微机原理与接口技术

§8 人机接口技术与综合实例设计

主讲人：佘青山

Homepage: https://auto.hdu.edu.cn/2019/0403/c3803a93084/page.htm
Email: qsshe@hdu.edu.cn
Mob：13758167196
Office: 第二教研楼南楼308室

2024年11月19日

一、键盘基础知识

1. 键盘的组织

键盘可分为**编码式键盘**或**非编码式键盘**。

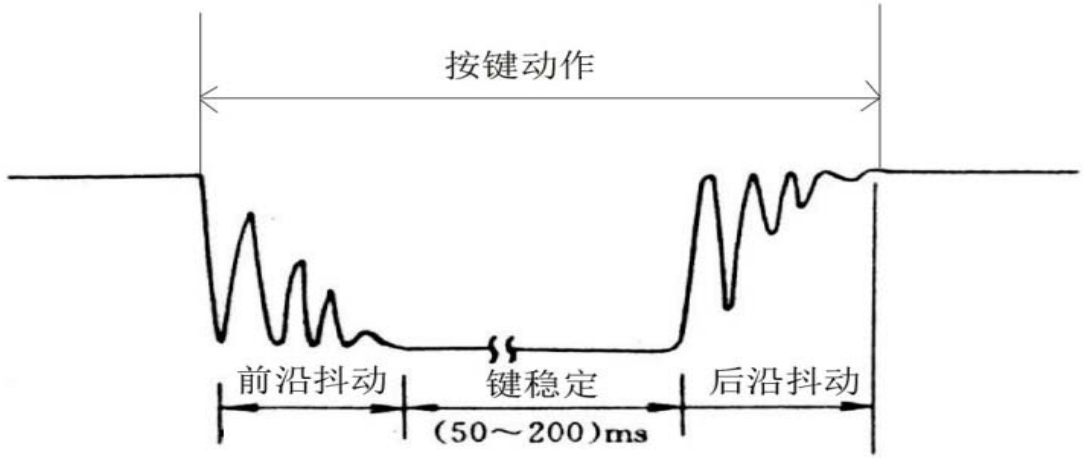
- **编码式键盘**：由键盘和专用键盘编码器（键盘管理芯片）两部分构成。键盘管理芯片自动完成键盘的扫描和译码。编码式键盘使用很方便，成本相对较高。常用的大规模集成电路键盘管理芯片如HD7279等。
- **非编码式键盘**：只简单地提供按键的通断信号，但某键按下时，键盘送出一个闭合（低电平）信号。该按键键值的确定必须借助于软件来实现。所以非编码式键盘的软件比较复杂，占用CPU时间多。但成本低、使用灵活，在微机系统中，得到广泛应用。

非编码式键盘可分为独立式键盘和矩阵式键盘。

一、键盘基础知识

2. 按键抖动与消除

触点式按键在闭合和断开瞬间存在抖动过程，即存在抖动现象，前后沿抖动时间一般在5ms～10ms。按键的稳定时间与按键动作有关，通常大于50ms。



按键抖动可能导致微机对一次按键操作做出多次响应，所以要**去抖动**。

（1）**硬件电路去抖动**：需要利用RS触发器等构成去抖动电路（很少使用）。

（2）**软件延时法**：当检测到有键按下时，用软件延时10ms～20ms，等待键稳定后重新再判一次，以躲过触点的抖动期。

一、键盘基础知识

3. 键盘的工作方式

微机系统中CPU对键盘进行扫描时，要兼顾两方面的问题：

- 要及时响应，保证系统对按键的每一次操作都能作出响应；
- 不能占用CPU过多的时间。

键盘的三种工作方式：

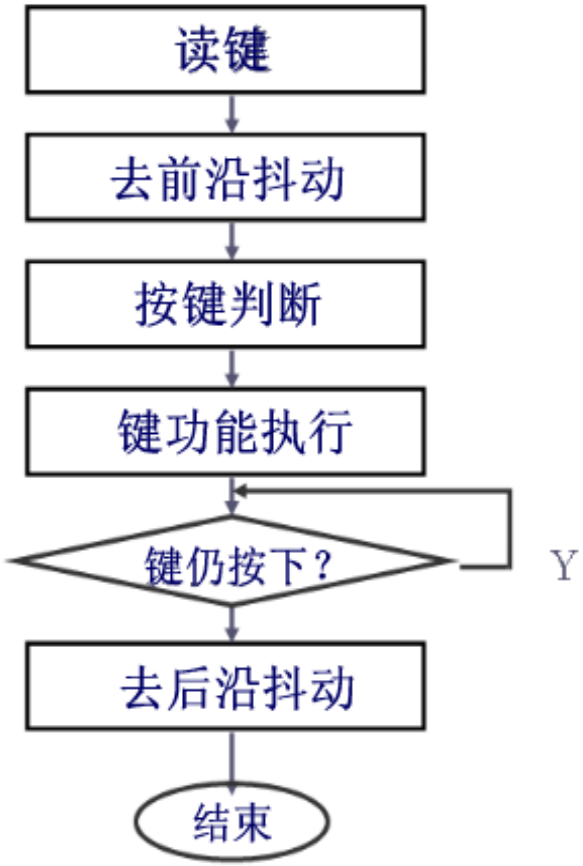
- **编程扫描方式（查询方式）**：利用CPU在完成其他工作的空闲，调用键盘扫描程序。当CPU在运行其他程序时，不会响应按键操作。
- **定时扫描方式**：用定时器产生定时中断，在定时中断中对键盘进行扫描。定时中断周期一般应≤50ms。该方式常会出现CPU常空扫描状态。
- **中断工作方式**：当有键按下时，利用硬件产生外部中断请求，CPU响应中断后对键盘进行扫描。**该方式优于定时扫描方式，既能及时响应按键操作，又节省CPU时间。**

一、键盘基础知识

4. 按键连击的消除与利用

连击：一次按键操作被多次执行的现象称为“连击”，有利有弊。

- **按键连击的消除：在程序中加入等待按键释放的处理，保证一次操作只被响应一次。**

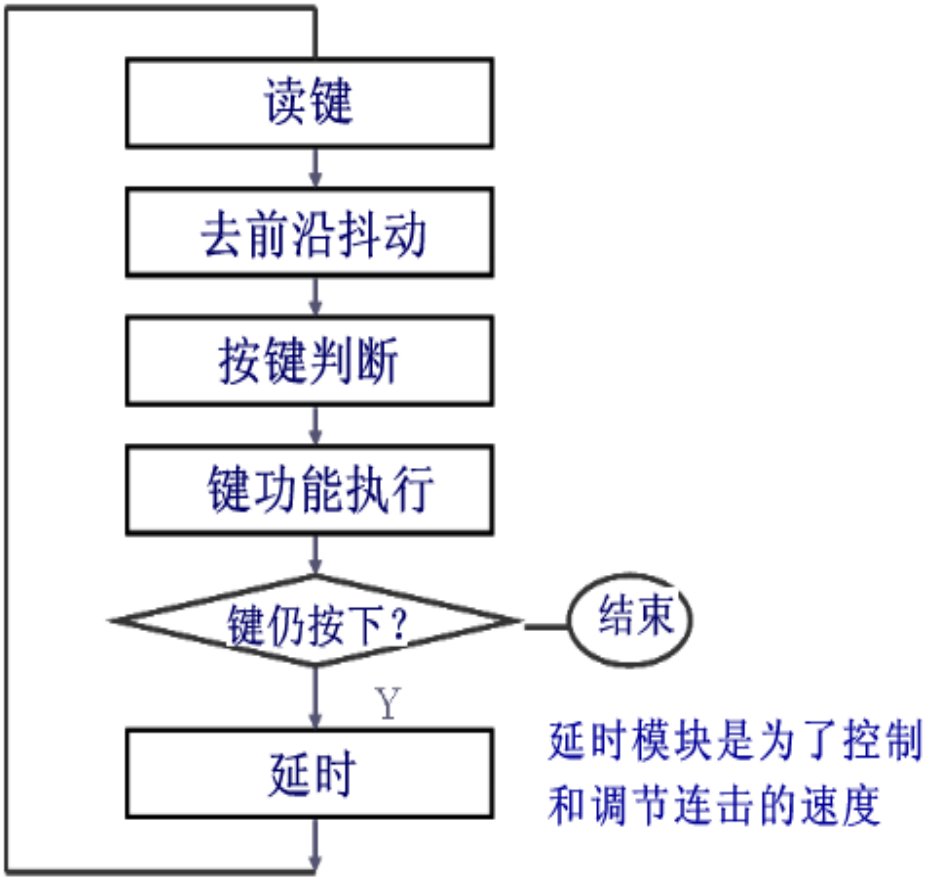


键连击现象的消除

一、键盘基础知识

4. 按键连击的消除与利用

- **按键连击的利用：如设置“增1”、“减1”两个按键，利用按键的连击，长按住“增1”、“减1”键，则参数会不断增加或减少。可以替代0~9的数字键，有效减少按键数量。**



键连击现象的利用

一、键盘基础知识

5. 重键保护与实现

重键：由于操作不慎，可能会造成同时有几个键被按下，这种现象称为重键。出现重键时，就产生了如何识别和作出响应的问题。

处理重键的技术：

(1) “N键锁定”

当扫描到有多个键被按下时，只把最后一个释放的按键作为有效键进行响应。

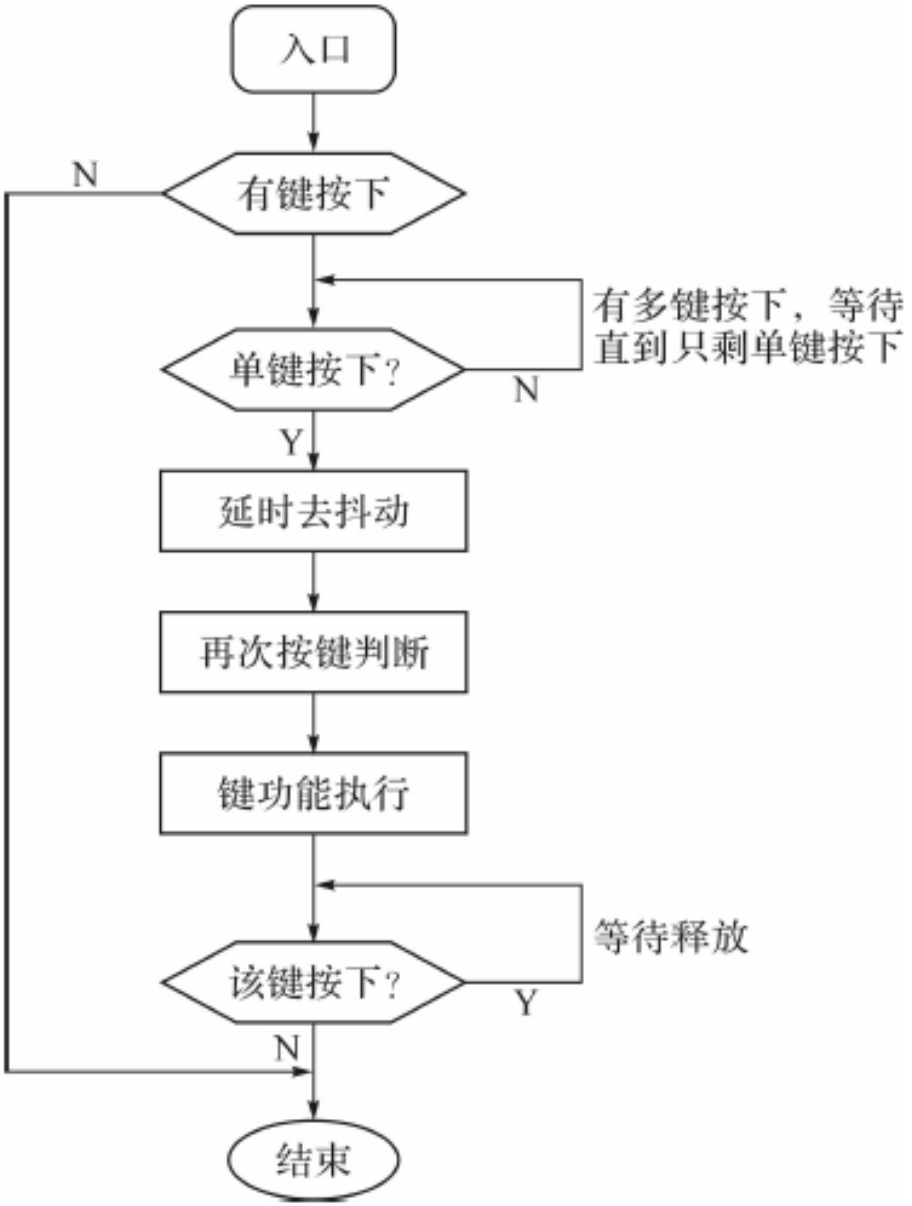
(2) “N键轮回”

