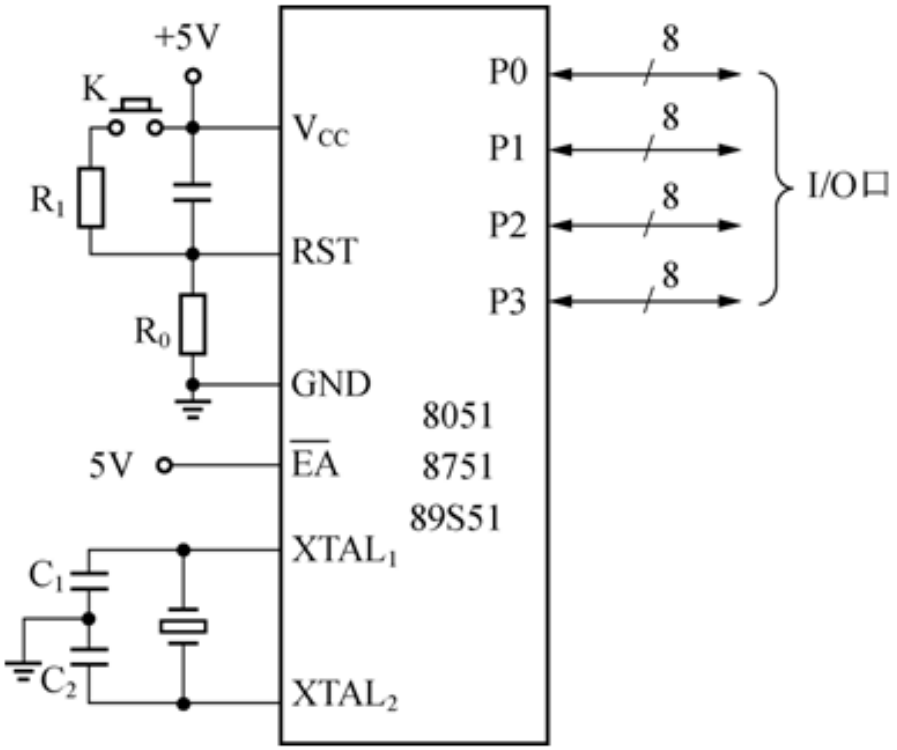


图 6-1　8051/8751/89S51 最小应用系统

6.1 51单片机最小系统

6.1.2 片内无ROM型单片机最小系统

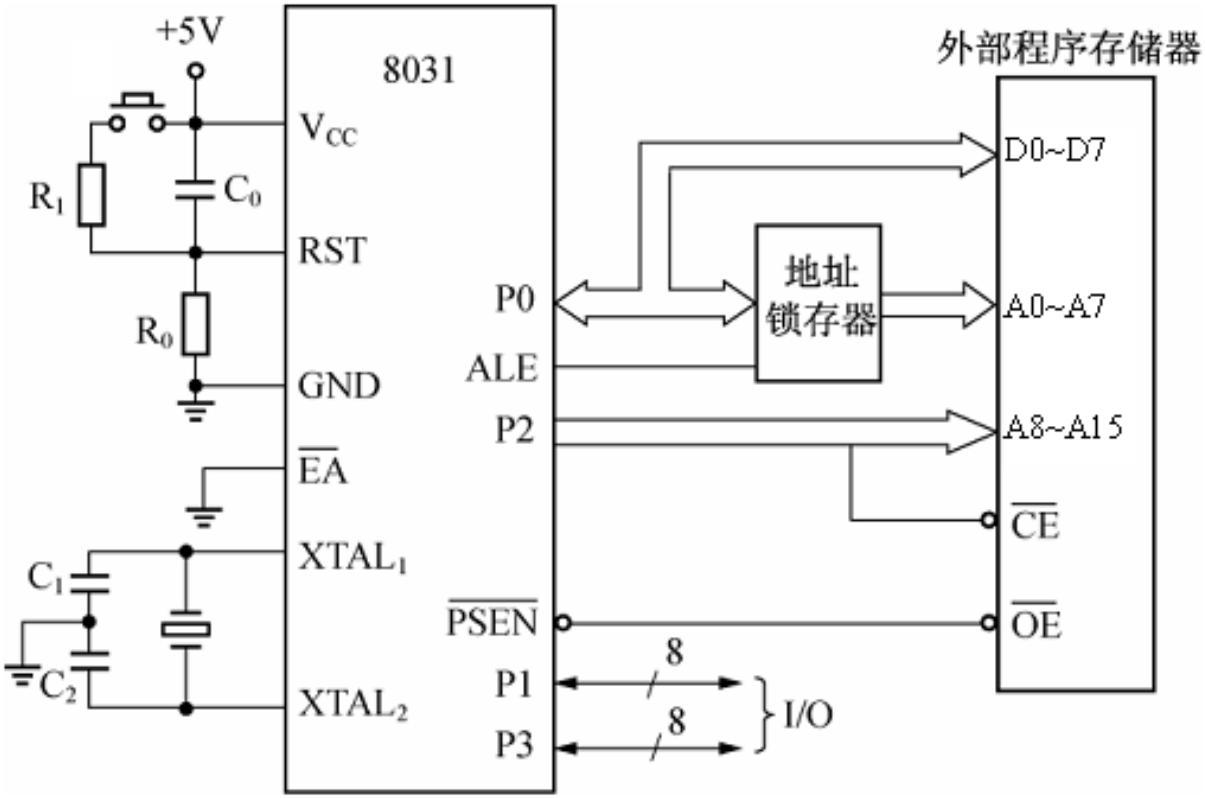


图 6-2　无 ROM 型单片机最小应用系统

6.1 51单片机最小系统

6.1.3 单片机系统总线

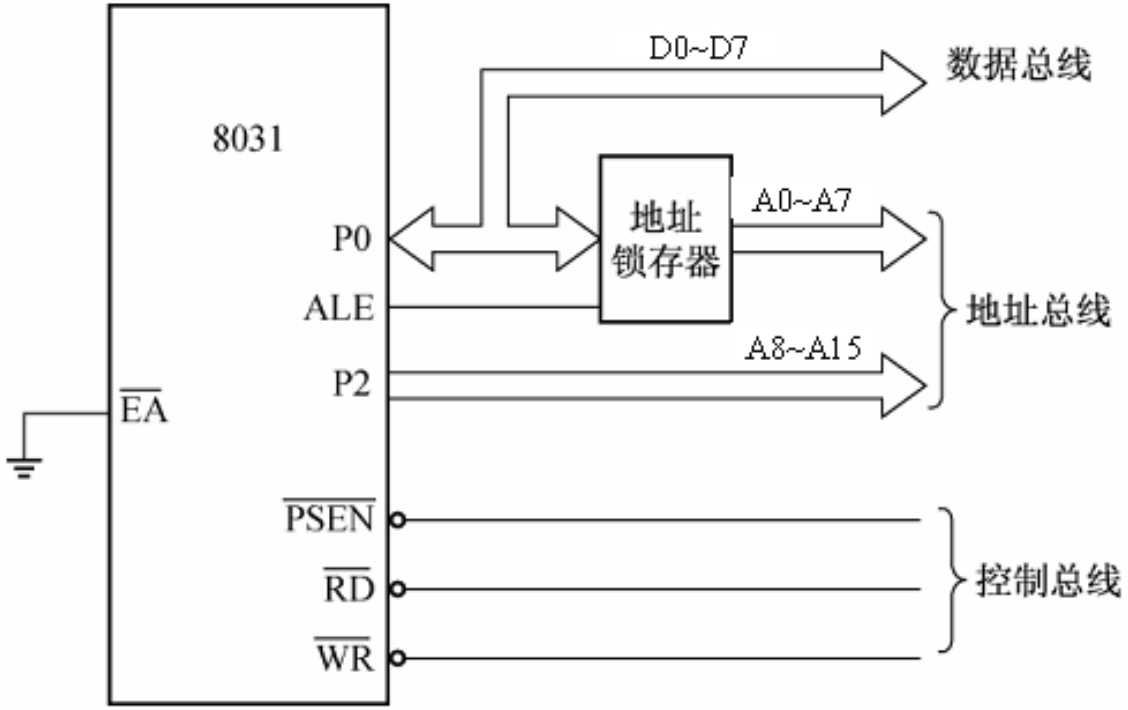


图 6-3　单片机的系统总线

6.2 存储器的扩展

6.2.1 单片机常用接口存储器的分类

（1）掩膜ROM。

（2）可编程ROM（PROM）。

（3）可擦除ROM （EPROM或EEPROM）。

（4）Flash ROM。

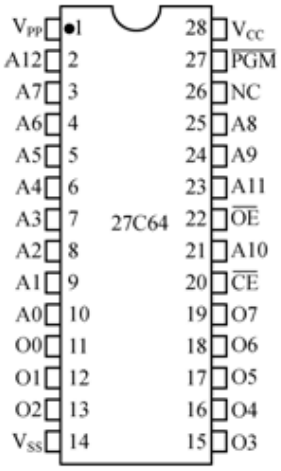
6.2 存储器的扩展

6.2.2 程序存储器的扩展

1. 常用的27和27C系列EPROM

表 6-1 常用的 27/27C 系列 EPROM 引脚排列情况

27512	27256	27128A	2732A	2716		2716	2732A	27128A	27256	27512
27C512	27C256	27C128						27C128	27C256	27C512
A15	V_{PP}	V_{PP}						V_{CC}	V_{CC}	V_{CC}
A12	A12	A12						$\overline{PGM}$	A14	A14
A7	A7	A7	A7	A7		V_{CC}	V_{CC}	A13	A13	A13
A6	A6	A6	A6	A6		A8	A8	A8	A8	A8
A5	A5	A5	A5	A5		A9	A9	A9	A9	A9
A4	A4	A4	A4	A4		A11	A11	A11	A11	A11
A3	A3	A3	A3	A3		$\overline{OE}$	$\overline{OE}$ /V_{PP}	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$ /V_{PP}
A2	A2	A2	A2	A2		A10	A10	A10	A10	A10
A1	A1	A1	A1	A1		$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{OE}$
A0	A0	A0	A0	A0		O7	O7	O7	O7	O7
O0	O0	O0	O0	O0		O6	O6	O6	O6	O6
O1	O1	O1	O1	O1		O5	O5	O5	O5	O5
O2	O2	O2	O2	O2		O4	O4	O4	O4	O4
GND	GND	GND	GND	GND		O3	O3	O3	O3	O3



6.2 存储器的扩展

6.2.2 程序存储器的扩展

2．EPROM 2764/27C64

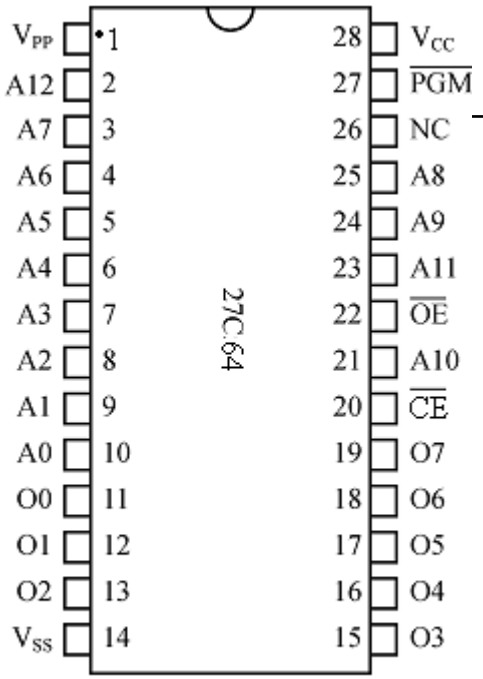


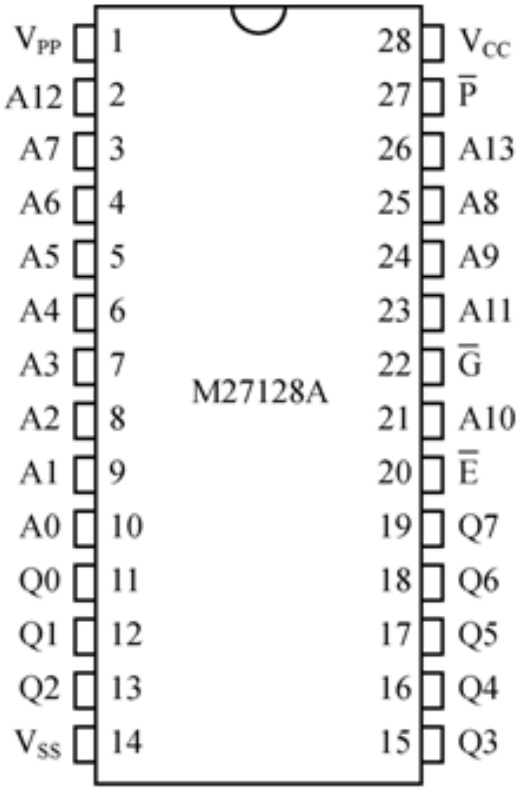
图 6-4　EPROM 2764 引脚图

表 6-2　EPROM 2764 引脚功能说明

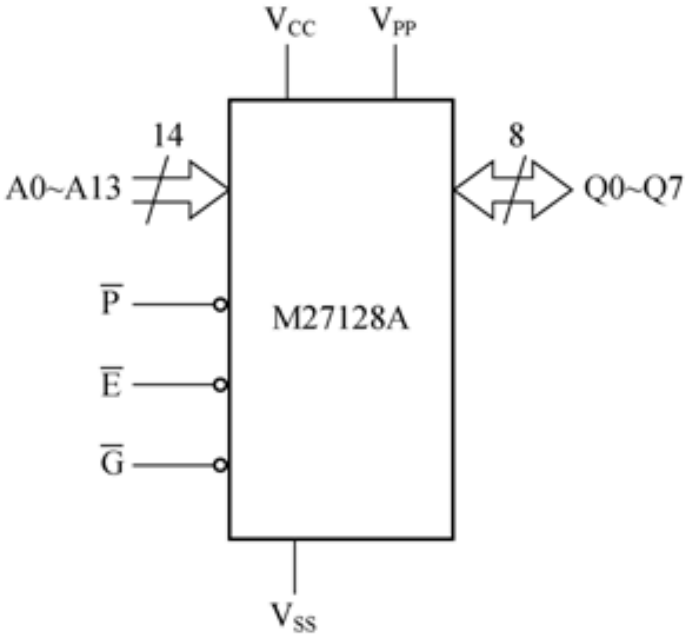
名　　称	功　　能	名　　称	功　　能
A0～A12	地址输入	O0～O7	数据输出
$\overline{CE}$	芯片使能	V_{cc}	+5V 电源
$\overline{OE}$	输出使能	V_{ss}	地线
$\overline{PGM}$	编程使能	NC	不连接
V_{PP}	+25V 或+12V 编程电压		

6.2 存储器的扩展

3. EPROM 27128/27C128



（a）引脚图



（b）逻辑图

图 6-5　EPROM 27128 的引脚与逻辑图

6.2 存储器的扩展

3. EPROM 27128/27C128

表 6-3 EPROM 27128 引脚功能说明

名　　称	功　　能
A0～A13	地址输入
$\overline{CE}$	芯片使能
$\overline{OE}$	输出使能
$\overline{PGM}$	编程使能
V_{PP}	+25V 或+12V 编程电压
Q0～Q7	数据输出
V_{cc}	+5V 电源
V_{ss}	地线
NC	不连接

