ECE3120: Computer Systems Chapter 7: Interfacing with I/P Devices

Manjeera Jeedigunta
http://blogs.cae.tntech.edu/msjeedigun21
Email: msjeedigun21@tntech.edu
Tel: 931-372-6181, Prescott Hall 120

- Prev
  - Interfacing with Switches
- Today
  - Interfacing with Keypad

Interfacing to a Keyboard

- A keyboard is arranged as an array of switches,
  - mechanical
  - membrane
  - capacitors
  - Hall-effect in construction.
- Mechanical switches are most popular for keyboards.
  - Mechanical switches have a problem called contact bounce. Closing a mechanical switch generates a series of pulses because the switch contacts do not come to rest immediately.
  - In addition, a human cannot type more than 50 keys in a second. Reading the keyboard more than 50 times a second will read the same key stroke too many times.

Keypad Input Process

- A keyboard input is divided into three steps:

- Scan the keyboard to discover which key has been pressed

  - Debounce the keyboard to determine if a key is indeed pressed. Both hardware and software approaches for key debouncing are available.

  - Lookup the ASCII table to find out the ASCII code of the pressed key.

Keypad Scanning

- PA7~PA4 → O/P, Row selection, row being [(0,1,2,3),(4,5,6,7)..]
- Row being scanned is driven low→ either one of PA7~PA4=0
- PA3~PA0→I/P, Decide which key is pressed
  - Initially High , when pressed the corr row and column will be shorted
  - When pressed the corresponding PA Pin would be low

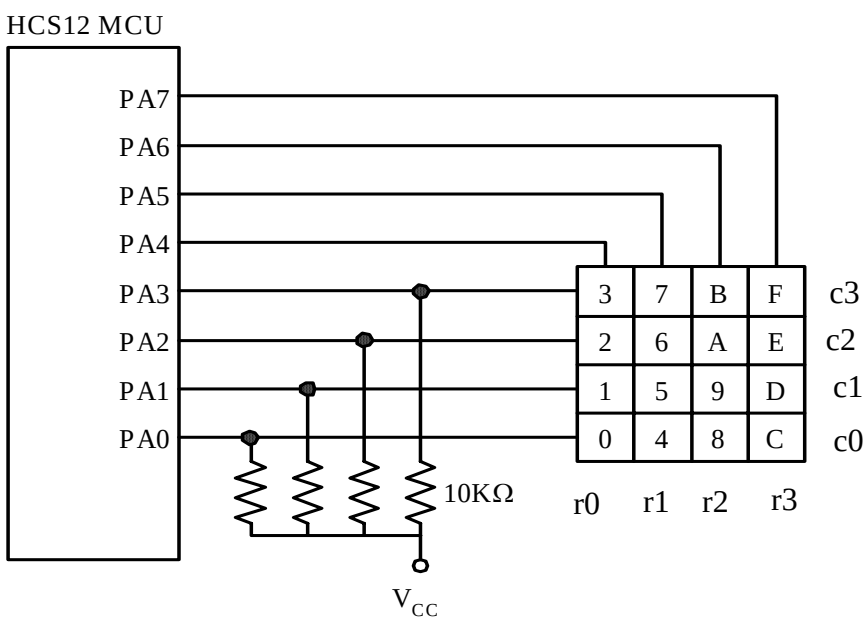

Figure 7.41 Sixteen-key keypad connected to the HCS12

PA7	PA6	PA5	PA4	Selected keys
1	1	1	0	0, 1, 2, and 3
1	1	0	1	4, 5, 6, and 7
1	0	1	1	8, 9, A, and B
0	1	1	1	C, D, E, and F

Table 7.16 Sixteen-key keypad row selections

Hardware Debouncing Techniques

Debouncer will recognize that the switch is closed after the voltage is low for around 10ms and that the switch is open after the voltage is high for about 10 ms

