单片机教学第二十四课：键盘接口与编程

键盘是由若干按键组成的开关矩阵，它是微型计算机最常用的输入设备，用户可以通过键盘向计算机输入指令、地址和数据。一般单片机系统中采和非编码键盘，非编码键盘是由软件来识别键盘上的闭合键，它具有结构简单，使用灵活等特点，因此被广泛应用于单片机系统。

按键开关的抖动问题

组成键盘的按键有触点式和非触点式两种，单片机中应用的一般是由机械触点构成的。在下图中，当开

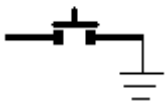


图 1

图 2

关 S 未被按下时，P1。0 输入为高电平，S 闭合后，P1。0 输入为低电平。由于按键是机械触点，当机械触点断开、闭合时，会有抖动动，P1。0 输入端的波形如图 2 所示。这种抖动对于人来说是感觉不到的，但对计算机来说，则是完全可以感应到的，因为计算机处理的速度是在微秒级，而机械抖动的时间至少是毫秒级，对计算机而言，这已是一个“漫长”的时间了。前面我们讲到中断时曾有个问题，就是说按键有时灵，有时不灵，其实就是这个原因，你只按了一次按键，可是计算机却已执行了多次中断的过程，如果执行的次数正好是奇数次，那么结果正如你所料，如果执行的次数是偶数次，那就不对了。

为使 CPU 能正确地读出 P1 口的状态，对每一次按键只作一次响应，就必须考虑如何去除抖动，常用的去抖动的方法有两种：硬件方法和软件方法。单片机中常用软件法，因此，对于硬件方法我们不介绍。软件法其实很简单，就是在单片机获得 P1。0 口为低的信息后，不是立即认定 S1 已被按下，而是延时 10 毫秒或更长一些时间后再次检测 P1。0 口，如果仍为低，说明 S1 的确按下了，这实际上是避开了按键按下时的抖动时间。而在检测到按键释放后（P1。0 为高）再延时 5-10 个毫秒，消除后沿的抖动，然后再对键值处理。不过一般情况下，我们通常不对按键释放的后沿进行处理，实践证明，也能满足一定的要求。当然，实际应用中，对按键的要求也是千差万别，要根据不同的需要来编制处理程序，但以上是消除键抖动的原则。

键盘与单片机的连接

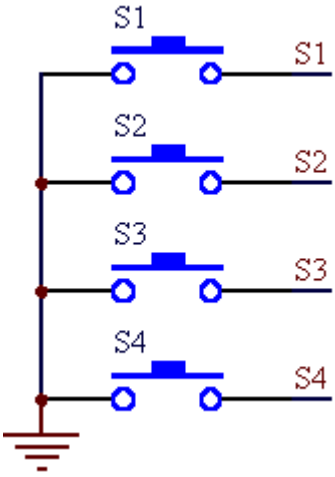


图 3

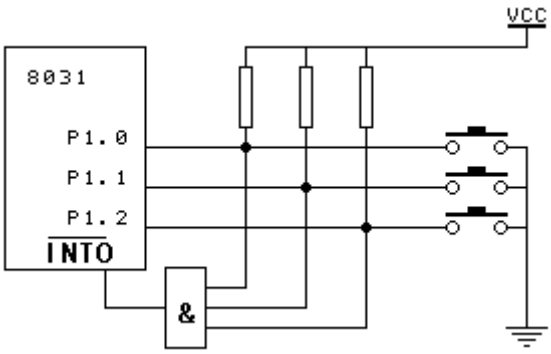


图 4

1、通过 1/0 口连接

将每个按键的一端接到单片机的 I/O 口，另一端接地，这是最简单的方法，如图 3 所示是实验板上按键的接法，四个按键分别接到 P3.2 、P3.3、P3.4 和 P3.5。对于这种键各程序可以采用不断查询的方法，功能就是：检测是否有键闭合，如有键闭合，则去除键抖动，判断键号并转入相应的键处理。下面给出一个例程。其功能很简单，四个键定义如下：

