第7章 键盘、显示接口技术

本章学习要求：

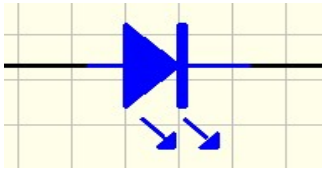
1. 掌握数码管显示原理、电路连接及编程；
2. 掌握键盘工作原理、电路连接及编程。

7.4 LED显示器及接口技术

7.4.1 LED显示接口技术

LED: Light Emiting Diode

电路符号：



LED是利用PN结把电能转换成光能的固体发光器件,根据制造材料的不同可以发出红、黄、绿、白等不同色彩的可见光来.

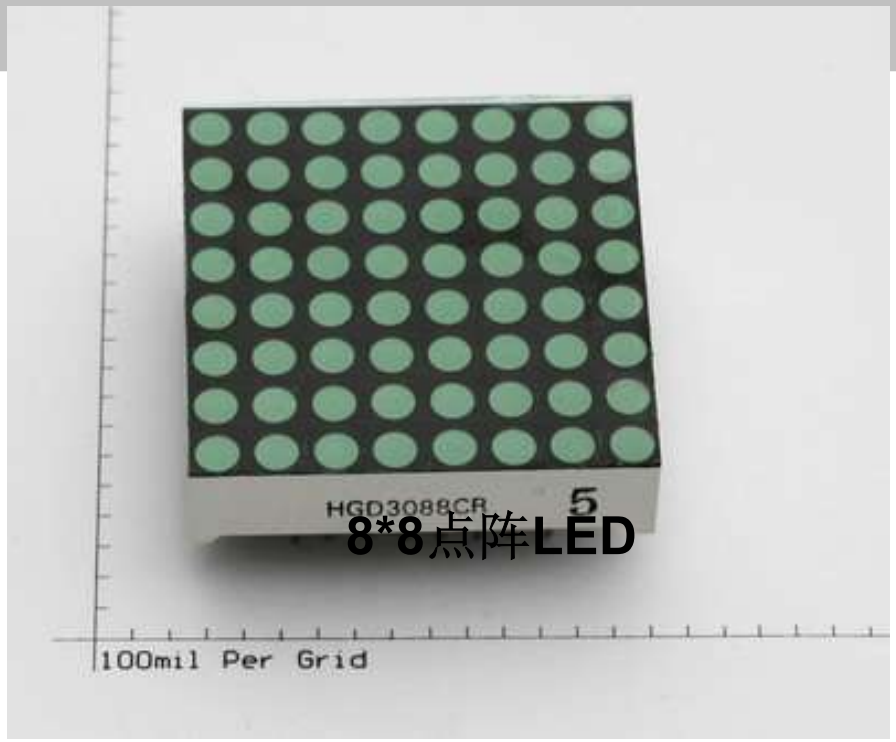
LED的伏安特性类似于普通二极管,正常工作电流Ig为5-20mA,压降Vg为1.5-2.0V左右.

(a) 单段

(b) 8段

指示灯

数码管

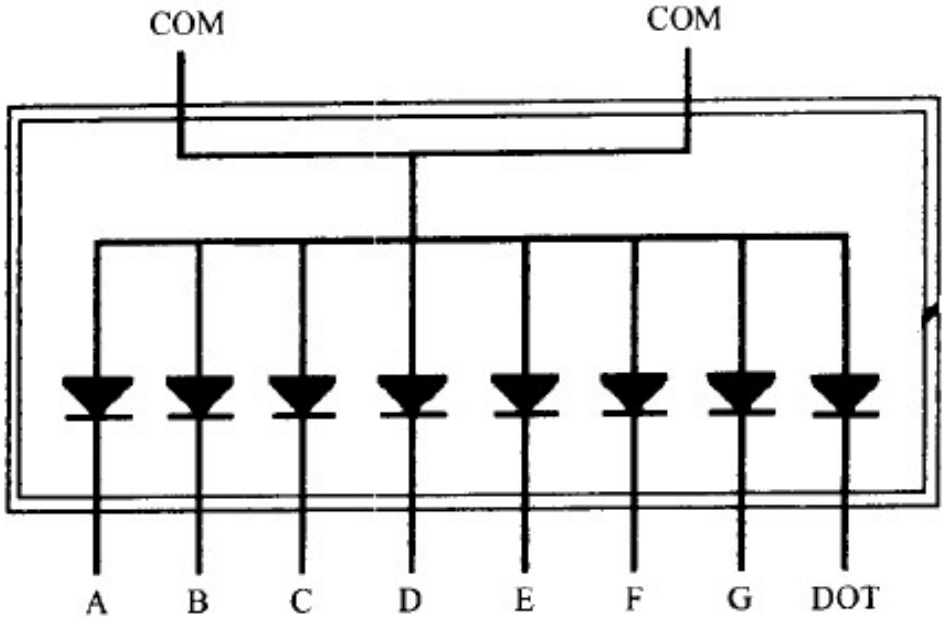


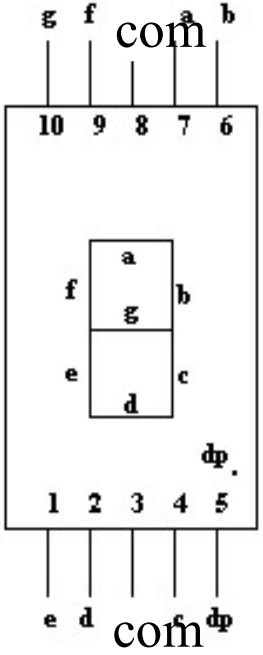
8*8点阵LED

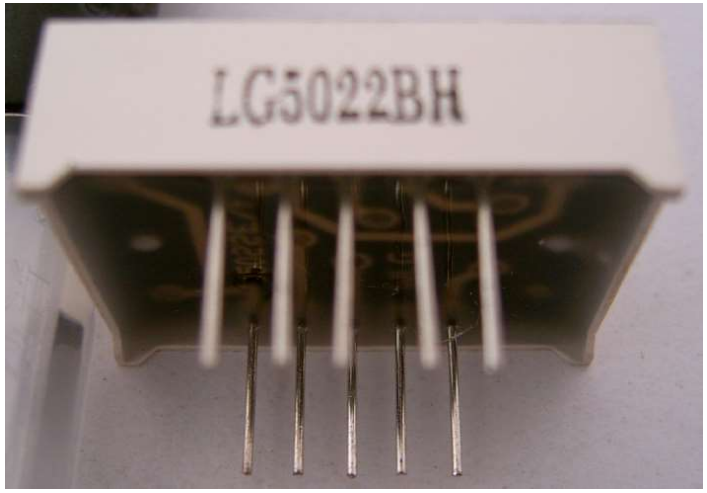
LED发光二极管显示器有多种结构形式,单段的圆形或方形LED常用来显示设备的运行状态,8段LED可以显示各种数字和字符,所以也称为LED数码管， 8段LED在控制系统中应用最为广泛。