当扫描到有多个键被按下时，对所有按下的按键依次产生键值并作出响应。

一、键盘基础知识

5. 重键保护与实现

在微机系统中，通常采取单键按下有效、多键按下无效的策略，即采用N键锁定方法。

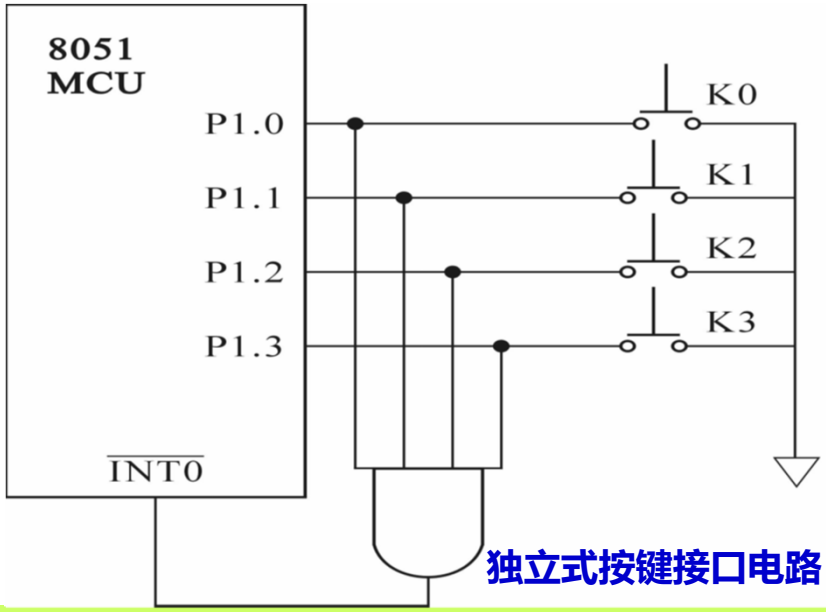


二、独立式键盘接口

独立式键盘：每个按键占用一根I/O口线。

无按键按下时，各I/O口线输入状态为高电平；当有按键按下时，I/O口线变为低电平。只要CPU检测到某一I/O口线为“0”，便可判别对应按键被按下。

- **优点：**结构简单，按键识别容易。
- **缺点：**当按键较多时，占用I/O口线多，只适用于按键较少的系统。

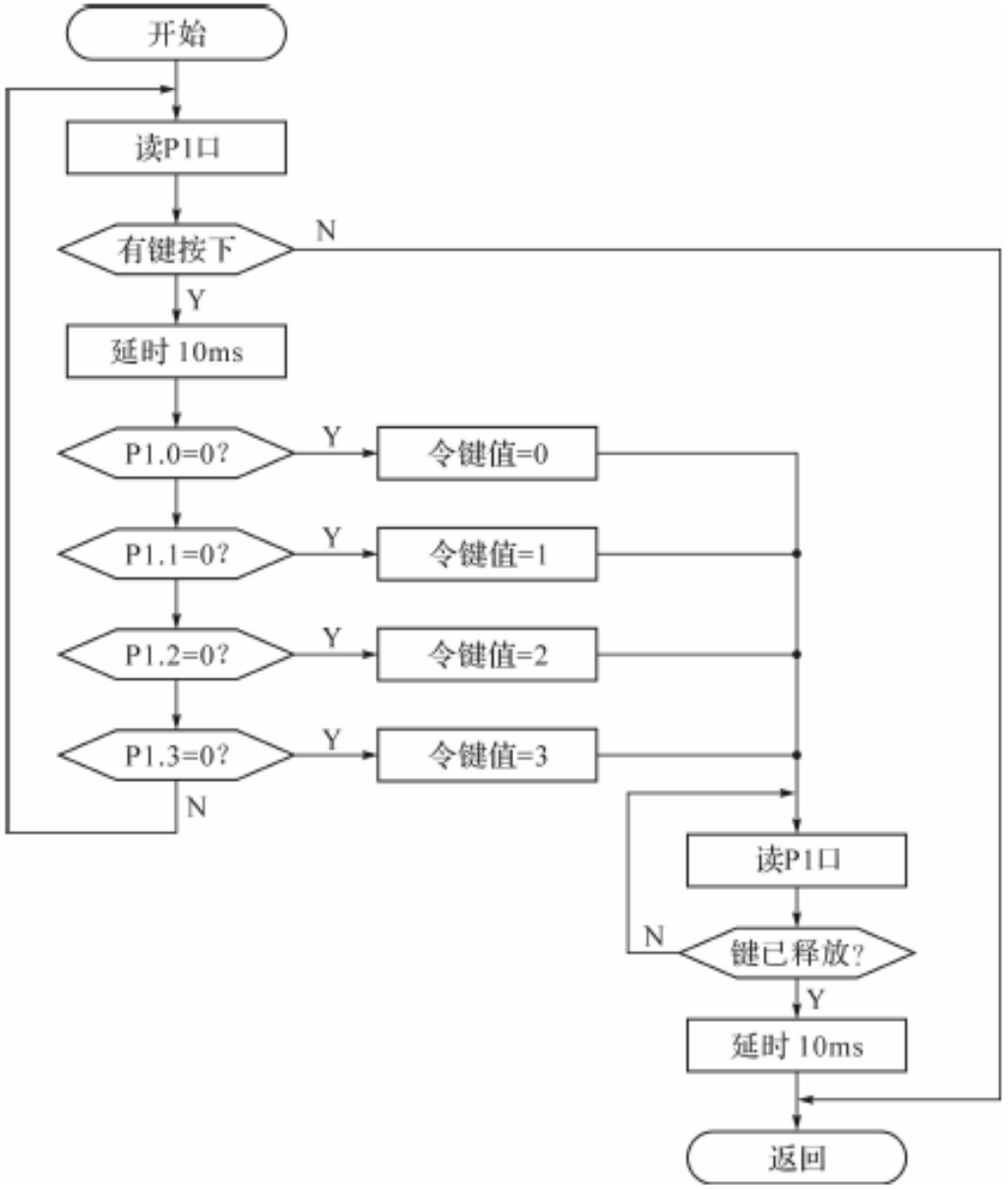


独立式按键接口电路

二、独立式键盘接口

程序流程（查询式）：

- 首先判断有无键按下，若检测到有键按下，延时10ms去抖动；
- 再逐位查询是哪个按键按下，并执行相应按键的处理程序；
- 最后等待按键释放，并延时10ms去除后沿抖动。



二、独立式键盘接口

中断扫描思路：

无键按下时，4与门输入全为高电平，不会产生中断。当任一键按下时，/INT0变为低电平，向MCU请求中断。MCU响应中断，扫描按键，得到键值。

汇编主程序：

```
        ORG     0000H
        SJMP    MAIN
        ORG     0003H
        LJMP    INT0SUB          ;外部中断0中断程序
        ORG     0100H
MAIN:   SETB    IT0              ;设置INT0为下降沿触发方式
        SETB    EX0              ;允许INT0中断
        SETB    EA               ;允许CPU中断
        CLR     KEYFLAG          ;清“有键按下”标志
LOOP:   JNB     KEYFLAG, LOOP    ;等待中断
        CLR     KEYFLAG
        LCALL   KEYPROCESS       ;执行按键处理程序(省略)
        SJMP    LOOP
```

键盘接口技术

汇编中断程序：

扫描按键，键值存放在R3中。

- K0的键值=0；
- K1的键值=1；
- K2的键值=2；
- K3的键值=3；

				K3	K2	K1	K0
0	0	0	0	P1.3	P1.2	P1.1	P1.0

A

C

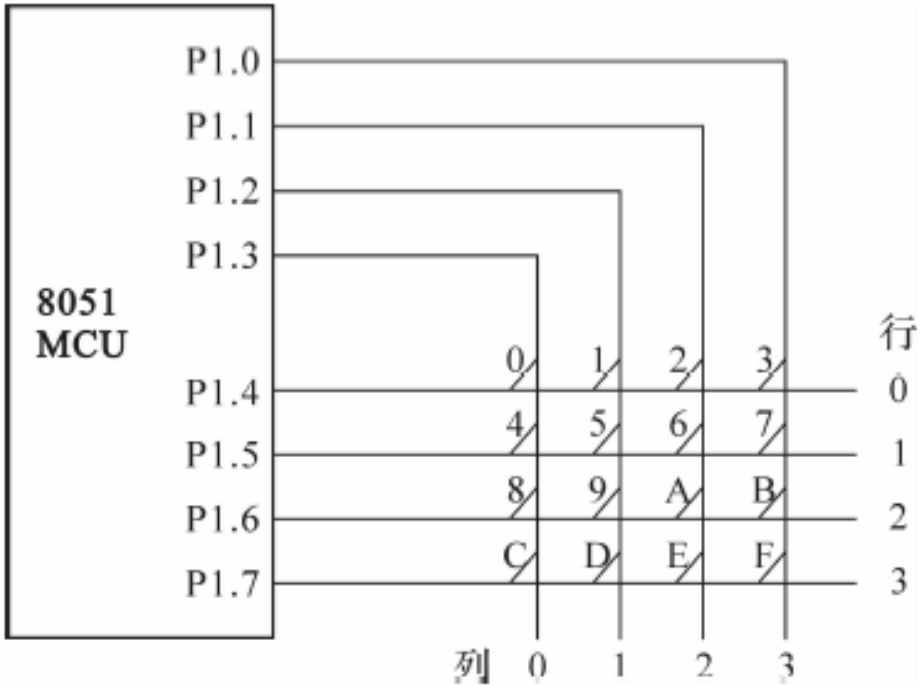
```
        ORG     0200H
INT0SUB:LCALL   DELAY10ms       ;去前沿抖动
        MOV     R3,#00H         ;设置键值初值
        MOV     A,P1
        ANL     A,#0FH
        CJNE    A,#0FH,SCAN     ;判断是否有按键按下
        MOV     A,#0FFH         ;不是正常的按键操作
        SJMP    NOKEY           ;键值为FFH，返回
SCAN:   MOV     R2,#4           ;设置查询按键数
SCAN1:  RRC     A
        JNC     FNDKEY          ;找到闭合的键
        INC     R3
        DJNZ    R2,SCAN1
        MOV     A,#0FFH         ;没有扫描到有效按键
        SJMP    NOKEY

FNDKEY: MOV     A,R3            ;A?
        SETB    KEYFLAG         ;建立“有键按下”标志
WAIT:   MOV     A,P1
        ANL     A,#0FH
        CJNE    A,#0FH,WAIT     ;等待按键释放
        LCALL   DELAY10ms       ;去后沿抖动
NOKEY:  RETI
```