6.2 存储器的扩展

4．程序存储器的控制信号

- ALE：用于低 8 位地址锁存控制信号。
- $\overline{\text{PSEN}}$：片外程序存储器选通控制信号。常直接连接 EPROM 的 $\overline{\text{OE}}$ 脚。
- $\overline{\text{EA}}$：片内或外程序存储器访问的控制信号。当 $\overline{\text{EA}}$ =1 时，允许访问片内程序存储器；当 $\overline{\text{EA}}$ =0 时，允许访问片外程序存储器。

6.2 存储器的扩展

5．程序存储器与单片机的连接

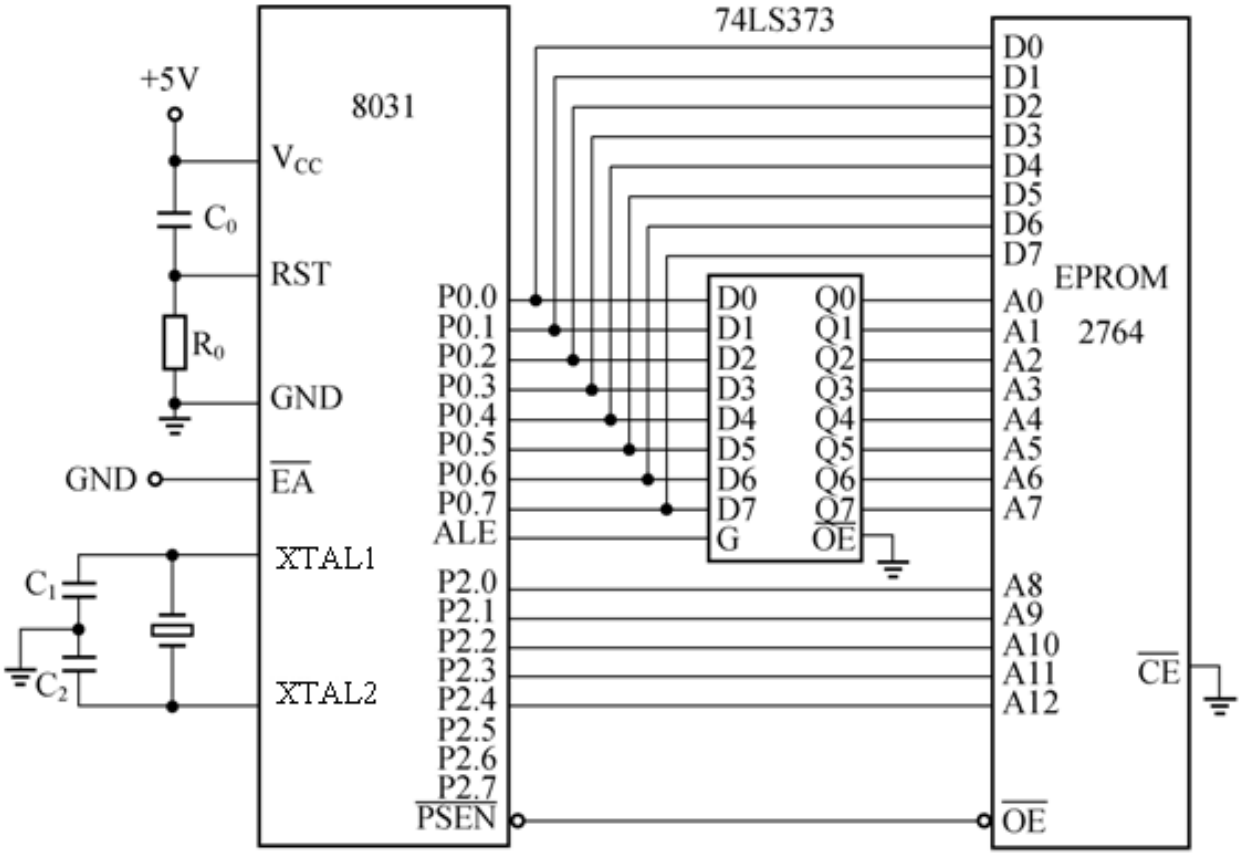


图 6-6　用 2764 扩展 8KB 程序存储器

6.2 存储器的扩展

5．程序存储器与单片机的连接

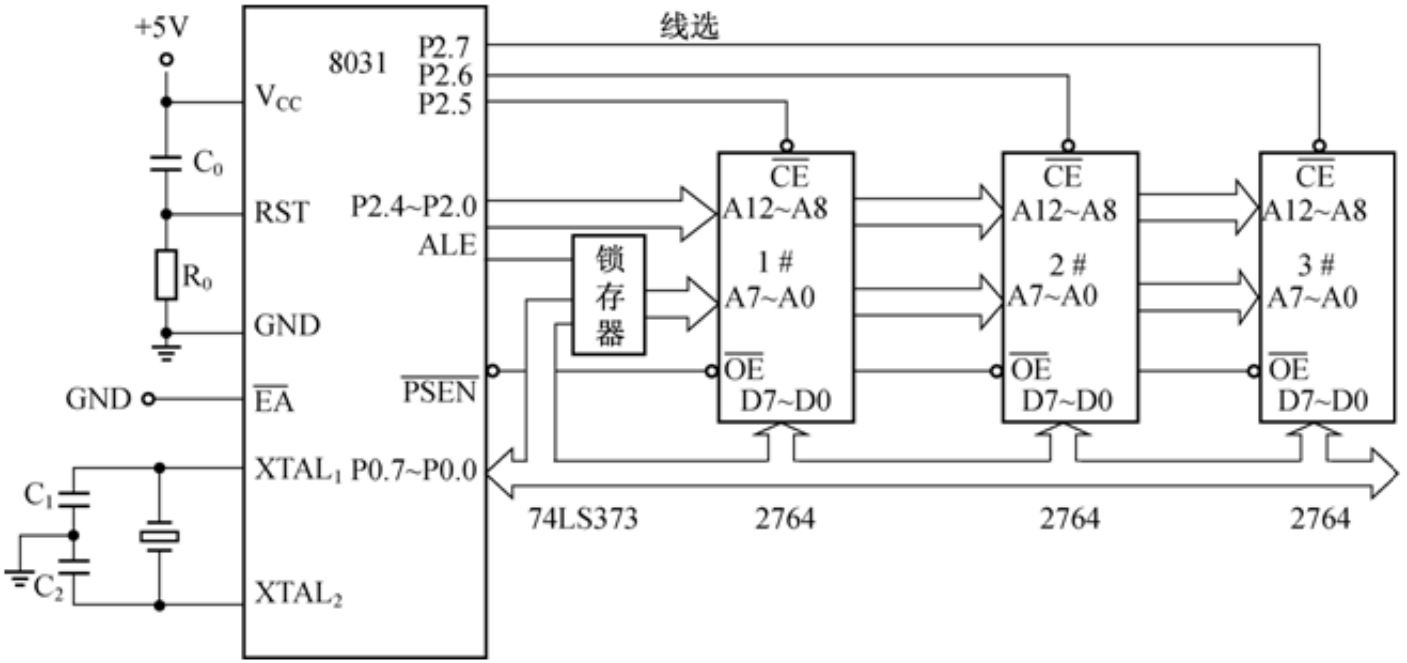


图 6-7　采用线选法的存储器扩展电路

6.2 存储器的扩展

5．程序存储器与单片机的连接

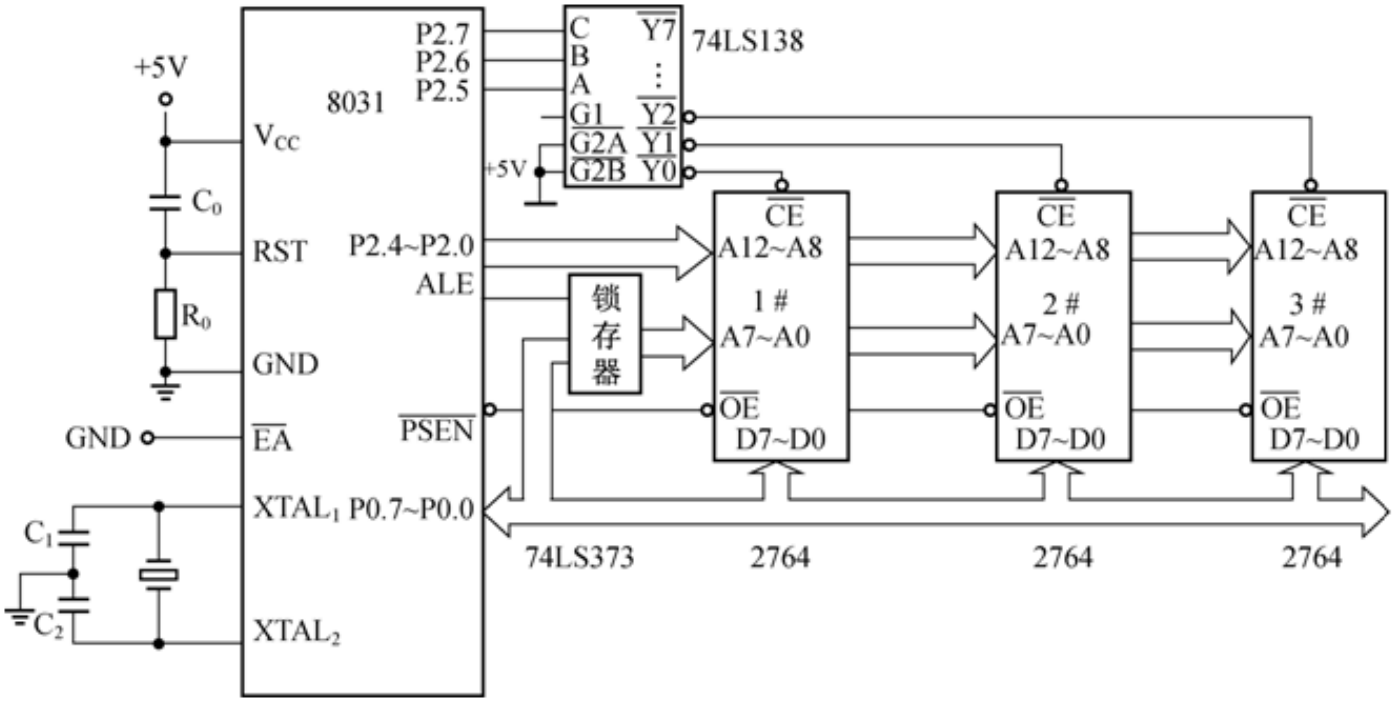


图 6-8　全译码存储器扩展电路

6.2 存储器的扩展

5．程序存储器与单片机的连接

表 6-4　地址线与各 EPROM 2764 的选通关系

端口名称	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
地址	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1#	0	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×
2#	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×
3#	0	1	0	×	×	×	×	×	×	×	×	×	×	×	×	×

6.2 存储器的扩展

5．程序存储器与单片机的连接

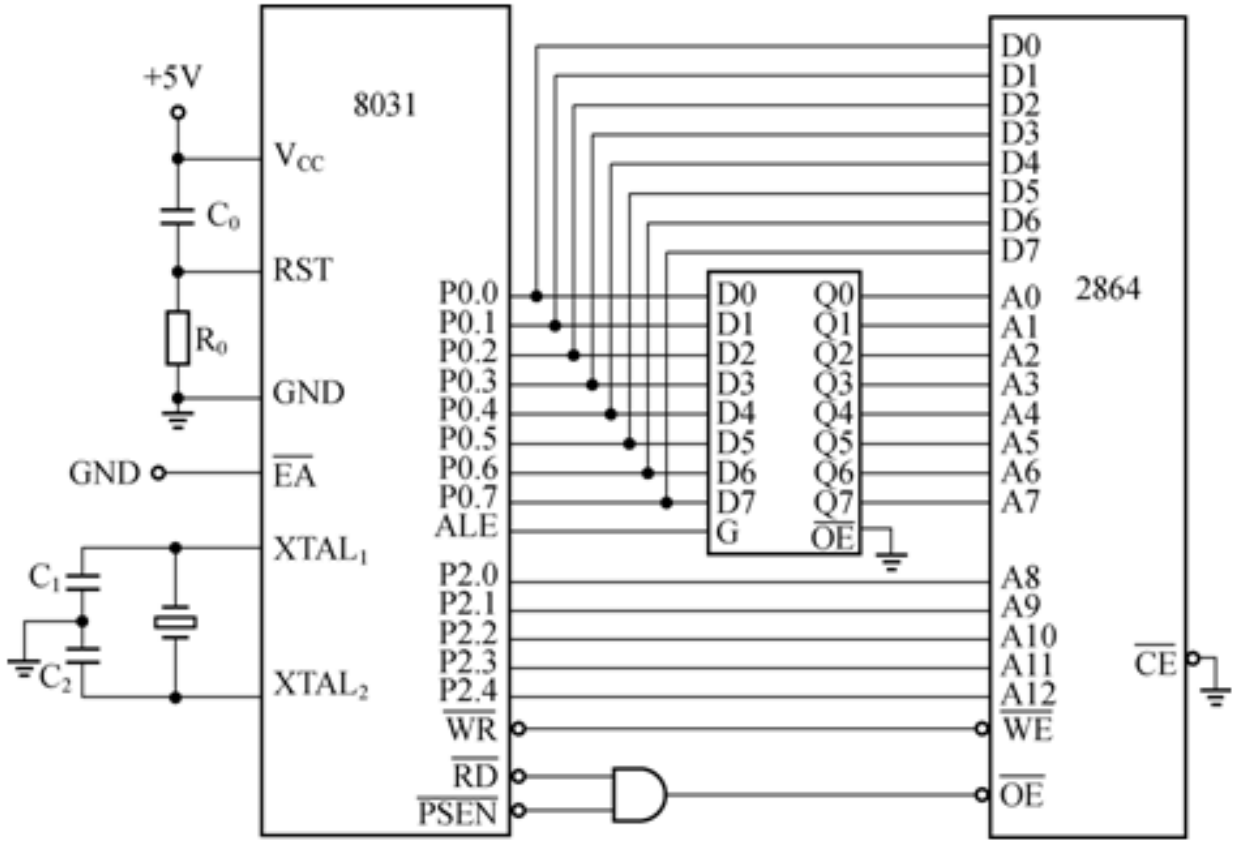


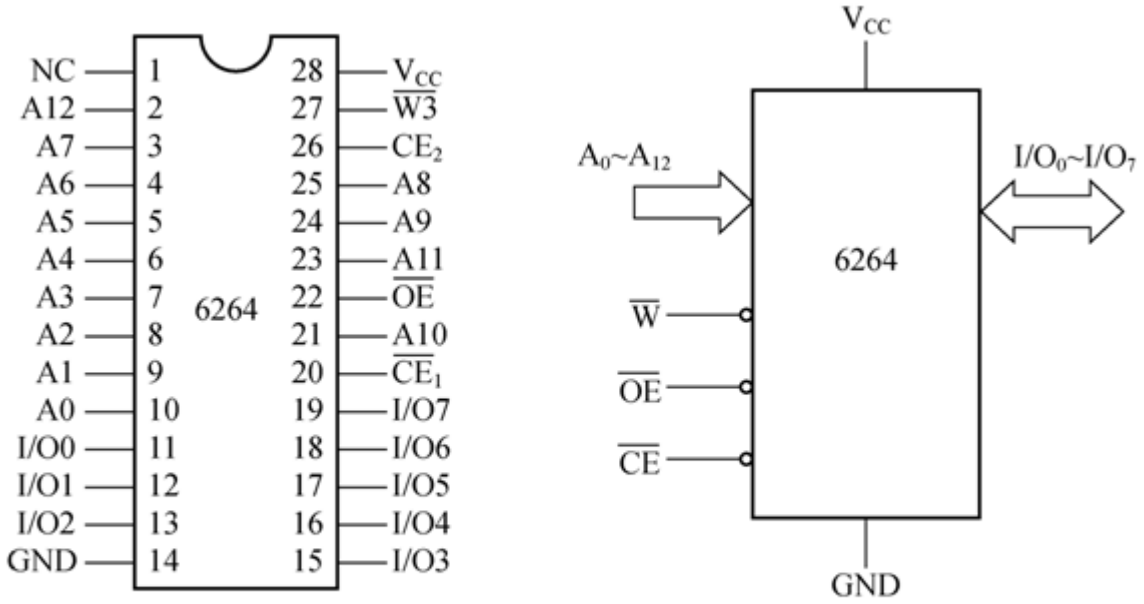
图 6-9　EEPROM 与 8031 的连接

6.2 存储器的扩展

6.2.3 随机存储器RAM的扩展

1. 常用静态数据存储器芯片

2. SRAM 6264的组成特点



（a）引脚图　　（b）逻辑图

图 6-10　SRAM 6264 引脚图和逻辑图

6.2 存储器的扩展

2. SRAM 6264的组成特点

表 6-5　6264 的工作方式

CE_2	$\overline{CE_1}$	$\overline{WE}$	$\overline{OE}$	方　式	功　能
1	0	0	0	禁止	不允许同时为低电平
1	0	1	0	读出	读出数据
1	0	0	1	写入	写入数据
1	0	1	1	选通	选通，输出高阻态
1	1	×	×	未选通	输出高阻态