点阵屏

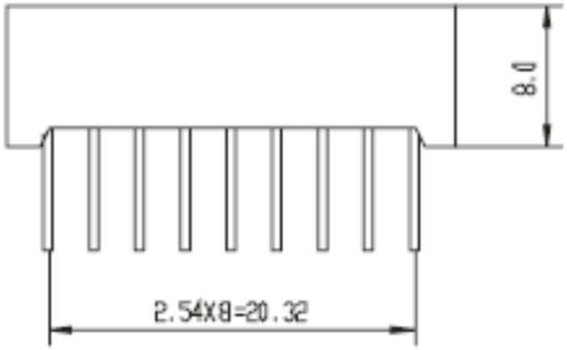
单联LED数码管

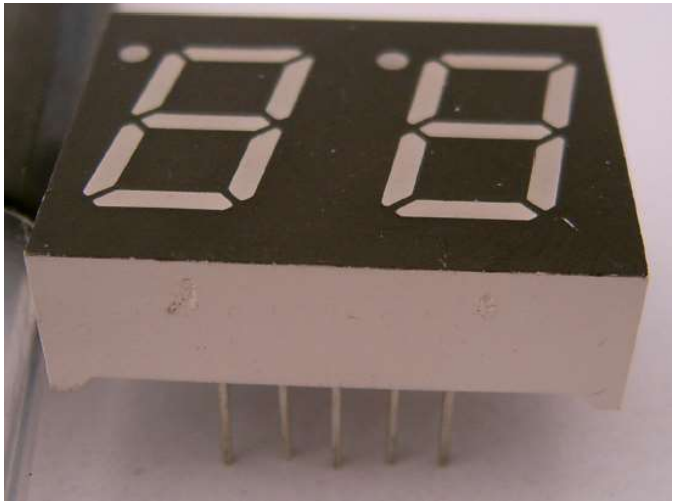

COM
COM
A
B
C
D
E
F
G
DOT

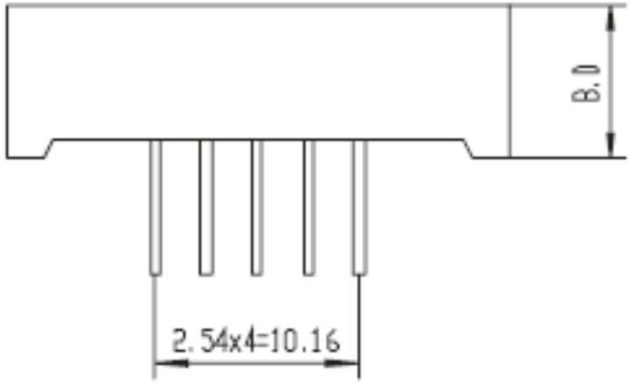

g
f
com
a
b
10
9
8
7
6
f
a
b
g
e
c
d
dp
1
2
3
4
5
e
d
com
dp

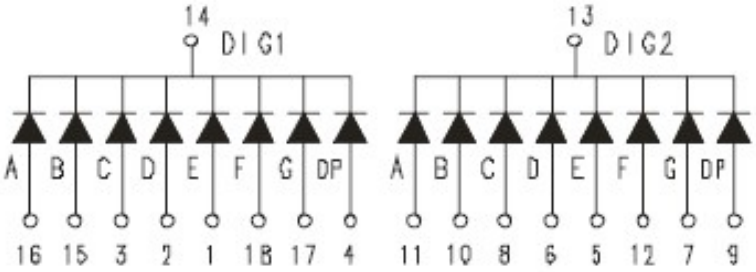

LG5022BH

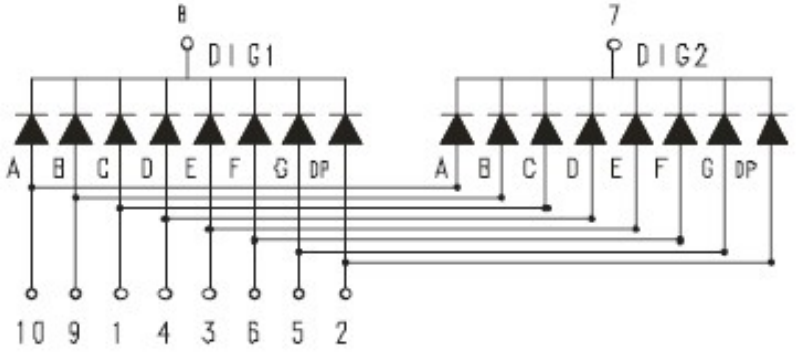
双联LED数码管

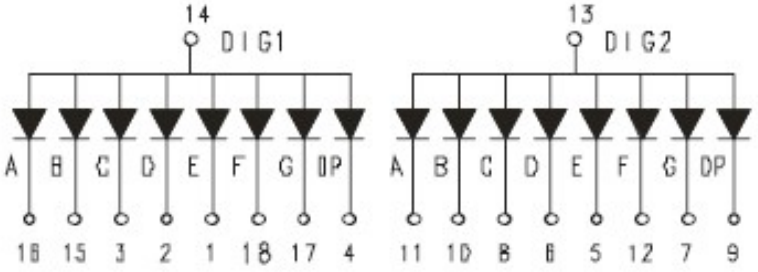

8.0
2.54X8=20.32

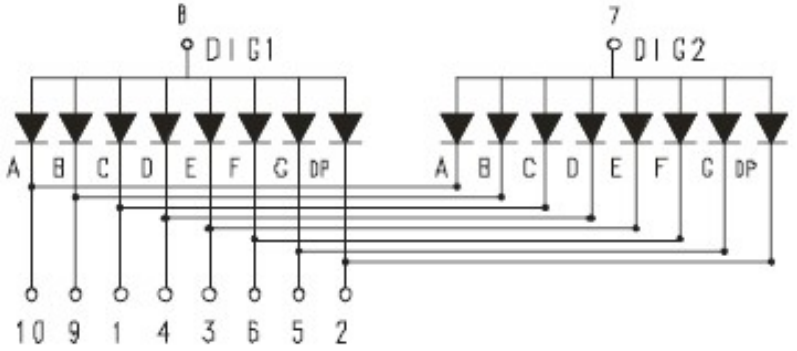


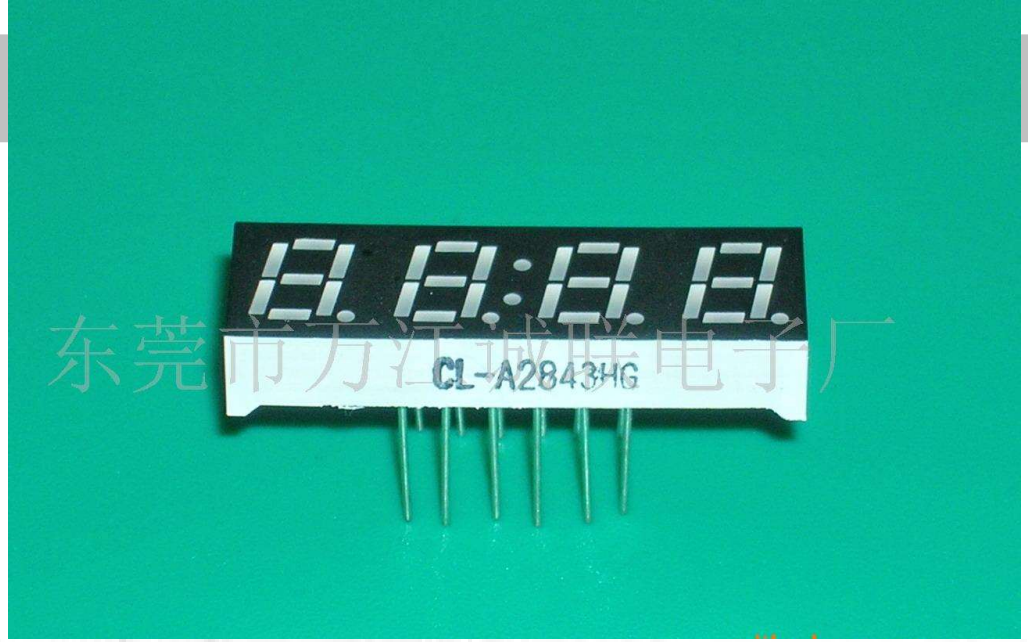

8.0
2.54x4=10.16


14
DIG1
13
DIG2
A B C D E F G DP
16 15 3 2 1 18 17 4
11 10 8 6 5 12 7 9

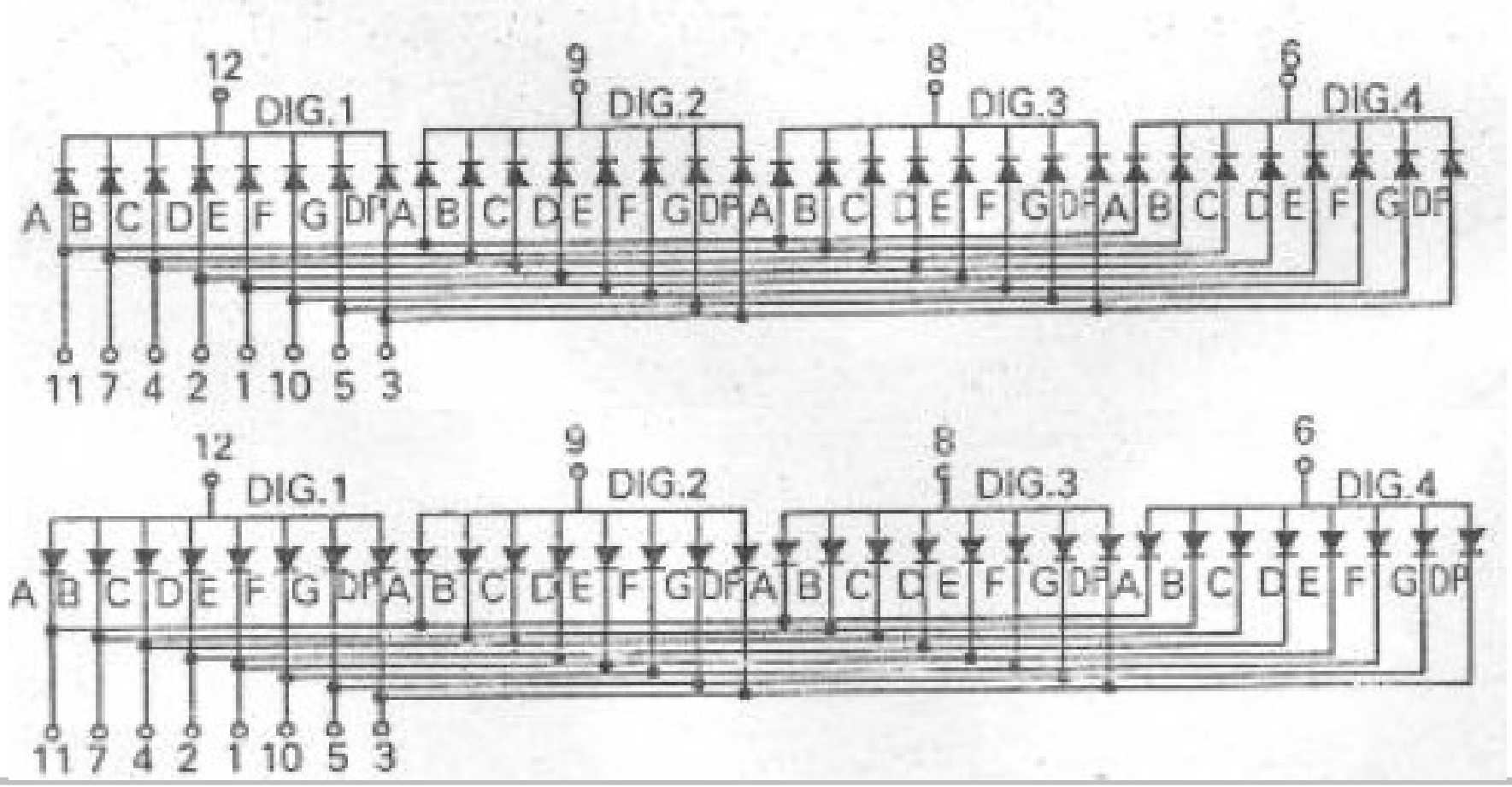

8
DIG1
7
DIG2
A B C D E F G DP
10 9 1 4 3 6 5 2


14
DIG1
13
DIG2
A B C D E F G DP
16 15 3 2 1 18 17 4
11 10 8 6 5 12 7 9


8
DIG1
7
DIG2
A B C D E F G DP
10 9 1 4 3 6 5 2



四联LED数码管



在使用发光二极管时,限流电阻的选择尤为重要,阻值过大或过小二极管都将不能正常发光,甚至烧毁器件.限流电阻Rx应满足如下条件:

$$Rx=(Vcc-Vg)/Ig$$

其中:Vcc —电压;

Vg—发光二极管工作时的管压降电压值(1.5-2.0V)

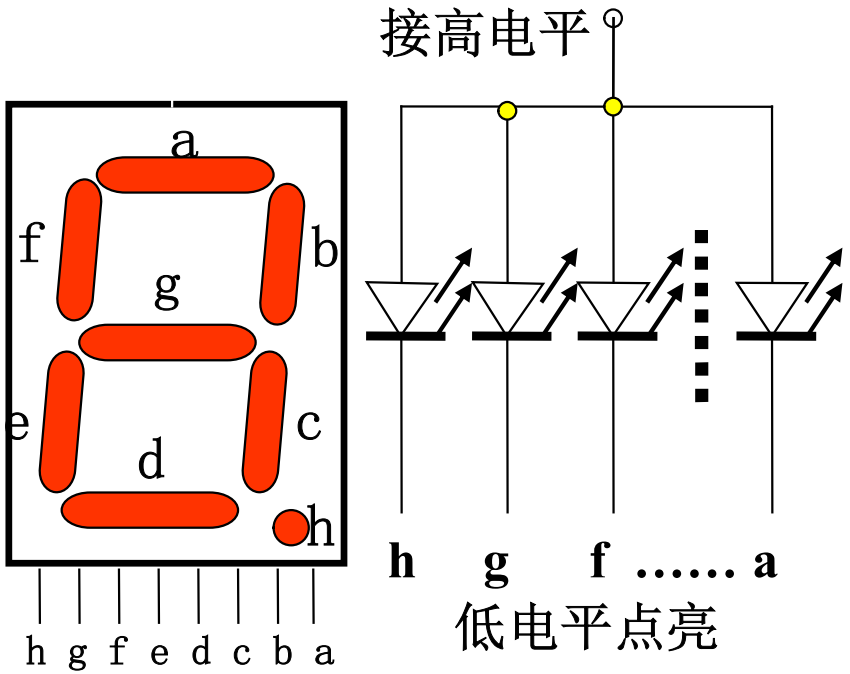
Ig—发光二极管工作电流范围(5mA-20mA)

这样,在5V电源电压下,限流电阻Rg的取值范围是:

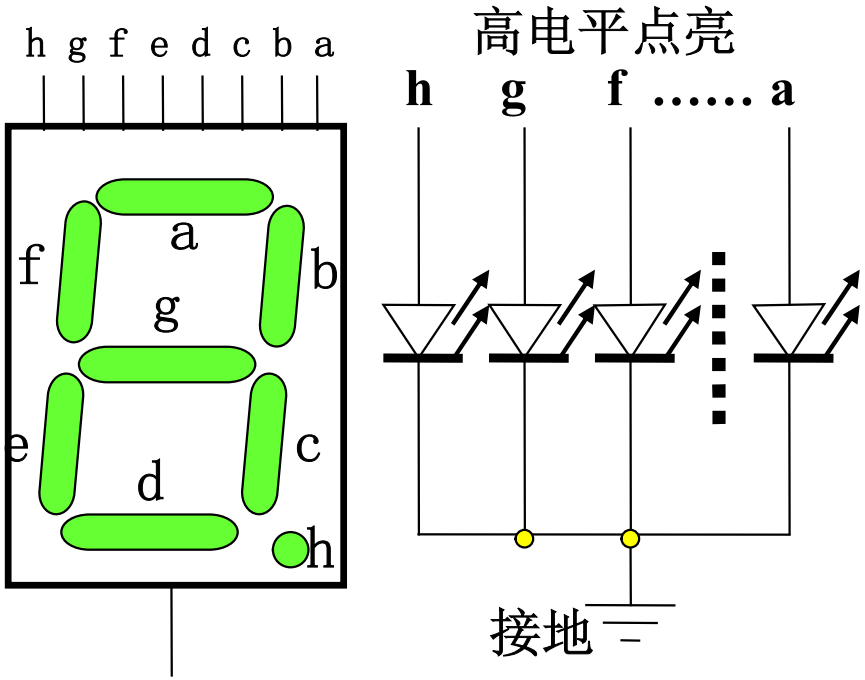
$$(5.0V-2.0V)/20mA<Rx<(5.0V-1.5V)/5mA$$

即R二应取值为150--700欧之间.

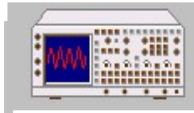
7.4.2 数码管结构:共阳与共阴



共阳极



共阴极



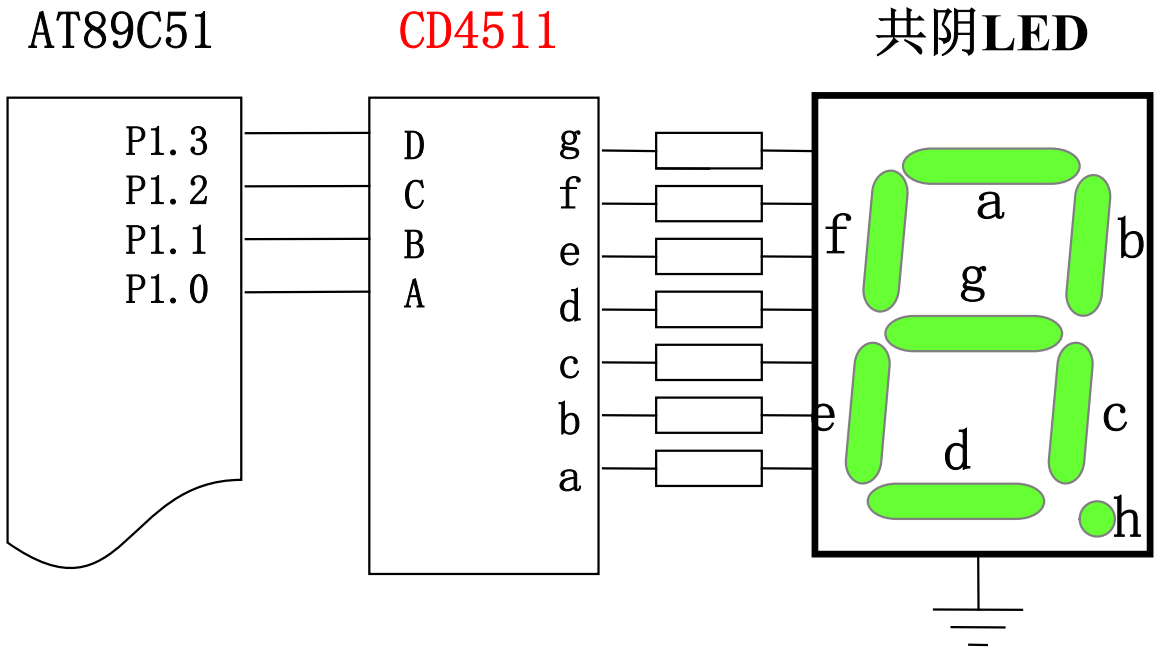
单片机系统扩展LED数码管时多用共阳LED:共阳数码管每个段笔画是用低电平(“0”)点亮的,要求驱动功率很小;

共阴数码管段笔画是用高电平(“0”)点亮的,要求驱动功率较大.

通常每个段码要串一个数百欧姆的限流电阻.

7.4.3 译码:

一、硬件译码



(a) 以硬件为主的接口方法

必须使用专用的译码驱动器，通过译码器把一位十六进制数（四位二进制）译码为相应的字形代码，然后由驱动器提供足够的功率去驱动LED。

CD4511/74LS48 是“BCD码→七段共阴译码/驱动”IC;

74LS47 是“BCD码→七段共阳译码/驱动”IC

硬件译码特点:

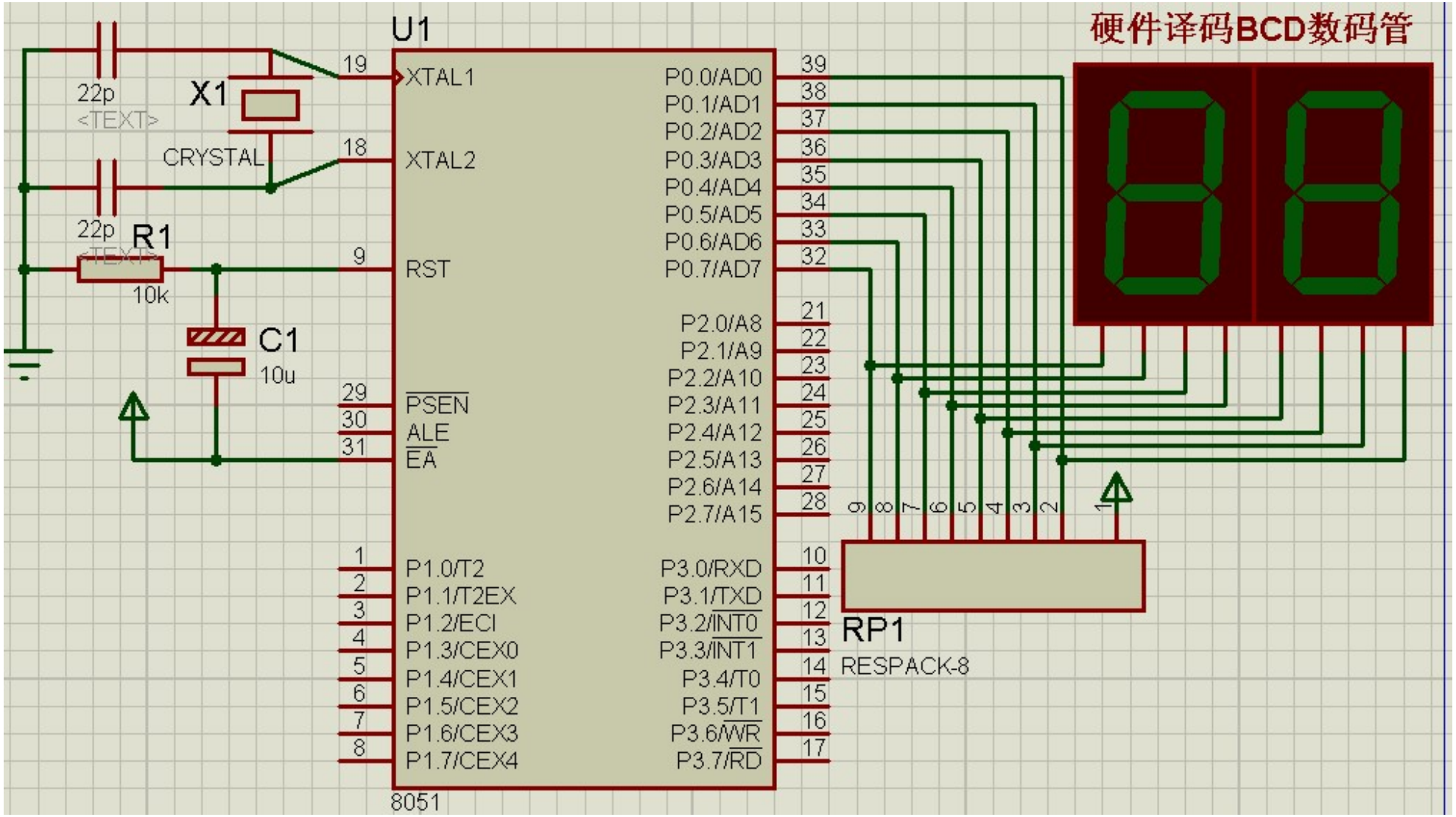
采用专用的译码/驱动器件,驱动功率较大;

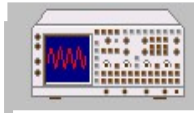
增加了硬件的开销;

软件编程简单;

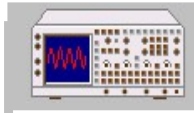
字型固定(比如:只有七段,只可译数字…).