三、矩阵式键盘接口

矩阵式键盘：需要行线和列线，按键位于行线和列线的交叉点上；m×n矩阵键盘只需要m + n条口线。

按键数目较多的系统中，矩阵式键盘比独立式按键要节省很多I/O口线。

矩阵式键盘判别按键的方法有行扫描法和线反转法。



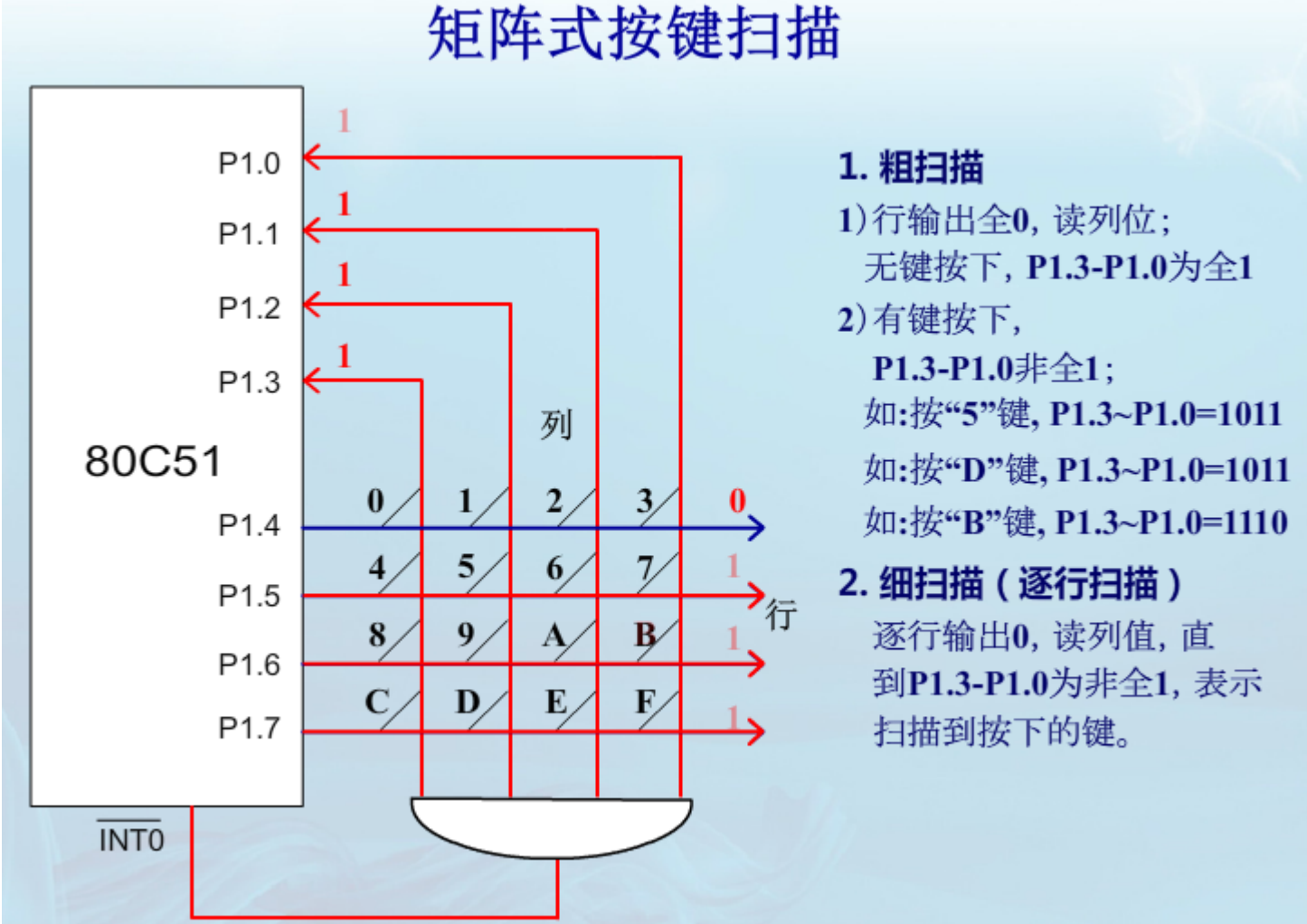
P1.4-P1.7为行扫描输出线；

P1.0-P1.3是列输入线。

若将4个列信号连接到一个4输入的与门，与门输出连接到外部中断引脚，则有键按下时，就会向CPU请求中断。

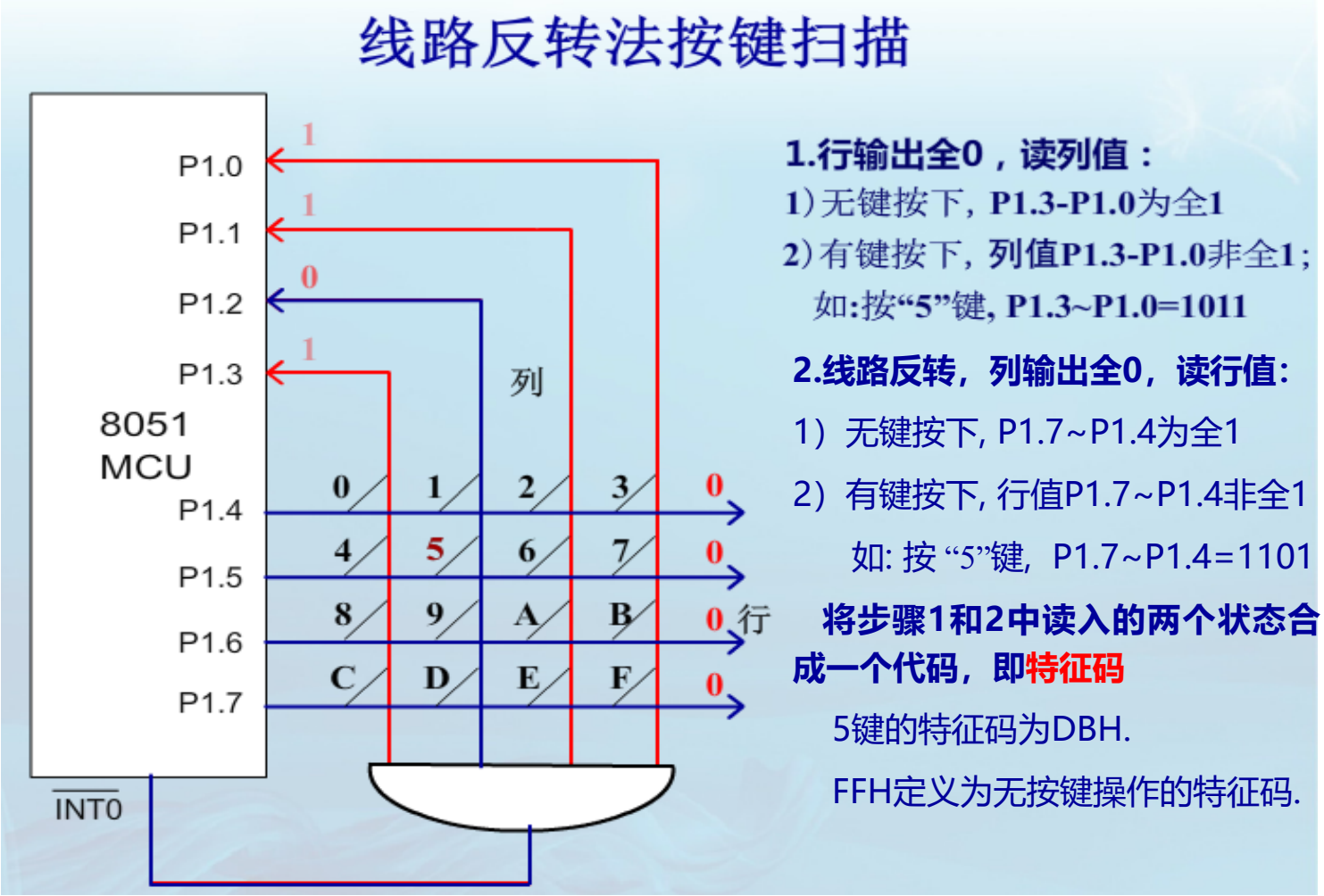
三、矩阵式键盘接口

1. 行扫描法



三、矩阵式键盘接口

2. 线路反转法



四、LED显示原理

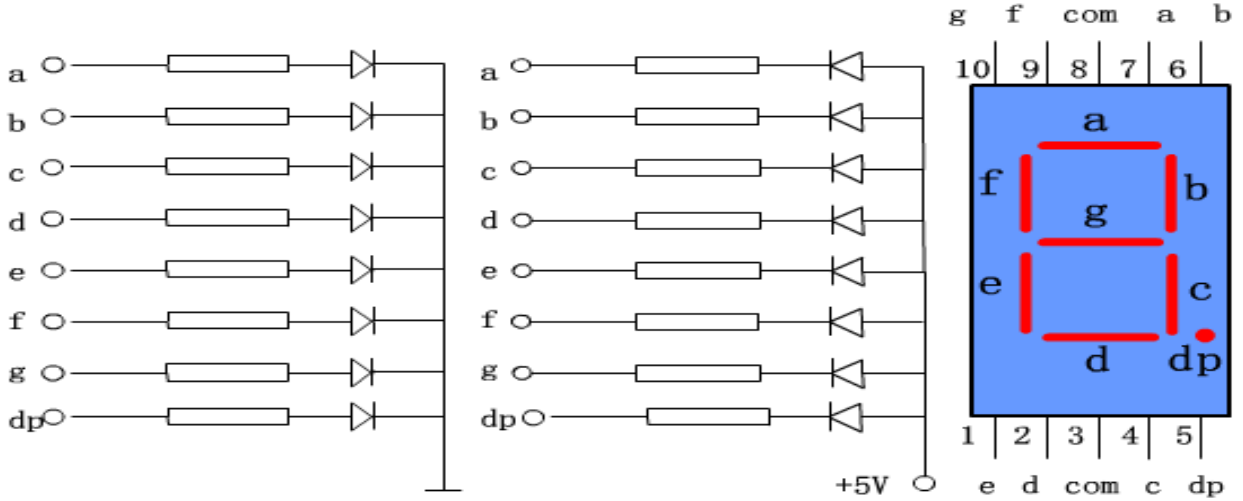
1. LED显示原理

LED（Light Emitting Diode）即发光二极管，是微机系统中最常用的显示器。LED显示器有单个LED、8个LED组成的数码管和点阵式（5×7、8×8）LED显示器等几种类型。

（1）段码式LED显示器（数码管）

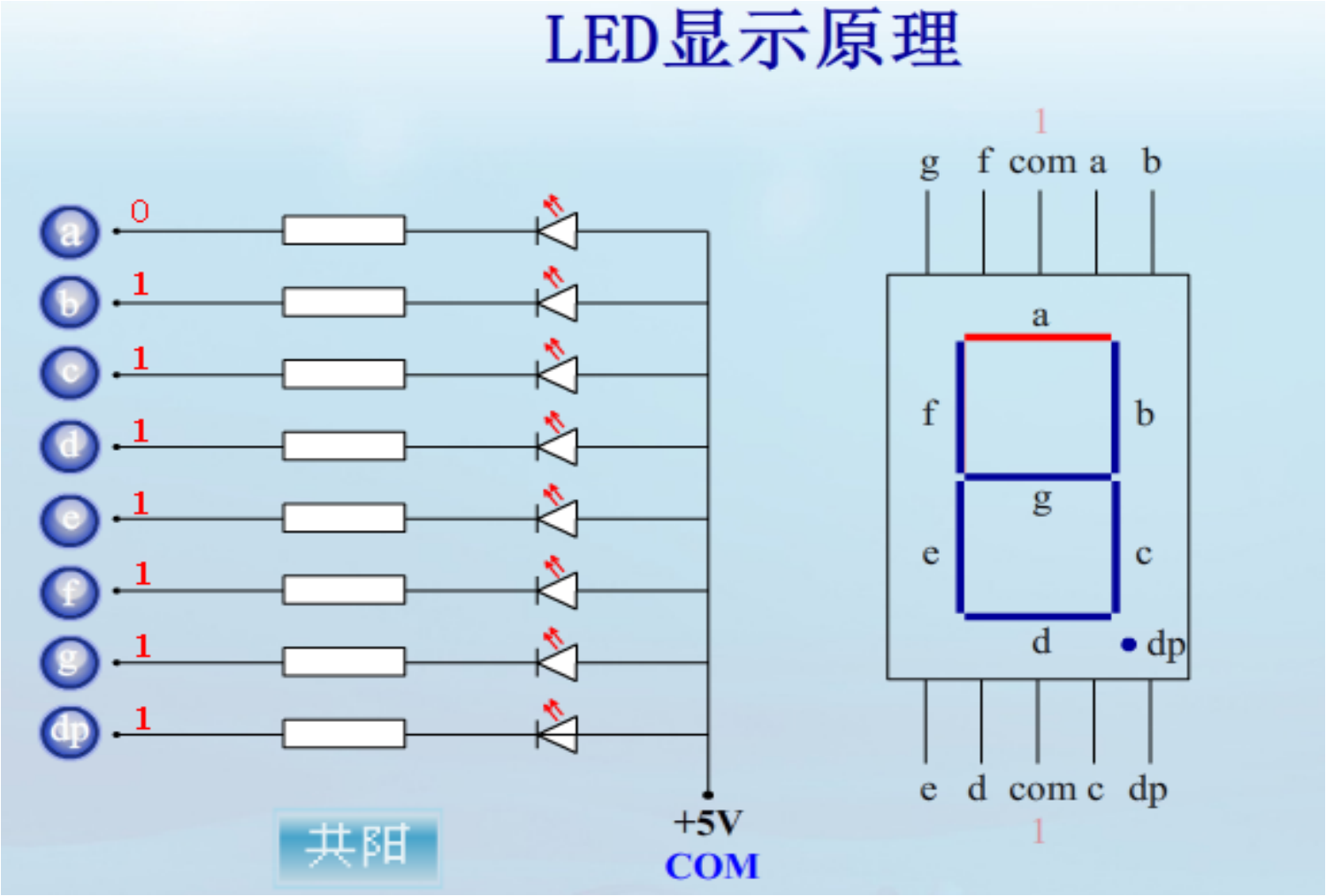
共阴数码管：公共端COM接地或具有较大灌电流的输入口线，阳极接高电平时点亮。

共阳数码管：共阳极接电源或具有强高电平驱动输出口线，阴极接低电平时点亮。



共阴　　共阳

(1) 段码式LED显示器（数码管）



思考：如果点亮a段，共阳极段码是多少呢?

(2) 段码式LED显示器（数码管）的段码表

数码管段码表

字符	共阴极段码	共阳极段码	字符	共阴极段码	共阳极段码
0	3FH	C0H	A	77H	88H
1	06H	F9H	B	7CH	83H
2	5BH	A4H	C	39H	C6H
3	4FH	B0H	D	5EH	A1H
4	66H	99H	E	79H	86H
5	6DH	92H	F	71H	8EH
6	7DH	82H	H	76H	09H
7	07H	F8H	P	73H	8CH
8	7FH	80H	U	3EH	C1H
9	6FH	90H	灭	00H	FFH