P3.2：开始，按此键则灯开始流动（由上而下）
P3.3：停止，按此键则停止流动，所有灯为暗
P3.4：上，按此键则灯由上向下流动
P3.5：下，按此键则灯由下向上流动

```
UpDown EQU 00H ;上下行标志
StartEnd EQU 01H ;起动及停止标志
LAMPCODE EQU 21H ;存放流动的数据代码
ORG 0000H
AJMP MAIN
ORG 30H
MAIN:
MOV SP,#5FH
MOV P1,#0FFH
CLR UpDown ;启动时处于向上的状态
CLR StartEnd ;启动时处于停止状态
MOV LAMPCODE,#0FEH ;单灯流动的代码
LOOP:
ACALL KEY ;调用键盘程序
JNB F0,LNEXT ;如果无键按下，则继续
ACALL KEYPROC ;否则调用键盘处理程序
```

```
LNEXT:
ACALL LAMP ;调用灯显示程序
AJMP LOOP ;反复循环，主程序到此结束
;---------------------------------------
DELAY:
MOV R7,#100
D1: MOV R6,#100
DJNZ R6,$
DJNZ R7,D1
RET
;----------------------------------------延时程序，键盘处理中调用
KEYPROC:
MOV A,B ;从 B 寄存器中获取键值
JB ACC.2,KeyStart ;分析键的代码，某位被按下，则该位为 1（因为在键盘程序中已取反）
JB ACC.3,KeyOver
JB ACC.4,KeyUp
JB ACC.5,KeyDown
AJMP KEY_RET
KeyStart:
SETB StartEnd ;第一个键按下后的处理
AJMP KEY_RET
KeyOver:
CLR StartEnd ;第二个键按下后的处理
AJMP KEY_RET
KeyUp: SETB UpDown ;第三个键按下后的处理
AJMP KEY_RET
KeyDown:
CLR UpDown ;第四个键按下后的处理
KEY_RET:RET
KEY:
CLR F0 ;清 F0，表示无键按下。
ORL P3,#00111100B ;将 P3 口的接有键的四位置 1
MOV A,P3 ;取 P3 的值
ORL A,#11000011B ;将其余 4 位置 1
CPL A ;取反
JZ K_RET ;如果为 0 则一定无键按下
ACALL DELAY ;否则延时去键抖
ORL P3,#00111100B
MOV A,P3
ORL A,#11000011B
CPL A
JZ K_RET
MOV B,A ;确实有键按下，将键值存入 B 中
SETB F0 ;设置有键按下的标志
```

```
K_RET:
ORL P3,#00111100B ;此处循环等待键的释放
MOV A,P3
ORL A,#11000011B
CPL A
JZ K_RET1 ;直到读取的数据取反后为 0 说明键释放了，才从键盘处理程序中返回
AJMP K_RET
K_RET1:
RET
;-----------------------------------
D500MS: ;流水灯的延迟时间
PUSH PSW
SETB RS0
MOV R7,#200
D51: MOV R6,#250
D52: NOP
NOP
NOP
NOP
DJNZ R6,D52
DJNZ R7,D51
POP PSW
RET
;-----------------------------------
LAMP:
JB StartEnd,LampStart ;如果 StartEnd=1，则启动
MOV P1,#0FFH
AJMP LAMPRET ;否则关闭所有显示，返回
LampStart:
JB UpDown,LAMPUP ;如果 UpDown=1，则向上流动
MOV A,LAMPCODE
RL A ;实际就是左移位而已
MOV LAMPCODE,A
MOV P1,A
LCALL D500MS
AJMP LAMPRET
LAMPUP:
MOV A,LAMPCODE
RR A ;向下流动实际就是右移
MOV LAMPCODE,A
MOV P1,A
LCALL D500MS
LAMPRET:
RET
```

END

以上程序功能很简单，但它演示了一个键盘处理程序的基本思路，程序本身很简单，也不很实用，实际工作中还会有好多要考虑的因素，比如主循环每次都调用灯的循环程序，会造成按键反应“迟钝”，而如果一直按着键不放，则灯不会再流动，一直要到松开手为止，等等，大家可以仔细考虑一下这些问题，再想想有什么好的解决办法。