任务4 硬件译码BCD数码管





```
        ORG     0000H
        LJMP START
        ORG 0030H
START:  MOV     P0,#12H             ;从P0口输出BCD码12
        LCALL   DELAY               ;延时
        MOV     P0,#34H             ;从P0口输出BCD码34
        LCALL   DELAY
        MOV     P0,#56H             ;从P0口输出BCD码56
        LCALL   DELAY
        MOV     P0,#78H             ;从P0口输出BCD码78
        LCALL   DELAY
        SJMP    START   ;循环
DELAY:  MOV     R5,#20  ;延时子程序
D2:     MOV     R6,#80
D1:     MOV     R7,#248
        DJNZ    R7,$
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET
        END
```

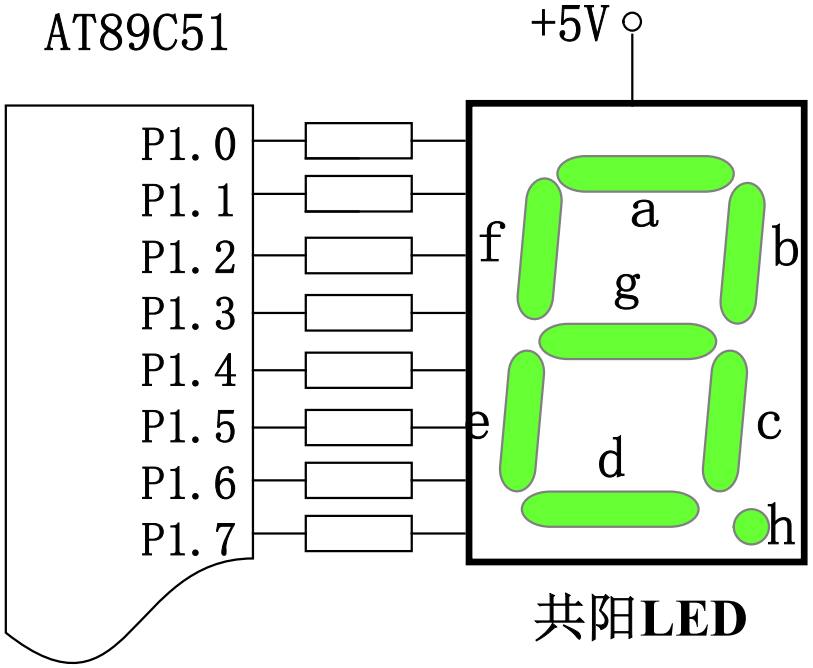


- 演 示

二、软件译码(常用)

（b）以软件为主的接口方法

主要以软件查表来代替硬件译码，也需简单的硬件电路配合。



软件译码特点:

1.不用专用的译码/驱动器件,驱动功率较小;

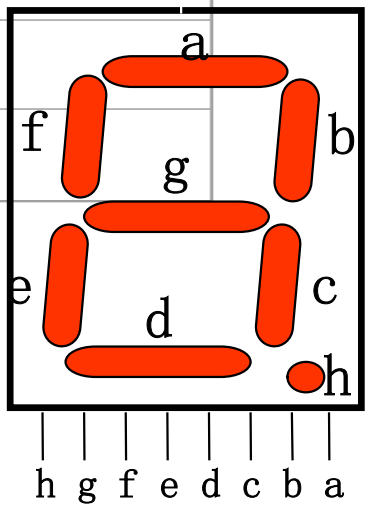
2.不增加硬件的开销;

3.软件编程较复杂;

4.字型灵活(比如:有八段,可译多种字符……).

七段LED的段选码

显示字符	共阴极段选码	共阳极段选码	显示字符	共阴极段选码	共阳极段选码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	D	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	8EH
4	66H	99H	P	73H	8CH
5	6DH	92H	U	3EH	C1H
6	7DH	82H	Γ	31H	CEH
7	07H	F8H	y	6EH	91H
8	7FH	80H	8.	FFH	00H
9	6FH	90H	“灭”	00H	FFH
A	77H	88H			
B	7CH	83H			



0-9共阳段码表

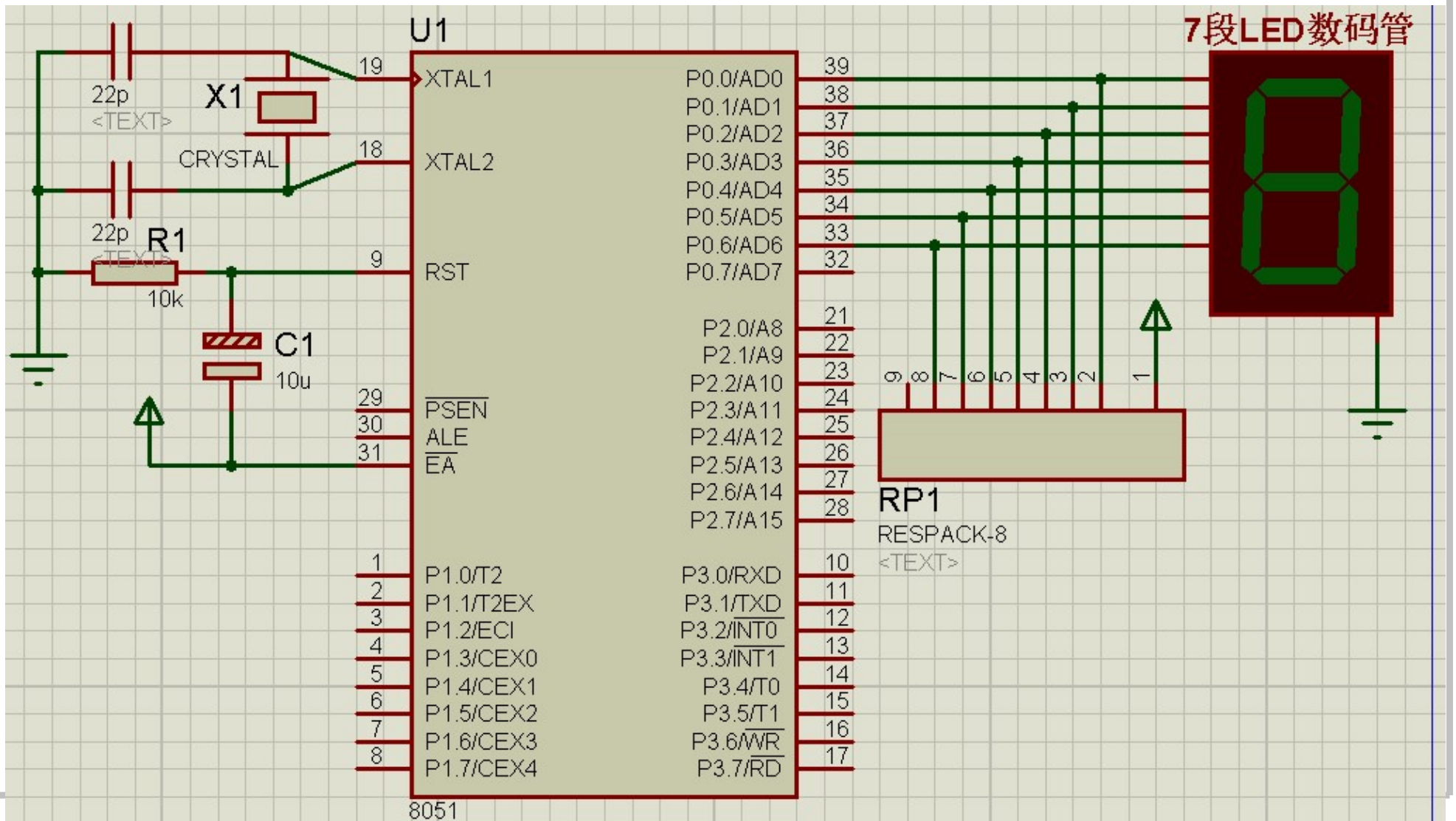
TAB: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H

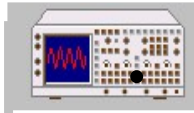
0-9共阴段码表

TAB1:DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH

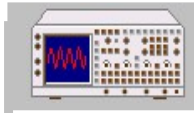
任务4　7段LED数码管

循环显示0~9

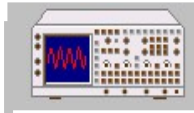




```
        ORG     0000H
        LJMP START
        ORG 0030H
START:          MOV     DPTR,#TABLE     ;DPTR指向段码表首地址
S1:     MOV     A,#00H
        MOVC    A,@A+DPTR       ;查表取得段码
        CJNE    A,#01H,S2       ;判断段码是否为结束符
        SJMP    START
S2:     MOV     P0,A            ;段码送数码管显示
        LCALL   DELAY           ;延时
        INC     DPTR
        SJMP    S1
DELAY:          MOV     R5,#20          ;延时子程序
D2:     MOV     R6,#20
D1:     MOV     R7,#248
        DJNZ    R7,$
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET
TABLE:          DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
                ;0-9共阴段码表
        DB      01H                             ;结束符
        END
```



- 演 示



7.1.4 LED数码管显示方式:静态与动态

1.静态显示:

LED数码管采用静态显示与单片机相接时，共阴接法的LED的公共端接地；共阳接法的LED公共端接高电平。每个显示器的段码线（a-h）分别与一个8位的锁存器输出口相连。各位的显示字符一经确定，相应锁存的输出将维持不变。

特点:原理简单,显示亮度强,无闪烁,占用I/O资源较多，适用于显示位数不多的情况。

8位锁存器：

A、直接采用并行I/O口

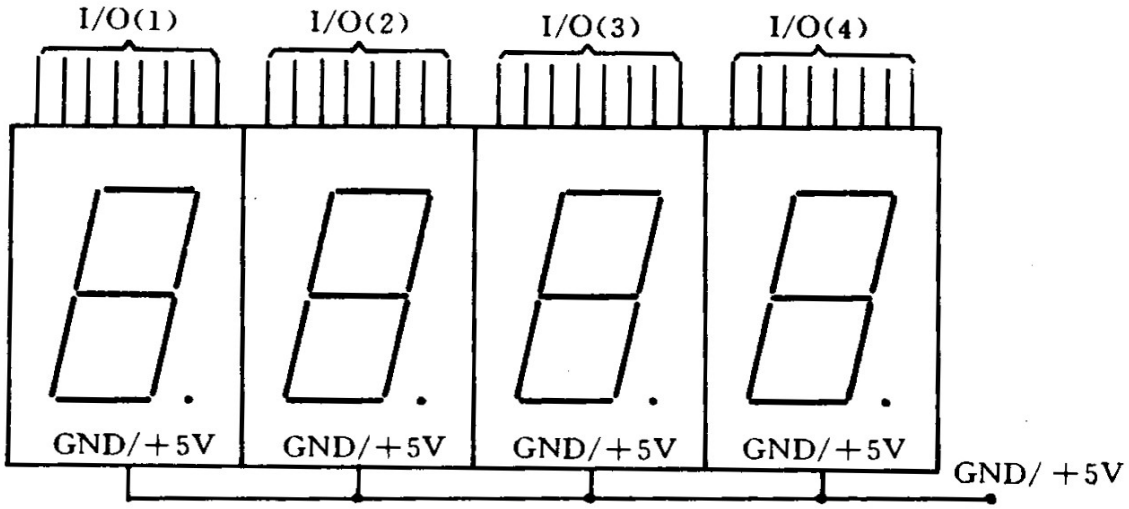
B、采用串入/并出的移位寄存器

C、采用具有三态功能的锁存器，如CD4511、MC14495

A、直接采用并行I/O口

显示过程中持续得到送显信号,与各数码管接口的I/O口线是专用的.无闪烁,使用的元器件较多,占I/O线多,无须扫描,节省CPU时间,编程简单.

连接:所有LED的位选均共同连接到+Vcc或GND,每个LED的8根段选线分别连接一个8位并行I/O口,从该I/O口送出相应的字型码显示字型.



分析:说明4个共阴极LED静态显示3456数字的工作过程.

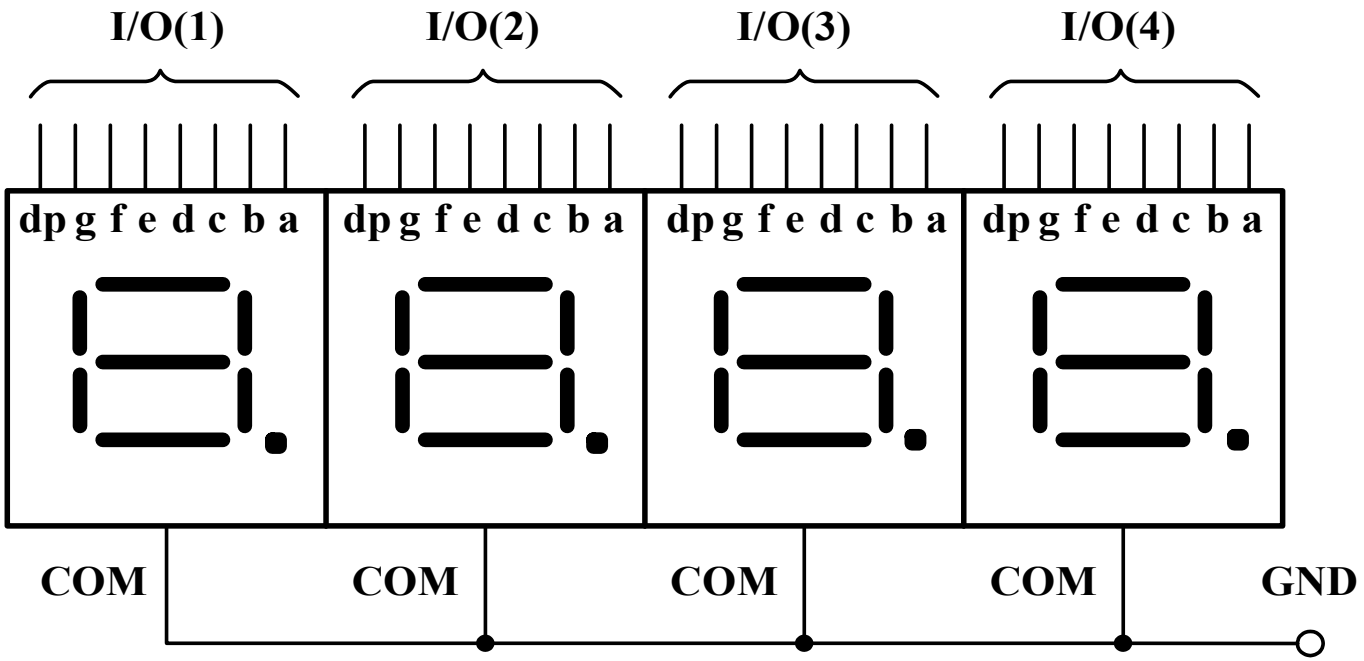


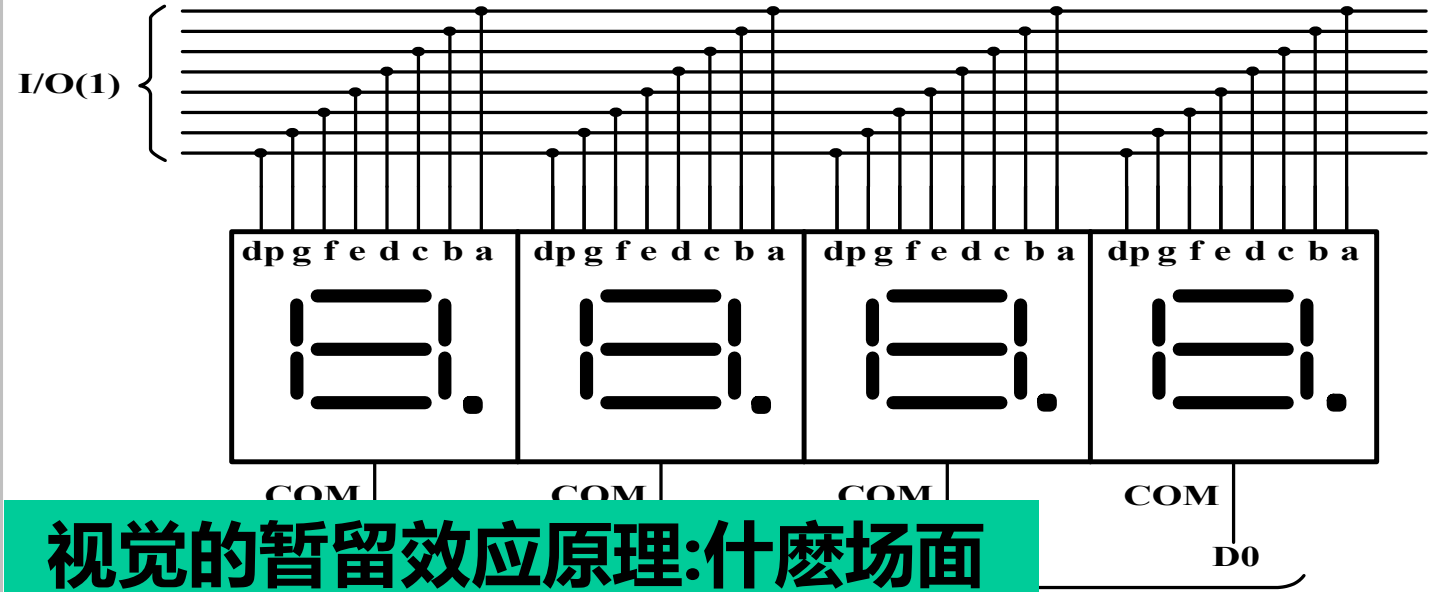
图　LED静态显示方式

所有COM端连接在一起并接地:

- 首先由I/O口(1)送出数字3的段选码4FH, 左边第一个LED显示3;
- 接着由I/O口(2)送出数字4的段选码66H, 左边第二个LED显示4;
- 同理,由I/O口(3)送出数字5的段选码6DH,左边第三个LED显示5
- 最后由I/O口(4)送出数字6的段选码7DH, 第四个LED显示6.