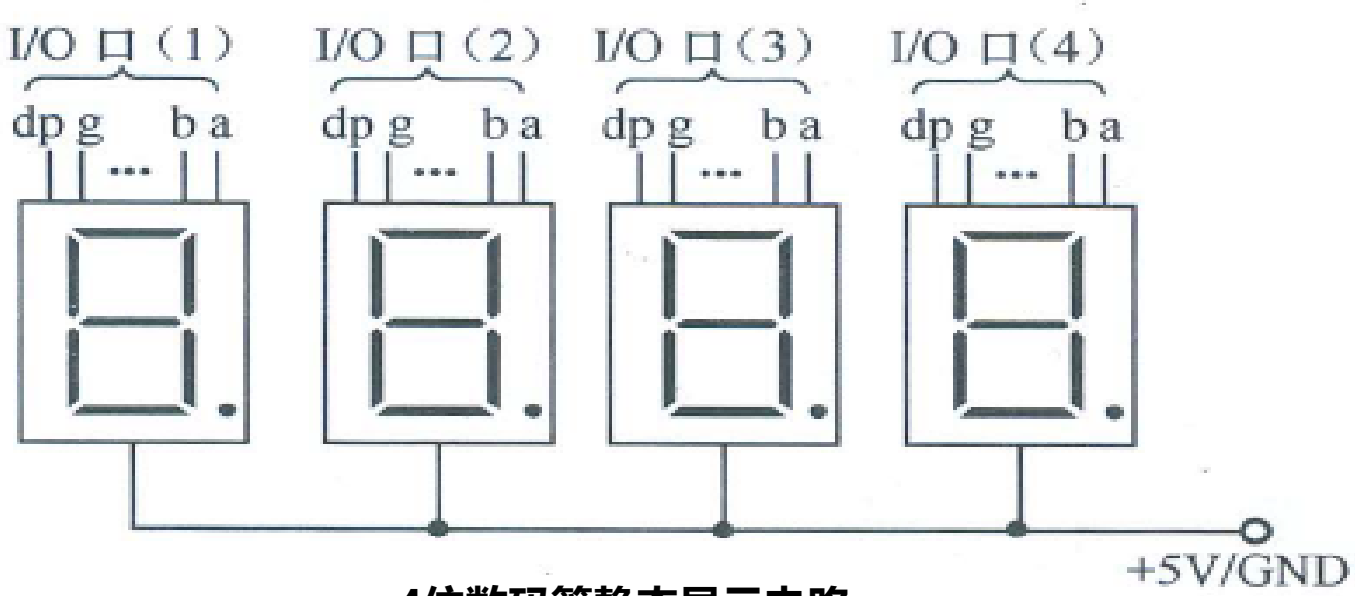
共阴：dp=0　　共阳：dp=1

五、段码式LED显示技术

1. LED静态显示技术

- 对于静态显示方式，一个数码管需要一个8位输出口连接其8个LED的段控制端。
- 特点是*程序简单、显示稳定可靠*，但当显示的位数较多时，需要的输出口较多。

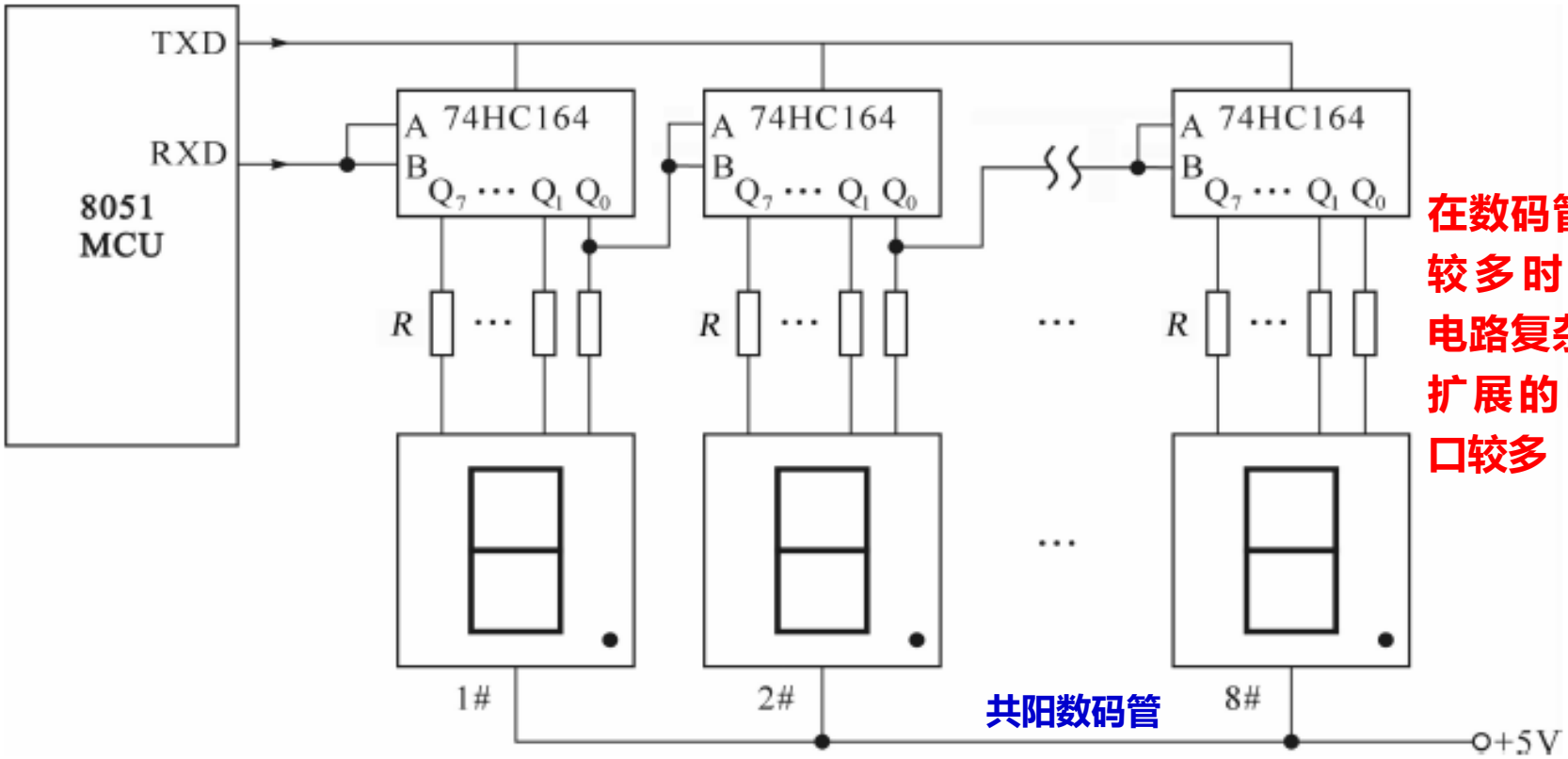
（1）采用并行接口



4位数码管静态显示电路

（2）采用串行扩展

可采用串行口的方式0或用普通I/O口线（如P1.0、P1.1）模拟SPI串行接口，通过外接“串入并出”移位寄存器(如74HC164或74HC595等芯片)来扩展输出接口。



在数码管位数较多时，硬件电路复杂，要扩展的I/O接口较多

串行扩展的数码管静态显示电路

LED接口技术

要显示某字符，首先要得到该字符的7段码，再通过串行口输出，控制数码管的显示。
建立一个0、1、2……E、F等字符的7段码表，用查表法获得要显示字符的7段码。

```
DISPLAY:    SETB    RS0          ;选用第1组工作寄存器
            PUSH    ACC          ;保护现场
            PUSH    DPH
            PSUH    DPL
            MOV     R2, #08H     ;设置显示数据个数
            MOV     R0, #DIS7    ;R0作为显示缓冲区指针，先指向末地址；先输出8#LED的段码
DL0:        MOV     A, @R0               ;取出显示数据，作查表偏移量
            MOV     DPTR, #TAB           ;DPTR指向段码表首地址
            MOV     A, @A+DPTR           ;查表得到7段码
            MOV     SBUF, A              ;串行发送出段码
DL1:        JNB     TI, DL1              ;等待发送完毕
            CLR     TI                   ;清发送完毕标志
            DEC     R0                   ;修改显示缓冲区指针
            DJNZ    R2, DL0              ;继续显示下一个数据
            CLR     RS0                  ;恢复主程序的第0组工作寄存器
            POP     DPL                  ;修复现场
            POP     DPH
            POP     ACC
            RET
TABLE:  DB  0C0H,0F0H, 0A4H, 0B0H, 99H, 92H, 82H, 0F8H, 80H, 90H ;0~9的段码
            88H, 83H, 0C6H, 0A1H, 86H, 8EH, 0BFH, 8CH, 0FFH          ;A~F, -, P, 全灭
```

五、段码式LED显示技术

2. LED动态显示技术

动态显示方式是多个数码管的各段连接在一起由一个输出口（**段码输出口**）控制，而每个数码管的COM端由另一个输出口的1位（**位码输出口**）控制，用2个输出口就可以控制8个数码管的显示。

动态显示方式下，多个数码管实际上是**轮流分时显示**的，即**同一时刻只有1个数码管显示**。具体而言，多位数码管的位控信号（**位码**）在任一时刻只能一位有效，当某位数码管的位控信号有效时，**段码**输出口输出该数码管要显示字符的7段码，而其他数码管均因位控制信号无效而不显示。

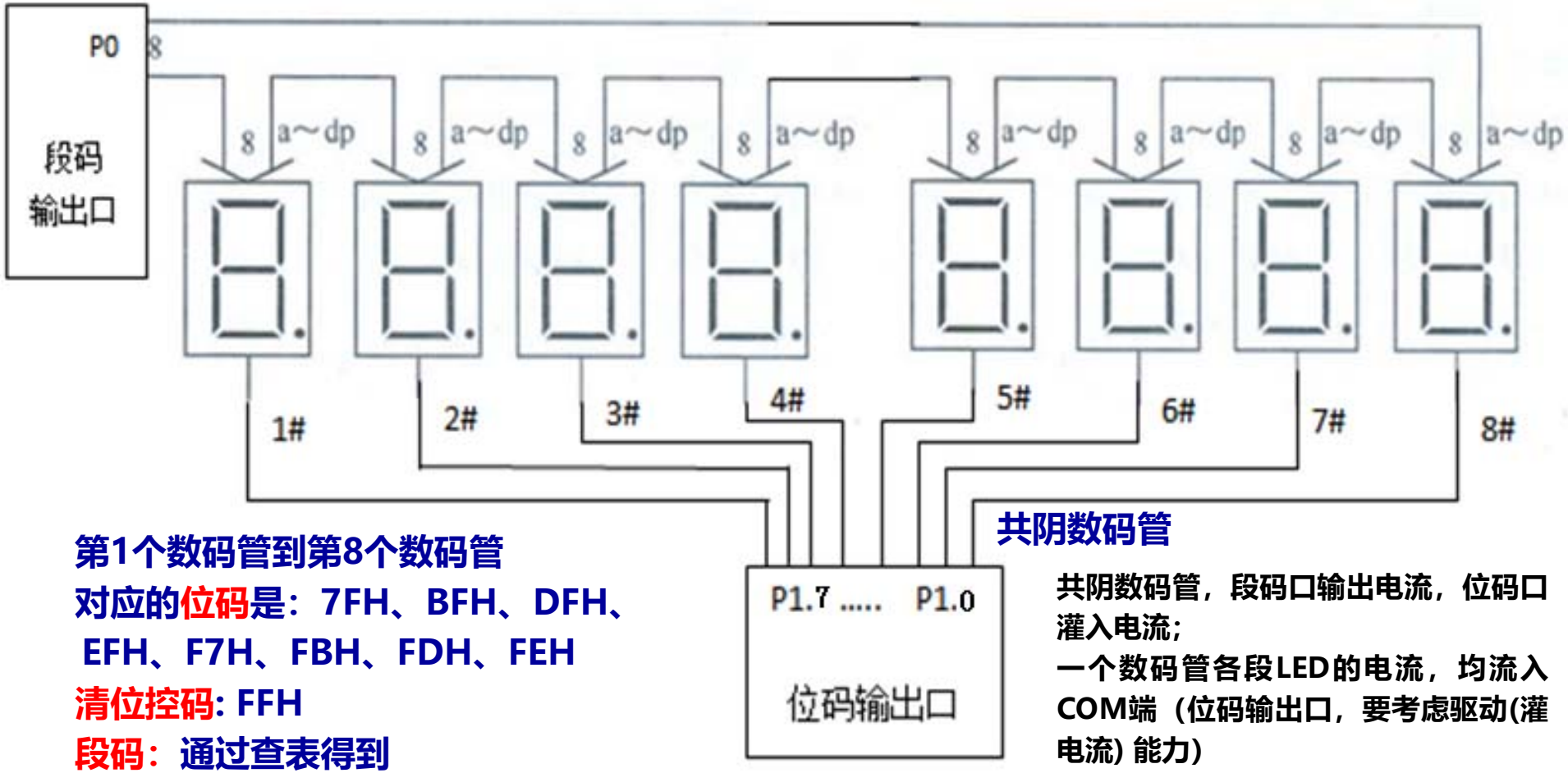
为达到全部数码管“**同时稳定显示**”的效果，需要不断重复输出8位数码管的显示内容（即要进行**显示扫描**），通常显示扫描周期不大于20ms。