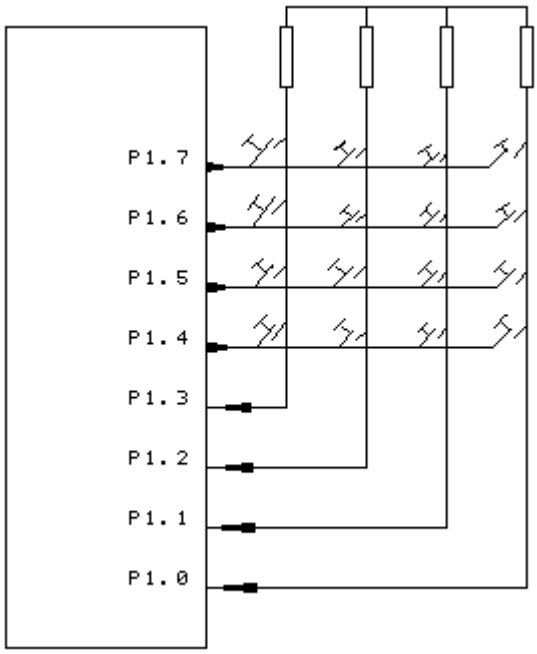
2、采用中断方式

如图 4 所示。各个按键都接到一个与非上，当有任何一个按键按下时，都会使与门输出为低电平，从而引起单片机的中断，它的好处是不用在主程序中不断地循环查询，如果有键按下，单片机再去做相应的处理。

单片机教学第二十五课：矩阵式键盘接口技术及编程

矩阵式键盘的结构与工作原理：

在键盘中按键数量较多时，为了减少 I/O 口的占用，通常将按键排列成矩阵形式，如图 1 所示。在矩阵式键盘中，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接。这样，一个端口（如 P1 口）就可以构成 4*4=16 个按键，比之直接将端口线用于键盘多出了一倍，而且线数越多，区别越明显，比如再多加一条线就可以构成 20 键的键盘，而直接用端口线则只能多出一键（9 键）。由此可见，在



需要的键数比较多时，采用矩阵法来做键盘是合理的。

矩阵式结构的键盘显然比直接法要复杂一些，识别也要复杂一些，上图中，列线通过电阻接正电源，并将行线所接的单片机的 I/O 口作为输出端，而列线所接的 I/O 口则作为输入。这样，当按键没有按下时，所有的输出端都是高电平，代表无键按下。行线输出是低电平，一旦有键按下，则输入线就会被拉低，这样，通过读入输入线的状态就可得知是否有键按下了。具体的识别及编程方法如下所述。

矩阵式键盘的按键识别方法

确定矩阵式键盘上何键被按下介绍一种“行扫描法”。

行扫描法

行扫描法又称为逐行（或列）扫描查询法，是一种最常用的按键识别方法，如上图所示键盘，介绍过程如下。

判断键盘中有无键按下 将全部行线 Y0-Y3 置低电平，然后检测列线的状态。只要有一

列的电平为低，则表示键盘中有键被按下，而且闭合的键位于低电平线与 4 根行线相交叉的 4 个按键之中。若所有列线均为高电平，则键盘中无键按下。

判断闭合键所在的位置 在确认有键按下后，即可进入确定具体闭合键的过程。其方法是：依次将行线置为低电平，即在置某根行线为低电平时，其它线为高电平。在确定某根行线位置为低电平后，再逐行检测各列线的电平状态。若某列为低，则该列线与置为低电平的行线交叉处的按键就是闭合的按键。

下面给出一个具体的例子：

图仍如上所示。8031 单片机的 P1 口用作键盘 I/O 口，键盘的列线接到 P1 口的低 4 位，键盘的行线接到 P1 口的高 4 位。列线 P1.0-P1.3 分别接有 4 个上拉电阻到正电源+5V，并把列线 P1.0-P1.3 设置为输入线，行线 P1.4-P.17 设置为输出线。4 根行线和 4 根列线形成 16 个相交点。

检测当前是否有键被按下。检测的方法是 P1.4-P1.7 输出全“0”，读取 P1.0-P1.3 的状态，若 P1.0-P1.3 为全“1”，则无键闭合，否则有键闭合。