2.动态显示(常用,有特色):

各数码管在显示过程中轮流得到送显信号,与各数码管接口的I/O口线是共用的.有闪烁,使用的元器件较少,占I/O线少,必须扫描,花费CPU时间,编程复杂.(有多个LED时尤为突出)



硬件连接:所有LED的段选线共同连接在一起共用一个8位I/O口,而每个LED的位选线分别由一根相应的I/O口线控制.

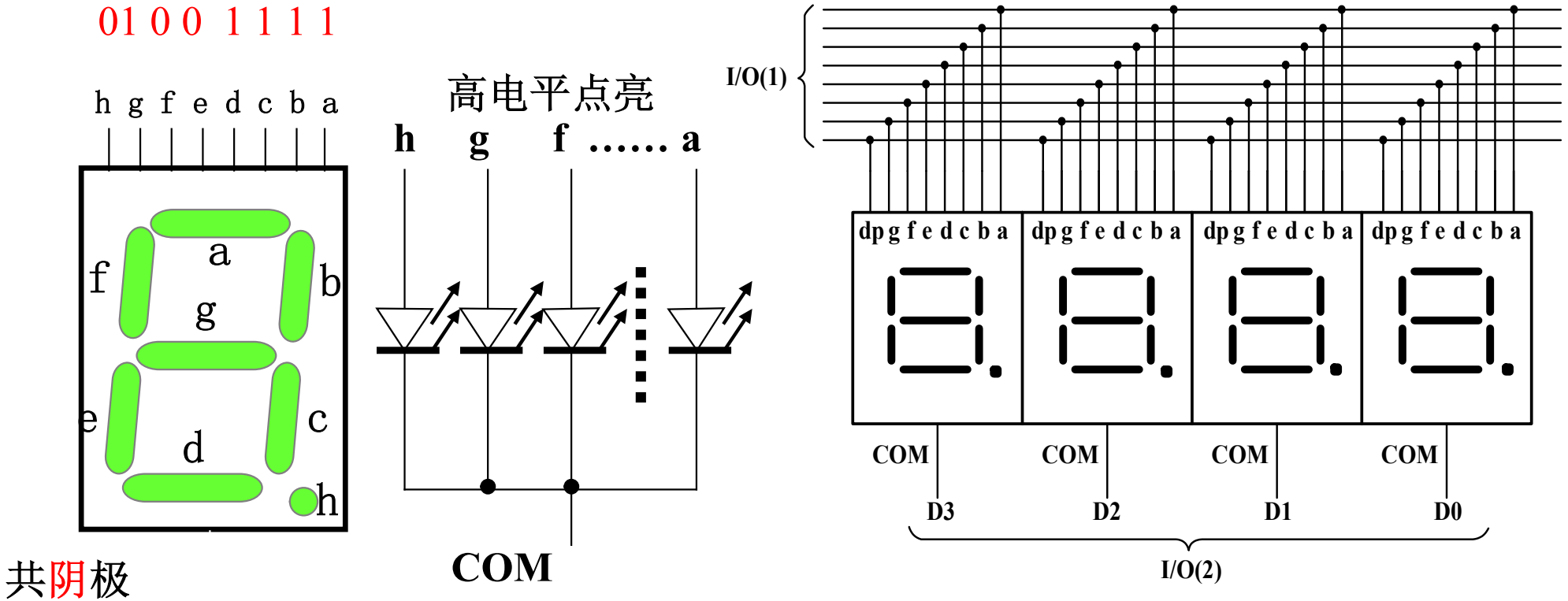
视觉的暂留效应原理:什麽场面被人看了一眼,这场面都会在人的眼里停留0.1秒,电影在1秒中拍摄24张照片,所以在播放时,里面的事物总是移动的.

逐位扫描显示方式:从段选口(段控)送出某位LED的字型码,然后选通该位LED(位控),并保持一段延时时间(1ms 记忆).然后选通下一位,直到所有位扫描完.

分析:说明4位共阴极LED动态显示3456数字的工作过程

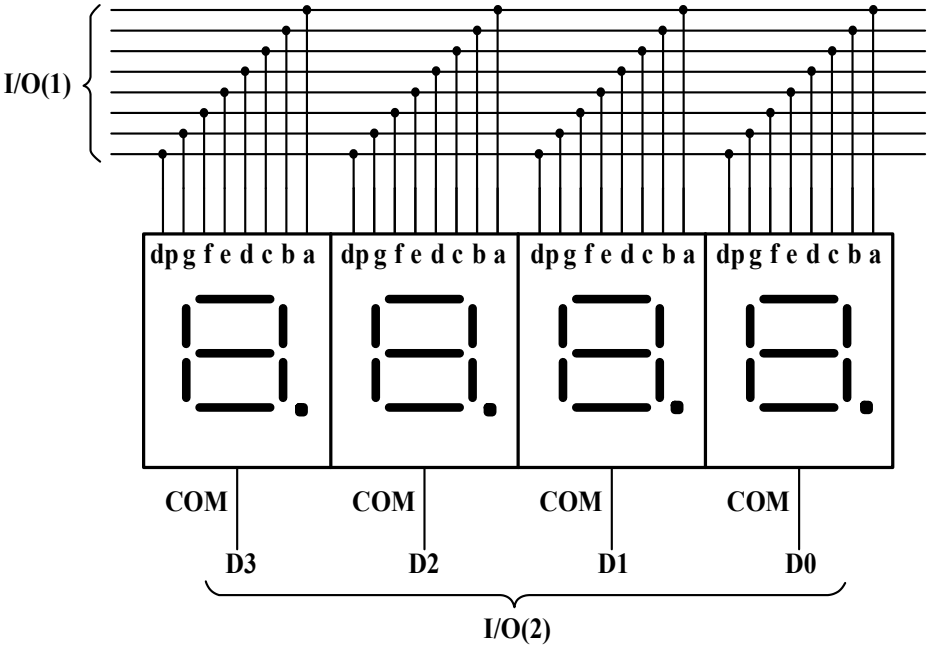
如图,首先由I/O口(1)送出数字3的段码4FH(0100 1111), 到4个LED共同的段选线上,

接着由I/O口(2)送出位码××××0111到位选线上,其中数据的高4位为无效的×,唯有D_3的COM端为低电平“0”,因此只有该LED的发光管因阳极接收到高电平“1”的g、d、c、b、a段有电流流过而被点亮,也就是显示出数字3,而其余3个LED因其COM端均为高电平“1”而无法点亮;

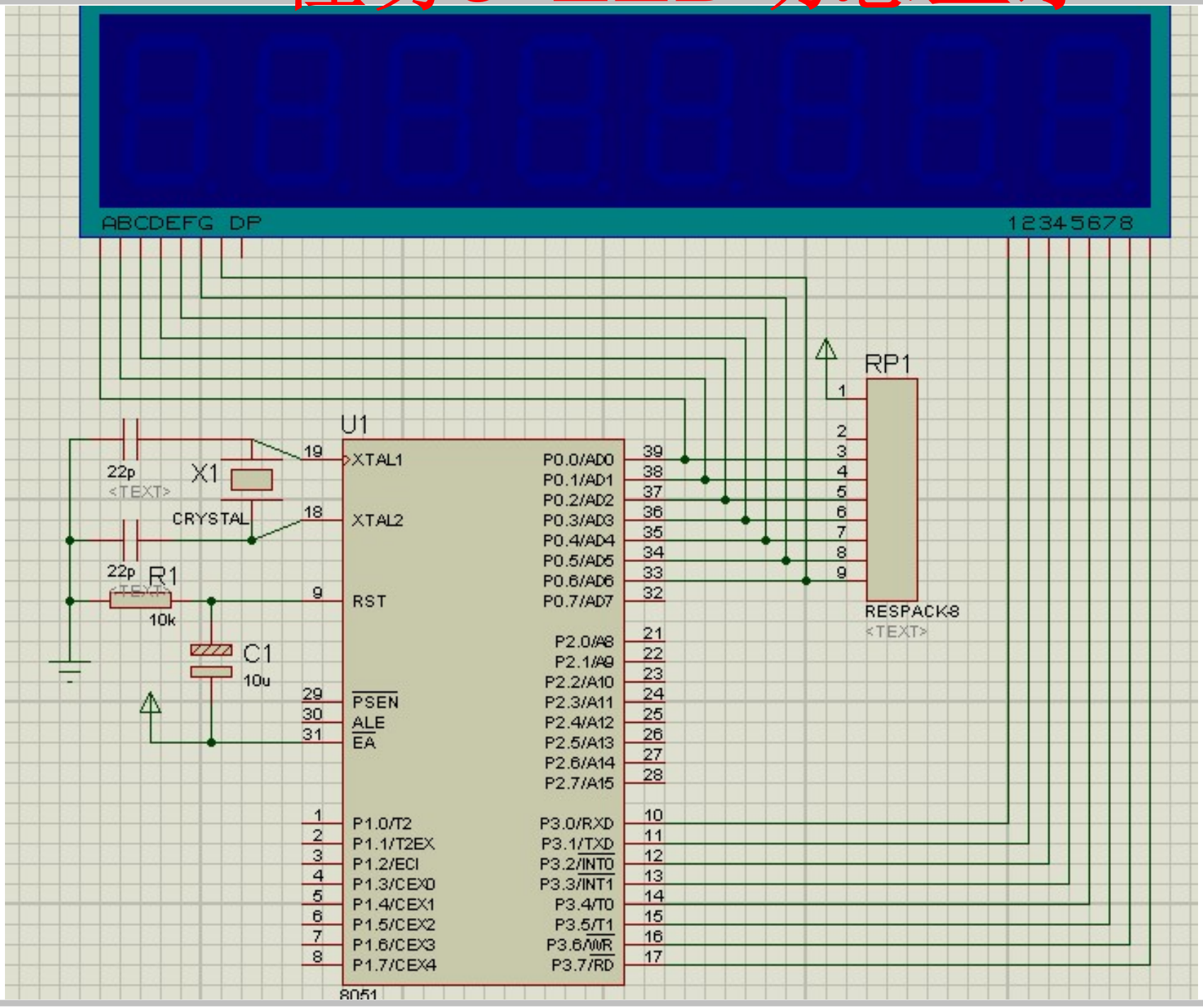


显示一定时间后,再由I/O口(1)送出数字4的段选码66H到段选线上,接着由I/O口(2)送出点亮D2的位选码××××1011到位选线上,此时只有该LED的发光管因阳极接受到高电平“1”的g、f、c、b段有电流流过因而被点亮,也就是显示出数字4,而其余3位LED不亮;

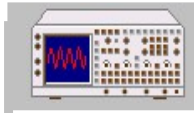
如此再依次送出第三个LED、第四个LED的段选与位选的扫描代码,就能一一分别点亮各个LED,使4个LED从左至右依次显示3、4、5、6.



任务5 LED动态显示

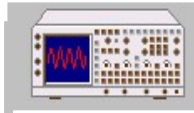


要求：显示数字0-7

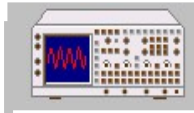


程序设计

```
        ORG     0000H
        LJMP START
        ORG 0030H
START:  MOV     DPTR,#TABLE     ;DPTR指向段码表首地址
        MOV     R7,#7FH         ;设置动态显示扫描初值
S1:     MOV     A,#00H
        MOVC    A,@A+DPTR       ;查表取得段码
        CJNE    A,#01H,S2       ;判断段码是否为结束符
        SJMP    START
S2:     MOV     B,A             ;段码送B保存
        MOV     A,R7
        RL      A               ;显示位扫描值左移1位
        MOV     P3,A            ;显示位扫描值送P3口
        MOV     R7,A
        MOV     P0,B            ;显示段码送P0显示
        LCALL   DELAY           ;延时
        INC     DPTR
        SJMP    S1
```



```
DELAY:    MOV    R5,#20              ;延时子程序
D2:   MOV    R6,#20
D1:   NOP
      DJNZ   R6,D1
      DJNZ   R5,D2
      RET
TABLE:    DB     3FH,06H,5BH,4FH,66H,6DH,7DH,07H        ;0-7共阴段码表
      DB     01H                                    ;结束符
      END
```



7.1 键盘与计算机接口

涉及如下问题：

1. 为什么要软件延时去抖动；
2. 独立式和矩阵式键盘接口电路的设计方法；
3. 独立式和矩阵式键盘程序的设计方法。

7.1.1 键盘结构与工作原理

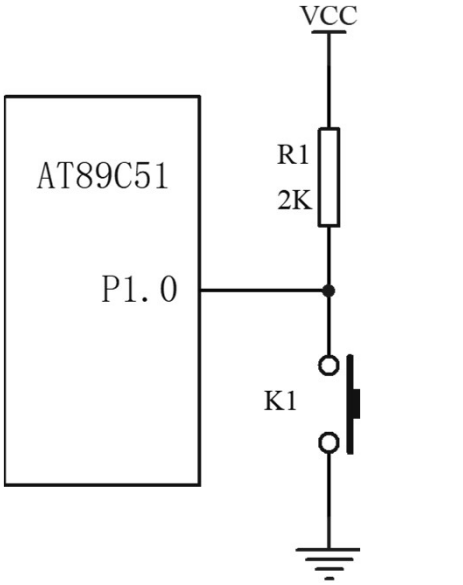
按键开关介绍

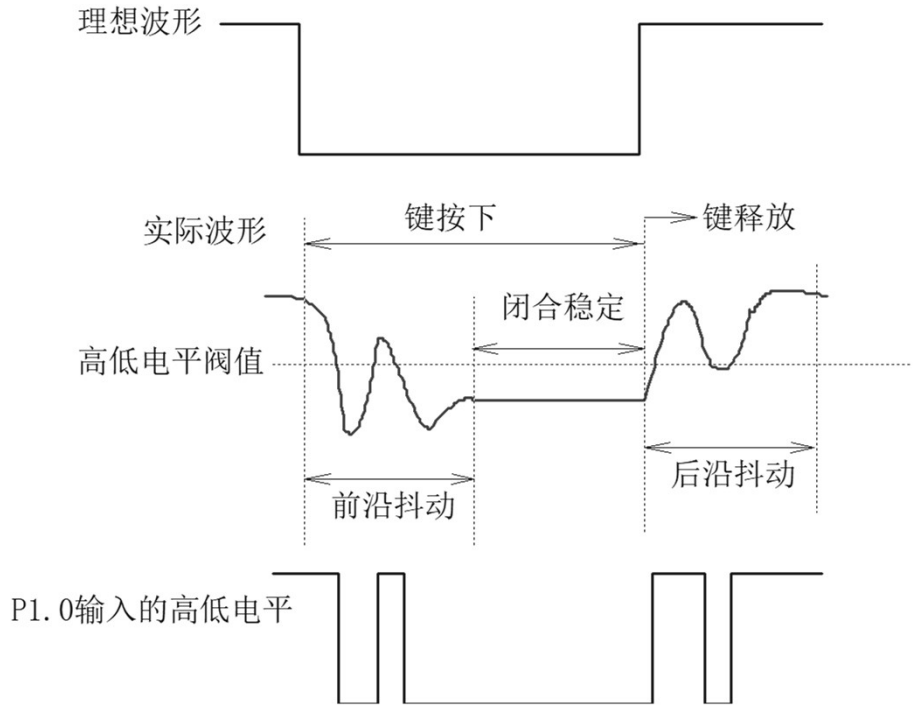
单片机中的键盘通常由按键开关组成，按键开关的外形和参数如下图所示，它是一种常开型按键开关，为了便于安装固定，它有四个管脚，其管脚说明如下图中的文字所示，在常态时开关触点（1和2）处于断开状态，只有按下按键时开关触点才闭合短路，所以可以用万用表检测开关的管脚排列、好坏和质量。

