单片机键盘接口全接触

作者：杜洋

2005-10-25

一般的具有人机对话的单片机系统少不了会有键盘。键盘接口的原理与应用许多的教材都有介绍，但通常各有各的方法，各有各的优劣。下面就我现有的对单片机键盘接口的了解和应用将众家的单片机直接驱动键盘的接口原理及应用作一个总结，并附加相应键盘的汇编子程序和 C 语言子函数。希望大家可以从中受益。

本文我们以键盘的数目来选择键盘最适合的接法和最佳的编程方法，对各键盘接口的方法的优缺点加以说明。本文旨在让大家掌握一种方法，而不要读死书，举一反三创造更新更好的接口方式才是学习的极致。我最喜欢爱因斯坦的一句话：“想象力比知识更重要，想象力可以创造知识，而知识却只是别人的想象”。

1~4 按键的单片机键盘接口：

当键盘的数目最多为 4 个时，我们最佳的接口方案当然是独立式接法了，即每一个 I/O 口上只接一个按键，按键的另一端接电源或接地（一般接地）。占用的 I/O 口数最大为 4 条。（注意：1~4 按键的键盘的接法许多，如果接成扫描式可以占用更少的 I/O 口，但从程序复杂性和系统稳定性的综合考虑的话，独立式键盘接法应该是首选）

独立式键盘的实现方法是利用单片机 I/O 口读取口的电平高低来判断是否有键按下。例如，我们将常开按键的一端接地，另一端接一个 I/O 口，程序开始时将此 I/O 口置于高电平，平时无键按下时 I/O 口保护高电平。当有键按下时，此 I/O 口与地短路迫使 I/O 口为低电平。按键释放后，单片机内部的上拉电阻使 I/O 口仍然保持高电平。我们所要做的就是在程序中查寻此 I/O 口的电平状态就可以了解我们是否有按键动作了。

值得注意的事，我们在用单片机对键盘处理的时候涉及到了一个重要的过程，那就是键盘的去抖动。这里说的抖动是机械的抖动，是当键盘在未按到按下的临界区产生的电平不稳定正常现象，并不是我们在按键时通过注意可以避免的。这种抖动一般在 10~200 毫秒之间，这种不稳定电平的抖动时间对于人来说太快了，而对于时钟是微秒级的单片机而言则是慢长的。为了提高系统的稳定，我们必须去除或避开它。目前的技术有硬件去抖动和软件去抖动，硬件去抖动就是用部分电路对抖动部分加之处理，但是实现的难度较大又会提高了成本。软件去抖动不是去掉抖动，而是避开抖动部分的时间，等键盘稳定了再对其处理。这里我们只研究软件去抖动，实现方法是先查寻按键当有低电平出现时立即延时 10~200 毫秒以避开抖动（经典值为 20 毫秒），延时结束后再读一次 I/O 口的值，这一次的值如果为 1 表示低电平的时间不到 10~200 毫秒，视为干