优点： 占用输出接口少。

缺点： 需要**定时**对各个数码管进行**显示扫描**，占用CPU时间资源。

2. LED动态显示技术

（1）硬件连接：P0口作为段码输出口，P1口作为位码输出口。



第1个数码管到第8个数码管对应的位码是：7FH、BFH、DFH、EFH、F7H、FBH、FDH、FEH

清位控码: FFH

段码：通过查表得到

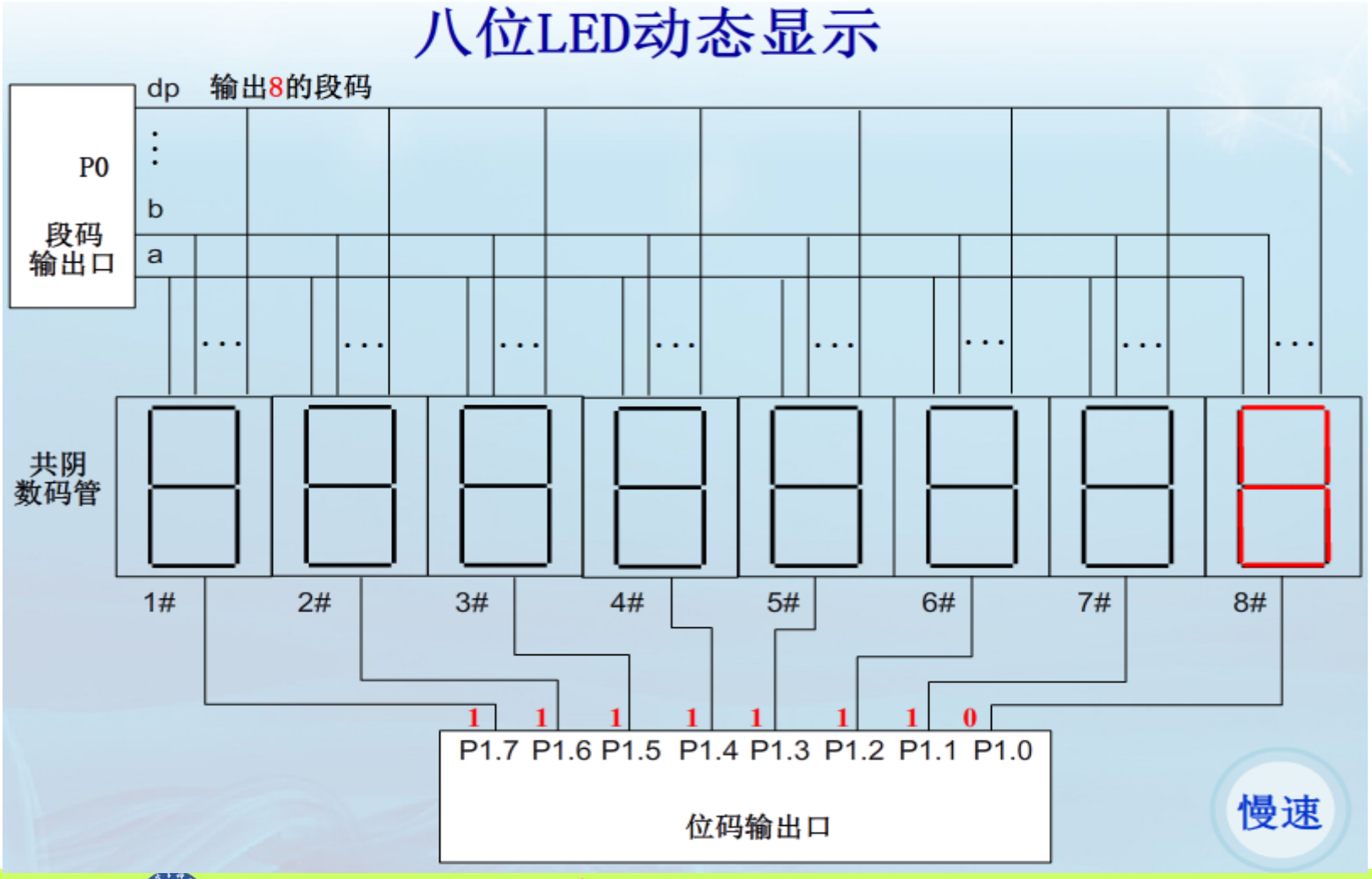
共阴数码管

共阴数码管，段码口输出电流，位码口灌入电流；

一个数码管各段LED的电流，均流入COM端（位码输出口，要考虑驱动(灌电流) 能力）

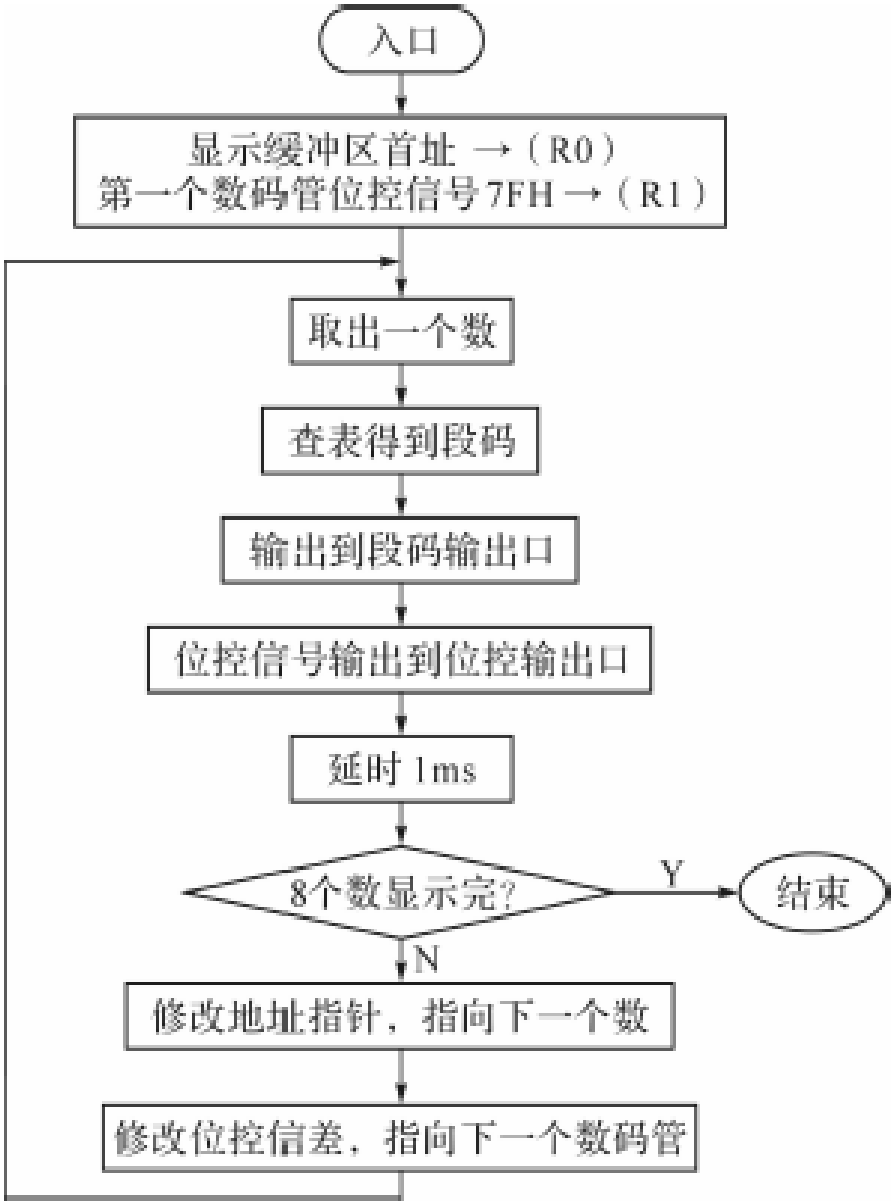
2. LED动态显示技术

（1）硬件连接：P0口作为段码输出口，P1口作为位码输出口。



2. LED动态显示技术

(2) 程序流程



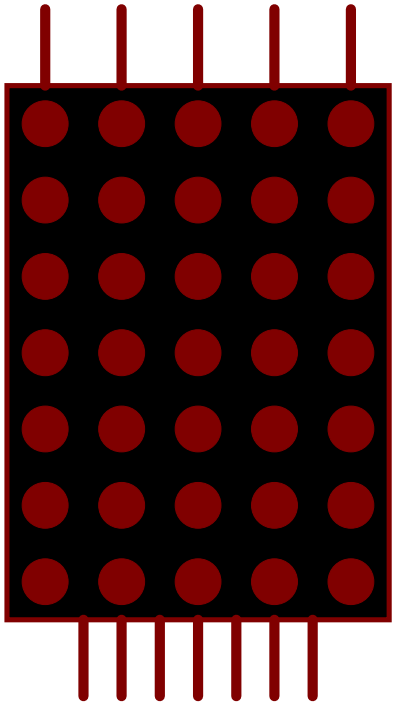
将内部RAM 30H开始的8个显示缓冲单元中的BCD数显示在8位数码管上，其动态扫描显示的汇编程序:

```
        ORG    0000H
        SJMP   MAIN
        ORG    0040H
MAIN:   MOV    R0, #30H          ; R0指向显示数据存放首址
        MOV    R1, #7FH          ; R1为位控信号寄存器，指向1#数码管
        MOV    R2, #08H
NEXT:   MOV    A, @R0            ; 取出一个数
        MOV    DPTR, #TABLE      ; DPTR指向段码表首地址
        MOVC A, @A+DPTR          ; 取出该数的段码
        MOV    P0, A             ; 将段码输出到段码输出口
        MOV    A, R1
        MOV    P1, A             ; 位控信号输出到位码输出口
        LCALL  DELAY1MS          ; 延时1ms
        INC    R0                ; 指针指向下一个数地址
        MOV    A, R1
        RR     A                 ; 循环右移，起何作用?
        MOV    R1, A             ; 修改位控信号，指向下一个数码管
        DJNZ   R2, NEXT          ; 没有显示完毕，继续
        RET
TABLE: DB  3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH      ; 0-9的段码
```

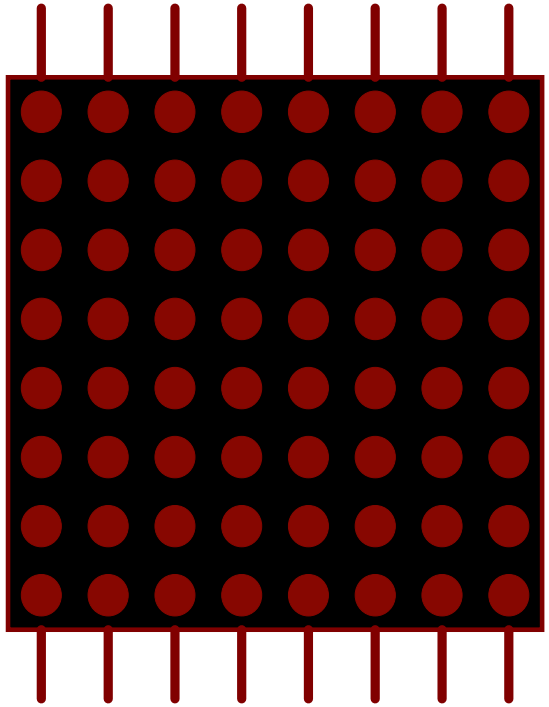
六、点阵式LED显示技术(自学)

1. 点阵式LED

点阵式LED显示器由多个圆形LED组成，有5×7、8×8等多种结构，能够显示字母和较多的字符。

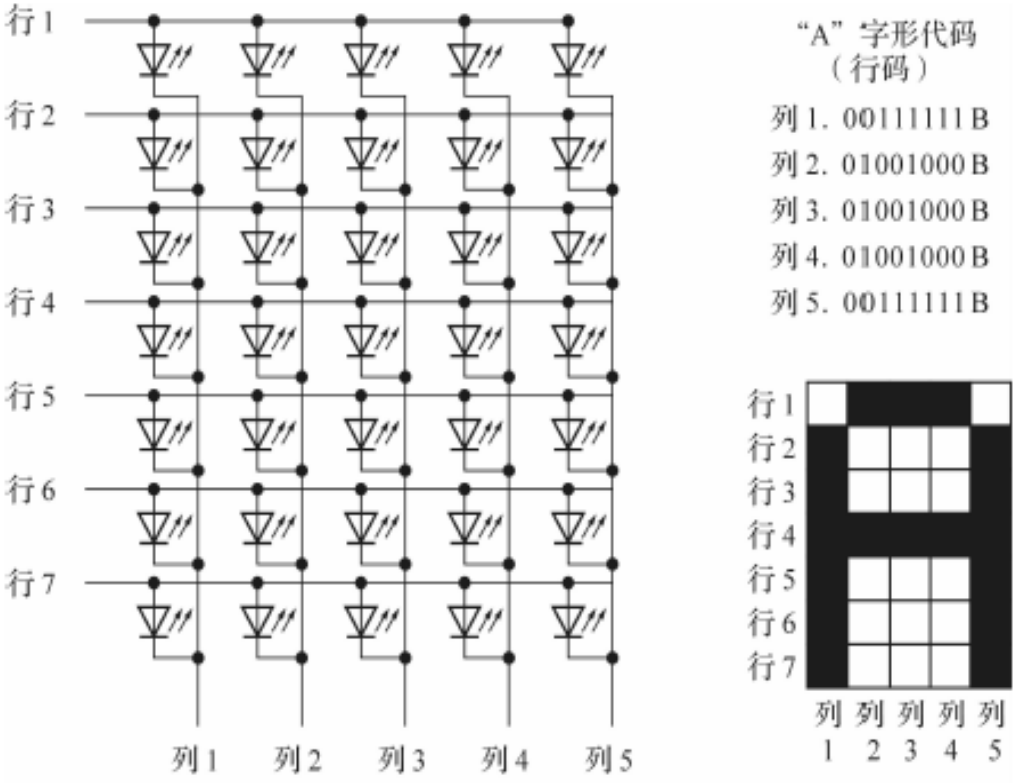


5×7点阵LED



8×8点阵LED

点阵式LED原理



5×7点阵LED原理图

- 每行上的5个LED按共阳方式连接，每列上的7个LED按共阴方式连接，可以把每列看成是一个共阴数码管。
- 列线看作为COM端，行线为段码控制端，控制一个点阵式LED，需要2个输出接口；其显示原理同数码管的动态显示方式。

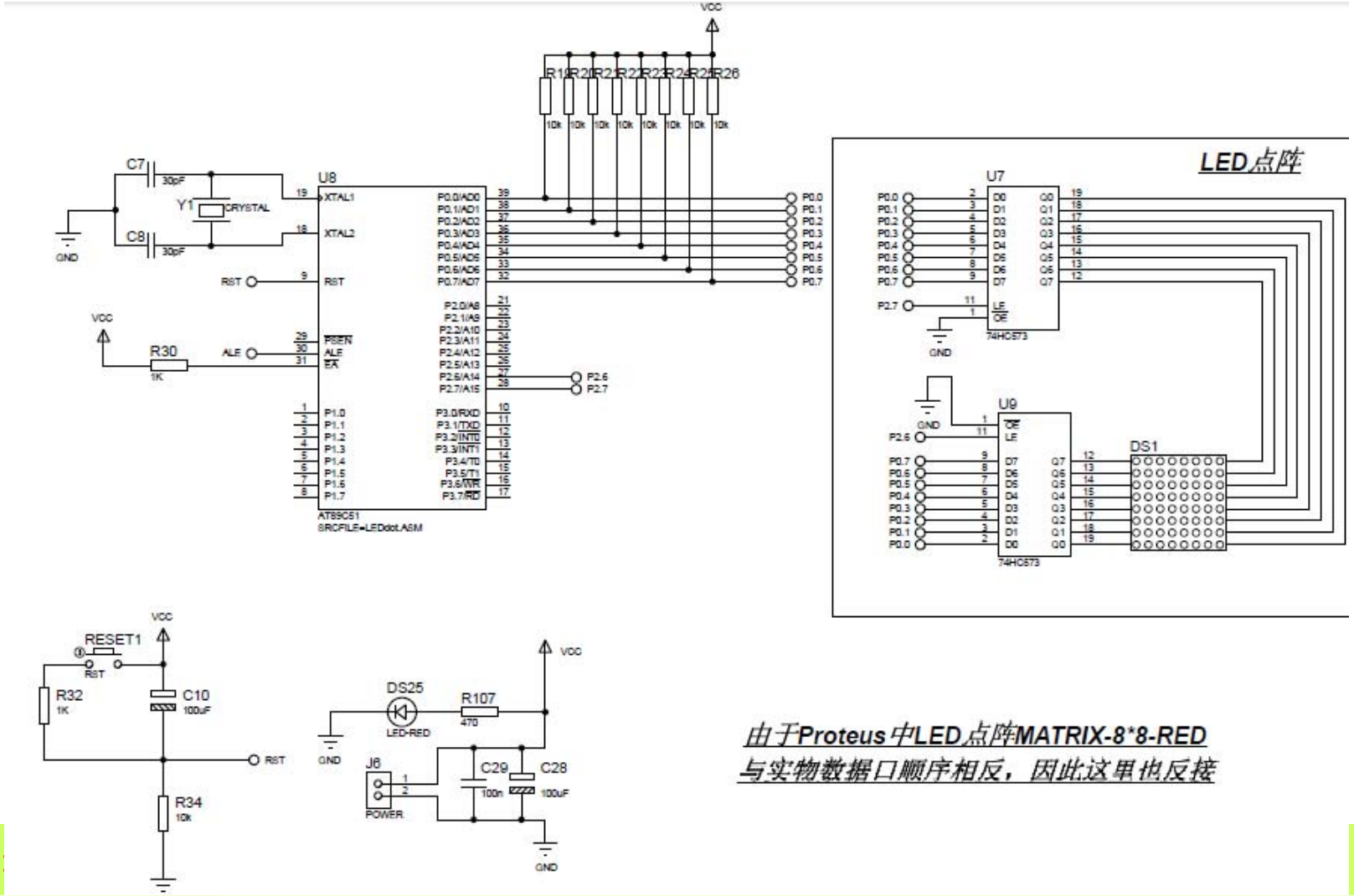
点阵式LED原理

D_7	0	0	0	0	0	0	0	0
D_6	0	0	0	0	1	0	0	0
D_5	0	0	0	1	1	0	0	0
D_4	0	0	0	0	1	0	0	0
D_3	0	0	0	0	1	0	0	0
D_2	0	0	0	0	1	0	0	0
D_1	0	0	0	0	1	0	0	0
D_0	0	0	0	1	1	1	0	0
	00	00	00	21	7F	01	00	00

```
GljlwbDuud|#
DB 00H,00H,3EH,41H,41H,41H,3EH,00H    ; 0
DB 00H,00H,00H,00H,21H,7FH,01H,00H    ; 1
DB 00H,00H,27H,45H,45H,45H,39H,00H    ; 2
DB 00H,00H,22H,49H,49H,49H,36H,00H    ; 3
DB 00H,00H,0CH,14H,24H,7FH,04H,00H    ; 4
DB 00H,00H,72H,51H,51H,51H,4EH,00H    ; 5
DB 00H,00H,3EH,49H,49H,49H,26H,00H    ; 6
DB 00H,00H,40H,40H,40H,4FH,70H,00H    ; 7
DB 00H,00H,36H,49H,49H,49H,36H,00H    ; 8
DB 00H,00H,32H,49H,49H,49H,3EH,00H    ; 9
```