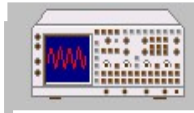
按键闭合、断开时的抖动

在单片机中，按键通常与I/O端口相连，如下图左边所示。

当按键开关K1未按下时，P1.0输入为高电平，而当按键K1闭合时，P1.0输入为低电平，由于开关为机械弹性开关，当机械触点断开、闭合时，由于机械触点的弹性作用，一个机械开关闭合时不会马上稳定地闭合接通，断开时也不会马上断开，而是在闭合、断开的瞬间伴随有一连串的抖动，如下图右边所示，抖动时间的长短与开关的机械特性决定，一般为5～10ms。





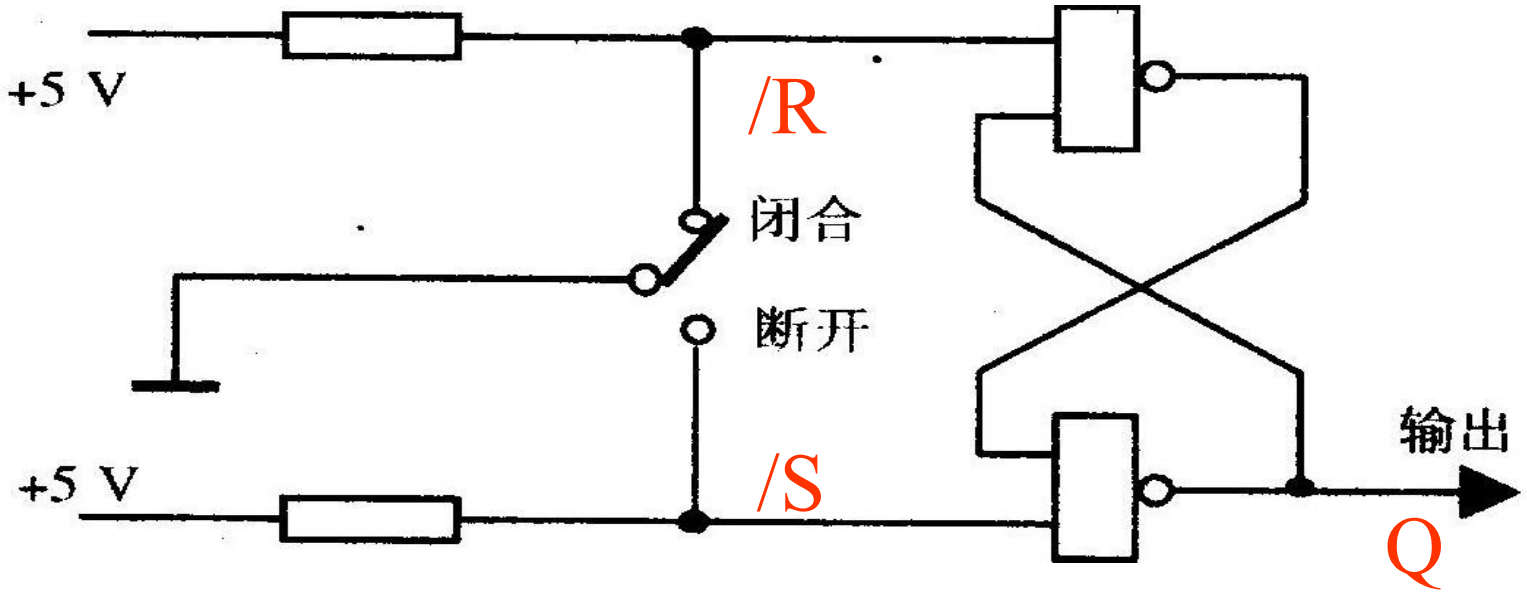


按键抖动消除办法

按键抖动是一种普遍的现象。如上页图所示，抖动将造成I/O端口输入的高低电平多次变化，使单片机系统误动作，一次按键产生多次按键效果，因此必须采取措施消除。

按键抖动消除可以采用硬件和软件方法消除，**硬件去抖动电路**如RS触发器等，由于硬件去抖动电路需要额外的硬件电路，使产品成本增加，硬件电路复杂，所以一般应用较少。在单片机中广泛采用的是**软件延时去抖动**，由图可知，按键闭合时存在前沿抖动，一般时间为5～10ms，因此我们可在按键按下后，延时10ms左右避开前沿抖动，然后再判断按键是否按下，即P1.0是否仍为低电平，如果仍为低电平，此时才确认为一次完整有效的按键闭合，否则认为只是抖动或干扰，系统对此不作出响应。

硬件：采用消除键抖动电路



原理：基本RS触发器，/S＝0，Q＝1

/R＝0，Q＝0

抖动时，/R=/S=1,Q不变。

软件：采用延时判别程序

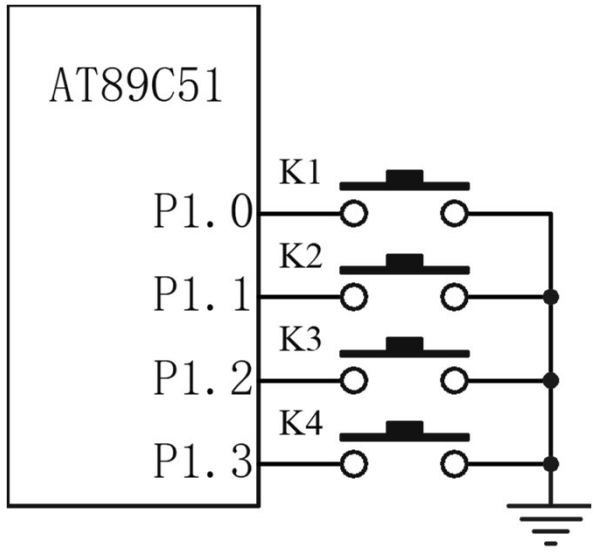
（具体程序后叙）

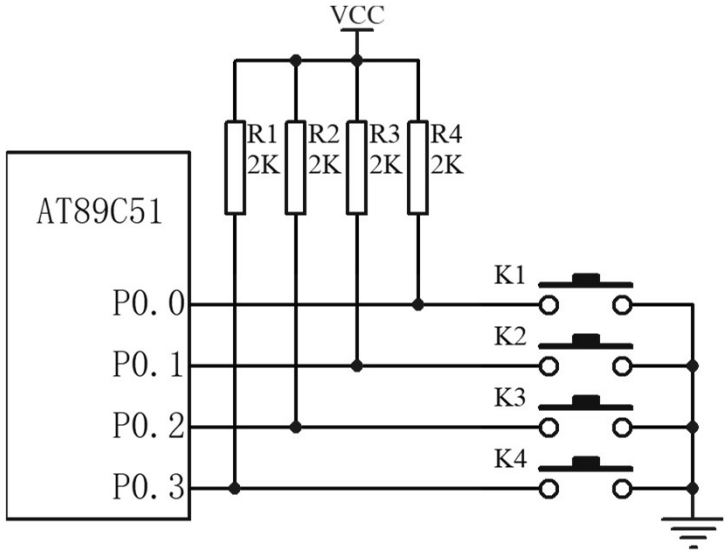
键盘的分类

- 键盘分编码键盘和非编码键盘。键盘上闭合键的识别由专用的硬件编码器实现，并产生键编码号或键值的称为编码键盘，如计算机键盘.
- 而靠软件编程来识别的称为非编码键盘；
- 在单片机组成的各种系统中，用的最多的是非编码键盘。也有用到编码键盘的。
- 非编码键盘分为：独立式键盘和行列式（又称为矩阵式）键盘。

§ 7.1.2 独立式键盘接口设计

独立式键盘的结构如下图所示，**组成键盘的各按键相互独立，每个按键独立地与一个I/O端口相连**，结构简单，其中左图适合于端口内部有上拉电阻的端口，如P1、P2、P3口，所以外部不用上拉电阻，电路更简单，成本更低。右图适合于端口内部没有上拉电阻地端口，如P0口，所以外部必须使用上拉电阻，成本稍高，所以一般尽量使用左图的形式 。





独立式按键的软件结构

设计独立式键盘的控制程序，必须解决以下几个问题：

（1）检测有无按键按下。

先将各按键相连的I/O端口置为高电平1，然后检测各I/O端口是否仍全为高电平，如果不是，表明有按键按下。

（2）如果有键按下，运用软件去抖动

在有键按下的情况下，延时10ms，再次检测是否有键按下，如果是，表明确实有键按下，否则表示只是干扰或抖动。

（3）确认有键按下，暂存键值，等键释放

这主要是为了保证一次按键仅执行一次按键功能，防止按住按键不放时，执行多次按键功能。

（4）判断按键情况，执行相应的按键功能

任务6 独立式键盘控制LED灯

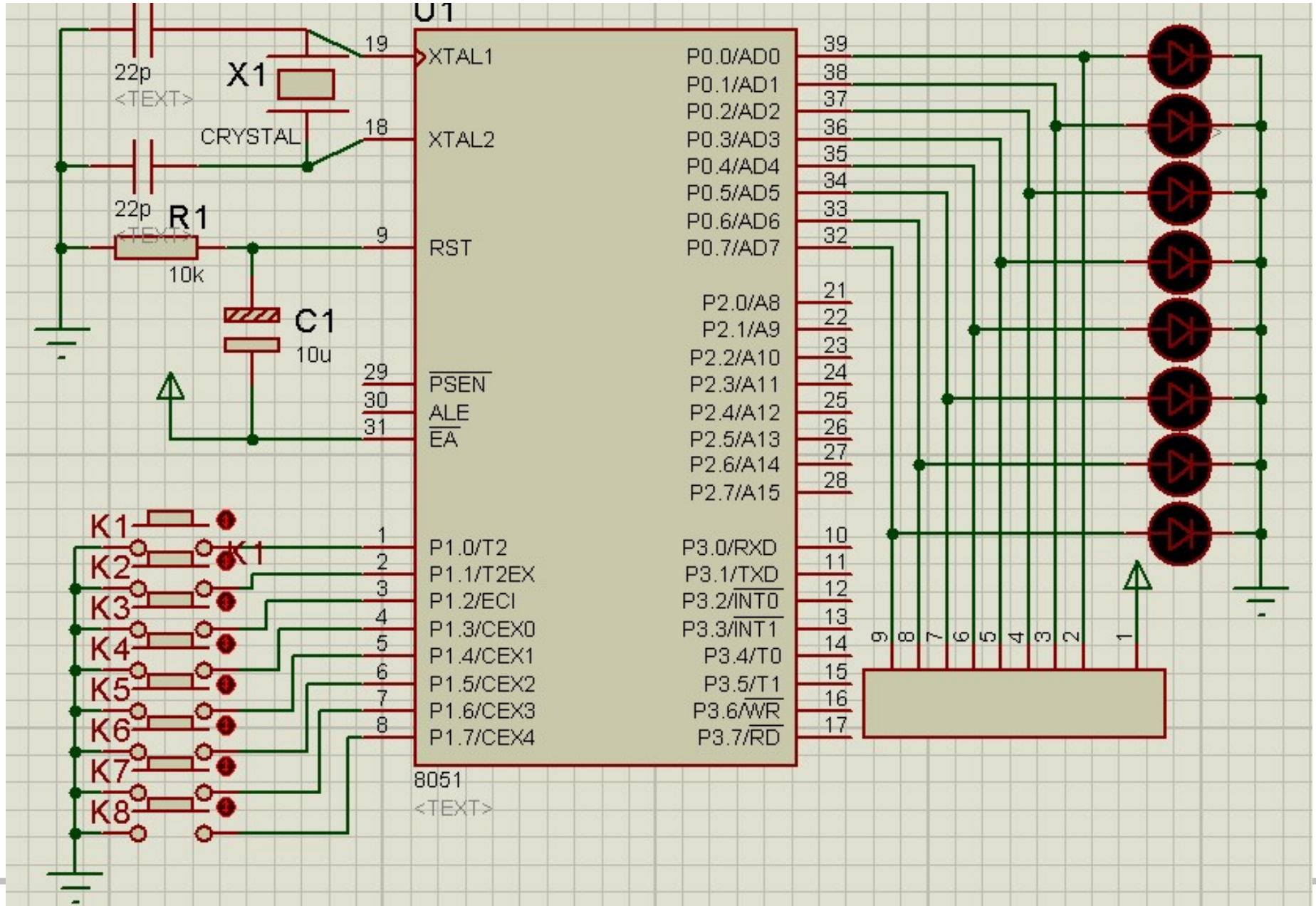
- 要求：

P1口作为输入口：接键盘电路K1~K8

P0口作为输出口：接LED灯D1~D8

P1口键盘按下，控制P0口相应位的灯的亮灭：

按第一次，灯亮，按第二次，灯灭.....


U1
8051
<TEXT>
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0/T2
P1.1/T2EX
P1.2/ECI
P1.3/CEX0
P1.4/CEX1
P1.5/CEX2
P1.6/CEX3
P1.7/CEX4
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
X1
CRYSTAL
22p
<TEXT>
22p
<TEXT>
R1
10k
C1
10u
K1
K2
K3
K4
K5
K6
K7
K8
K1

本任务采用查询方式判断是否有键盘按下，程序思路大致如下：

（1）首先进行程序初始化。

置位P1口作为输入口； P0口清零，使LED灯灭。

（2）查询是否有键按下。

按位查询P1口电位，有低电平表明有按键按下。

（3）延时去抖动。

若有按键按下，延时10ms，再次读入判断相应位状态，判断是否仍然有键按下，如果有，表明确实有键按下，否则表明只是抖动或干扰信号。

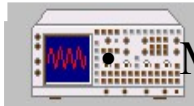
（4）判断按键情况，执行相应的按键功能

即如果按键K1按下，D1亮； K2按下，D2亮…… 。

```
ORG 0000H
START:      LJMP 0030H
        ORG 0030H
      MOV  P1,#0FFH
      MOV P0,#00H                ；初始化
MAIN:
        JNB P1.0,M0              ;查询是否有键按下
        JNB P1.1,M1              ;查询K2键是否按下
        JNB P1.2,M2              ;查询K3键是否按下
        JNB P1.3,M3              ;查询K4键是否按下
        JNB P1.4,M4              ;查询K5键是否按下
        JNB P1.5,M5              ;查询K6键是否按下
        JNB P1.6,M6              ;查询K7键是否按下
        JNB P1.7,M7              ;查询K8键是否按下
```

（1）首先进行程序初始化。

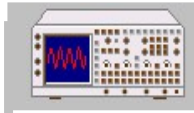
（2）查询是否有键按下。



```
M0:   LCALL DELAY          ;延时，反弹跳
      JNB P1.0,P10         ;K1键压下处理
      SJMP MAIN
M1:   LCALL DELAY          ;延时，反弹跳
      JNB P1.1,P11         ;K2键压下处理
      SJMP MAIN
M2:   LCALL DELAY          ;延时，反弹跳
      JNB P1.2,P12         ;K3键压下处理
      SJMP MAIN
M3:   LCALL DELAY          ;延时，反弹跳
      JNB P1.3,P13         ;K3键压下处理
      SJMP MAIN
M4:   LCALL DELAY          ;延时，反弹跳
      JNB P1.4,P14         ;K5键压下处理
      SJMP MAIN
M5:   LCALL DELAY          ;延时，反弹跳
      JNB P1.5,P15         ;K6键压下处理
      SJMP MAIN
M6:   LCALL DELAY          ;延时，反弹跳
      JNB P1.6,P16         ;K7键压下处理
      SJMP MAIN
M7:   LCALL DELAY          ;延时，反弹跳
      JNB P1.7,P17         ; K8键压下处理
      SJMP MAIN
```

