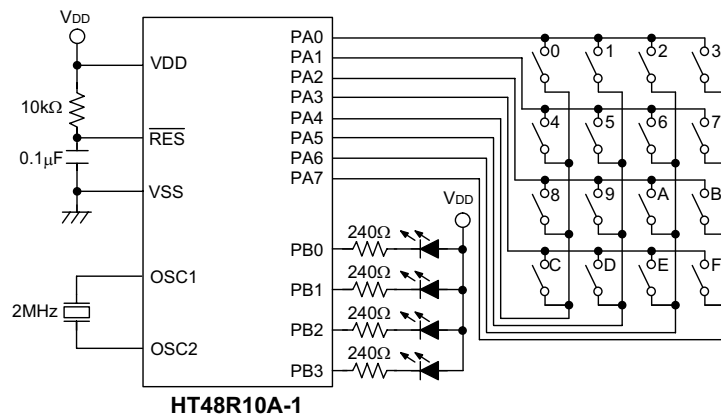


# HT48 & HT46 Keyboard Scan Program

D/N : HA0011E

## Introduction

The keyboard application used here is a 4x4 keyboard array with 16 keys, each one having its own hexadecimal code. The keyboard scan program scans the rows and columns of the keyboard array to determine which key is pressed, and then shows its binary code value on the LEDs. The four LEDs display a value from 0000H to 1111H. During the scan process, if two keys are pressed simultaneously, only the key which is scanned first, will be shown. This kind of coding keyboard can designate a figure to the key.



## Circuit

Set PA0~PA3 as outputs and PA4~PA7 as inputs to make the 4x4 array. The program scans which key is pressed and looks up its pre-defined value. Define PB0~PB3 as output ports that output 4 bit binary codes. Each of the 16 figures corresponds to a key.

### Scan Process

Take the first line and first row as an example. First output FEH to PA to scan the first line. If any key on this line is pressed, the high byte PA key figure that is read will not return a value of F. If the returned value is not F due to a pressed key than a row scan will begin. If one key is pressed in the first line, PA.4 will be 0, otherwise it will be 1. In this way, those keys in lines and rows that are pressed can be verified and their corresponding codes determined.

### Program Example

```
#include ht48r10a-1.inc
;-----data
data .section 'datat' ;data section
    temp      db  ?      ;temporary register
    temp2     db  ?      ;keyboard scan code to detect the row value
    disp      db  ?      ;display register for key value
    count1    db  ?      ;delay accumulator
    mask      db  ?      ;mask register
    matrix    db  ?      ;keyboard array register
;-----code
code .section at 0 'code' ;program section
    org 00h
    jmp start
start: ;program start
    clr pbc ;set PB as an output port
    mov a,0f0h ;(1) ;set PA highest 4 bits as inputs, lowest 4
bits as outputs
    mov pac,a
    clr pa ;clear PA
    clr pb ;clear PB
keyloop: ;keyboard scan loop
    mov a,0feh ;(2) ;scan if line 1 is pressed
    mov matrix,a ;send line 1 codes to matrix
    mov pa,a ;output scan code to PA
    mov a,pa ;read PA status to ACC
    xor a,0feh ;check if highest 4 bits are 0 or not,
;if yes, key
;pressed and ACC will change
    sz acc ;check if a key in line 1 is pressed; if yes,
;ACC will not be 0
    jmp get_key ;key pressed, jump to read the key code
    mov a,0fdh ;(2) ;scan if line 2 is pressed
    mov matrix,a ;send line 2 code to matrix
    mov pa,a
    mov a,pa
    xor a,0fdh
```

```

sz    acc
jmp   get_key
mov   a,0fbh    ;(2)    ;scan if line 3 is pressed
mov   matrix,a    ;send line 3 code to matrix
mov   pa,a
mov   a,pa
xor   a,0fbh
sz    acc
jmp   get_key
mov   a,0f7h    ;(2)    ;scan if line 4 is pressed
mov   matrix,a    ;send line 4 code to matrix
mov   pa,a
mov   a,pa
xor   a,0f7h
sz    acc
jmp   get_key
jmp   keyloop    ;jump back to key loop scan
get_key:    ;read the key value
call  key_I    ;(3)    ;call key_in subprogram
mov   pb,a    ;(11)   ;show key value from PB
jmp   keyloop    ;jump back to key loop scan
key_in proc    ;read key value to subprogram
mov   a,pa    ;read PA data
mov   temp,a    ;(4)    ;read PA status into temp register
mov   temp2,a    ;send scan value to temp2 to detect row value
call  delays    ;(5)   ;call delay subprogram
get_release:    ;wait for key release
mov   a,pa    ;send PA main status to ACC
and   a,0f0h    ;mask ACC high 4 bit, read key status
sz    acc    ;(6)    ;wait for key release, if released acc=0
jmp   get_release
mov   a,0fh    ;read mask register low 4 bit
andm  a,matrix
mov   a,0
get_row:    ;read lines
rrc   matrix    ;(7)   ;move matrix pointer to the right
sz    status.0    ;check and read the key line
jmp   get_column1    ;if key line is found jump to get_next
clr   c    ;if key line is not found, clear carry_c
add   a,4h    ;(8)   ;add 4 to display pointer
jmp   get_row    ;jump back to get_row
get_colmn1:
mov   temp,a
mov   a,0f0h
andm  a,temp2    ;read keyboard scan high 4 bit code and
                ;detect the row value
swap  temp2    ;exchange, put the low bit to the row value
mov   a,0h    ;(9)
get_column:

```

```

        rrc  temp2           ;check one by one until the byte shows 0
        snz  status.0       ;if 0, some key is pressed in that row
        jmp  next
        clr  c
        add  a,1h
        jmp  get_column     ;read row value
next:
        add  a,temp
        xor  a,0ffh        ;count the key value and read display code
key_in endp
delays proc                ;delay subroutine
        mov  a,0ffh
        mov  count1,a
d1:
        sdz  count1
        jmp  d1
        ret
delays endp

```

### Program Instruction

Section ( 1 ) defines which ports are inputs and outputs. The program loop scans to see if there is any key that is pressed. There being four keys on the same line, the program should tell which key on that line is pressed In section (2), 1-3 keys of the first line are pressed and then the program jumps to the loop scanning and jumps to section ( 3 )code to determine which key on that line is pressed. If no key is pressed in line 1, the program will move on and look at keys 4-7 and so on. When the program enters this section code, it will save the key status in a register (4)and go with a time delay period (5). The codes will detect if the keys are released ( 6 ) and the program will loop until no keys are pressed. The next program section will confirm which key on which line is pressed (7). Scanning is controlled by a 4 bit binary code program that will be executed line by line (8). Once the key position is read, line x 4 value plus the key position will confirm the key value (9). To find the actual key value, run the procedures from line to row. The look up instructions in (1) are based on key values to show their display codes and then sent to PB to show the value of the pressed key on the LEDs.