2. 点阵式应用案例

要求：LED点阵以每隔8秒右向左移动一个字符的速度显示“0、1、2、3、4、5、6、7、8、9”，数据口通过P0扩展口作为点阵扫描接口。



由于Proteus中LED点阵MATRIX-8*8-RED与实物数据口顺序相反，因此这里也反接

2. 点阵式应用案例

程序：

```
; LED点阵以每隔8秒右向左移动一个字符的速度显示"0、1、2、3、4、5、6、7、8、9"
; 数据口通过P0扩展口作为点阵扫描接口
; 显示内容放到主程序中，用50H~57H作为显示缓冲单元

SAN         EQU 30H     ; 扫描码单元
TIM         EQU 31H     ; 定时循环次数
CNTC        EQU 32H     ; 字符列循环
LEDData0    EQU 50H     ; 8个LED点阵数据缓存器
LEDData1    EQU 51H
LEDData2    EQU 52H
LEDData3    EQU 53H
LEDData4    EQU 54H
LEDData5    EQU 55H
LEDData6    EQU 56H
LEDData7    EQU 57H
LEDSEG_LE   BIT P2.6    ; LED点阵行数据锁存位
LEDSEG_ROW  BIT P2.7    ; LED点阵列数据锁存位
```

2. 点阵式应用案例

主程序：

```
        ORG     0000H
        AJMP    MAIN
        ORG     000BH
        JMP     PITO

        ORG     0100H
MAIN:   MOV     SP, #60H
        MOV     TMOD, #01H      ; 将定时器0设置为方式1
        MOV     TL0, #0DCH      ; 定时125ms
        MOV     TH0, #0BH

        MOV     TIM, #08H       ; 设置定时循环次数
        MOV     LEDData0, #00H  ; 首次显示“0”
        MOV     LEDData1, #00H
        MOV     LEDData2, #3EH
        MOV     LEDData3, #41H
        MOV     LEDData4, #41H
        MOV     LEDData5, #41H
        MOV     LEDData6, #3EH
        MOV     LEDData7, #00H
        MOV     CNTC, #00H
        CLR     LEDSEG_LE       ; LED点阵行数据锁存位

        SETB    EA              ; 开总中断
        SETB    ET0             ; 定时0中断允许
        SETB    TR0
HERE:   ACALL   DISP
        AJMP    HERE
```

LED接口技术

显示子程序DISP:

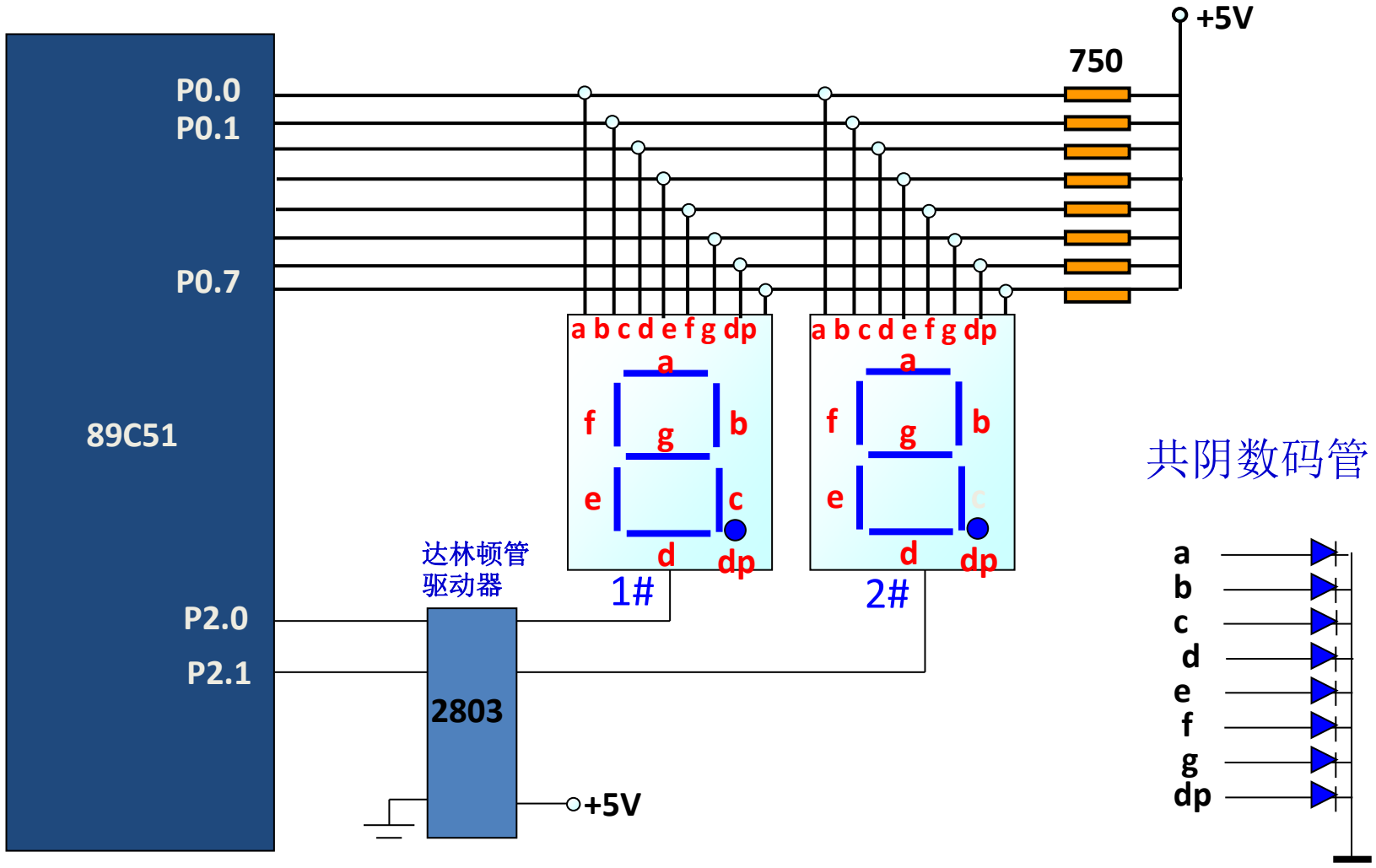
```
DISP: PUSH  DPH
      PUSH  DPL
      PUSH  PSW
      PUSH  ACC
      MOV   R0, #LEDData0   ; 显示缓冲单元首地址
      MOV   SAN, #0FEH      ; 位控码
LD0:  MOV   P0, #0FFH       ; 清位控口,修改
      SETB  LEDSEG_ROW      ; LED点阵列数据锁存位
      NOP
      CLR   LEDSEG_ROW
      MOV   A, @R0          ; 50~57H中直接存放显示码
      MOV   P0, A           ; 送段码
      SETB  LEDSEG_LE       ; LED点阵行数据锁存位
      NOP
      CLR   LEDSEG_LE
      MOV   A, SAN
      MOV   P0, A           ; 送位控码
      SETB  LEDSEG_ROW      ; LED点阵列数据锁存位
      NOP
      CLR   LEDSEG_ROW
      ACALL DELAY
      INC   R0
      MOV   A, SAN
      JNB   ACC.7, LD1
      RL    A
      MOV   SAN, A
      AJMP  LD0
LD1:  MOV   P0, #00         ; 清段控口,修改
      SETB  LEDSEG_LE       ; LED点阵行数据锁存位
      NOP
      CLR   LEDSEG_LE
      POP   ACC
      POP   PSW
      POP   DPL
      POP   DPH
      RET
```

延时子程序DELAY:

```
DELAY: MOV  R3,#02H     ;延时约1ms
DEL0:  MOV  R2, #00H
DEL1:  DJNZ R2, DEL1
       DJNZ R3, DEL0
       RET
```

PITO 子程序、表格DIGIT见泛雅平台上传资料

七、实例设计1—LED显示



汇编程序综合实例设计

程序一(显示“12”)

```
        ORG    0000H
        AJMP   MAIN
        ORG    0100H
MAIN:   MOV    SP,  #60H
DISP:   MOV    P0,  #06H     ; 1#LED显示“1”
        SETB   P2.0
        CLR    P2.1
        ACALL DELAY          ; 延时50ms
        MOV    P0, #5BH      ; 2#LED显示“2”
        SETB   P2.1
        CLR    P2.0
        ACALL DELAY
        AJMP   DISP
```

```
; 延时子程序
DELAY:  MOV  R1, #61H   ;延时约50ms，在Products中对于独立的数码管显示需要加长延时
DEL0:   MOV  R2, #00H
DEL1:   DJNZ R2, DEL1
        DJNZ R1, DEL0
        RET
        END     ;结束整个源程序
```

程序二(显示“12”)

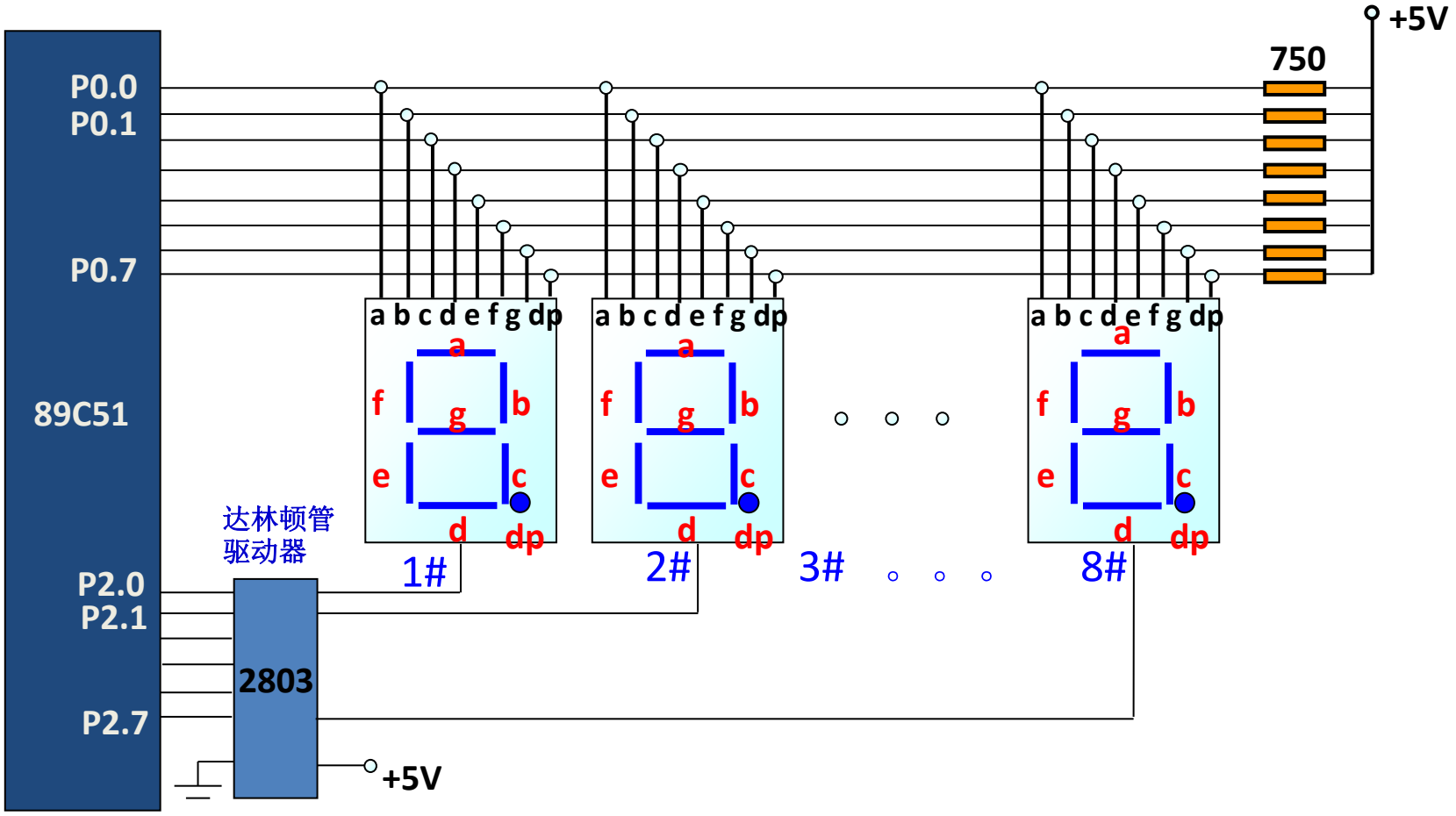
在程序一的基础上做了改进简单，解决了闪动、变化过程中未清0等问题，但该程序内容仍无法修改.

```
;主程序
        ORG     0000H
        AJMP    MAIN

        ORG     0100H
MAIN:   MOV     SP,  #60H
        MOV     50H, #06H
        MOV     51H, #5BH

HERE:   ACALL   DISP
        AJMP    HERE
```

```
;显示子程序
DISP: MOV  P2, #00H     ;添加清0
      MOV  P0, 50H      ;送段码
      MOV  P2, #01H     ;送位控码
      ACALL DELAY
      MOV  P2, #00H     ;添加清0
      MOV  P0, 51H
      MOV  P2, #02H
      ACALL DELAY
      RET
;延时子程序
DELAY: MOV    R1, #02H   ;延时约1ms
DEL0:  MOV    R2, #00H
DEL1:  DJNZ   R2, DEL1
       DJNZ   R1, DEL0
       RET
       END    ;勿忘!!!
```

程序三(显示12345678)

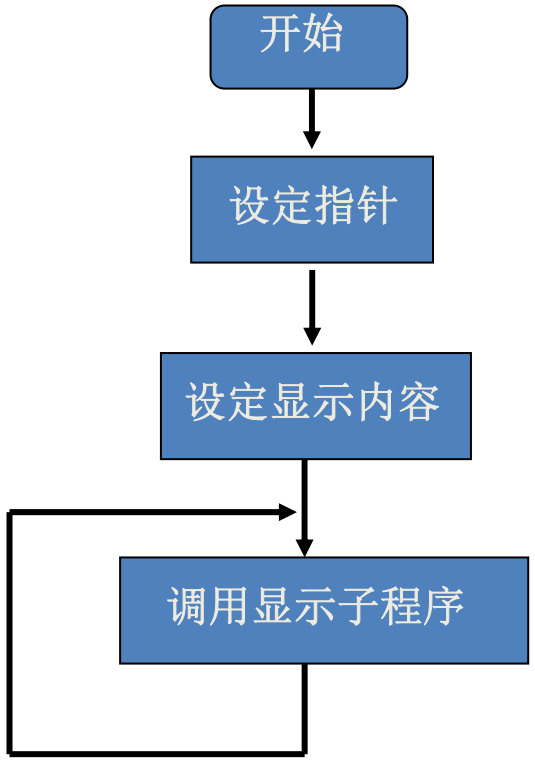
增加到8个数码管设计



汇编程序综合实例设计

```
        ORG     0000H
        AJMP    MAIN
        ORG     0100H
MAIN:   MOV     SP,  #5FH
        MOV     50H, #01H
        MOV     51H, #02H
        MOV     52H, #03H
        MOV     53H, #04H
        MOV     54H, #05H
        MOV     55H, #06H
        MOV     56H, #07H
        MOV     57H, #08H
HERE:   ACALL   DISP
        AJMP    HERE
```