（3）延时去抖动。

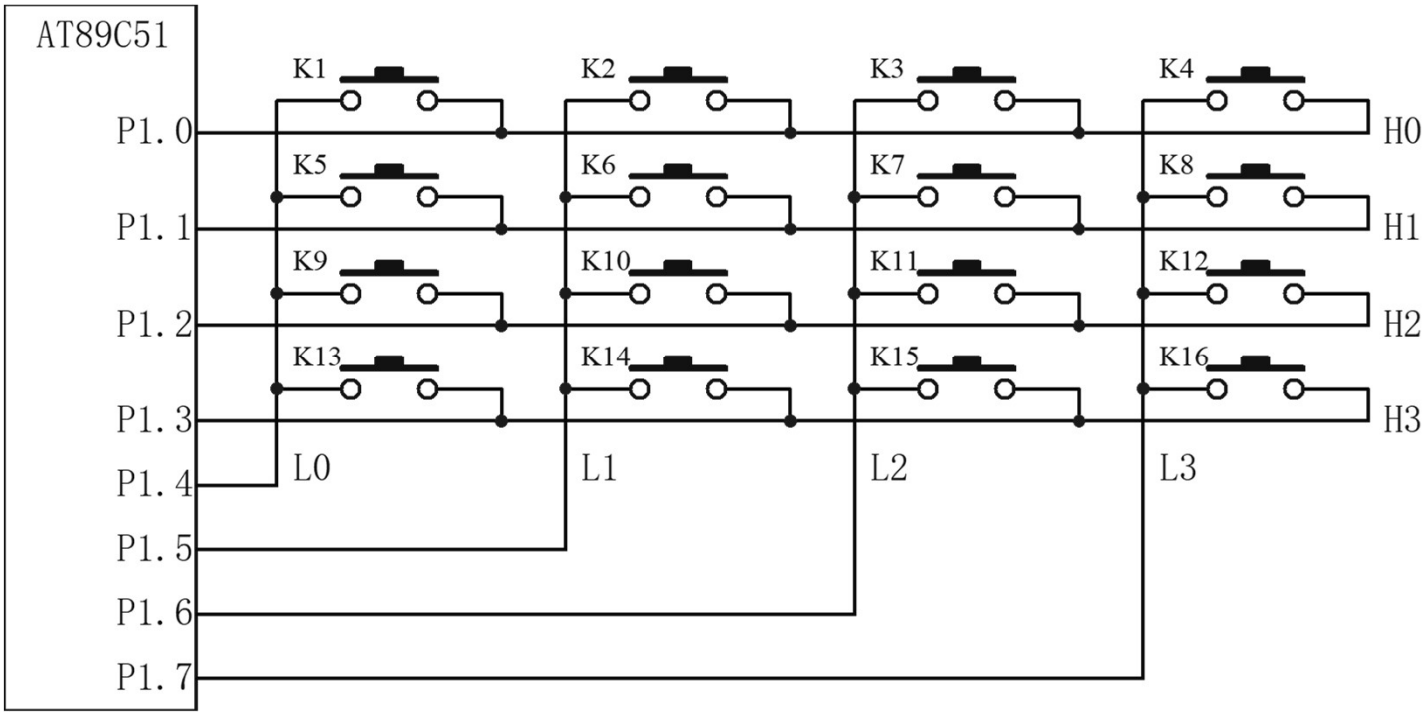
（4）判断按键情况，执行相应的按键功能

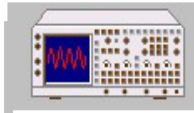


```
P10:    CPL P0.0
        SJMP MAIN
P11:    CPL P0.1
        SJMP MAIN
P12:    CPL P0.2
        SJMP MAIN
P13:    CPL P0.3
        SJMP MAIN
P14:    CPL P0.4
        SJMP MAIN
P15:    CPL P0.5
        SJMP MAIN
P16:    CPL P0.6
        SJMP MAIN
P17:    CPL P0.7
        SJMP MAIN
DELAY:          MOV     R5,#50H                 ;延时子程序
D2:     MOV     R6,#0F0H
D1:     NOP
        DJNZ    R6,D1
        DJNZ    R5,D2
        RET
        END
```

§7.1.3 矩阵式键盘接口设计

独立式键盘虽然硬件、软件结构简单，但在按键数量较多的情况下，将占有较多的I/O端口，所以在按键数量较多的情况下，一般采用可以有效减少I/O端口数量的矩阵式键盘。矩阵式键盘又称为行列式键盘，采用行、列线结构，按键设置在行列线的交叉点上，如下图所示，H0～H3为四条行线，L0～L3为四条列线，在行列相交的每个交点上通过按键来连通，按键开关的一个触点连行线，一个触点连列线，从而组成4×4矩阵16键键盘

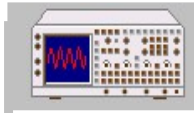




独立式和矩阵式键盘比较

独立式键盘硬件结构简单，软件编程较简单，但每个按键独自占用一个I/O端口，在按键数量较多的情况下，将占有较多的I/O端口。所以，独立式键盘一般运用于按键数量不多的场合。

矩阵式键盘能有效的减少I/O端口的占用量，但因为各按键不是单独的占有I/O端口，从而给按键的判断带来难度，造成编程难度加大。



1、矩阵式键盘的接口

矩阵式键盘的接口方法有许多，例如直接接口于单片机的I/O口上；利用扩展的并行I/O接口；用串行口扩展并行I/O口接口；利用一种可编程的键盘、显示接口芯片8279进行接口等。其中，利用扩展的并行I/O接口方法方便灵活，在单片机应用系统中比较常用。

矩阵式键盘结构

- ✓ 矩阵式键盘由行线和列线构成
- ✓ 行线接到输入口上，通过上拉电阻接到+5V
- ✓ 列线接到输出口上
- ✓ 按键位于行、列的交叉点上
- ✓ 适用于需要按键较多的场合

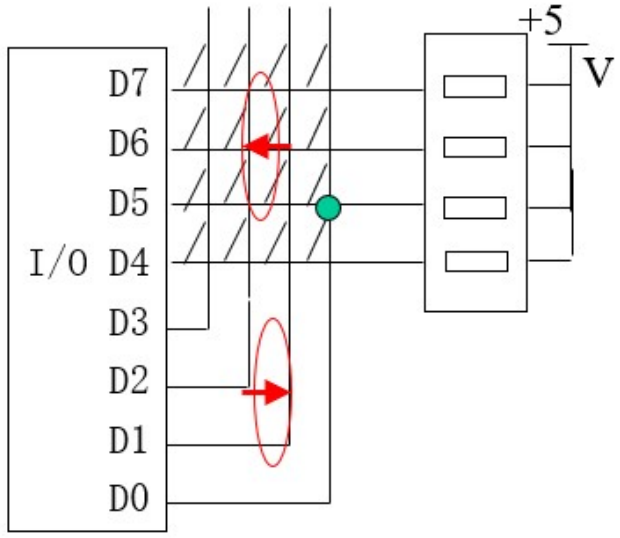
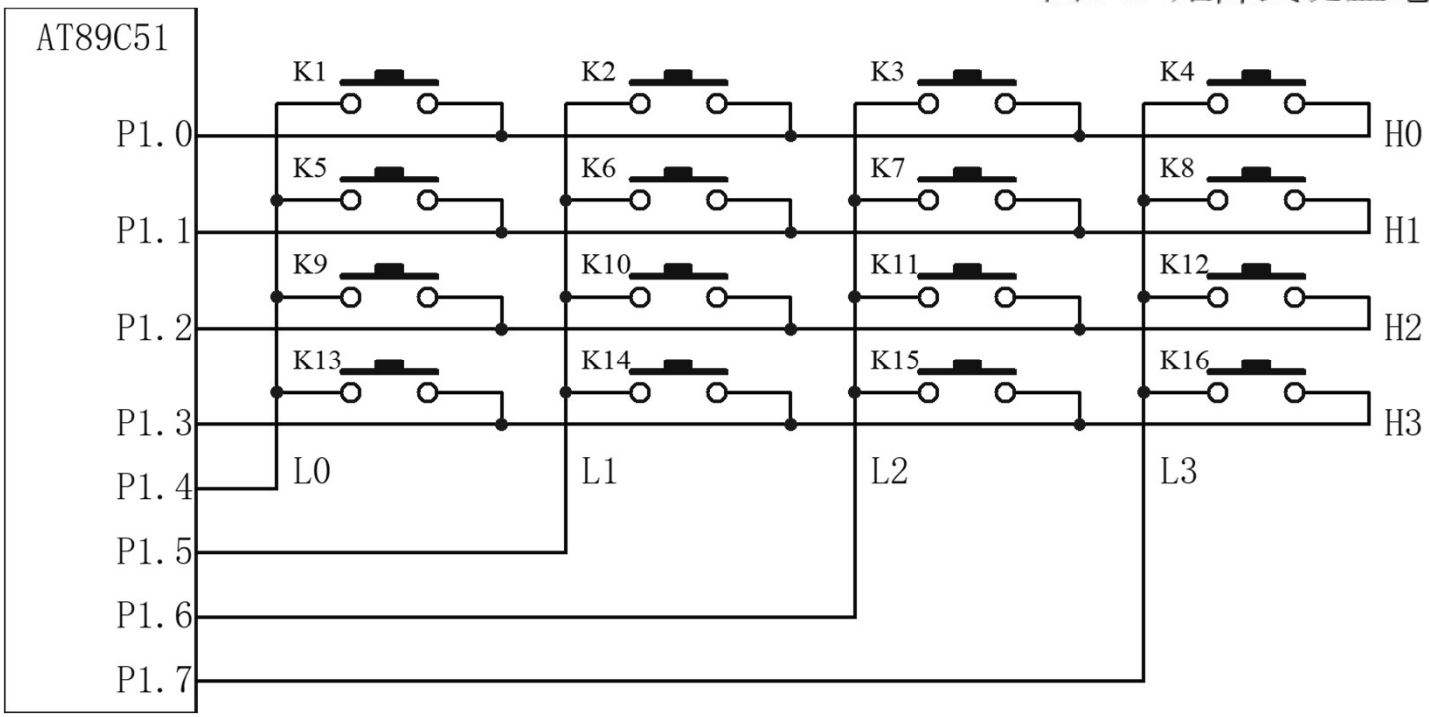