扰信号。当读出的值是 0 时则表示有按键按下，调用相应的处理程序。

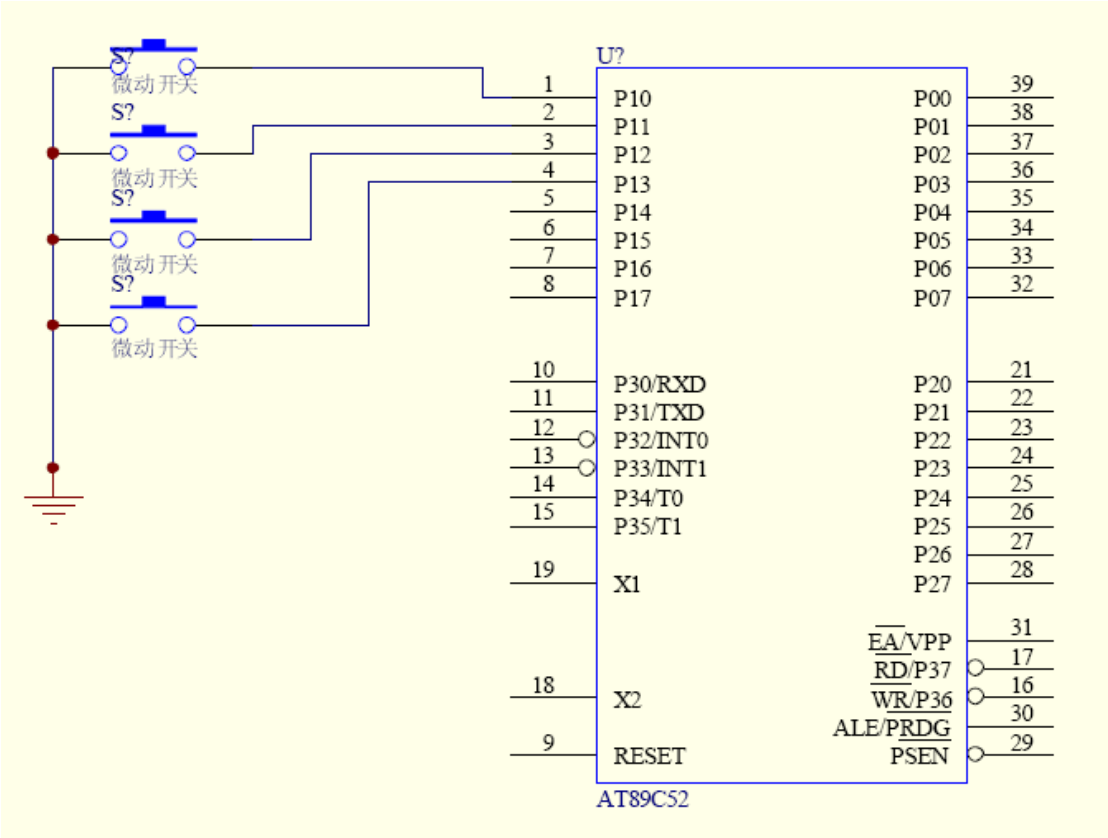

图 1

4~16 按键的单片机键盘接口：

当键盘数为 4~16 键时再用独立方法实现就得占用 4~16 的 I/O 口，显然是一种浪费。这时我们就用行列扫描式方案，行数加上列数就等于我们的按键总数，这样我们要 16 个按键时只要 8 个 I/O 口，也就是占用了一个字节 I/O 口，这种 16 键行列扫描是最常用的一种了。下面就 16 个按键的键盘为例，说一下行列键盘的扫描方法。

逐点扫描法：

设 P1 口的低 4 位置为 0，高 4 位置为 1，当无按键时 P1 口的数据始终保持 F0H 不变。当有按键时，高位中的高电平会和低位中的低电平短路。此时高位中的高电平就会被拉低（任何高电平遇到低电平都为低），即高 4 位中有 0 出现。当单片机读 P1 口的值不为 F0H 时则表示有键按下，经过去抖处理后就开始扫描这一个键的位置了。先把 P1 的高 4 位置 0，低 4 位置 1，下面置低第 1 列线并置高第 1 行，之后再读第 1 行的电平，为高说明不是第 1 行第 1 列的键被按下，跳到下一个点的扫描（第 1 列第 2 列）。当其为 0 时表示第 1 行第 1 列的键被按下，调用相应处理程序。

逐点扫描的优点是它的编程简单，易于理解，可以作同时按多个键的识别。缺点是它的速度慢，处理程序代码较长。

逐行扫描法：

和逐点扫描的方法相似，只是数据的处理是以一行的 4 位数据直接处理，也就是先使能第 1 行（置低电平），然后看哪一列的数据变成低电平了，如果高 4 位数据没有变为低电平则使能下一行。找到了按键所在的行并测出列数据就可以调用相应处理程序。

逐行扫描的照逐点扫描要好的多，程序相对简单，速度快，也支持同时多按键处理。一般的扫描键盘多用此法。

全局扫描法：

全局扫描是只先设 P1 的高 4 位为 1，低 4 位为 0，即 F0H，然后读取 P1 口的数据如果不为 F0H 说明有键按下，经过延时去抖处理后读出 P1 口的值，因为低 4 位是 0 无论按键如果都不会影响它，只有高 4 位被改变。将数据寄存起来后再把 P1 的状态反过来，将 P1 的高 4 位置 0，低 4 位置 1，即 0FH，再读一次数据。这时高 4 位的值是 0 依然不变。这样两次读取我们就得到了 2 个字节的数据（XXXX0000 和 0000XXXX，X 为读到的数据），最后我们将这两个数相或（将两个半字节数据溶合为一个字节），就得到了一个新的字节，用这个字节和我们的设定的数据比较来决定键值。