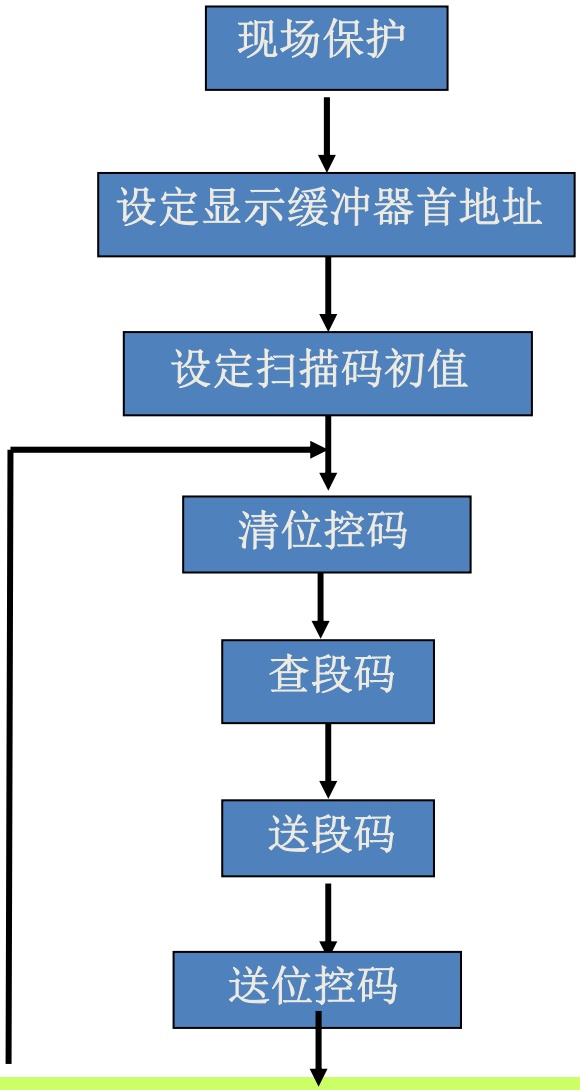
主程序流程



汇编程序综合实例设计

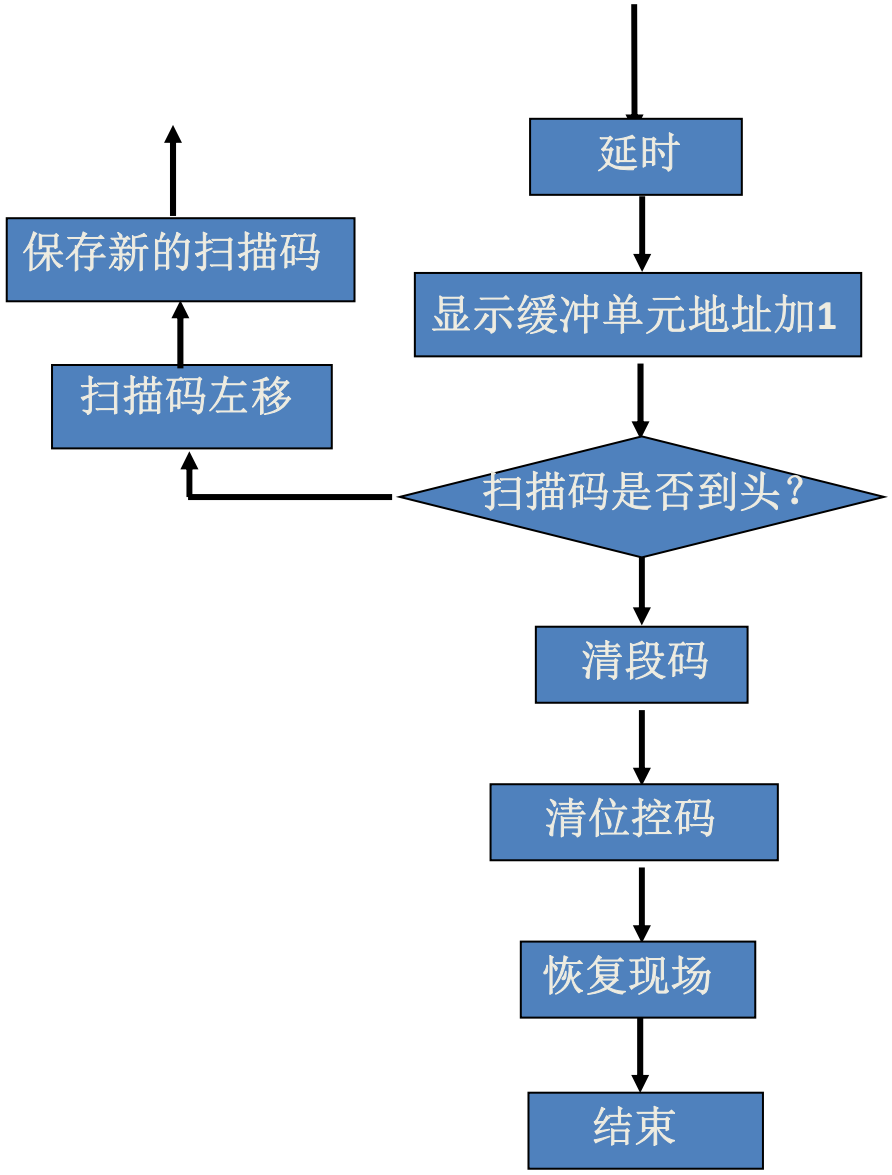
```
DISP:   PUSH   DPH
        PUSH   DPL
        PUSH   PSW
        PUSH   ACC
        SETB   PSW.3          ;第一组通用寄存器
        MOV    R0,  #50H
        MOV    30H,  #01H
LD0:    MOV    P2, #0C0H      ;清位控
        MOV    DPTR,  #TABLE
        MOV    A,   @R0
        MOVC   A,   @A+DPTR   ;查LED显示码
        MOV    P0,  A         ;送段码
        MOV    A,  30H
        ORL    P2,  A         ;送位控码
```

显示子程序流程



汇编程序综合实例设计

```
            ACALL   DELAY
            INC     R0
            MOV     A,  30H
            JB      ACC.7, LD1 ;位控码是否到头
            RL      A
            MOV     30H,  A
            AJMP    LD0
LD1:        MOV     P0, #00H    ;清段码
            MOV     P2, #00H    ;清位控
            POP     ACC
            POP     PSW
            POP     DPL
            POP     DPH
            RET
TABLE:      DB  3FH,06H,5BH,4FH,66H  ; 0 1 2 3 4
            DB  6DH,7DH,07H,7FH,6FH  ; 5 6 7 8 9
            DB  77H,7CH,39H,5EH,79H  ; A B C D E
            DB  71H,00H              ; F 灭（空）
DELAY:      MOV     R1, #02H ; 延时约1ms
DEL0:       MOV     R2, #00H
DEL1:       DJNZ    R2, DEL1
            DJNZ    R1, DEL0
            RET
            END     ; 勿忘!!!
```



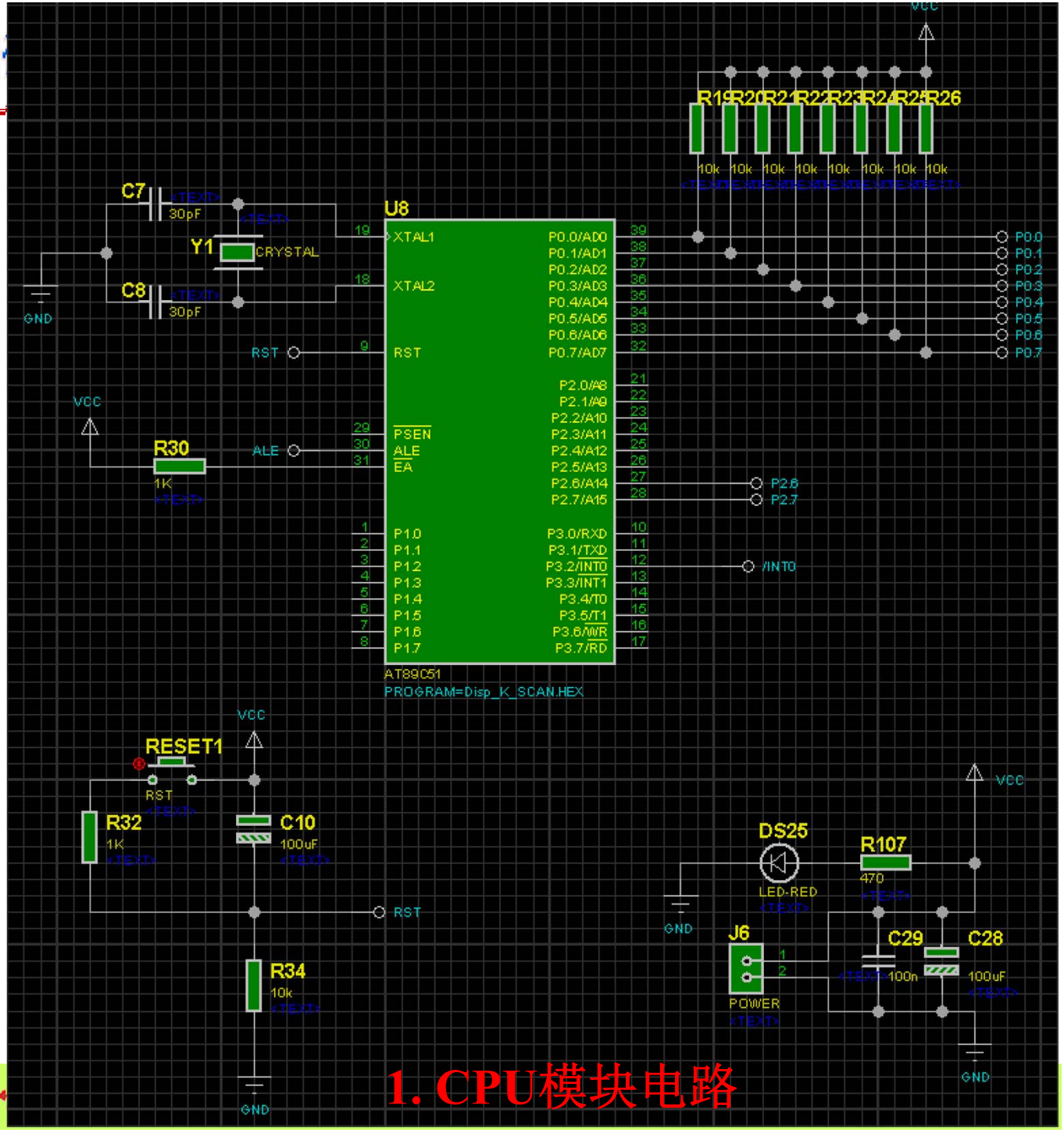
八、实例设计2—按键与显示

要求：

(1) P0口的2个扩展口作为段控口和位控口，通过使用P2.6和P2.7对SN74HC573芯片的使能.

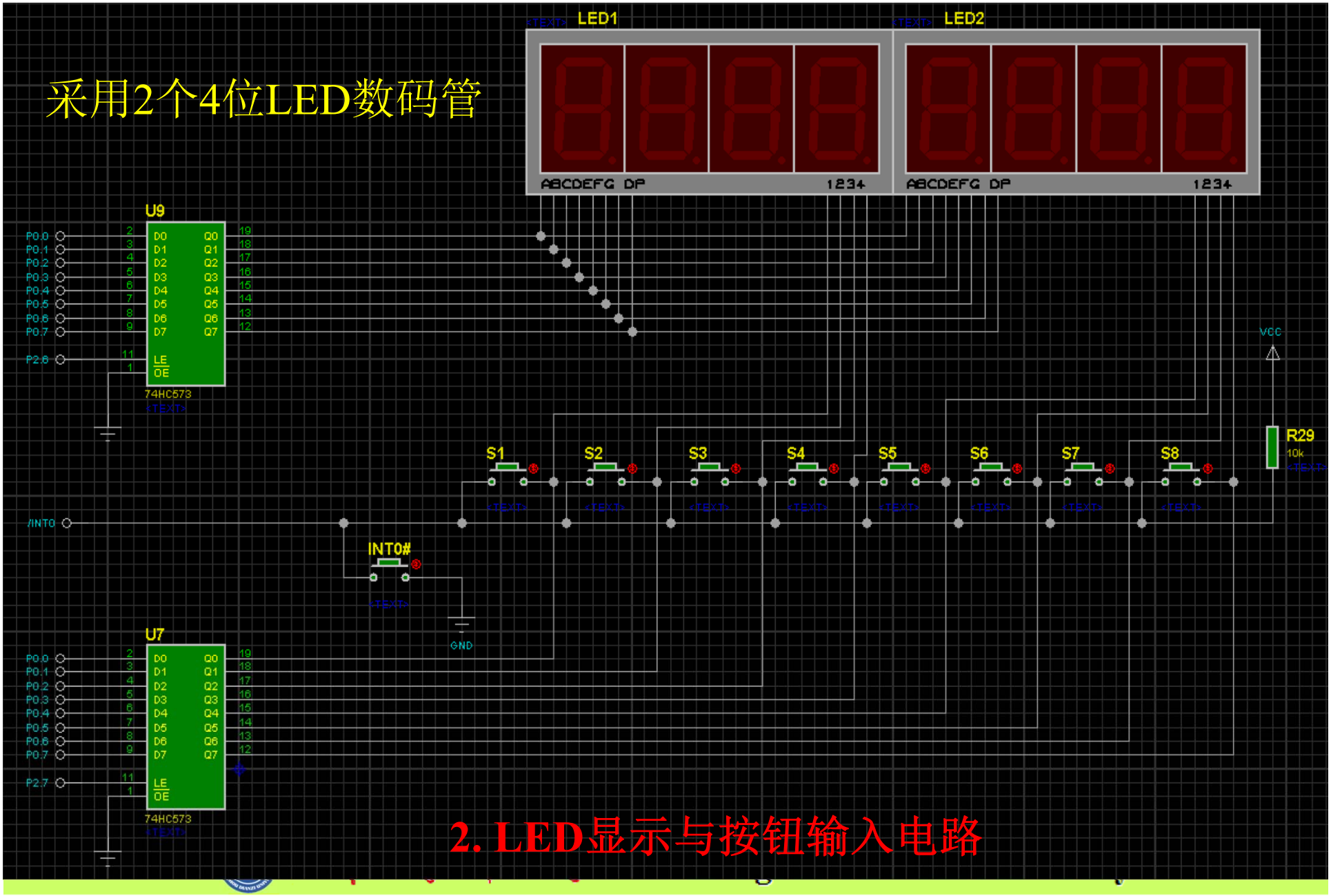
(2) P3.2 (/INT0)作为按钮输入口构成一个“0#～#7”的8个按钮和8个LED的显示按钮电路，系统复位时，显示“HELLO-51”.