图7-2 矩阵式键盘电路原理



2．键编码及键值

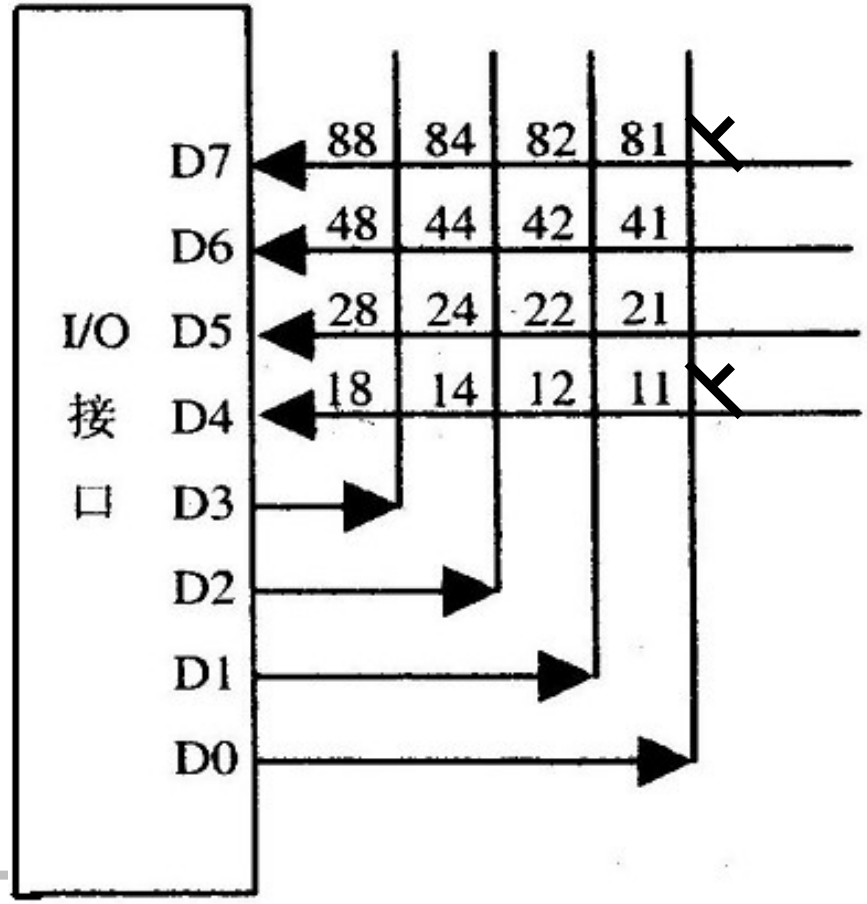
(1) 用键盘连接的I/O线的二进制组合表示键码

例如用4行、4列线构成的16个键的键盘，可使用一个8位I/O口线的高、低4位口线的二进制数的组合表示16个键的编码。

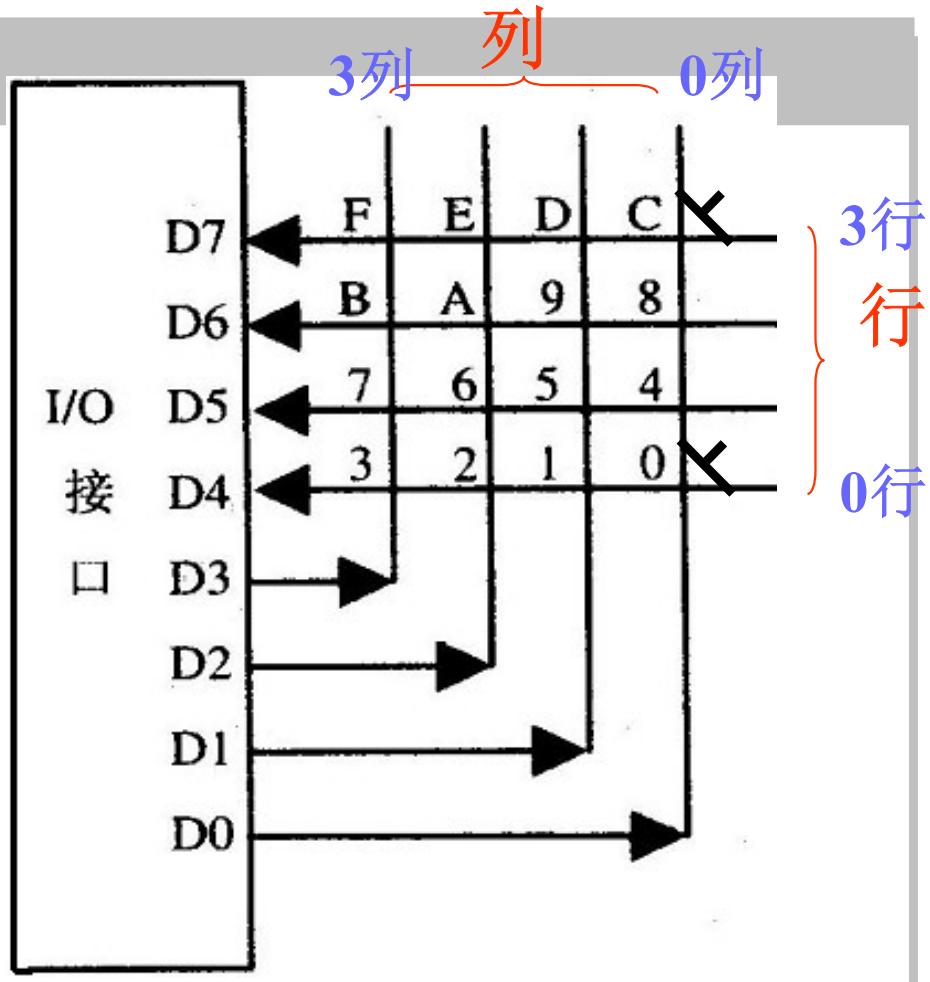
如图所示，各键相应的键值为：

88H、84H、82H、81H、
48H、44H、42H、41H、
28H、24H、22H、21H、
18H、14H、12H、11H。

这种键值编码软件较为简单直观，但离散性大，不便安排散转程序的入口地址。



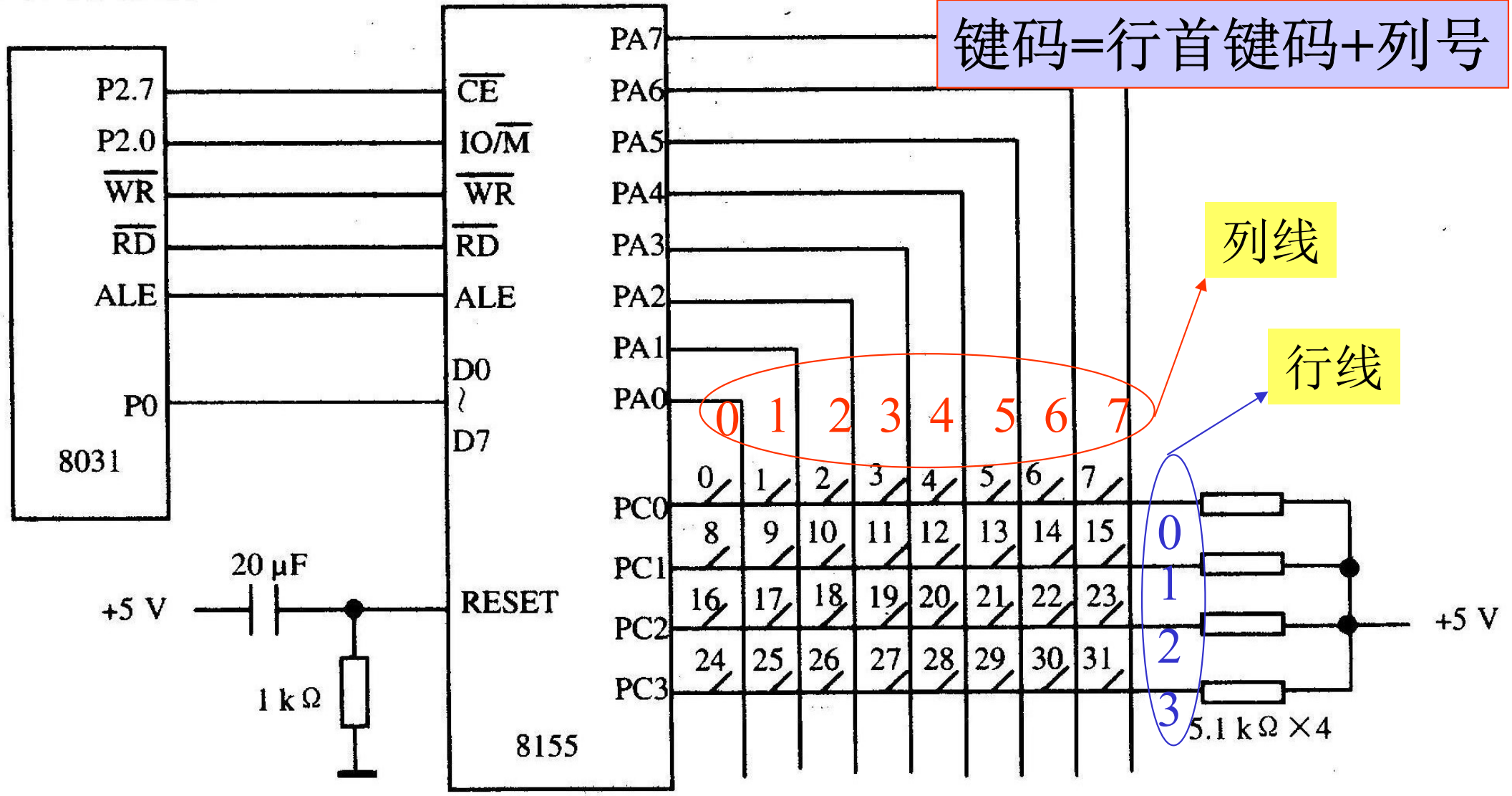
(2) 顺序排列键编码



如图所示，这种方法键值的形成要根据I/O线的状态作相应的程序处理。键码可按下式形成：

键码=行首键码(高位)+列号(低位)

行首键码	列号	键码
D4：0行→0000	D0：0列→0000	0000,0001,0010,0011
D5：1行→0100	D1：1列→0001	0100,0101,0110,0111
D6：2行→1000	D2：2列→0010	1000,1001,1010,1011
D7：3行→1100	D3：3列→0011	1100,1101,1110,1111



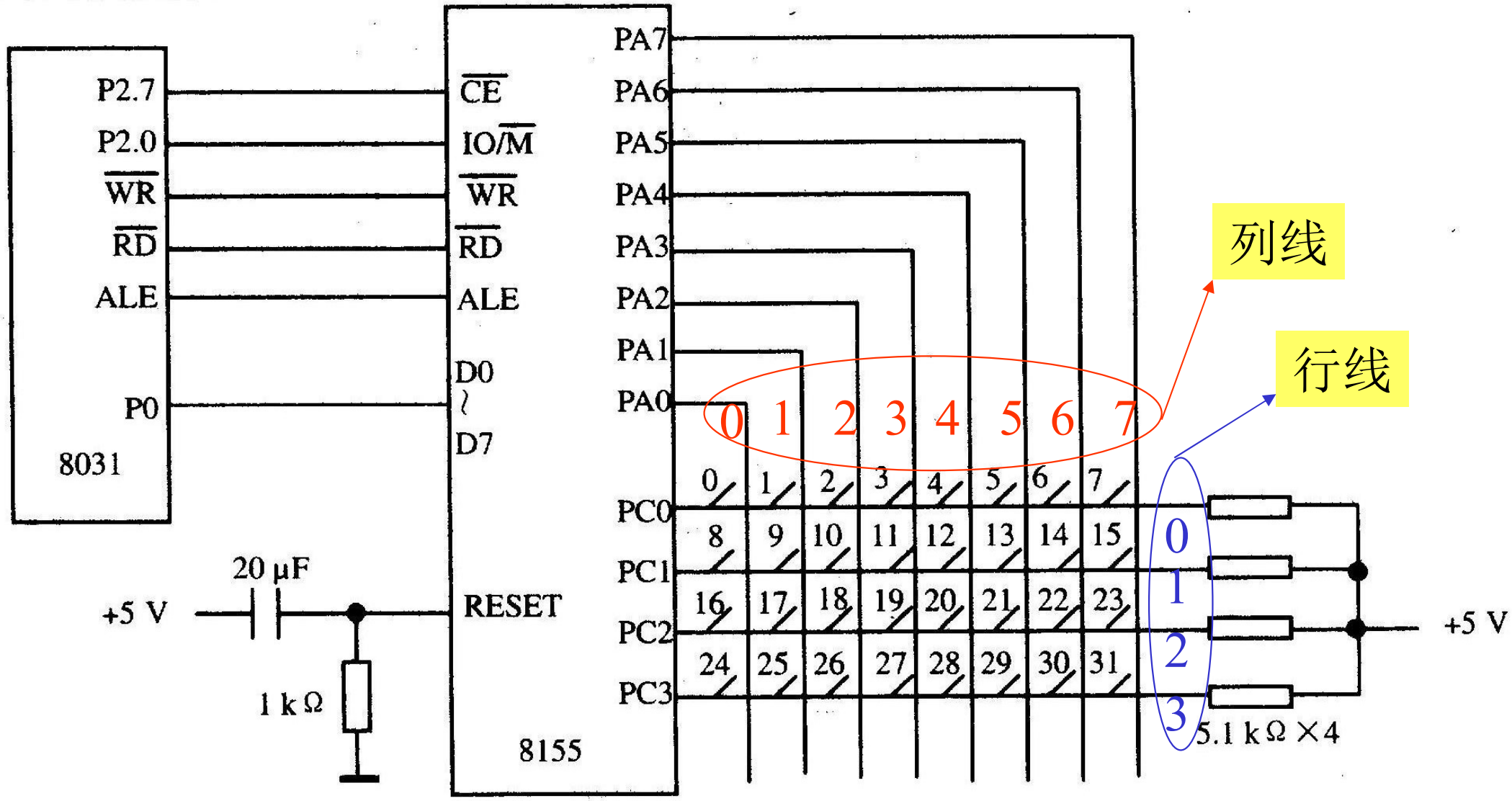
行首键码 0行: 00000 列号 000~111 键码: 0行: 00000~00111

1行: 01000 1行: 01000~01111

2行: 10000 2行: 10000~10111

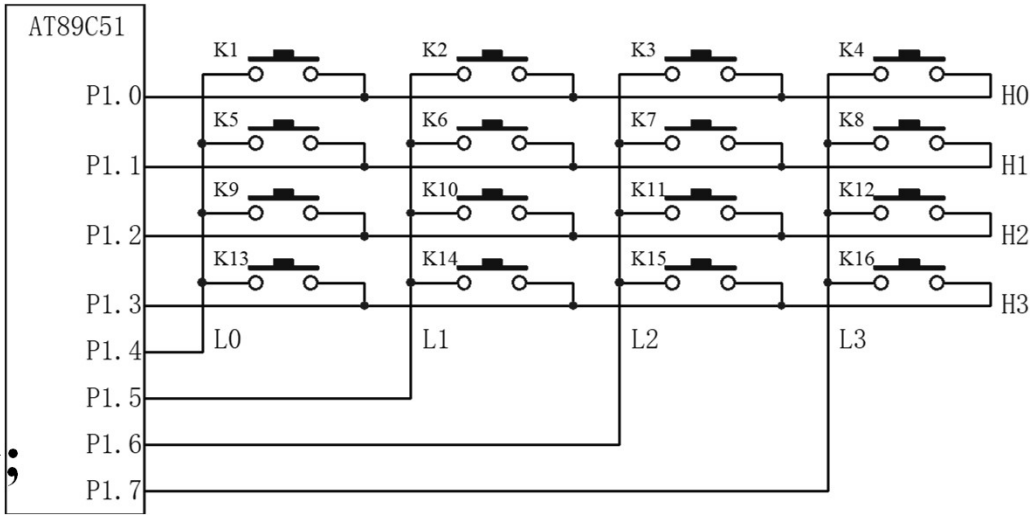
3行: 11000 3行: 11000~11111

PA口每位依次送出0，如有某键按下，则在PC口能读到相应的值，结合PA口的信息，则能确定键值。



8155扩展I/O口组成的行列式键盘
共32个键

3. 行列式键盘工作原理



矩阵式键盘工作原理

- ✓ 无按键按下时,行线输入高电平;
- ✓ 有按键按下时,行线输入电平由

与此行线相连的列线电平决定.

- 如果列线输出低电平,则行线电平为低
- 如果列线输出高电平,则行线电平为高

- ✓ 为了确认按键位置，必须将行、列线配合使用。

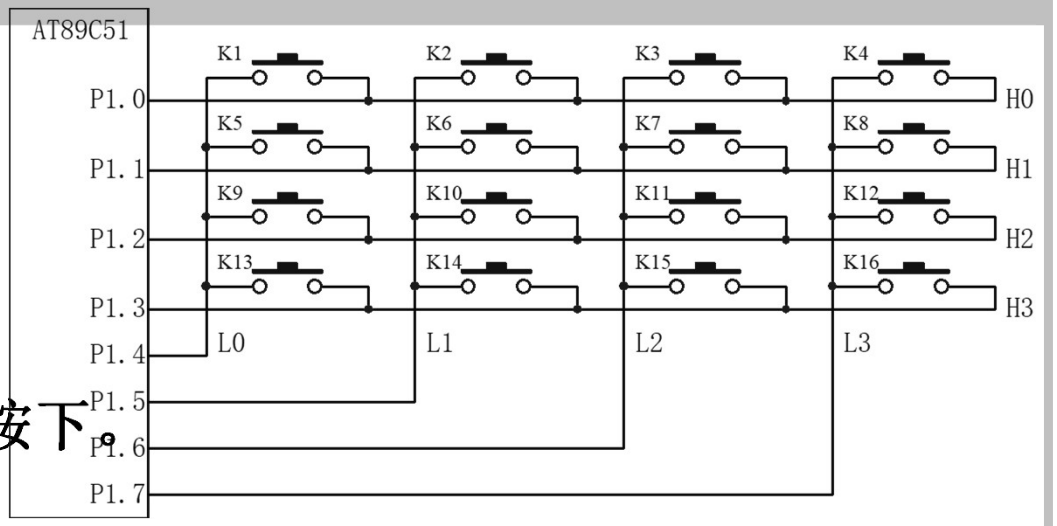
对键盘的工作过程可分两步：

第一步是**CPU**首先检测键盘上是否有键按下；

第二步是再识别是哪一个键按下。

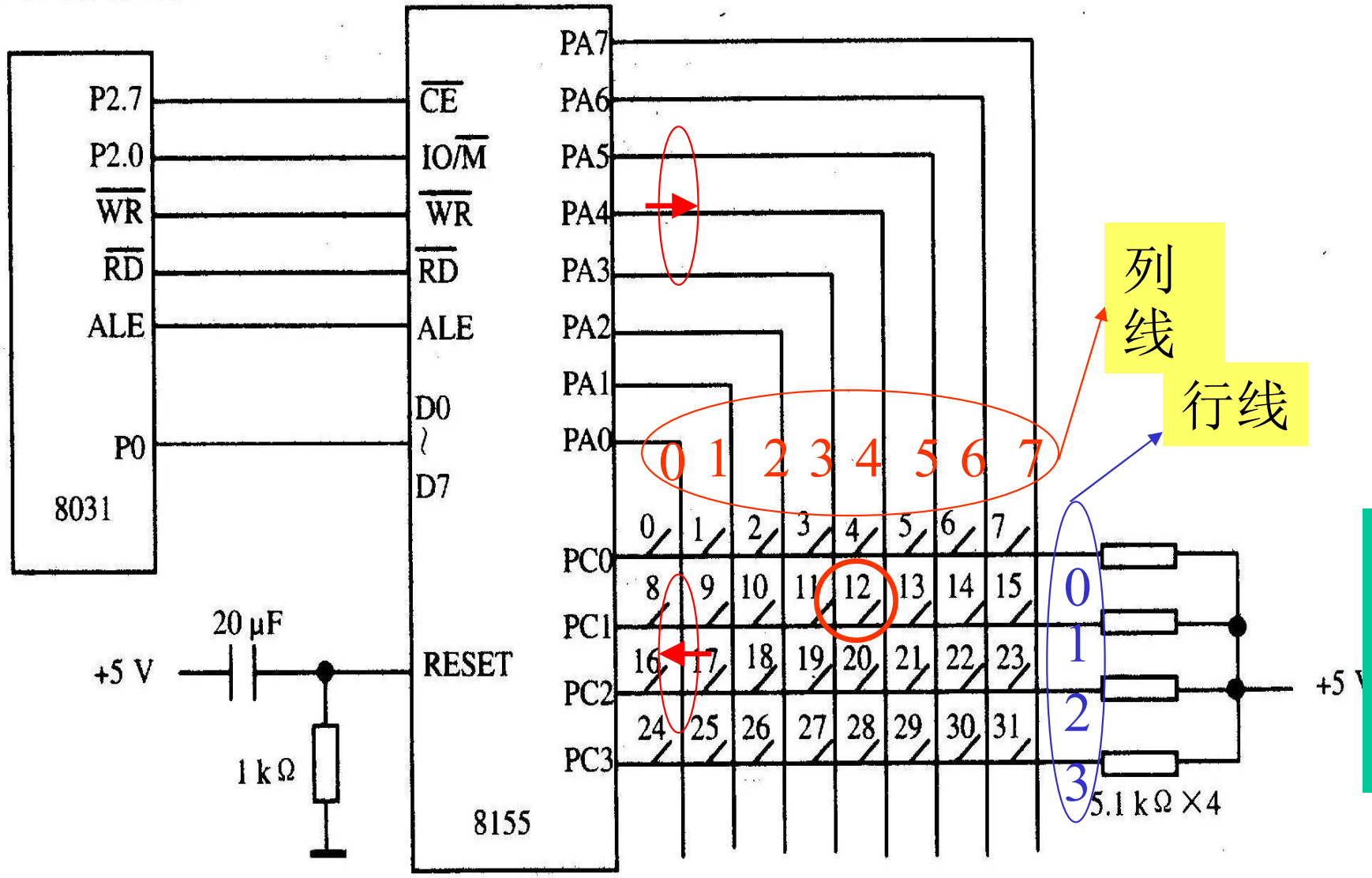
检测键盘上有无键按下可采用程序扫描方式和中断工作方式。

1)程序扫描方式



- ✓ 首先列线全输出0，判断是否有键按下。
  - 如果行线为全1，无按键按下
  - 如果行线非全1，有按键按下
- ✓ 然后，让列线P1.4输出0，其它3条列线输出1，读行线状态。
  - 如果行线为全1，第一列无按键按下，继续扫描。
  - 如果行线非全1，可以判断按键在第0列，再根据为0的行线序号，可以确定按键具体的行号，停止扫描。
- ✓ 如果第0列无按键按下，让列线P1.5口输出0，其它三条列线输出1，读行线状态，判断按键是否在第一列。