全局扫描只用两次扫描，速度快，易学易用，程序简单，可是它不支持同时多键处理，最佳适用 4*4 扫描键盘，可以用在一般的用途。

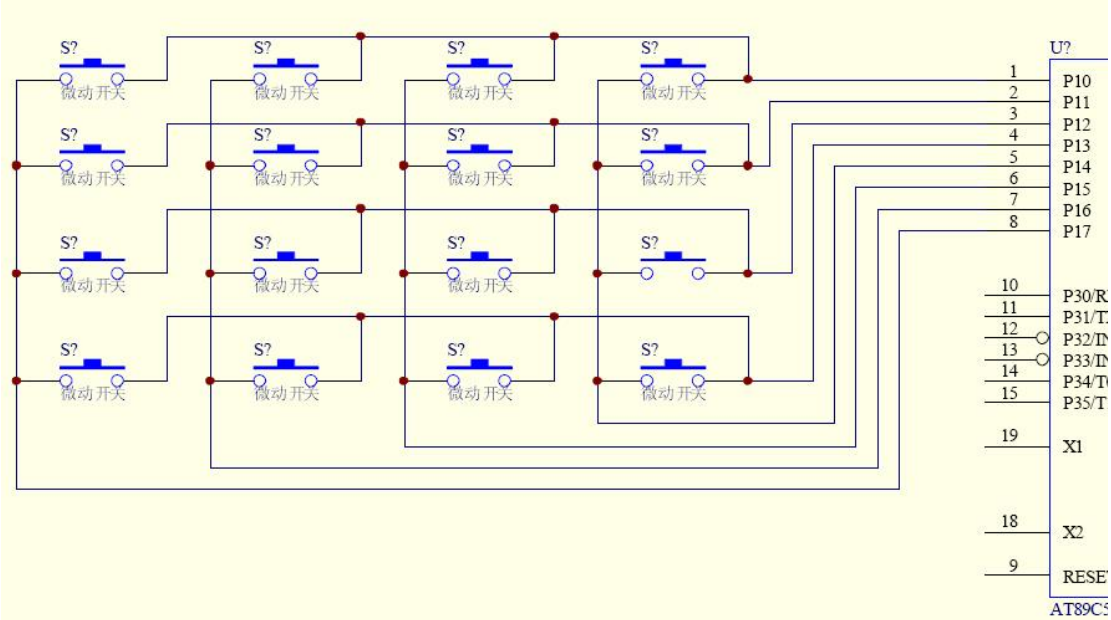

图 2

大于 16 按键的单片机键盘接口：

当按键数目大于 16 时用行列扫描方式驱动是比较常用的，增加 I/O 口来达到行列扫描的键盘数，扫描方法多为逐行扫描，这里不多介绍。下面来了解一个用 8 个 I/O 口接 36 个按键的方法（参考《电子制作》杂志 139 期的“扩展单片机输入键的另一种方法”）。

它采用了独特的硬件接法和逐点扫描的方法完成对 36 个按键的扫描，是一种很好的节省硬件资源的方法。8 个 I/O 口的任意两个都有一个按键的同时，其各 I/O 口对地也都有一个按键，可以说是一个非常有新意的方法。（详见原文）