6.2 存储器的扩展

3．单片机外部数据存储器的扩展

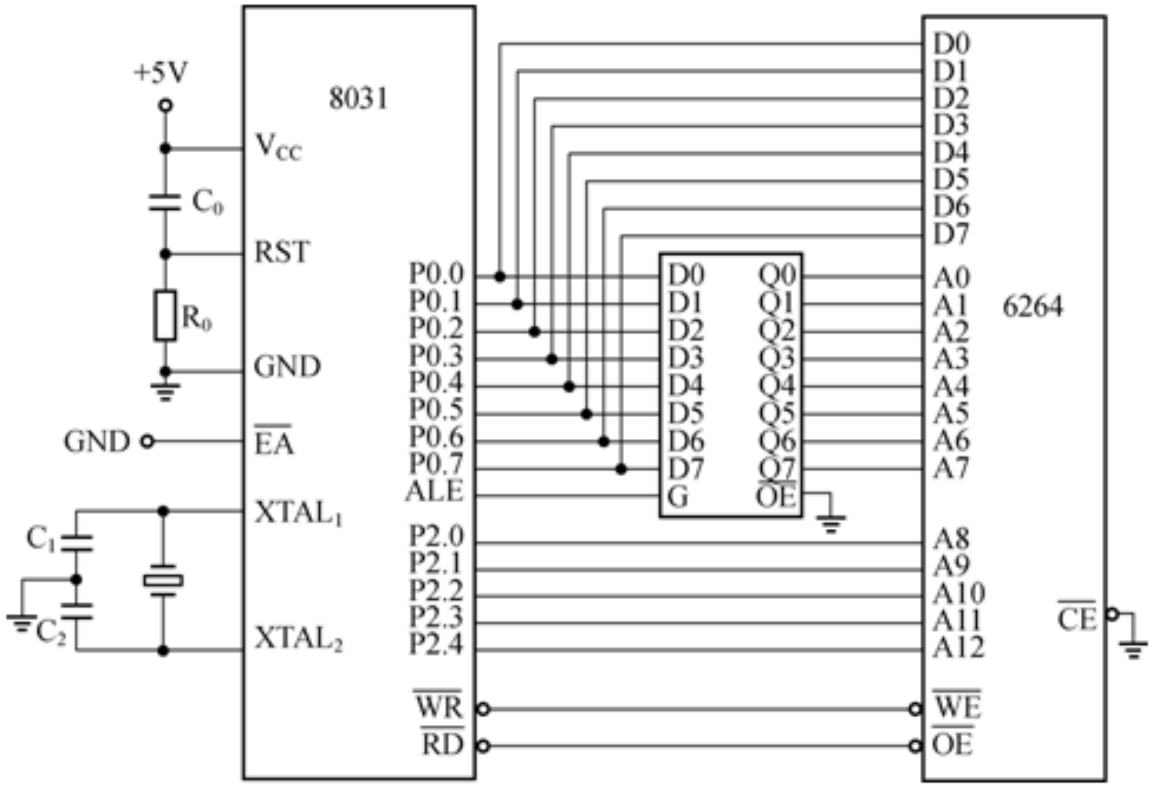


图 6-11　8031 单片机与 6264 的连接

6.2 存储器的扩展

4．扩展数据存储器的软件调试方法

【程序 6-1】

```
    MOV     DPTR，#ADRI       ；ADRI为某单元地址
    MOV     A，#DATA          ；DATA为验证数据
    MOVX    @DPTR ，A         ；写验证数据
    MOVX    A，@DPTR          ；读验证数据
    XRL     A，#DATA          ；验证数据比较
    JNZ     EROOR
    …                         ；正确
EROOR:  …                     ；错误
```

6.2 存储器的扩展

4．扩展数据存储器的软件调试方法

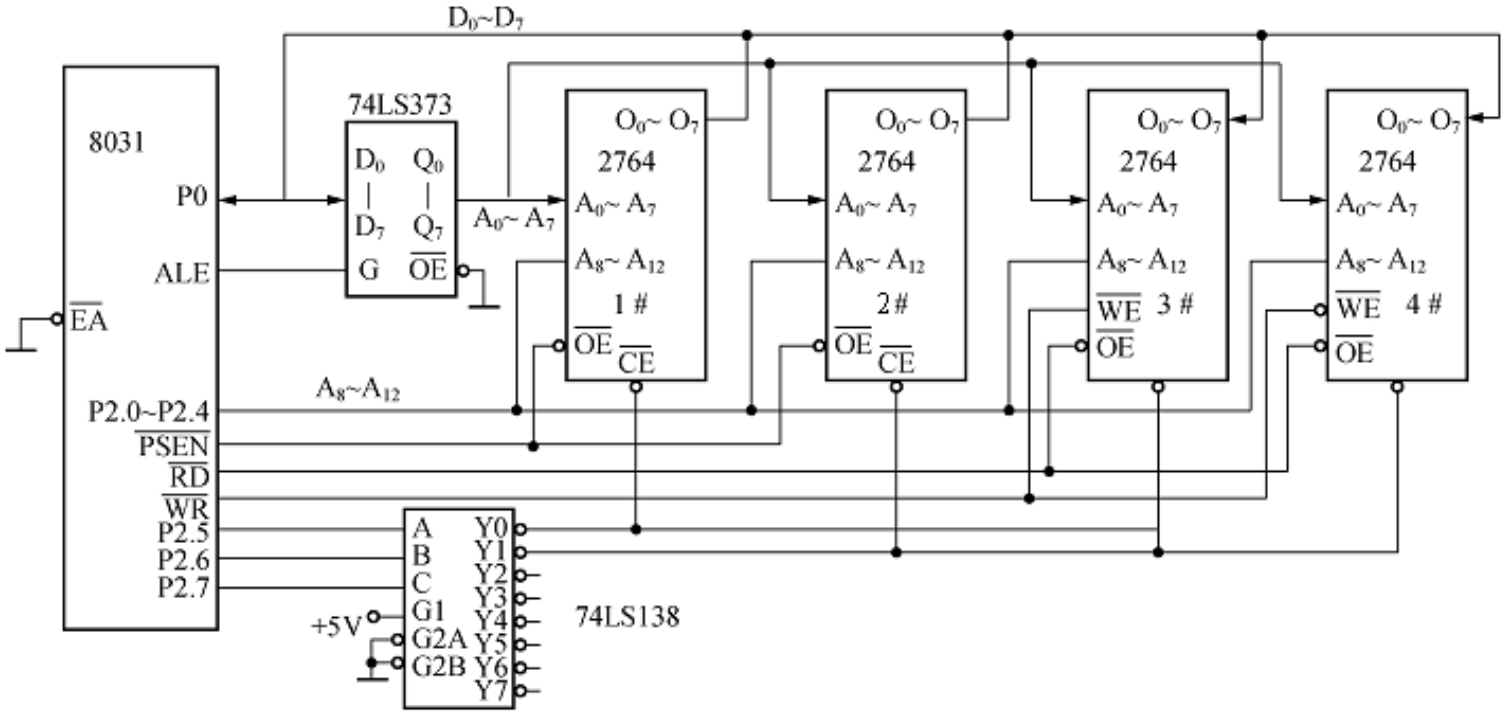


图 6-12　EPROM 和 SRAM 存储器扩展电路

6.2 存储器的扩展

4．扩展数据存储器的软件调试方法

表 6-6　地址线与各存储器选通关系

端口名称	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
地址	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1#（Y0）	0	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×
2#（Y1）	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×
3#（Y0）	0	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×
4#（Y1）	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×