(3) 当按下任意键时，在最右边LED上显示该键号，原显示内容自动左移.



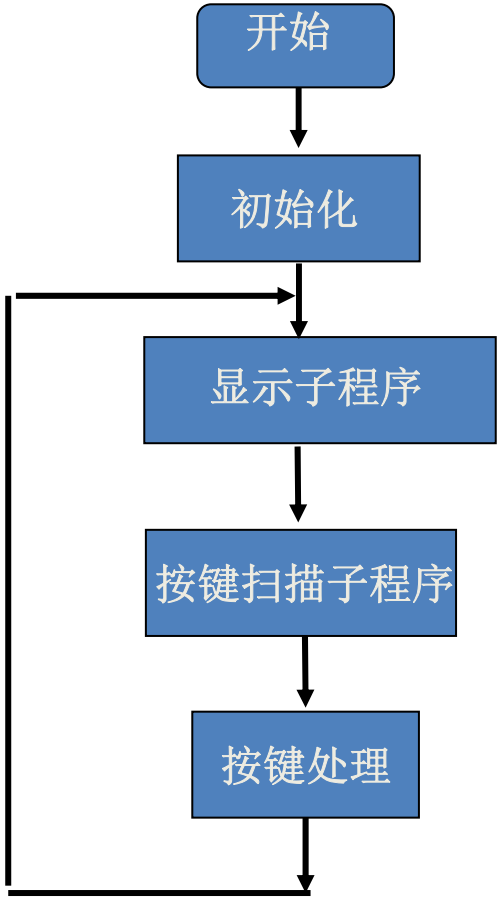
1. CPU模块电路

汇编程序综合实例设计


采用2个4位LED数码管
LED1
LED2
ABCDEFG DP
1234
U9
U7
74HC573
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.6
P2.7
LE
OE
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
S1
S2
S3
S4
S5
S6
S7
S8
INT0#
/INT0
GND
VCC
R29
10k
2. LED显示与按钮输入电路

汇编程序综合实例设计

要求：P0口的2个扩展口作为段控口和位控口，通过使用P2.6和P2.7对SN74HC573芯片的使能。P3.2（/INT0）作为按钮输入口构成一个“0#～#7”的8个按钮和8个LED的显示按钮电路，系统复位时，显示“HELLO-51”，当按下任意键时，在最右边LED上显示该键号，原显示内容自动左移。



SP设置；
显示初值设置等

← 显示缓冲单元地址 RAM 50H～ 57H

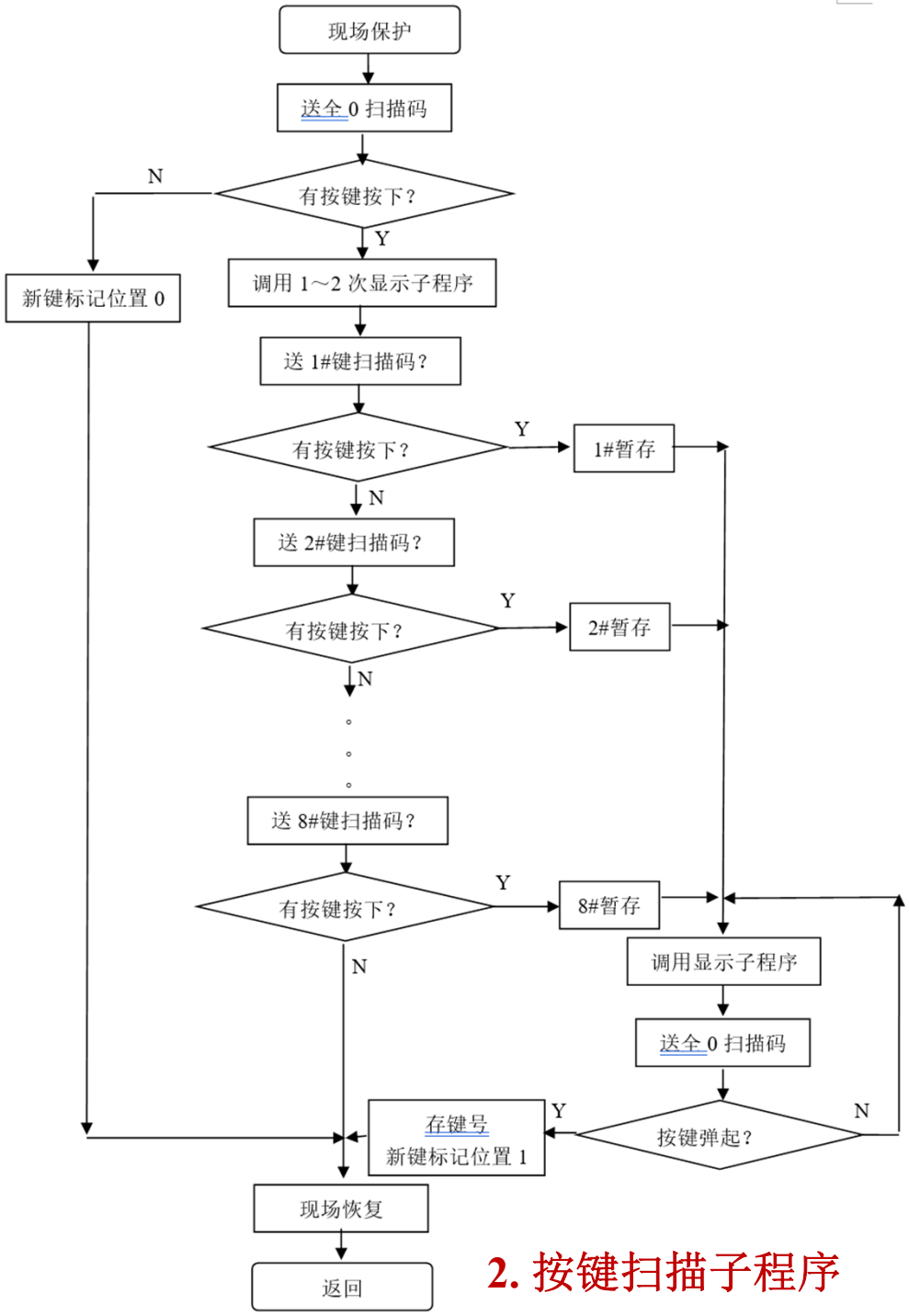
→ 新键号存储单元　RAM 38H，
新按键标记位　01H

→按键移位只需显示单元内容左移，并将新键号放入右边的显示缓冲

1. 主程序流程

汇编程序综合实例设计

要求：P0口的2个扩展口作为段控口和位控口，通过使用P2.6和P2.7对SN74HC573芯片的使能。P3.2（INT0）作为按钮输入口构成一个“0#～#7”的8个按钮和8个LED的显示按钮电路，系统复位时，显示“HELLO-51”，当按下任意键时，在最右边LED上显示该键号，原显示内容自动左移。



2. 按键扫描子程序

汇编程序综合实例设计

```
; 系统复位时，显示"HELLO-51"，当按下任何时，在最右边LED上显示该键号，原显示内容自动左移
; 显示程序采用子程序结构，使用查表指令; 显示内容放到主程序中，用50H~57H作为显示缓冲单元，
; 由30H放位控码；新键标记位01H，新键38H，新键暂存37H
; 该程序的关键在于按键处理，处理前先消除新键标记位
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 主程序
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            ORG    0000H
            AJMP   MAIN
            ORG    0100H
MAIN:       MOV    SP, #60H
            MOV    50H, #11H      ;设置了显示缓冲单元50H~57H
            MOV    51H, #0EH      ;显示"HELLO-51"
            MOV    52H, #12H
            MOV    53H, #12H
            MOV    54H, #00H
            MOV    55H, #13H
            MOV    56H, #05H
            MOV    57H, #01H
            CLR    P2.6
            CLR    P2.7
            CLR    01H            ;清新按钮标记位
HERE:       ACALL  DISP
            ACALL  KEYSCAN
            ACALL  KEYPRO
            AJMP   HERE
```

汇编程序综合实例设计

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 显示子程序
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISP: PUSH   DPH
      PUSH   DPL
      PUSH   PSW
      PUSH   ACC
      MOV    R0, #50H          ;显示缓冲单元首地址
      MOV    30H, #0FEH        ;位控码
LD0:  MOV    P0, #0FFH         ;清位控口,修改
      SETB   P2.7
      NOP
      CLR    P2.7
      MOV    DPTR, #TABLE      ;查段码
      MOV    A, @R0
      MOVC   A, @A+DPTR
      MOV    P0, A             ;送段码
      SETB   P2.6
      NOP
      CLR    P2.6
      MOV    A, 30H
      MOV    P0, A             ;送位控码
      SETB   P2.7
      NOP
      CLR    P2.7
      ACALL  DELAY
      INC    R0
      MOV    A, 30H
      JNB    ACC.7, LD1
      RL     A
      MOV    30H, A
      AJMP   LD0
LD1:  MOV    P0, #00H          ;清段控口,修改
      SETB   P2.6
      NOP
      CLR    P2.6
      MOV    P0, #0FFH         ;清位控口,修改
      SETB   P2.7
      NOP
      CLR    P2.7
      POP    ACC
      POP    PSW
      POP    DPL
      POP    DPH
      RET

TABLE: DB 3FH,06H,5BH,4FH,66H    ; 0  1  2  3  4
       DB 6DH,7DH,07H,7FH,6FH    ; 5  6  7  8  9
       DB 77H,7CH,39H,5EH,79H    ; A  B  C  D  E
       DB 71H,00H,76H,38H,40H    ; F 灭  H  L  -
```

汇编程序综合实例设计

```
; 键盘扫描子程序
KEYSCAN: PUSH   DPH
         PUSH   DPL
         PUSH   PSW
         PUSH   ACC
         MOV    P0, #00H   ;送全0码，检查是否有按钮按下
         SETB   P2.7
         NOP
         CLR    P2.7
         JNB    P3.2, ONE
         CLR    01H
KEY_END: POP    ACC
         POP    PSW
         POP    DPL
         POP    DPH
         RET
ONE:     ACALL  DISP       ;调用现实延时，消抖
         MOV    P0, #0FEH  ;送01#键扫描码
         SETB   P2.7
         NOP
         CLR    P2.7
         JB     P3.2, TWO
         MOV    37H, #01H
         AJMP   KEY_D
TWO:     MOV    P0, #0FDH   ;送02#键扫描码
         SETB   P2.7
         NOP
         CLR    P2.7
         JB     P3.2, THREE
         MOV    37H, #02H
         AJMP   KEY_D
THREE: MOV   P0, #0FBH          ;送03#键扫描码
       SETB  P2.7
       NOP
       CLR   P2.7
       JB    P3.2, FOUR
       MOV   37H, #03H
       AJMP  KEY_D
FOUR:  MOV   P0, #0F7H          ;送04#键扫描码
       SETB  P2.7
       NOP
       CLR   P2.7
       JB    P3.2, FIVE
       MOV   37H, #04H
       AJMP  KEY_D
FIVE:  MOV   P0, #0EFH          ;送05#键扫描码
       SETB  P2.7
       NOP
       CLR   P2.7
       JB    P3.2, SIX
       MOV   37H, #05H
       AJMP  KEY_D
SIX:   MOV   P0, #0DFH          ;送06#键扫描码
       SETB   P2.7
       NOP
       CLR    P2.7
       JB     P3.2, SEVEN
       MOV    37H, #06H
       AJMP   KEY_D
```

汇编程序综合实例设计

```
; 键盘扫描子程序(续)
SEVEN:   MOV   P0, #0BFH        ;送07#键扫描码
         SETB  P2.7
         NOP
         CLR   P2.7
         JB    P3.2, EIGHT
         MOV   37H, #07H
         AJMP  KEY_D
EIGHT:   MOV   P0, #7FH         ;送08#键扫描码
         SETB  P2.7
         NOP
         CLR   P2.7
         JB    P3.2, KEY_E
         MOV   37H, #08H

KEY_D:   ACALL DISP
         MOV   P0, #00H         ;送全0码，检查是否有按钮按下
         SETB  P2.7
         NOP
         CLR   P2.7
         JNB   P3.2, KEY_D      ;是否弹开
         MOV   A, 37H
         MOV   38H, 37H
         SETB  01H              ;设定新键标记
         AJMP  KEY_END
KEY_E:   CLR   01H
         AJMP  KEY_END
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;
; 按键处理子程序
; 处理前先消除新键标记位
;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;
KEYPRO:  JNB   01H, PROEND
         CLR   01H              ;移位处理
         MOV   50H, 51H
         MOV   51H, 52H
         MOV   52H, 53H
         MOV   53H, 54H
         MOV   54H, 55H
         MOV   55H, 56H
         MOV   56H, 57H
         MOV   57H, 38H
PROEND:  RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 延时子程序(约1ms)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY:   MOV   R1, #02H
DEL0:    MOV   R2, #00H
DEL1:    DJNZ  R2, DEL1
         DJNZ  R1, DEL0
         RET

         END                    ; 结束整个源程序
```

课后思考

仍然针对该键盘与显示综合设计：P0口的2个扩展口作为段控口和位控口，通过使用P2.6和P2.7对SN74HC573芯片的使能。P3.2 (/INT0)作为按钮输入口构成一个“0#～#7”的8个按钮和8个LED的显示按钮电路，系统复位时，显示“HELLO-51”，当按下任意键时，在最右边LED上显示该键号，原显示内容自动左移。

如果采用基于总线的控制方式，该如何进行硬件设计和软件编程呢？

THE END

THANK YOU