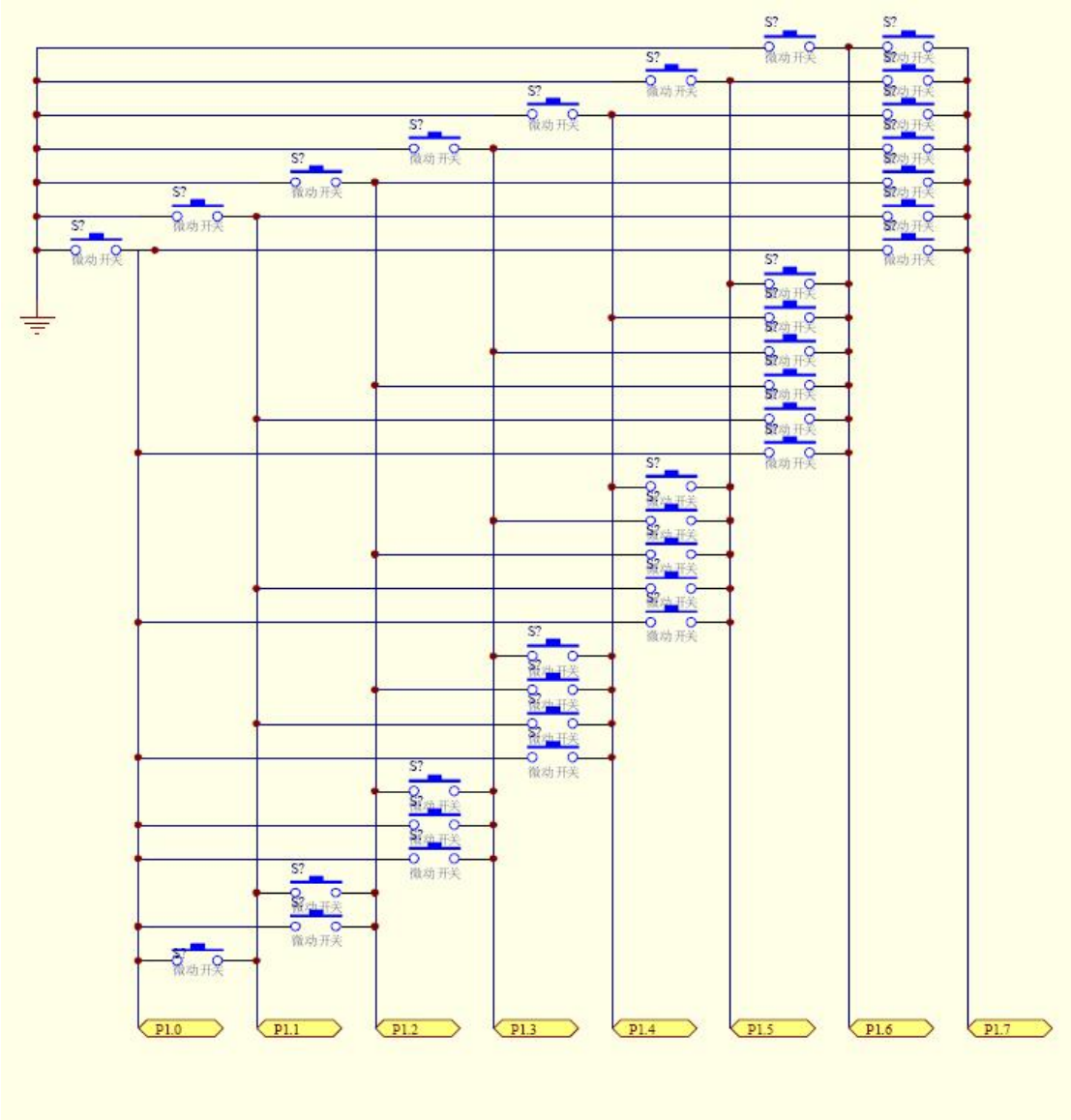

图 3

1-1、 独立键盘接口程序：

```
ORG   0000H
JMP    MAIN      程序开始，从 MAIN 执行
ORG   0030H
MAIN:
      JNB    P1.0, KEY1      查寻 P1.0 接的独立键盘是否有键接下，为低电平按下。
      JNB    P1.1, KEY2      同上
      JNB    P1.2, KEY3      同上
      ……
      JMP    MAIN            查寻键盘的循环体
KEY1:
       CALL   DL20MS       调用延时 20 毫秒，去除机械抖动
       JB     P1.0, MAIN     重新查寻，亦去抖。
      CALL   PRO1            确认有键按下，调用真正处理程序
      JMP    MAIN            执行结束，返回查寻循环
KEY2:
       CALL   DL20MS       同上
       JB     P1.1, MAIN
      CALL   PRO2
      JMP     MAIN
 …….
DL20MS:                        20 毫秒延时子程序，去抖用。
      MOV    R6,#100
DL20MS_1:
      MOV    R7,#100
      DJNZ   R7,$
      DJNZ   R6, DL20MS_1
      RET
```