6.3 单片机并行I/O扩展

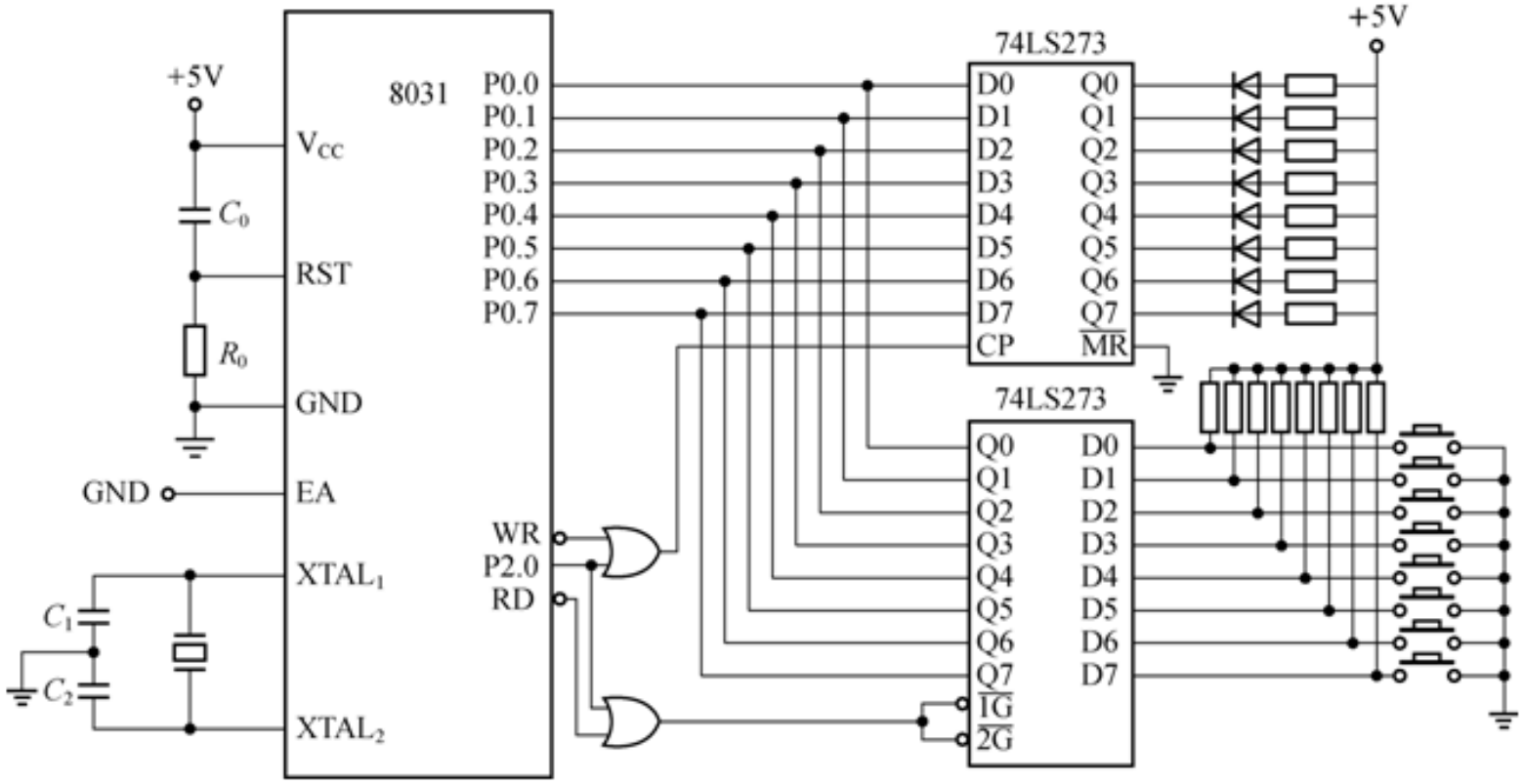


图 6-13　用 TTL 器件扩展简单 I/O 口

6.4 单片机键盘接口技术

6.4.1 按键抖动问题

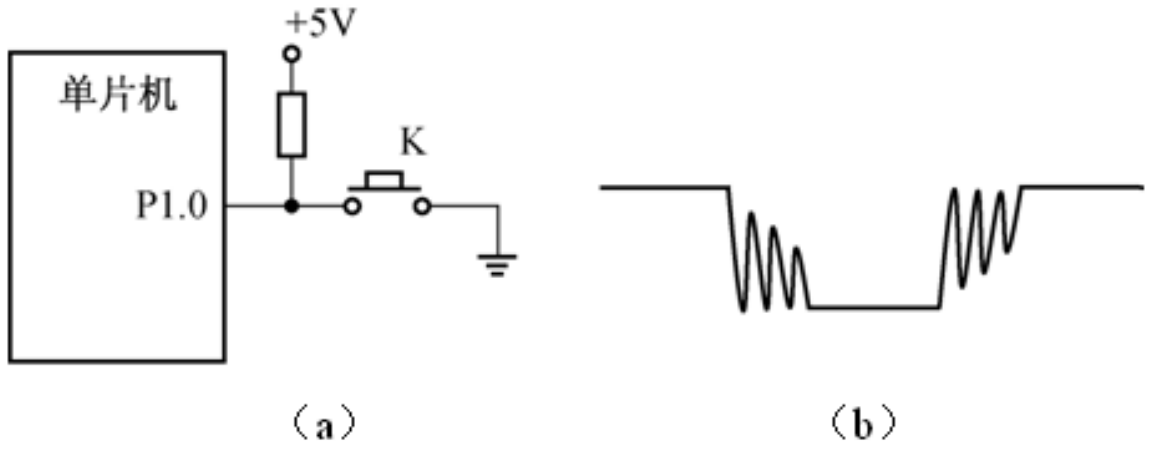


图 6-14　按键时的抖动情况

6.4 单片机键盘接口技术

6.4.2 独立式按键接口和键盘消抖动程序编写

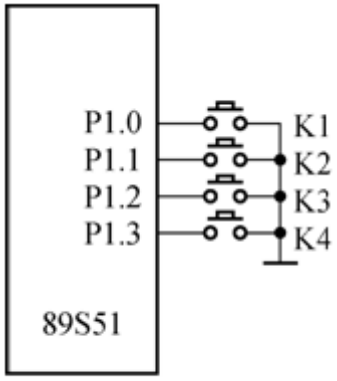


图 6-15　独立按键接口

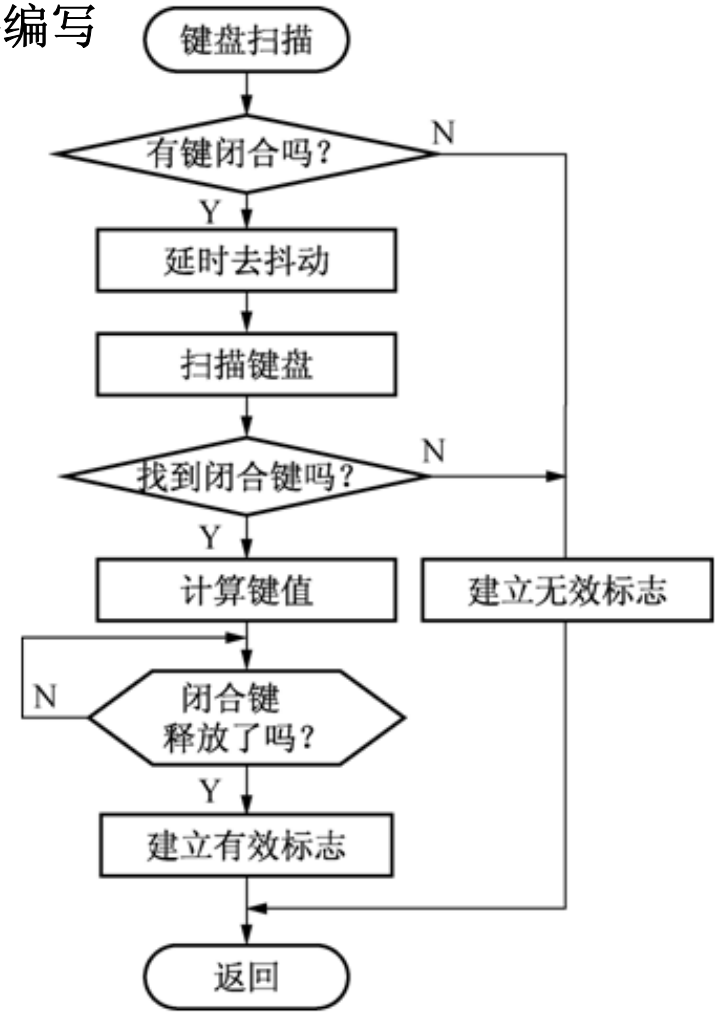


图 6-16　键盘扫描流程图

6.4 单片机键盘接口技术

6.4.2 独立式按键接口和键盘消抖动程序编写

【程序 6-2】查询方式的键盘扫描程序

```
        MOV     P1，#0FFH     ；P1口锁存器置1，准备输入
NEXT:   MOV     A，P1         ；键状态输入
        CPL     A             ；累加器内容取反
        ANL     A，#0FH       ；高4位置0，保护低4位数据
        JZ      NEXT          ；累加器A若为0，判为无键，返回
        LCALL   DELAY         ；有键闭合，调用延时程序，延时数毫秒
        MOV     A，P1         ；继续检测是否有按键
        CPL     A             ；累加器内容取反
        ANL     A，#0FH       ；高4位置0，保护低4位数据
        JZ      NEXT          ；A=0，无键返回
        MOV     A，P1         ；有键闭合，读无抖动原始数据
        MOV     B，A          ；保存数据
KEY:    MOV     A，P1         ；读键
        CPL     A             ；取反
        ANL     A，#0FH
        JNZ     KEY           ；A不等于0，键未松，返回再测
        MOV     A，B          ；键已松开，以下判别键的具体位置
```

接下页

6.4 单片机键盘接口技术

6.4.2 独立式按键接口和键盘消抖动程序编写

接上页

```
        JNB     ACC.0, SS1   ; 检测0号键是否按下，若按下转SS1
        JNB     ACC.1, SS2   ; 检测1号键是否按下，按下转SS2
        JNB     ACC.2, SS3   ; 检测2号键是否按下，按下转SS3
        JNB     ACC.3, SS4   ; 检测3号键是否按下，按下转SS4
        JMP     NEXT         ; 无键按下返回，再顺次检测
SS1:    LJMP    PROC0        ; 转向键功能程序
SS2:    LJMP    PROC1
SS3:    LJMP    PROC2
SS4:    LJMP    PROC3
PROC0:    …                  ; S1号键功能程序
        LJMP    NEXT         ; S1号键功能程序执行完返回
PROC1:    …                  ; S2号键功能程序
        LJMP    NEXT         ; S2号键功能程序执行完返回
PROC2:    …                  ; S3号键功能程序
        LJMP    NEXT         ; S3号键功能程序执行完返回
PROC3:    …                  ; S4号键功能程序
        LJMP    NEXT         ; S4号键功能程序执行完返回
```

6.4 单片机键盘接口技术

6.4.3 矩阵式键盘接口编程

1．矩阵式按键接口电路

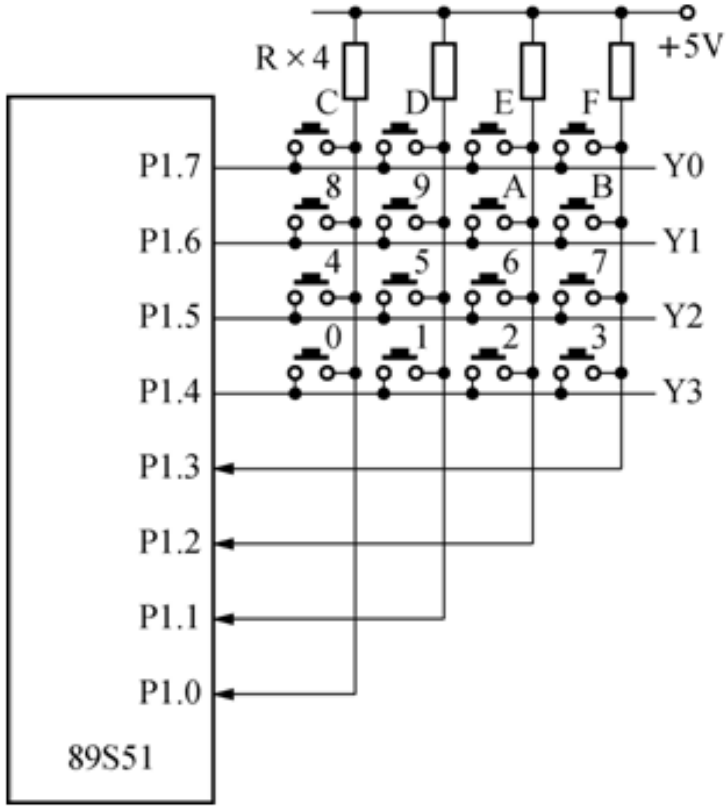


图 6-17　矩阵键盘电路图

2．矩阵式键盘的按键识别方法

6.4 单片机键盘接口技术

6.4.3 矩阵式键盘接口编程

3．矩阵式键盘编程技术

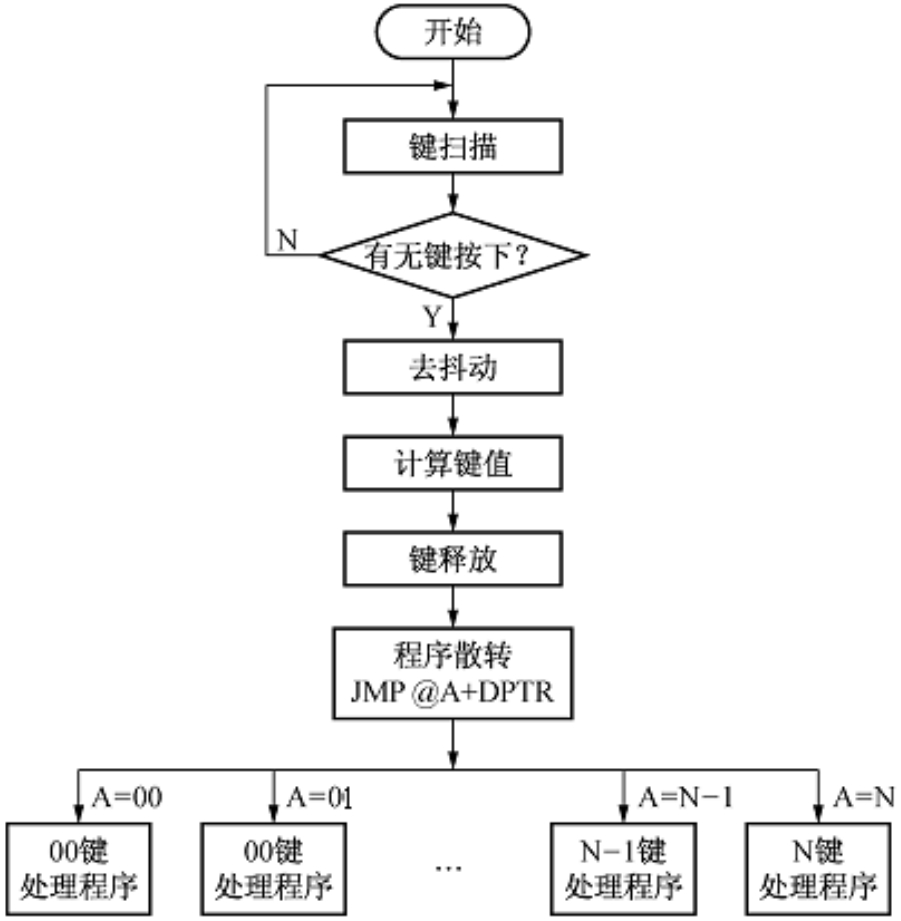


图 6-18　矩阵键盘处理流程

6.4 单片机键盘接口技术

4．键盘扫描子程序

【程序 6-3】键盘扫描程序

```
SCAN:   MOV     P1，#0FH          ；检查是否有键闭合
        MOV     A，P1
        ANL     A，#0FH
        CJNE    A，#0FH，NEXT1    ；有，转到NEXT1
        SJMP    NEXT3
NEXT1:  ACALL   DELAY             ；延时，去抖动
        MOV     A，#0EFH          ；检测第一行
NEXT2:  MOV     R1，A
        MOV     P1，A
        MOV     A，P1
        ANL     A，#0FH
        CJNE    A，#0FH，KCODE    ；若有键闭合，则转计算键码
        MOV     A，R1             ；检测下一行
        SETB    C
        RLC     A
        JC      NEXT2
NEXT3:  MOV     R0，#00H          ；建立无效标志，R0=00H
        RET     ；返回
```

接下页

6.4 单片机键盘接口技术

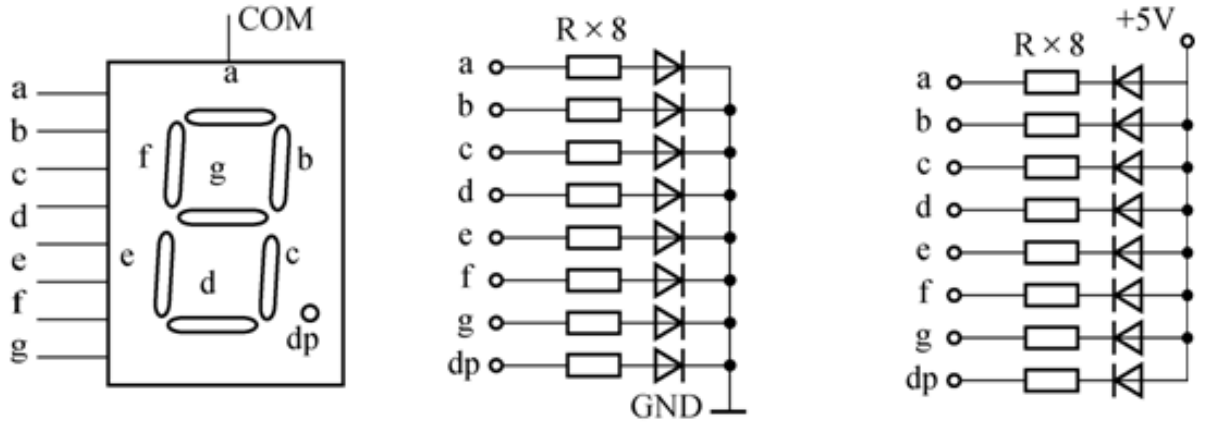
4．键盘扫描子程序

接上页

```
KCODE:  MOV     B，#0FBH          ；计算键码
NEXT4:  RRC     A
        INC     B
        JC      NEXT4
        MOV     A，R1
        SWAP    A
NEXT5:  RRC     A
        INC     B
        INC     B
        INC     B
        INC     B
        JC      NEXT5
NEXT6:  MOV     A，P1             ；检测键是否释放
        ANL     A，#0FH
        CJNE    A，#0FH，NEXT6
        MOV     R0，#0FFH         ；建立有效标志，R0=FFH
        MOV     A，B              ；键码放在累加器A中
        RET                       ；返回
```

6.5 LED显示器及其接口技术

6.5.1 LED数码显示器的结构



（a）符号与结构　　（b）共阴极接法　　（c）共阳极接法

图 6-19　七段 LED 数码管的结构

表 6-7　数据位与数码管各段的对应关系

数据位	D7	D6	D5	D4	D3	D2	D1	D0
显示段	dp	g	f	e	d	c	b	a

6.5 LED显示器及其接口技术

6.5.1 LED数码显示器的结构

表 6-8 七段 LED 显示器字形码

显 示 字 形	共阳极段码	共阴极段码	显 示 字 形	共阳极段码	共阴极段码
0	C0H	3FH	9	90H	6FH
1	F9H	06H	A	88H	77H
2	A4H	5BH	b	83H	7CH
3	B0H	4FH	C	C6H	39H
4	99H	66H	d	A1H	5EH
5	92H	6DH	E	86H	79H
6	82H	7DH	F	8EH	71H
7	F8H	07H	灭	FFH	00H
8	80H	7FH	P	8CH	73H