- SR latches
- Non-inverting CMOS gates
- Integrating debouncer

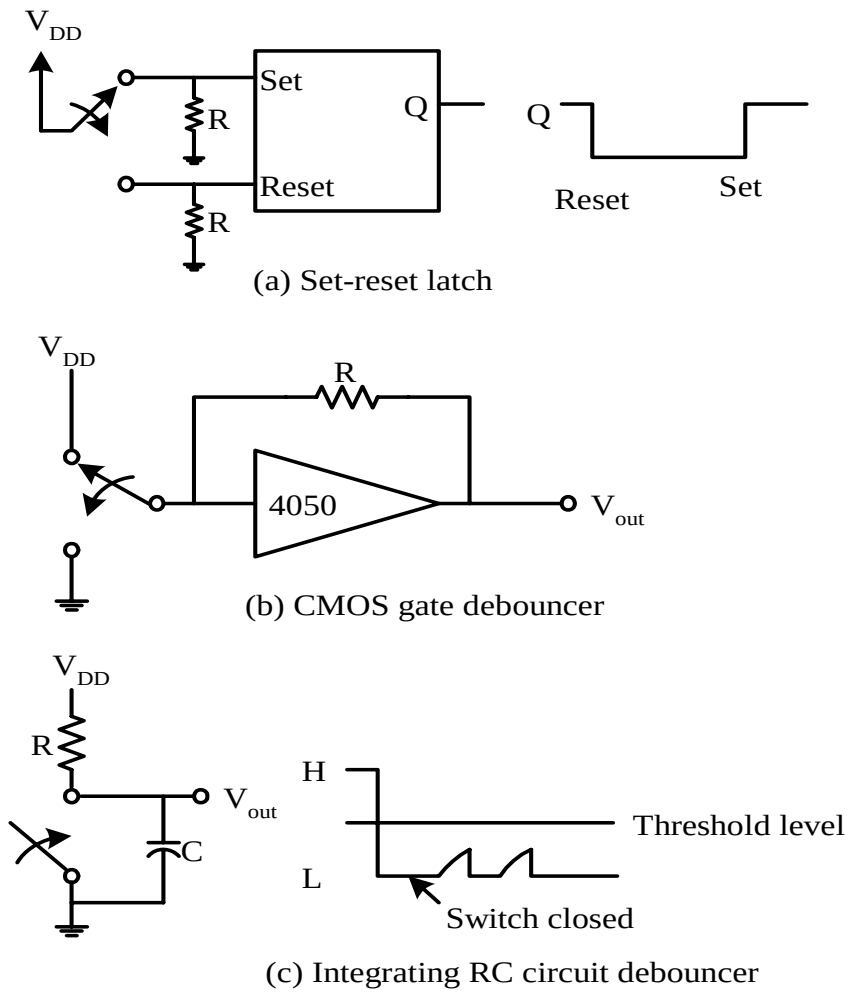

Figure 7.42 Hardware debouncing techniques

Software Debouncing Technique

- The most popular and simple one has been the **wait and see** method.
  - In this method, the program simply waits for about 10 ms and reexamines the same key again to see if it is still pressed.