1-2、独立式键盘接口程序：
键盘接在 P1 口，P2 口 LED 显示

```
ORG 0000H
AJMP MAIN                    真正的程序入口
ORG 030H
MAIN:MOV P2,#0FFH            显示初始化
MOV A,#0FFH                  设定累加器值
MOV P1,A                     P1 口全置“1”
MOV A,P1                     将 P1 口键盘值送入 A
CJNE A,#0FFH,GO1 查键值      查看 A 中是否有键按下，有键则跳到 GO1
AJMP MAIN                    P1 口全为“1”表示无键，返回循环查寻程序
GO1:ACALL DEL 延时去抖动     P1 口中有“0”出现，则延时去抖动。
CJNE A,#0FFH,GO2             再次查寻
AJMP MAIN                    为干扰则返回循环程序
GO2:MOV DPTR,#TAB            DPTR 装入散转表的首址
MOV R0,#00H 键值查寻         R0 初始化，以备累加
L1: RRC A                    累加器带 C 右移
JNC N1                       查每一个位，如果为 0 则跳出
INC R0                       键值寄存器
SJMP L1                      此位不为“0”继续查下一位
N1:MOV A,R0                  取得键值
RLC A                        乘 3 处理，用于散转
JMP @A+DPTR                  散转
TAB:AJMP PR0                 散转表
AJMP PR1
AJMP PR2
AJMP PR3
AJMP PR4
AJMP PR5
AJMP PR6
AJMP PR7
PR0:MOV P2,#0FEH             真正处理程序
ACALL DEL
AJMP MAIN
PR1:MOV P2,#0FCH
ACALL DEL
AJMP MAIN
PR2:MOV P2,#0F8H
ACALL DEL
AJMP MAIN
PR3:MOV P2,#0F0H
```

```
    ACALL DEL
    AJMP MAIN
    PR4:MOV P2,#0E0H
    ACALL DEL
    AJMP MAIN
    PR5:MOV P2,#0C0H
    ACALL DEL
    AJMP MAIN
    PR6:MOV P2,#80H
    ACALL DEL
    AJMP MAIN
    PR7:MOVP2,#00H
    ACALL DEL
    AJMP MAIN
    DEL:MOV R7,#0FFH          延时程序
    DEL1:MOV R6,#0FFH
    DEL2:DJNZ R6,DEL2
    DJNZ R7,DEL1
    RET
```

逐行扫描程序：

```
;===============;4*4 行列键盘程序;=================;
;
;       键盘接 P3 口，高 4 位接 5.1K 上拉电阻
;       用 2 号实验板完成，在按各键时，从 LED 屏上显示 0~F.
;       dy 20050511pm
;================================================;


ORG 0000H
JMP MAIN
ORG 30H

MAIN:;----------------------------------------------------初始化定义
MOV P2,#0H
MOV P0,#00H
MOV R0,#00H
MOV R1,#00H
START:;---------------------------------------------------扫描程序开始
MOV P3,#0FH
MOV A,P3

CJNE A,#0FH,GO1                 ;是否有键接下
JMP START                       ;无键时返回

GO1:
CALL    DL20MS                  ;延时去抖
CJNE    A,#0FH,GO2              ;再次查寻
JMP    START                    ;为干扰则返回

GO2:
MOV R2,#0EFH                     ;行列扫描初始化
MOV R0,#00H

ST:
MOV P3,R2                       ;送入首列使能信号
MOV A,P3                        ;得到数据送入 A
JB ACC.0,ONE                    ;行读键值，为“0”表示有键
MOV A,#01H
JMP LKP
```

```
ONE:
JB ACC.1,TWO                    ;同上
MOV A,#05H
JMP LKP

TWO:
JB ACC.2,THR
MOV A,#09H
JMP LKP

THR:
JB ACC.3,NEXT                   ; 如行无键时，则使能下一列
MOV A,#0DH

LKP:
ADD A,R0                        ; 得到键值处理
CJNE A,#0FH,LKP1
MOV A,#00H

LKP1:
MOV R1,A                        ;将键值送入 R1 并退出
JMP START

NEXT:                           ;列使能控制
INC R0
CJNE R0,#04H,L1
MOV R0,#00H
JMP ST

L1:                             ;列初值键码
CJNE R2,#0EFH,L2
MOV R2,#0DFH
JMP ST

L2:
CJNE R2,#0DFH,L3
MOV R2,#0BFH
JMP ST

L3:
CJNE R2,#0BFH,L4
MOV R2,#7FH
JMP ST
```

```
L4:
CJNE R2,#7FH,RE
MOV R2,#0EFH

RE:                                    ;未扫描到键值则返回
JMP START


TAB_TT:;-------------------------------------------------散转表处理
JMP PPY9
JMP PPY00
JMP PPY15
JMP PPY14
JMP PPY13
JMP PPY1
JMP PPY4
JMP PPY7
JMP PPY10
JMP PPY2
JMP PPY5
JMP PPY8
JMP PPY0
JMP PPY3
JMP PPY6
JMP PPY9
JMP PPY11

RET

PPY:                                   ;真正处理程序
PPY0:MOV P0,#0C0H
LJMP START

PPY1:MOV P0,#0F9H
LJMP START

PPY2:MOV P0,#0A4H
LJMP START

PPY3:MOV P0,#0B0H
LJMP START
.......
PPY15:MOV P0,#8EH
```

```
LJMP START

PPY00:MOV P0,#0FFH

LJMP START


DL20MS: MOV     R6,#100;----------------------延时子程序
        DL1:    MOV     R7,#100
        DL2:    DJNZ    R7,DL2
                DJNZ    R6,DL1
                RET

END
```

==END==