6.5 LED显示器及其接口技术

6.5.2 单片机与LED数码管的接口电路设计

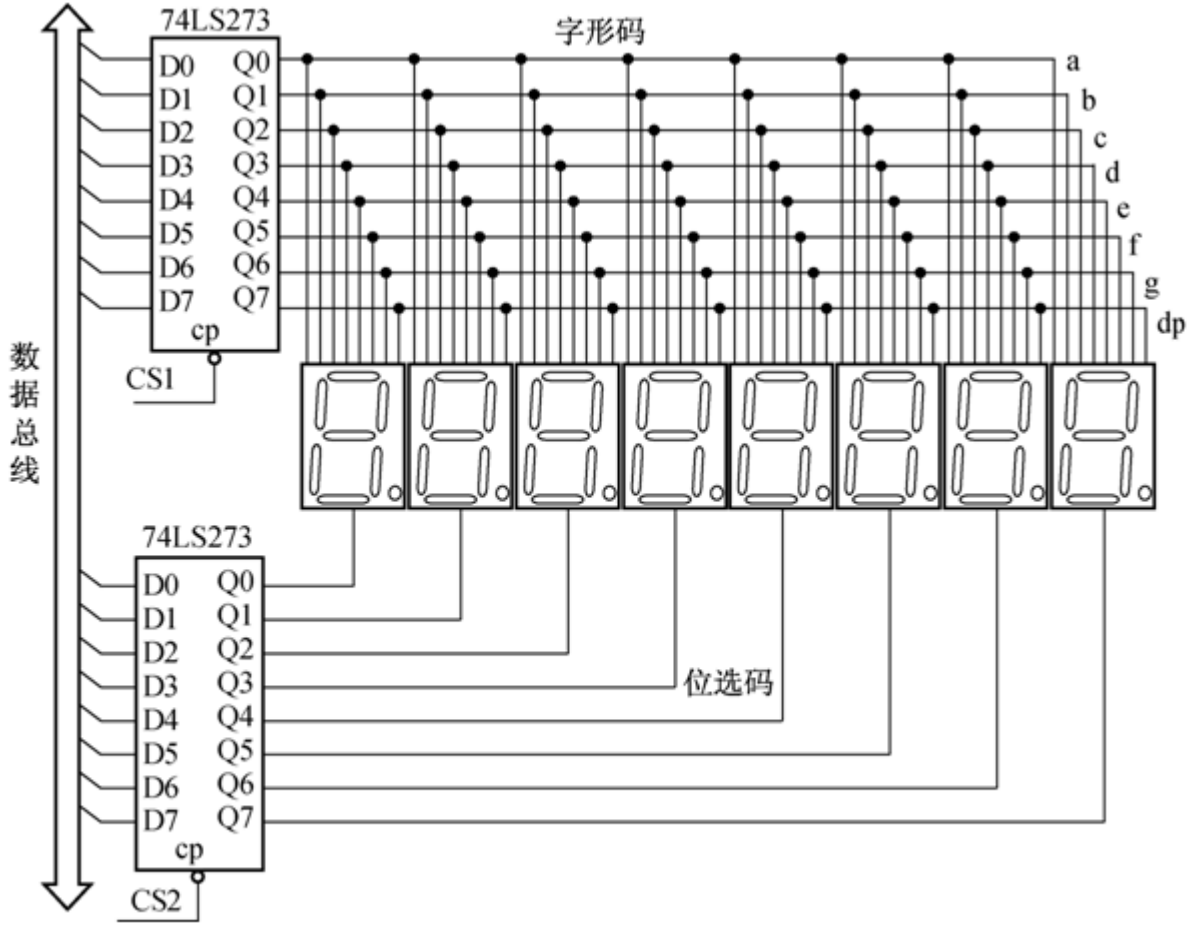


图 6-20　利用扩展并行口动态扫描显示

6.5 LED显示器及其接口技术

6.5.3 键盘、LED显示器组合接口电路设计

1．硬件电路设计

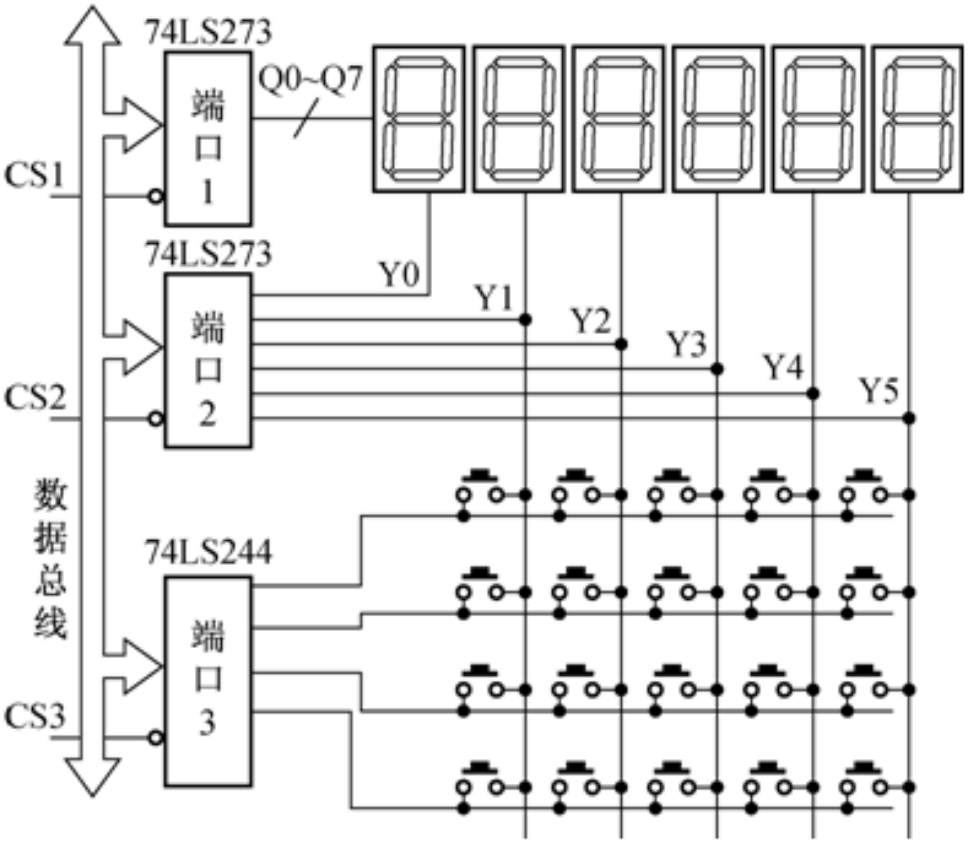


图 6-21　矩阵键盘显示器组合接口电路

2．程序设计

【程序 6-4】键盘扫描和 LED 数据管动态显示程序

```
MAIN:   MOV     R1, #70H        ; 显示缓存区清0
        CLR     A
        MOV     R2, #8
QD0:    MOV     @R1, A
        INC     R1
        DJNZ    R2, QD0
        MOV     79H, #70H       ; 显示缓存地址
        MOV     7AH, #0FEH      ; 显示缓存位地址
        MOV     20H, #00
KK:     LCALL   DIR             ; 调用显示子程序
        LCALL   KS              ; 调用判别是否有键按下子程序
        JZ      KK              ; 没有键按下转到KK处
        ACALL   K2              ; 调用键识别子程序
        JNB     00H, KK         ; 判别是否找到键值
        MOV     A, R3           ; 键散转处理
        RL      A               ; 序号×2
        CLR     00H
        MOV     DPTR, #TBB      ; 散转表首地址
        JMP     @A+DPTR         ; 散转
```

接下页

6.5 LED显示器及其接口技术

2．程序设计

接上页

```
TBB:    AJMP    KW1          ；转到键1处理程序
        AJMP    KW2          ；转到键2处理程序
        AJMP    KW3          ；转到键3处理程序
        …
        AJMP    KW20         ；转到键20处理程序
KW1:    …                    ；键1处理程序
        AJMP    KK
KW2:    …                    ；键2处理程序
        AJMP    KK
KW3:    …                    ；键3处理程序
        AJMP    KK
        …
KW20:                        ；键20处理程序
        AJMP    KK
```

6.5 LED显示器及其接口技术

6.5.4 串行I/O口扩展技术

1. 用74LS165扩展并行输入口电路设计

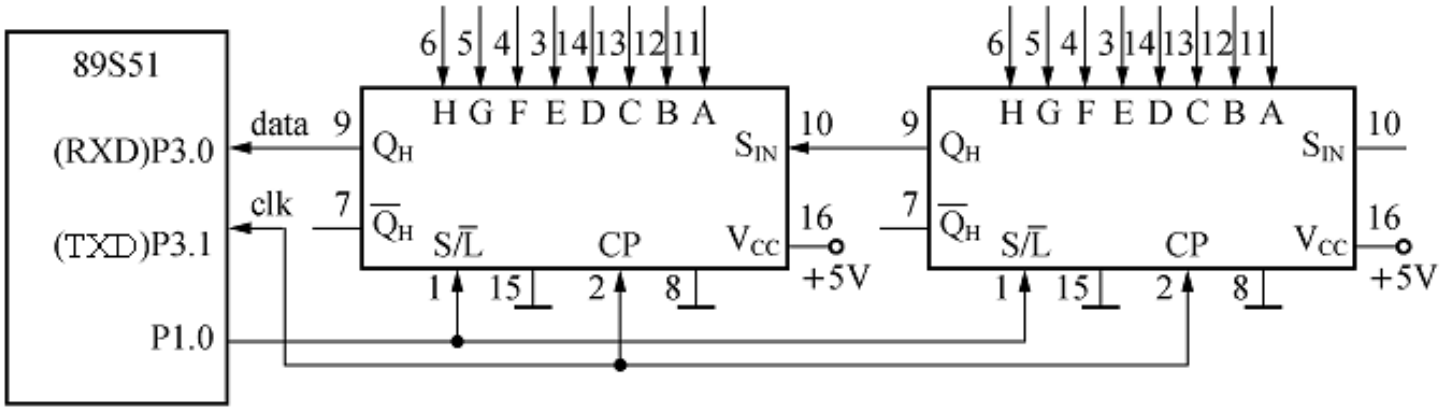


图 6-22　利用 74LS165 扩展并行输入口

6.5 LED显示器及其接口技术

6.5.4 串行I/O口扩展技术

1. 用74LS165扩展并行输入口电路设计

【程序 6-5】串行口扩展并行输入口程序

```
MOV      R7, #10        ; 设置读入组数
MOV      R0, #50H       ; 设置内部RAM数据区首址
START:   CLR     P1.0           ; 并行置入数据, S/L=0
         SETB    P1.0           ; 允许串行移位, S/L=1
         MOV     R1, #02H       ; 设置每组字节数, 即外扩74LS165的个数
RXDATA:  MOV     SCON, #10H     ; 设串行口方式0, 允许接收, 启动接收过程
WAIT:    JNB     RI, WAIT       ; 未接收完一帧, 循环等待
         CLR     RI             ; 清RI标志, 准备下次接收
         MOV     A, SBUF        ; 读入数据
         MOV     @R0, A         ; 送至RAM缓冲区
         INC     R0             ; 指向下一个地址
         DJNZ    R1, RXDATA     ; 未读完一组数据, 继续
         DJNZ    R7, START      ; 10组数据未读完重新并行置入
         RET
```

6.5 LED显示器及其接口技术

6.5.4 串行I/O口扩展技术

2. 用74LS164扩展为静态数码显示电路设计

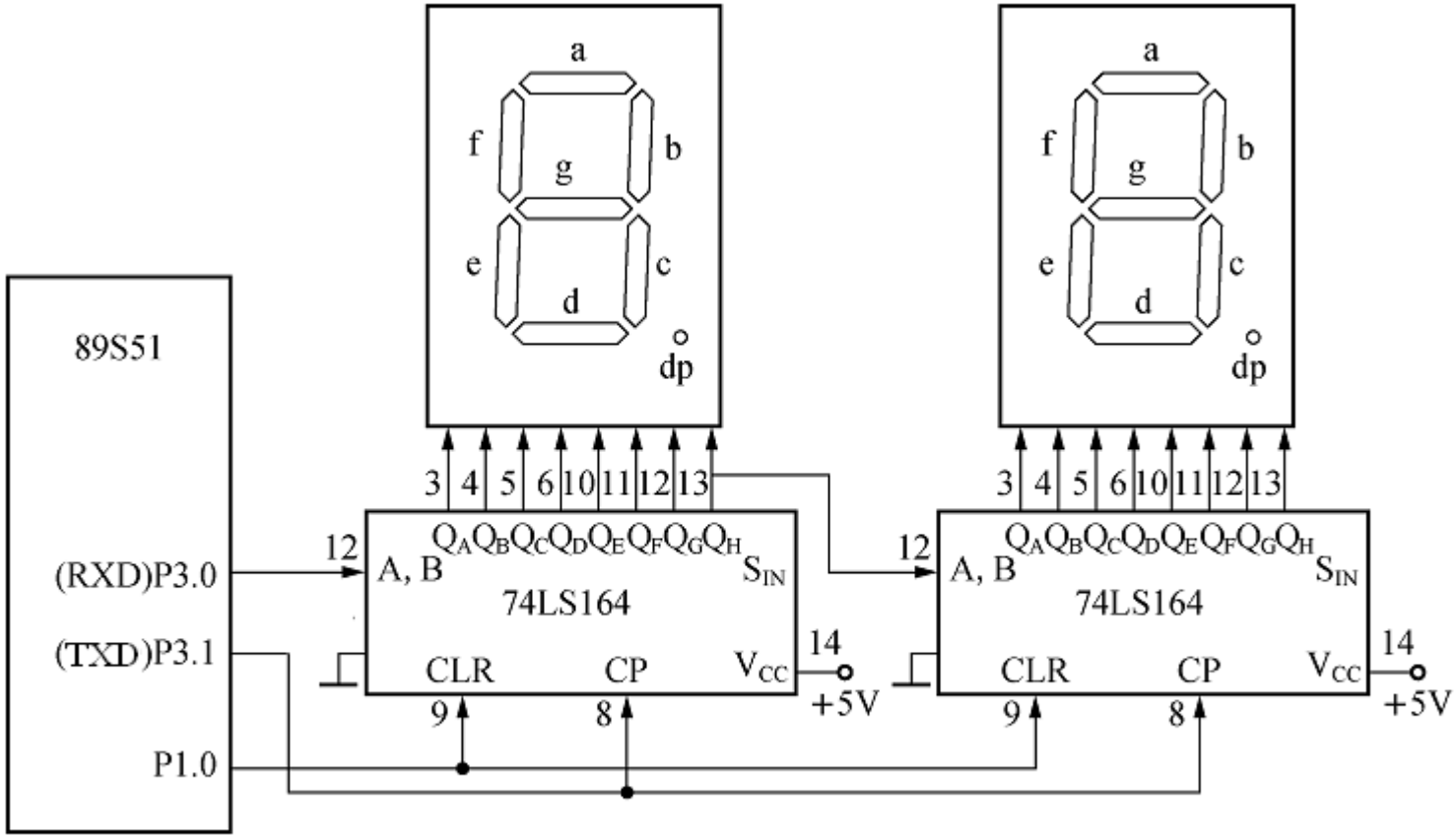


图 6-23　利用 74LS164 的扩展实现静态数码管显示

6.5 LED显示器及其接口技术

6.5.4 串行I/O口扩展技术

2．用74LS164扩展为静态数码显示电路设计

【程序 6-6】串行口静态输出显示程序

```
START:    MOV    R7,     #02H     ; 设置要发送的字节个数
          MOV    R0,     #40H     ; 设置地址指针
          MOV    SCON,   #20H     ; 设置串行口为方式0
SEND:     MOV    A,      @R0      ; 取发送数据
          MOV    SBUF,   A        ; 启动发送数据
WAIT:     JNB    TI,     WAIT     ; 一帧数据未发送完，循环等待
          CLR    TI               ; 清除发送中断标志TI
          INC    R0               ; 取下一个数
          DJNZ   R7,     SEND     ; 循环
          RET                     ; 返回
```