- ✓ 为求取键码，在逐列扫描时，可用计数器记录下当前扫描列的列号，然后用行首键码加列号的办法计算。



行首键码

0行: 00000
1行: 01000
2行: 10000
3行: 11000

如12号键按下, 列扫描号为0100,

PC口读入时PC1=0, 可确定行首键码为**01000**,

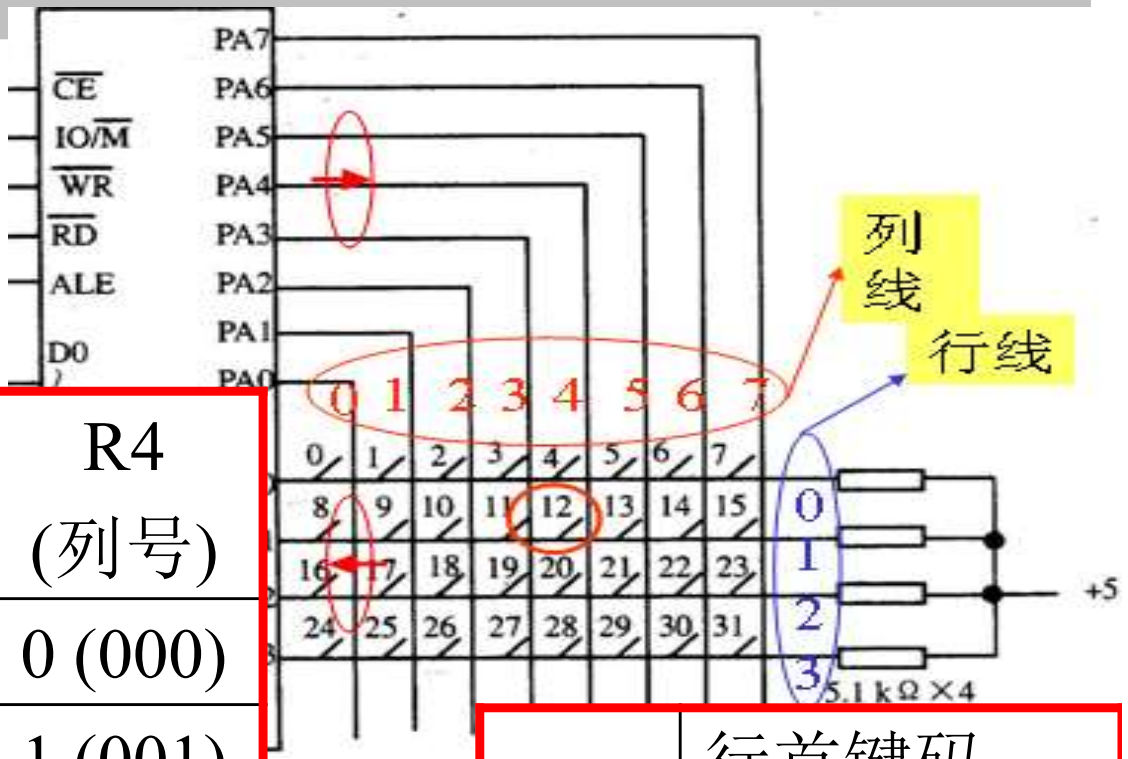
故键值为:**01000+0100=01100**

键盘显示扫描程序编写

课本P169例:

PA口输出（列）

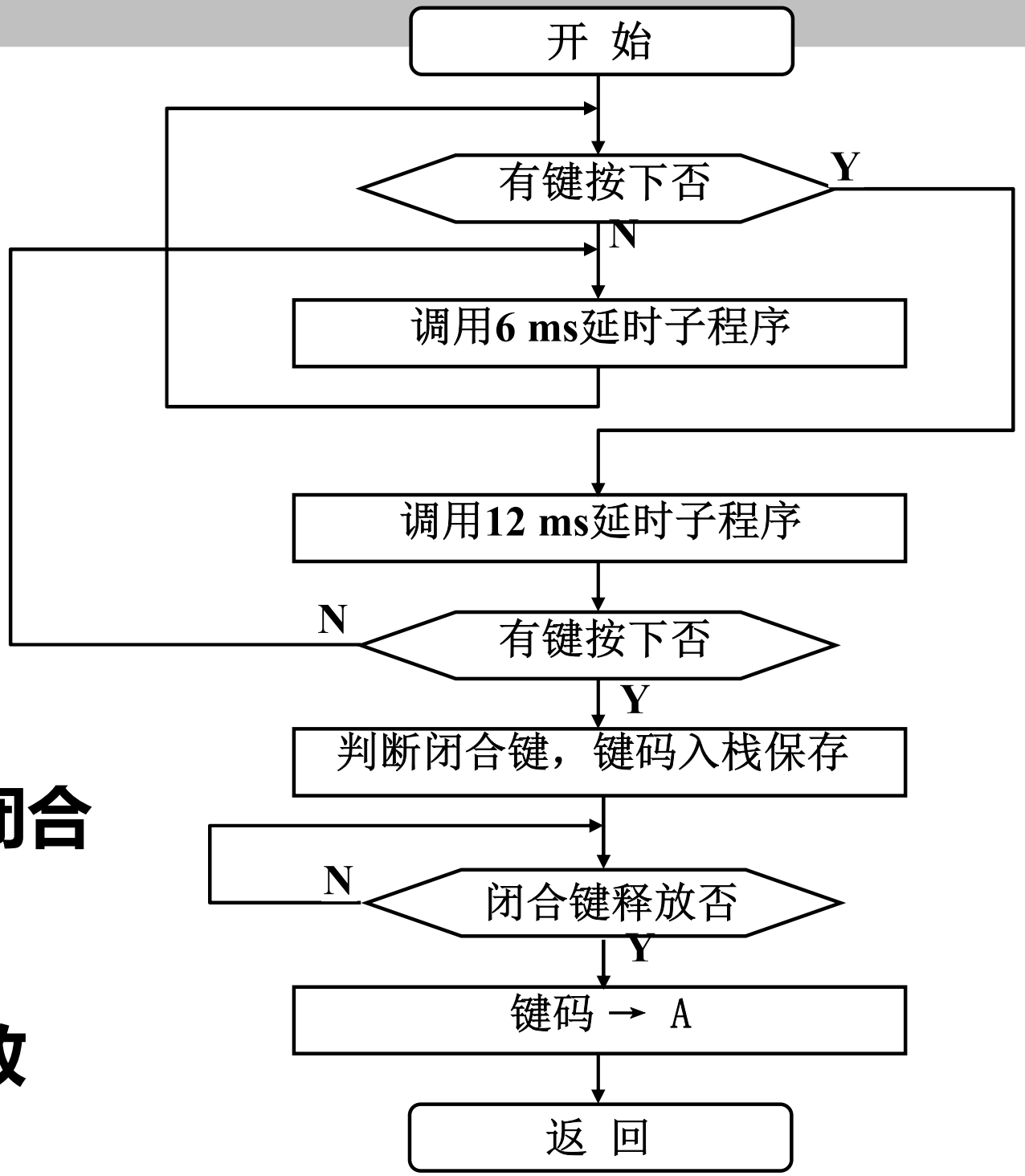
PC口输入（行）



	R2(PA7~PA0) 列扫描字	R4 (列号)
扫描第0列	FE （1111 1110）	0 (000)
扫描第1列	FD （1111 1101）	1 (001)
扫描第2列	FB （1111 1011）	2 (010)
扫描第3列	F7 （1111 0111）	3 (011)
扫描第4列	EF （1110 1111）	4 (100)
扫描第5列	DF （11011111）	5 (101)
扫描第6列	BF （1011 1111）	6 (110)
扫描第7列	7F （0111 1111）	7 (111)

	行首键码
0行	00 000 (00H)
1行	01 000 (08H)
2行	10 000 (10H)
3行	11 000 (18H)

键盘扫描子程序流程框图



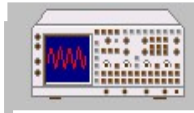
编写键盘程序四步

(1)判断是否有键闭合

(2)去抖动

(3)求键值

(4)等待按键的释放



程序清单:

1 子程序1——判断是否有键按下

2 子程序2——键盘扫描程序,确认按下的键

课本P169例:

PA口输出（列）

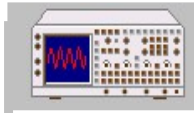
PC口输入（行）

- ✓ 首先列线全输出0，判断是否有键按下。
  - 如果行线为全1，无按键按下
  - 如果行线非全1，有按键按下

子程序1：判断是否有键按下

占用资源：A

```
KS:  MOV   DPTR，  #PA          ；A口地址送入DPTR
     MOV   A，  #00H             ；全扫描字送A
     MOVX  @DPTR，  A             ；全扫描字送PA口，列全部输出0
     INC     DPTR                ；指向C口
     INC     DPTR
     MOVX  A，  @DPTR            ；读入PC口（行）状态
     CPL     A                   ；变正逻辑：高电平表示有键按下
     ANL     A，  #0FH           ；屏蔽高4位
     RET
```



子程序1：判断是否有键按下

占用资源：A

```
KEY:    LCALL  KS          ；判别有无键按下
        JNZ    K1          ；(A)不等于0，有键按下，转K1
        LCALL  DELAY       ；无键按下调用延时子程序
        AJMP   KEY         ；返回重新查询
K1:     LCALL  DELAY       ；加长延时，消除键抖动
        LCALL  DELAY
        LCALL  KS          ；再次查询有无键按下
        JNZ    K2          ；(A)不等于0，有键按下，转逐列扫描
        AJMP   KEY         ；误读键，返回
```

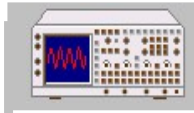
R2＝ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

✓ 然后，让列线PA0输出0，其它七条列线输出1，读行线状态。

- 如果行线为全1，第一列无按键按下，继续扫描。
- 如果行线非全1，可以判断按键在第一列，再根据为0的行线序号，可以确定按键具体的行号，停止扫描。

子程序2：键盘扫描程序,确认按下的键　　占用资源：A, R2, R4

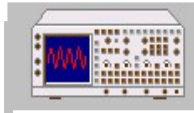
```
                                ；从第0列开始扫描
K2:     MOV   R2，#0FEH    ；首列扫描字送R2
        MOV   R4，#00H     ；首列号送R4
K3:     MOV   DPTR，#PA    ；A口地址送DPTR
        MOV   A，R2
        MOVX  @DPTR，A     ；列扫描字送8155A口
        INC   DPTR
        INC   DPTR
        MOVX  A，@DPTR          ；读取行扫描值
        JB    ACC.0，L1；（PC.0）=(ACC.0)=1，第0行无键按下，转查第一行
        MOV   A，#00H           ；第0行有键按下，行首键号送A
        AJMP  LK                ；转求键号
```



```
L1:   JB    ACC.1，L2  ；（PC.1）=(ACC.1)=1，第1行无键按下，转查第2行
      MOV   A，#08H                ；第1行有键按下，行首键号送A
      AJMP  LK                     ；转求键号
L2:   JB    ACC.2，L3  ；（PC.2）=(ACC.2)=1，第2行无键按下，转查第3行
      MOV   A，#10H                ；第2行有键按下，行首键号送A
      AJMP  LK                                ；转求键号
L3:   JB    ACC.3，NEXT；（PC.3）=(ACC.3)=1，第3行无键按下，转查下一列
      MOV   A，#18H                ；第3行有键按下，行首键号送A
```

```
                                  ;开始求键码
LK:   ADD   A，R4                  ；形成键码送A:键码=行首键码+列号
      PUSH  ACC                    ；键码入栈保存
K4:   LCALL DELAY
      LCALL KS            ；等待键释放
      JNZ   K4                     ；(A)不等于0,有键按下，按键未释放，等待

      POP   A                      ；键释放，弹出键码
      RET
```



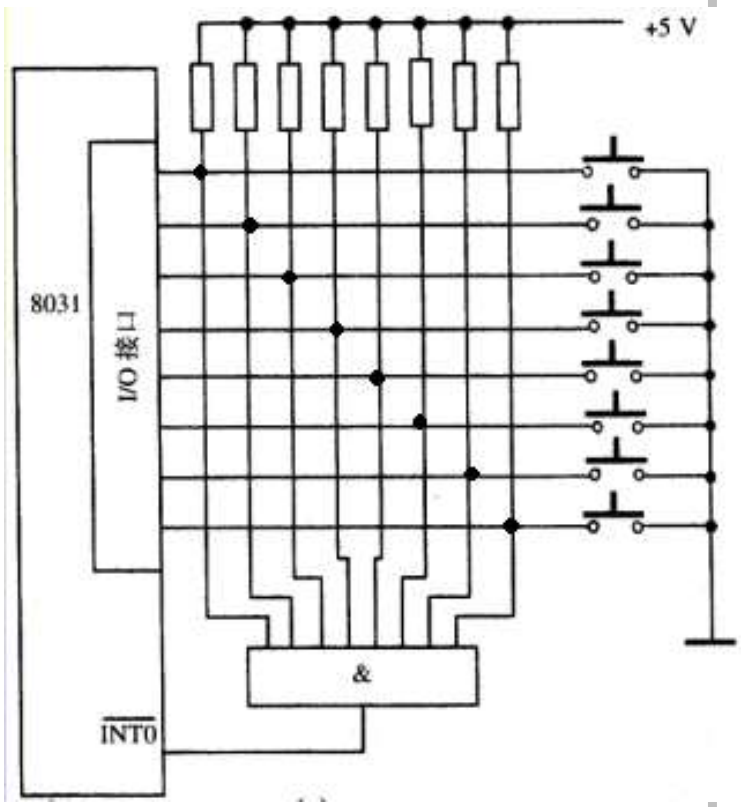
✓ 如果第一列无按键按下，让列线PA1口输出0，其它七条列线输出1，读行线状态，判断按键是否在第二列。

```
                                    ;准备扫描下一列
NEXT：INC   R4                      ；修改列号，扫描下一列
      MOV   A，  R2                 ；PA口（列）数据送A
      JNB   ACC.7，  KEY            ；（PA.7）=(ACC.7)=0，8列扫描完，返回KEY
       RL   A                       ；未扫描完，扫描字左移一位
      MOV   R2，  A                 ；扫描字存R2
       AJMP  K3
```

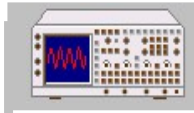
```
                                    ；延时子程序
DELAY：MOV    R7，  #0FFH
LP0：    MOV    R6，  #0FFH
LP：     NOP
         DJNZ   R6，  LP
         DJNZ   R7，  LP0
         RET
```

2) 中断工作方式

计算机应用系统工作时，并不经常需要键输入。但无论是查询工作方式还是定时扫描工作方式，**CPU**经常处于空扫描状态。为了提高**CPU**的效率，可采用中断工作方式。这种工作方式是当键盘上有键按下时，向**CPU**发一个中断请求信号，**CPU**响应中断后，在中断服务程序中扫描键盘，执行键功能程序。



中断服务程序中应完成键识别、消除抖动、排除多次执行键功能操作等功能，可参考查询工作方式键盘程序。



PBL练习

- 1、实现矩阵式键盘输入、单LED数码管显示
- 2、实现矩阵式键盘输入、多LED数码管动态输出显示