去除键抖动。当检测到有键按下后，延时一段时间再做下一步的检测判断。

若有键被按下，应识别出是哪一个键闭合。方法是对键盘的行线进行扫描。P1.4-P1.7 按下述 4 种组合依次输出：

P1.7 1 1 1 0

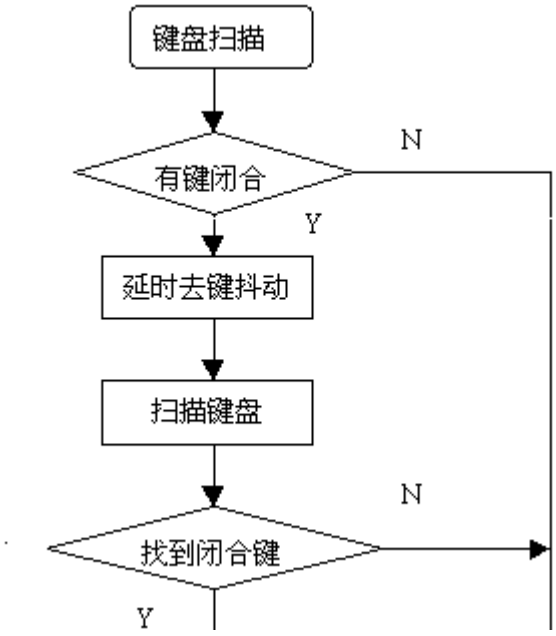
P1.6 1 1 0 1

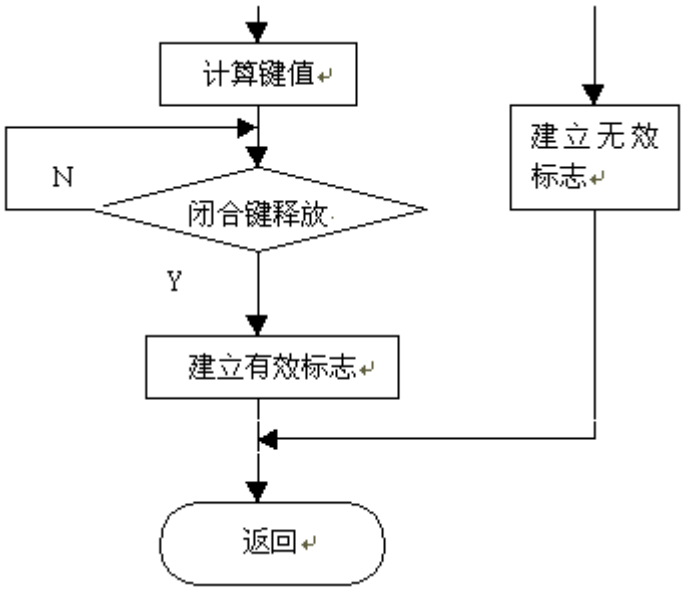
P1.5 1 0 1 1

P1.4 0 1 1 1

在每组行输出时读取 P1.0-P1.3，若全为“1”，则表示为“0”这一行没有键闭合，否则有键闭合。由此得到闭合键的行值和列值，然后可采用计算法或查表法将闭合键的行值和列值转换成所定义的键值

为了保证键每闭合一次 CPU 仅作一次处理，必须却除键释放时的抖动。





键盘扫描程序：

从以上分析得到键盘扫描程序的流程图如图 2 所示。程序如下

```
SCAN: MOV P1,#0FH
MOV A,P1
ANL A,#0FH
CJNE A,#0FH,NEXT1
SJMP NEXT3
NEXT1: ACALL D20MS
MOV A,#0EFH
NEXT2: MOV R1,A
MOV P1,A
MOV A,P1
ANL A,#0FH
CJNE A,#0FH,KCODE;
MOV A,R1
SETB C
RLC A
JC NEXT2
NEXT3: MOV R0,#00H
RET
KCODE: MOV B,#0FBH
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
ANL A,#0FH
CJNE A,#0FH,NEXT6
MOV R0,#0FFH
RET
```

键盘处理程序就作这么一个简单的介绍，实际上，键盘、显示处理是很复杂的，它往往占到一个应用程序的大部份代码，可见其重要性，但说到，这种复杂并不来自于单片机的本身，而是来自于操作者的习惯等等问题，因此，在编写键盘处理程序之前，最好先把它从逻辑上理清，然后用适当的算法表示出来，最后再去写代码，这样，才能快速有效地写好代码。

到本课为止，本站教程暂告一个段落！感谢大家的关心和支持！