3．用串行口实现数码管动态扫描显示

6.5 LED显示器及其接口技术

6.5.5 利用串行口实现键盘/显示器接口

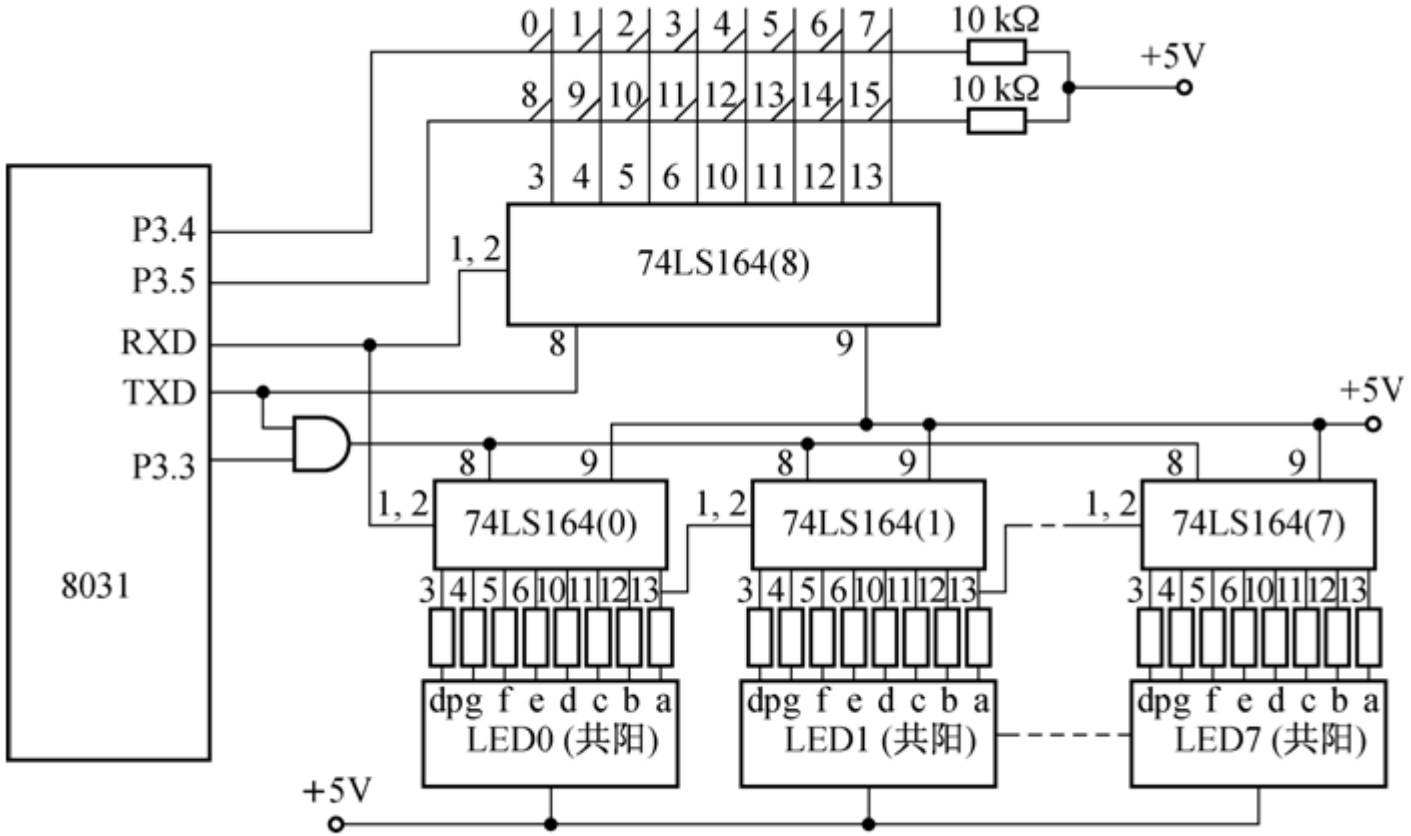


图 6-24　用串行口控制键盘/显示器电路

6.5 LED显示器及其接口技术

6.5.5 利用串行口实现键盘/显示器接口

【程序 6-7】串行输出静态显示子程序

```
DIR:    SETB    P3.3        ; 开放显示输出
        MOV     R2, #08H    ; 送出的段码个数，R2为段码个数计数器
        MOV     R0, #7FH    ; 7FH～78H为显示缓冲区
DIR1:   MOV     A, @R0      ; 取出待显示的数
        ADD     A, #0DH     ; 加偏移量
        MOVC    A, @A+PC    ; 查段码表TAB，取出段选码数据
        MOV     SBUF, A     ; 输出段选码
DIR2:   JNB     TI, DIR2    ; 1个字节的段码是否输出完
        CLR     TI          ; 完，清中断标志
        DEC     R0          ; 指向下一个数据单元
        DJNZ    R2, DIR1    ; 段码计数器R2是否为0，不为0，则继续送段码
        CLR     P3.3        ; 返回
        RET
TAB:    DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
                            ; 0～9
        DB      88H,83H,0C6H,0A1H,86H,8EH,0BFH,8CH,0FFH
                            ; A～F,－,P,"暗"
```

【程序 6-8】键盘扫描子程序

```
KEY1:   MOV     A, #00H
        MOV     SBUF, A         ; 扫描键盘74LS164（8）的输出为00H，使所有列线为0
KL0:    JNB     TI, KL0         ; 串行输出完否
        CLR     TI              ; 清0中断标志
KL1:    JNB     P3.4, PK1       ; 第一行键中是否有闭合键。如果有，跳PK1进行处理
        JB      P3.5, KL1       ; 在第二行键中是否有闭合键
PK1:    ACALL   DL10            ; 调用延时10ms子程序DL10
        JNB     P3.4, PK2       ; 是否抖动引起的
        JB      P3.5, KL1       ; 不是抖动引起的
PK2:    MOV     R7, #08H        ; 判别是哪一个键按下
        MOV     R6, #0FEH
        MOV     R3, #00H
        MOV     A, R6
KL5:    MOV     SBUF, A
KL2:    JNB     TI, KL2         ; 等待串行口发送完
        CLR     TI
        JNB     P3.4, PKONE     ; 是第一行某键否
        JB      P3.5, NEXT      ; 是第二行某键否
        MOV     R4, #08H        ; 第二行键中有键被按下
        AJMP    PK3
```

接下页

接上页

```
PKONE:  MOV     R4, #00H        ; 第一行键中有键被按下
PK3:    MOV     SBUF, #00H      ; 等待键释放
KL3:    JNB     TI, KL3
        CLR     TI
KL4:    JNB     P3.4, KL4
        JNB     P3.5, KL4
        MOV     A, R4           ; 键释放，取得键码
        ADD     A, R3
        RET
NEXT:   MOV     A, R6           ; 判断下一列键是否按下
        RL      A
        MOV     R6, A
        INC     R3
        DJNZ    R7, KL5         ; 8列键都检查完否
        AJMP    KEY1            ; 扫描完毕，开始下一个扫描周期
DL10:   MOV     30H, #0AH       ; 延时10ms子程序（设fosc =6MHz）
DL:     MOV     31H, #0FFH
DL1:    DJNZ    31H, DL1
        DJNZ    30H, DL
        RET
```

6.6 LCD液晶显示器接口技术

6.6.1 LCM 1602简介

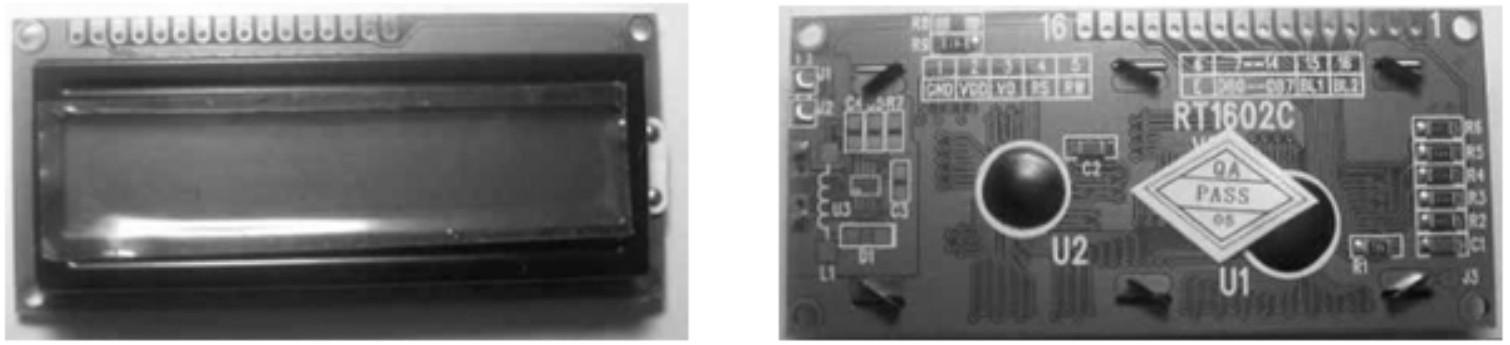


图 6-25　LCM 1602 液晶模块

表 6-9　LCM 1602 字符代码表

High 4BIT / LOW 4BIT	MSB 0000	0010	0011	0100	0101	0110	0111
LSB XXXX0000	RAM (1)		0	@	P	`	P
XXXX0001	(2)	!	1	A	Q	a	q
XXXX0010	(3)	"	2	B	R	b	r
XXXX0011	(4)	#	3	C	S	c	s
XXXX0100	(5)	$	4	D	T	d	t
XXXX0101	(6)	%	5	E	U	e	u
XXXX0110	(7)	&	6	F	V	f	v
XXXX0111	(8)	'	7	G	W	g	w
XXXX1000	(1)	(	8	H	X	h	x
XXXX1001	(2)	)	9	I	Y	i	y
XXXX1010	(3)	*	:	J	Z	j	z
XXXX1011	(4)	+	;	K	[	k	{
XXXX1100	(5)	,	<	L	¥	l	|
XXXX1101	(6)	-	=	M	]	m	}
XXXX1110	(7)	.	>	N	^	n	→
XXXX1111	(8)	/	?	O	_	o	←

6.6 LCD液晶显示器接口技术

6.6.1 LCM 1602简介

表 6-10　LCM 1602 的内部显示地址

显示位置	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
第 1 行	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
第 2 行	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

表 6-11　LCM 1602 指令表

指令	功　能	指　令　码										说　明
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
1	清屏	0	0	0	0	0	0	0	0	0	1	清除屏幕，置 AC 为 0，光标回位
2	光标返回	0	0	0	0	0	0	0	0	1	*	DDRAM 地址为 0，显示回原位，DDRAM 内容不变
3	设置输入方式	0	0	0	0	0	0	0	1	I/D	S	设置光标移动方向并指定显示是否移动
4	显示开关	0	0	0	0	0	0	1	D	C	B	设置显示开或关 D、光标开关 C、光标所在字符闪烁 B
5	移位	0	0	0	0	0	1	S/C	R/L	*	*	移动光标及整体显示，同时不改变 DDRAM 内容
6	功能设置	0	0	0	0	1	DL	N	F	*	*	设置接口数据位数 DL、显示行数 L、字符字体 F
7	CGRAM 地址设置	0	0	0	1	ACG						设置 CGRAM 地址。设置后发送接收数据
8	DDRAM 地址设置	0	0	1	ADD							设置 DDRAM 地址。设置后发送接收数据
9	忙标志/读地址计数器	0	1	BF	AC							读忙标志BF标志正在执行内部操作并读地址计数器内容
10	CGRAM/DDRAM 数据写	1	0	写数据								从 CGRAM 或 DDRAM 写数据
11	CGRAM/DDRAM 数据读	1	1	读数据								从 CGRAM 或 DDRAM 读数据
说明		I/D=1：增量方式；I/D=0：减量方式；S=1：移位 S/C=1：显示移位；S/C=0：光标移位 R/L=1：右移；R/L=0：左移 DL=1：8 位；DL=0：4 位 N=1：2 行；N=0：1 行 F=1：5×10 字体　F=0：5×7 字体 BF=1：执行内部操作；BF=0 可接收指令										DDRAM：显示数据 RAM CGRAM：字符发生器 RAM ACG：CGRAM 地址 ADD：DDRAM 地址及光标地址 AC：地址计数器，用于（DDRAM 和 CGRAM）

6.6 LCD液晶显示器接口技术

6.6.2 LCM 1602模块应用举例

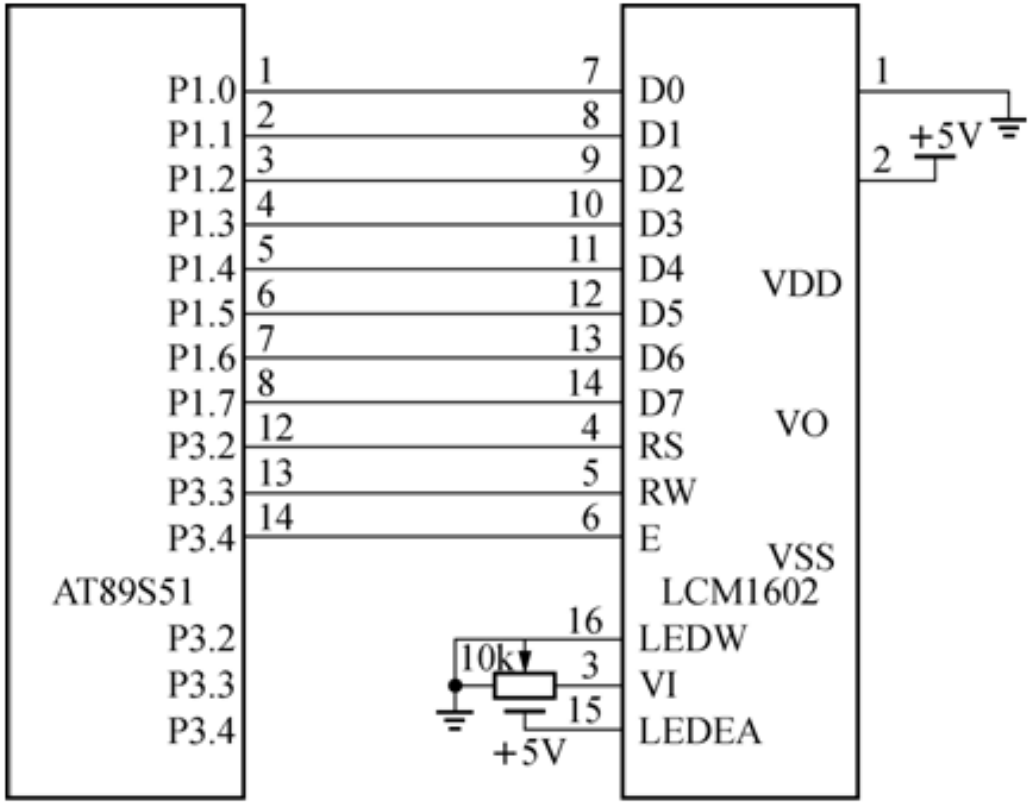


图 6-26　LCM 1602 与单片机的连接

【例 6-6】LCM l602 显示程序示例。

```
RS          BIT     P3.2              ; 定义RS为P3.2
R_W         BIT     P3.3              ; 定义R/W为P3.3
E           BIT     P3.4              ; 定义E为P3.4
DB0_7       EQU     P1                ; 数据口DB0-7接P1
;  ------------------------------------------------------------
            ORG     0000H             ; 程序地址0000H开始存放
            SJMP    START             ; 跳到标记START处执行程序
            ORG     30H               ; 程序从地址0030H开始
START:      MOV     SP，#60H          ; 设定MCS-51堆栈指针，从61H开始存放
            LCALL   Initial           ; 调用LCM的初始化程序
            LCALL   CLS               ; 调用清除显示器的子程序
            MOV     A，#10000100B     ; 显示地址从第1行第5列开始
            LCALL   Write_COM         ; 调用写指令码子程序
            MOV     DPTR，#STR1       ; 将第1行字串的起始地址存入DPTR
            LCALL   STRING            ; 调用PRSTRING子程序，将字串显示到LCM
            MOV     A，#11000010B     ; 显示地址移到第2行第3列
            LCALL   Write_COM         ; 调用写指令码子程序
            MOV     DPTR，#STR2       ; 将第2行字串的起始地址存DPTR
            LCALL   STRING            ; 调用PRSTRING子程序，将字串显示到LCM
LOOP:       SJMP    LOOP              ; 程序无限循环
STR1:       DB  " Hello"，00H         ; 在LCM第1行显示字串"Hello"
```

接下页

接上页

```
STR2:       DB " You are welcome ", 00H; 在LCM第2行显示字串"You are welcome"
                                       ; Initial初始化子程序
Initial:    MOV     A, #00111000B      ; 设置8位格式，2行，5×7字型
            LCALL   Write_COM          ; 调用写指令码子程序
            MOV     A, #00001110B      ; 显示器开，光标开，不闪烁
            LCALL   Write_COM          ; 调用写指令码子程序
            MOV     A, #00000110B      ; 文字不动，光标自动右移
            LCALL   Write_COM          ; 调用写指令码子程序
            RET
; CheckBusy检测LCM忙子程序
CheckBusy:  PUSH    ACC                ; 将累加器ACC的内容放到堆栈内
ChkLop:     CLR     E                  ; 设定E=0，禁止LCM读模式
            SETB    R_W                ; 设定R/W=1，选择读模式
            CLR     RS                 ; 设定RS=0，选择指令寄存器IR
            SETB    E                  ; 将E设定为1，使能LCM
            MOV     A, DB0_7           ; 由P1读出LCM的状态信息存入ACC中
            CLR     E                  ; 将E设定为0
            JB      ACC.7, ChkLop      ; 判断LCM的位BF是否为1， 若等于1
                                       ; 表示LCM忙，CPU跳到CheckBusyLoop继续等待
            POP     ACC                ; 将累加器ACC内容从堆栈区取出
            LCALL   DELAY              ; 调用延迟子程序，延时约数毫秒
            RET                        ; 返回主程序
```

接下页

接上页

```
; Write_COM写命令子程序。将ACC中命令输入到LCM的IR指令寄存器
Write_COM:  LCALL   CheckBusy        ; 调用CheckBusy子程序确定LCM可以执行指令
            CLR     E                ; 设定E=0，禁能LCM
            CLR     R_W              ; 设定R/W=0，选择写模式
            CLR     RS               ; 设定RS=0，选择指令寄存器IR
            SETB    E                ; 将E设定为1，使能LCM
            MOV     DB0_7，A         ; 将存在ACC内的指令码经由P1输出到LCM
            CLR     E                ; 将E设定为0，MCS-51向LCM存取数据后，
                                     ; 必须将LCM的E脚输出0，让LCM禁能（Disable）
            RET                      ; 返回主程序
; WriteLCDData子程序。将ACC内的数据输入到LCM的DR数据寄存器
WriteLCDData: LCALL   CheckBusy      ; 调用CheckBusy子程序，确定LCM可以执行指令
            CLR     E                ; 设定E=0，禁能LCM
            CLR     R_W              ; 设定R/W=0，选择写模式
            SETB    RS               ; 设定RS=1，选择数据寄存器DR
            SETB    E                ; 将E设定为1，使能LCM
            MOV     DB0_7，A         ; 将存在ACC内的指令码经由P1输出到LCM
            CLR     E
            RET                      ; 返回主程序
```

接下页

接上页

```
; CLS子程序。清除LCM的显示字幕
CLS:        MOV     A, #01H
            LCALL   Write_COM
            RET
; STRING写字符串子程序。将一个字符显示在LCM，字串首地址要存入DPTR，字串必须以00H结束
STRING:     PUSH    ACC
LOOP1:      CLR     A
            MOVC    A, @A+DPTR
            JZ      END_PR
            LCALL   WriteLCDData
            INC     DPTR
            SJMP    LOOP1
END_PR:     POP     ACC
            RET
; DELAY子程序。所延迟的时间约为2.5ms；延时时间约为R6*（500μs）
DELAY:      MOV     R6, #5
D1:         MOV     R7, #248
            DJNZ    R7, $
            DJNZ    R6, D1
            RET
            END
```

6.7 单片机串行总线扩展技术

6.7.1 单总线及单总线器件

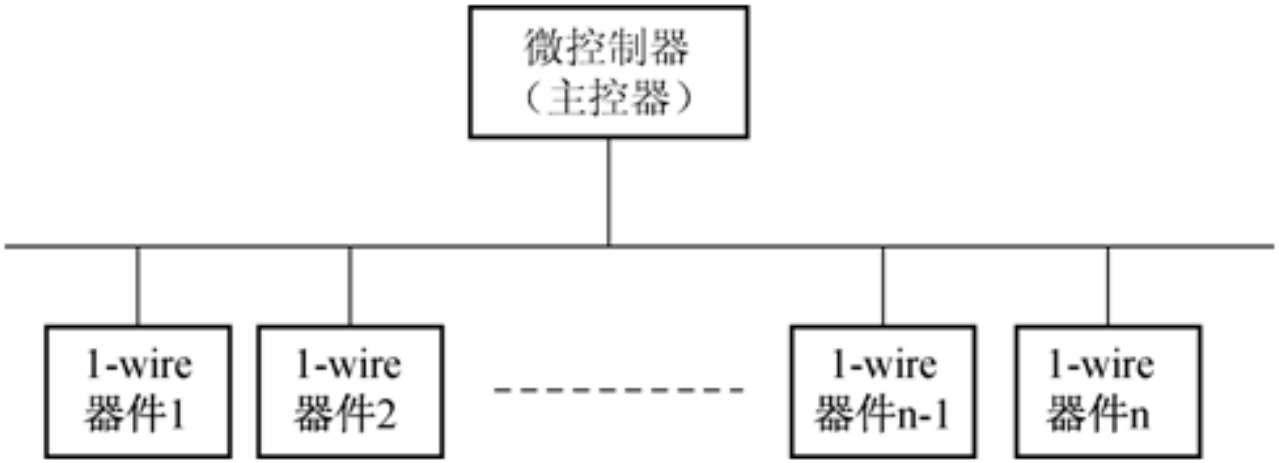


图 6-27　单总线多节点系统示意图

6.7 单片机串行总线扩展技术

6.7.1 单总线及单总线器件

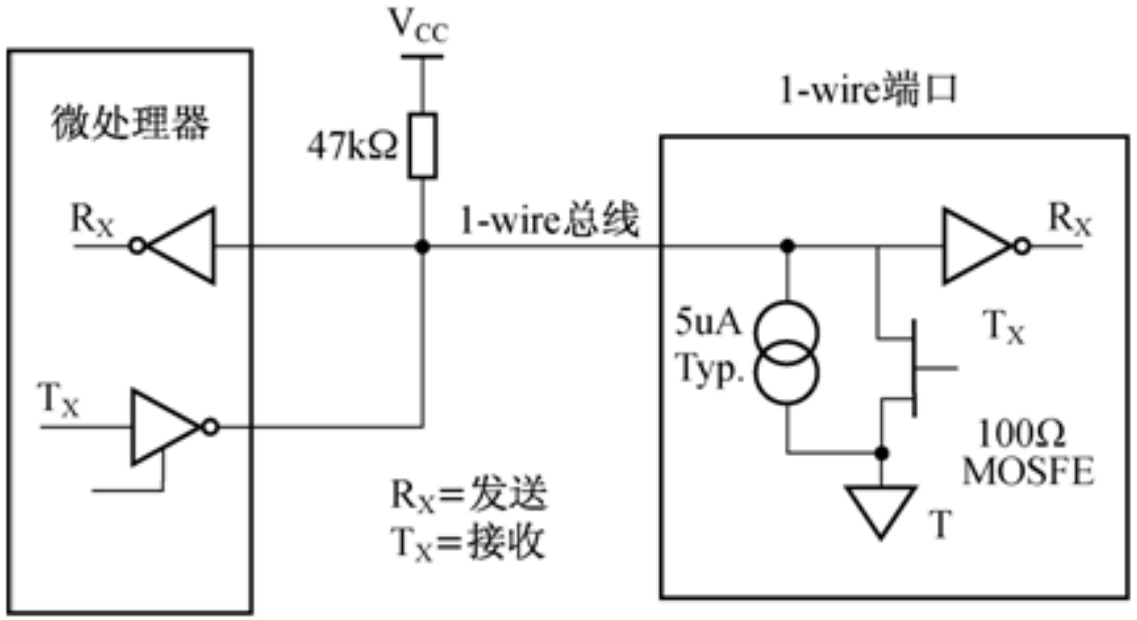


图 6-28　单总线接口示意图

6.7 单片机串行总线扩展技术

6.7.2 单总线温度传感器DS18B20

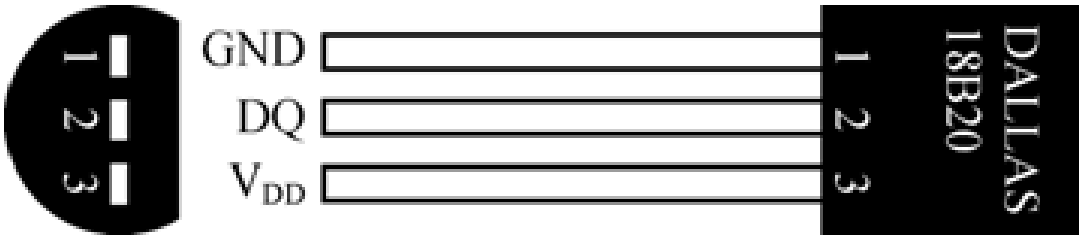


图 6-29　DS18B20 数字温度计外形示意图

1．DS18B20的主要特点概述

6.7 单片机串行总线扩展技术

2．DS18B20的组成结构与工作原理

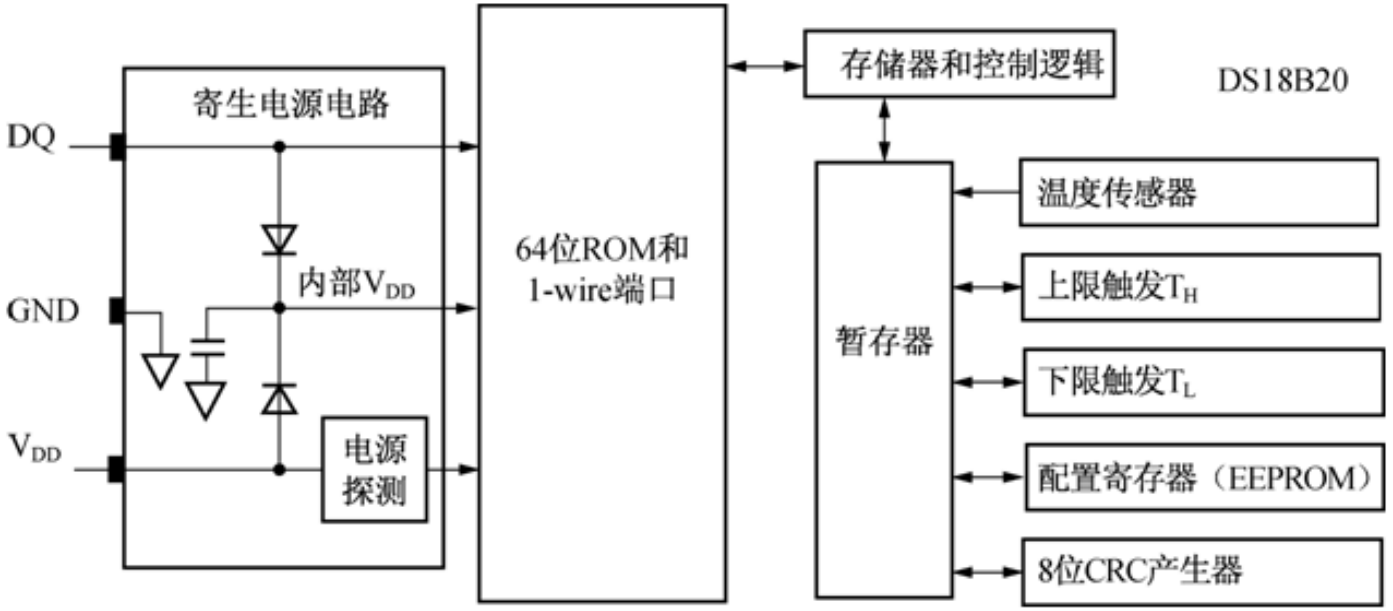


图 6-30　DS18B20 结构框图

6.7 单片机串行总线扩展技术

2．DS18B20的组成结构与工作原理

表 6-12　DS18B20 命令集

指　令	说　明	约 定 代 码	发出约定代码后单总线的操作
温度变换	启动温度变换	44h	读温度忙状态
读暂存存储器	从暂存存储器读字节	BEh	读 9 字节数据
写暂存存储器	写字节至暂存存储器地址 2 和 3 处 T_H 和 T_L 温度触发器	4Eh	写数据至地址 2 和地址 3 的两个字节
复制暂存存储器	把暂存存储器复制入非易性存储器，仅地址 2 和地址 3	43h	读复制状态
重新调出 E^2	把存储在非易失性存储器内的数值重新调入暂存存储器温度触发器	E3h	读温度“忙”状态
读电源	发 DS1820 电源方式的信号至主机	B4h	读电源状态

6.7 单片机串行总线扩展技术

3．DS18B20与单片机的接口与编程

【例 6-7】DS18B20 应用程序示例。

```
;  FLAG1 :  标志位，为"1"时表示检测到DS18B20 ;  DQ :  DS18B20的数据总线接脚
;  TEMPER_NUM :  保存读出的温度数据
;  本程序仅适合单个DS18B20和51单片机的连接，晶振为12MHz左右
TEMPER_L    EQU     36H       ; 检测温度低位存储单元
TEMPER_H    EQU     35H                ; 检测温度高位存储单元
FLAG1       BIT     F0                 ; 标志，若FLAG1=1，DS1820存在
TEMPER_NUM  EQU     37H                ; 显示温度存储单元
DQ          BIT     P3.6               ; 压缩BCD格式
LED         BIT     P1.4
MAIN1:      LCALL   INIT_1820          ; 调用DS18B20初始化程序
            LCALL   RE_CONFIG          ; 重写暂存存储器设定值
            LCALL   GET_TEMPER         ; 读出转换后的温度值
            LCALL   TEMPER_COV         ; 温度数据转换成BCD码
;           LCALL   DSPLAY_TMP         ; 显示温度数据
            LJMP    MAIN1              ; 循环
```

接下页

接上页

```
; DS18B20初始化程序
INIT_1820:  SETB    DQ
            NOP
            CLR     DQ
            MOV     R0, #0A0H
TSR1:       DJNZ    R0, TSR1          ; 延时
            SETB    DQ
            MOV     R0, #0A0H
TSR2:       JNB     DQ, TSR3
            DJNZ    R0, TSR2
            LJMP    TSR4              ; 延时
TSR3:       SETB    FLAG1             ; 置标志位，表示DS1820存在
            LJMP    TSR5
TSR4:       CLR     FLAG1             ; 清标志位，表示DS1820不存在
            LJMP    TSR7
TSR5:       MOV     R0, #0A0H
TSR6:       DJNZ    R0, TSR6          ; 延时
TSR7:       SETB    DQ
            RET
;  重新写DS18B20暂存存储器设定值
RE_CONFIG:  JB      FLAG1, RE_CONFIG1 ; 若DS18B20存在，转RE_CONFIG1
            RET
RE_CONFIG1: MOV     A, #0CCH          ; 发SKIP ROM命令
```

接下页

接上页

```
                LCALL   WRITE_1820
                MOV     A，#4EH          ；发写暂存存储器命令
                LCALL   WRITE_1820
                MOV     A，#00H          ；TH（报警上限）中写入00H
                LCALL   WRITE_1820
                MOV     A，#00H          ；TL（报警下限）中写入00H
                LCALL   WRITE_1820
                MOV     A，#1FH          ；选择9位温度分辨率
                LCALL   WRITE_1820
                RET
；  读出转换后的温度值
GET_TEMPER:     SETB    DQ               ；定时入口
                LCALL   INIT_1820
                JB      FLAG1，TSS2
                RET                      ；若DS18B20不存在则返回
TSS2:           MOV     A，#0CCH         ；跳过ROM匹配
                    LCALL   WRITE_1820
                    MOV     A，#44H          ；发出温度转换命令
                    LCALL   WRITE_1820
                    LCALL   INIT_1820
                    MOV     A，#0CCH         ；跳过ROM匹配
                    LCALL   WRITE_1820
                    MOV     A，#0BEH         ；发出读温度命令
                    LCALL   WRITE_1820
```

接下页

接上页

```
                LCALL   READ_18200
                MOV     TEMPER_NUM，A     ；保存读出的温度数据
                RET
；  写DS18B20的程序
WRITE_1820:     MOV     R2，#8
                CLR     C
WR1:            CLR     DQ
                MOV     R3，#6
                DJNZ    R3，$
                RRC     A
                MOV     DQ，C
                MOV     R3，#23
                DJNZ    R3，$
                SETB    DQ
                NOP
                DJNZ    R2，WR1
                SETB    DQ
                RET
；读DS18B20的程序，从DS18B20中读出两个字节的温度数据
READ_18200:     MOV     R4，#2            ；将温度高位和低位从DS18B20中读出
                MOV     R1，#TEMPER_L     ；低位存入36H（TEMPER_L）
RE00:           MOV     R2，#8            ；高位存入35H（TEMPER_H）
RE01:           CLR     C
                SETB    DQ
```

接下页

接上页

```
                NOP
                NOP
                CLR     DQ
                NOP
                NOP
                SETB    DQ
                MOV     R3，#7
                DJNZ    R3，$
                MOV     C，DQ
                MOV     R3，#23
                DJNZ    R3，$
                RRC     A
                DJNZ    R2，RE01
                MOV     @R1，A
                DEC     R1
                DJNZ    R4，RE00
                RET
；对从DS18B20中读出的温度数据进行转换
TEMPER_COV:     MOV     A，#0F0H
                ANL     A，TEMPER_L          ；舍去温度低位中小数点后的四位
                SWAP    A                    ；温度数值
                MOV     TEMPER_NUM，A
                MOV     A，TEMPER_L
                JNB     ACC.3，TEMPER_COV1   ；四舍五入取温度值
```

接下页

6.7 单片机串行总线扩展技术

3．DS18B20与单片机的接口与编程

接上页

```
                    INC     TEMPER_NUM
TEMPER_COV1:        MOV     A，TEMPER_H
                    ANL     A，#07H
                    SWAP    A
                    ORL     A，TEMPER_NUM
                    MOV     TEMPER_NUM，A          ；保存变换后的温度数据
                    LCALL   BIN_BCD
                    RET
；对十六进制的温度数据转换成压缩BCD码
BIN_BCD:            MOV     A，TEMPER_NUM
                    MOV     B，#10
                    DIV     AB
                    SWAP    A
                    ORL     A，B
                    MOV     TEMPER_NUM，A
                    RET
                    END
```

6.7 单片机串行总线扩展技术

6.7.3 单片机红外串行通信

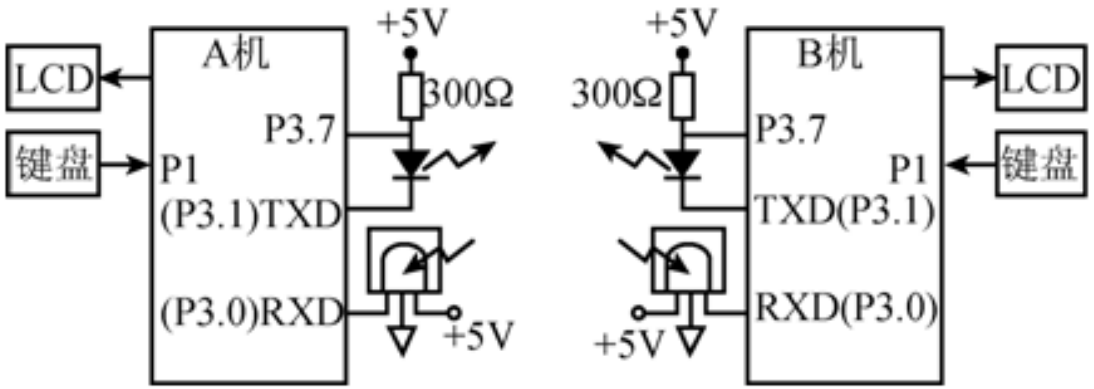
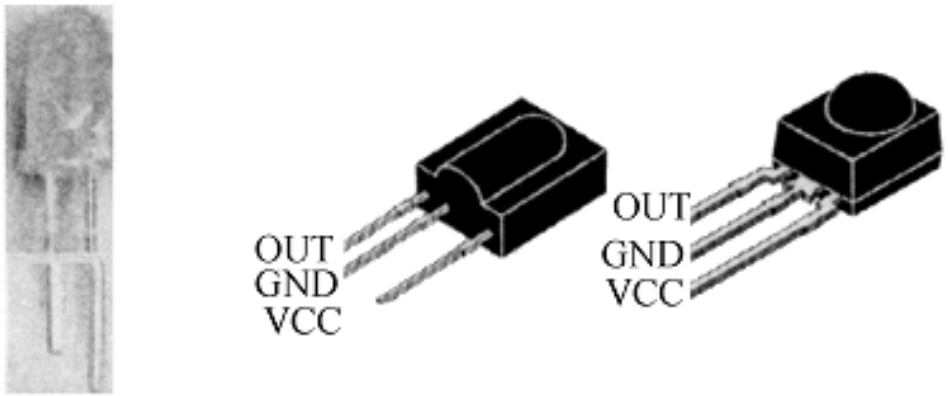


图 6-31　单片机红外双机通信接口

6.7 单片机串行总线扩展技术

6.7.3 单片机红外串行通信



（a）红外发光二极管 （b）一体化红外线接收器

图 6-32 红外线发射器/接收器

6.7 单片机串行总线扩展技术

6.7.3 单片机红外串行通信

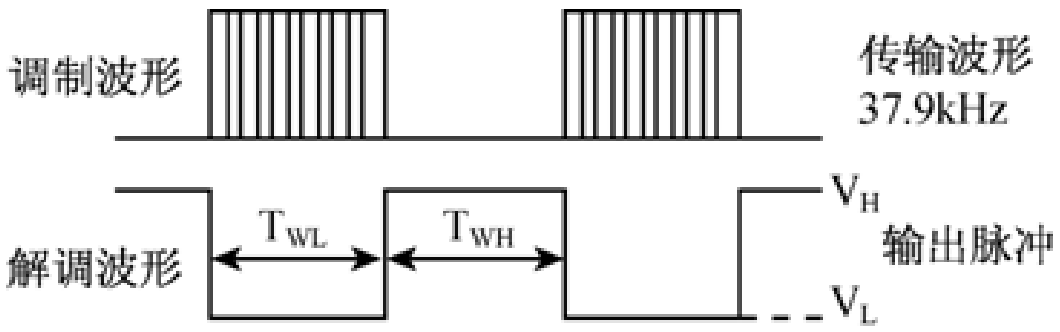


图 6-33　红外通信调制/解调波形

思考练习题

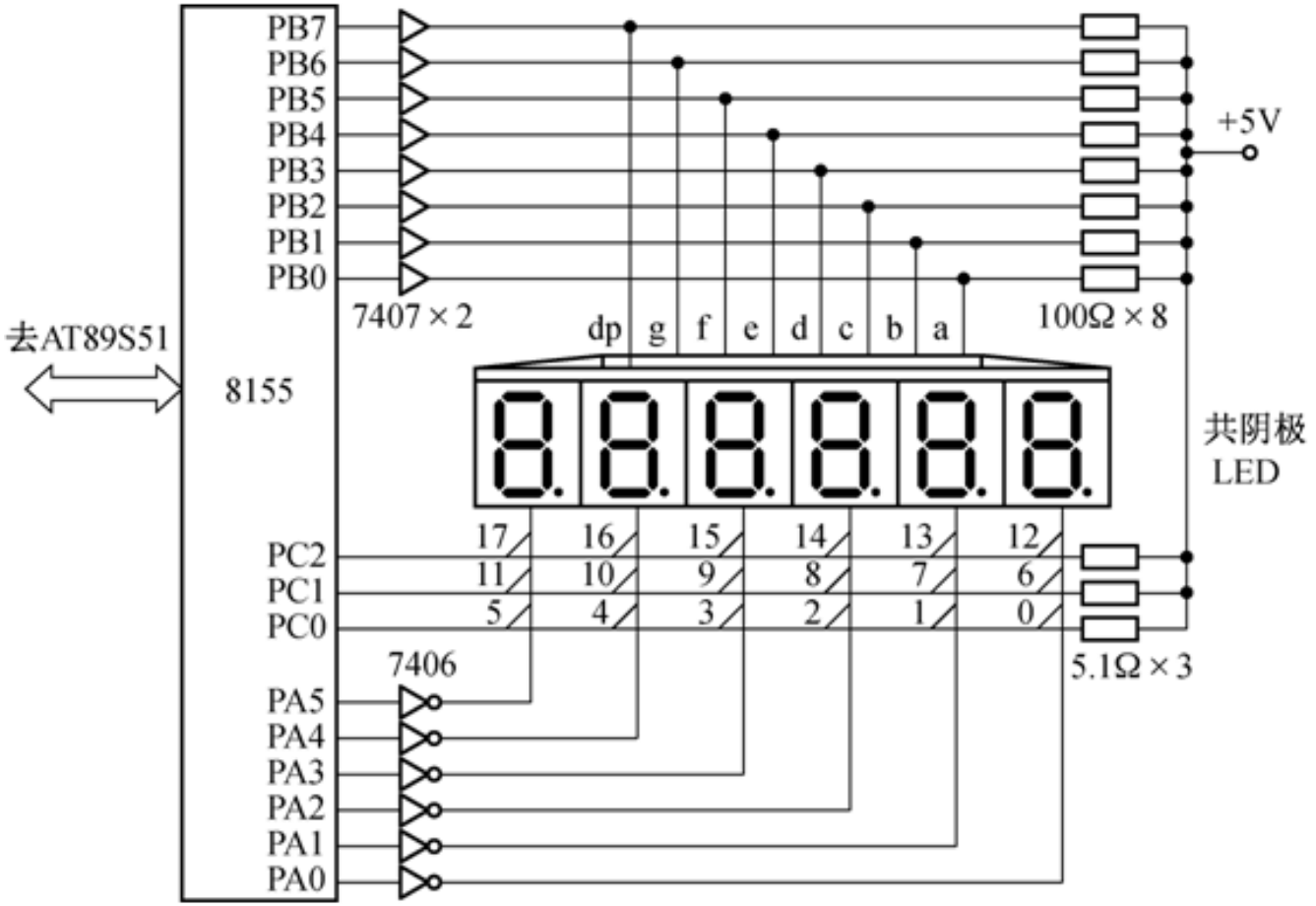


图 6-34　题 9 图

思考练习题

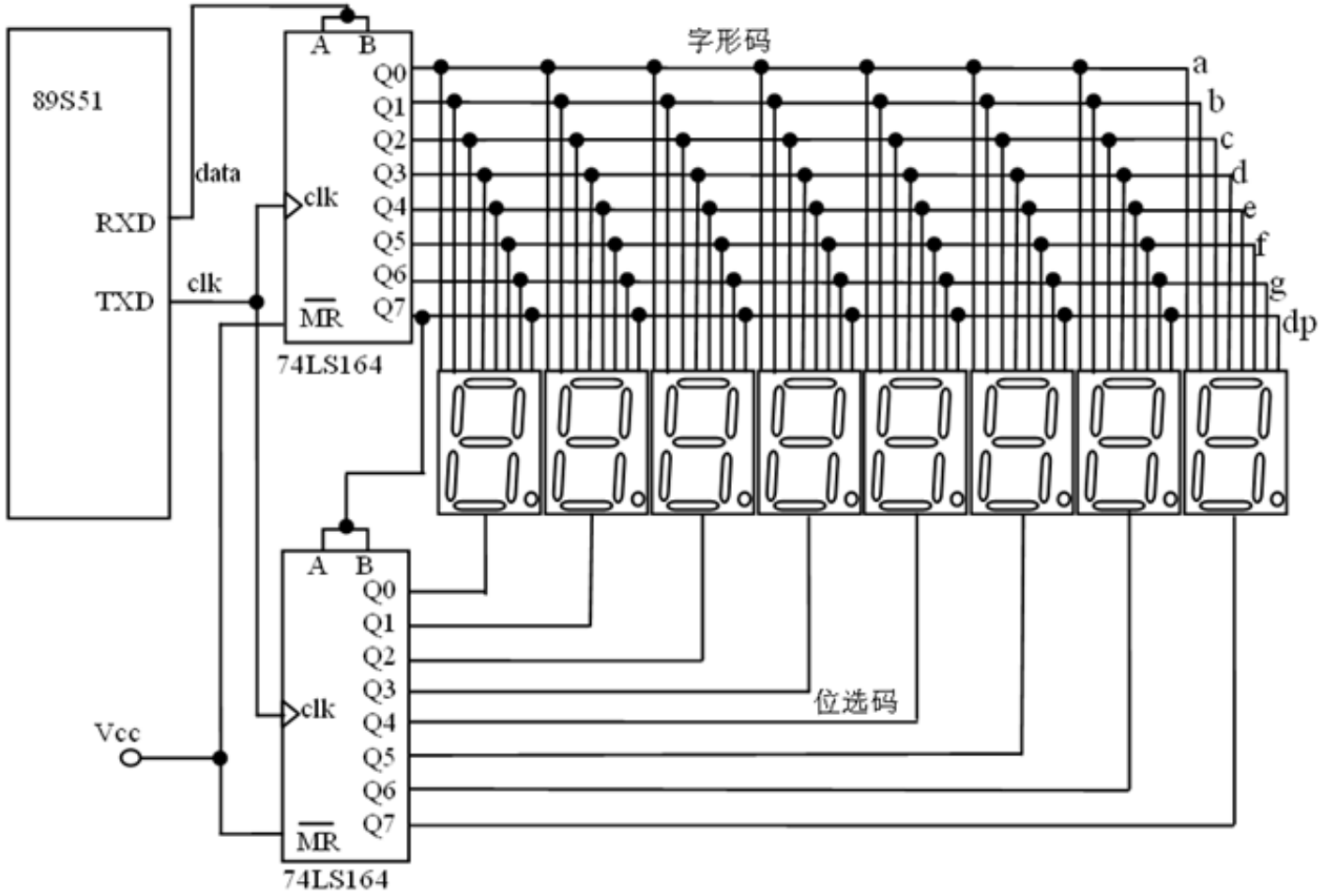


图 6-35　题 13 图