- Example 7.10 Write a program to perform keypad scanning, debouncing, and returns the ASCII code in accumulator A to the caller.
- Solution
  - Pins PA4..PA7 each control one row of four keys.
  - Scanning is performed by setting one of the PA7…PA4 pins to low, the other three pins to high and testing one key at a time.

```
#include   "c:\miniide\hcs12.inc"
keyboard equ     PTA

get_char  movb   #$F0,DDRA            ; set PA7~PA4 for output, PA3~PA0 for input
scan_r0   movb   #$EF,keyboard        ; scan the row containing keys 0123
scan_k0   brclr  keyboard,$01,key0    ; is key 0 pressed?
scan_k1   brclr  keyboard,$02,key1    ; is key 1 pressed?
scan_k2   brclr  keyboard,$04,key2    ; is key 2 pressed?
scan_k3   brclr  keyboard,$08,key3    ; is key 3 pressed?
          bra    scan_r1
key0      jmp    db_key0
key1      jmp    db_key1
```

```
key2      jmp     db_key2
key3      jmp     db_key3
scan_r1   movb    #$DF,keyboard       ; scan the row containing keys 4567
scan_k4   brclr   keyboard,$01,key4   ; is key 4 pressed?
scan_k5   brclr   keyboard,$02,key5   ; is key 5 pressed?
scan_k6   brclr   keyboard,$04,key6   ; is key 6 pressed?
scan_k7   brclr   keyboard,$08,key7   ; is key 7 pressed?
          bra     scan_r2
key4      jmp     db_key4
key5      jmp     db_key5
key6      jmp     db_key6
key7      jmp     db_key7
scan_r2   movb    #$BF,keyboard       ; scan the row containing keys 89AB
          bclr    keyboard,$40        ;  “
scan_k8   brclr   keyboard,$01,key8   ; is key 8 pressed?
scan_k9   brclr   keyboard,$02,key9   ; is key 9 pressed?
scan_kA   brclr   keyboard,$04,keyA   ; is key A pressed?
scan_kB   brclr   keyboard,$08,keyB   ; is key B pressed?
          bra     scan_r3
key8      jmp     db_key8
key9      jmp     db_key9
```

```
keyA      jmp     db_keyA
keyB      jmp     db_keyB
scan_r3   movb    #$7F,keyboard       ; scan the row containing keys CDEF
scan_kC   brclr   keyboard,$01,keyC   ; is key C pressed?
scan_kD   brclr   keyboard,$02,keyD   ; is key D pressed?
scan_kE   brclr   keyboard,$04,keyE   ; is key E pressed?
scan_kF   brclr   keyboard,$08,keyF   ; is key F pressed?
          jmp     scan_r0
keyC      jmp     db_keyC
keyD      jmp     db_keyD
keyE      jmp     db_keyE
keyF      jmp     db_keyF
; debounce key 0
db_key0   jsr     delay10ms
          brclr   keyboard,$01,getc0
          jmp     scan_k1
getc0     ldaa    #$30                ; return the ASCII code of 0
          rts
; debounce key 1
```

```
db_key1  jsr     delay10ms
         brclr   keyboard,$02,getc1
         jmp     scan_k2
getc1    ldaa    #$31                ; return the ASCII code of 1
         rts
db_key2  jsr     delay10ms
         brclr   keyboard,$04,getc2
         jmp     scan_k3
getc2    ldaa    #$32                ; return the ASCII code of 2
         rts
db_key3  jsr     delay10ms
         brclr   keyboard,$08,getc3
         jmp     scan_r1
getc3    ldaa    #$33                ; return the ASCII code of 3
         rts
db_key4  jsr     delay10ms
         brclr   keyboard,$01,getc4
```

```
        jmp     scan_k5
getc4   ldaa    #$34            ; return the ASCII code of 4
        rts
db_key5 jsr     delay10ms
        brclr   keyboard,$02,getc5
        jmp     scan_k6
getc5   ldaa    #$35            ; return the ASCII code of 5
        rts
db_key6 jsr     delay10ms
        brclr   keyboard,$04,getc6
        jmp     scan_k7
getc6   ldaa    #$36            ; return the ASCII code of 6
        rts
db_key7 jsr     delay10ms
        brclr   keyboard,$08,getc7
        jmp     scan_r2
```

```
getc7     ldaa    #$37                ; return the ASCII code of 7
          rts
db_key8   jsr     delay10ms
          brclr   keyboard,$01,getc8
          jmp     scan_k9
getc8     ldaa    #$38                ; return the ASCII code of 8
          rts
db_key9   jsr     delay10ms
          brclr   keyboard,$02,getc9
          jmp     scan_kA
getc9     ldaa    #$39                ; return the ASCII code of 9
          rts
db_keyA   jsr     delay10ms
          brclr   keyboard,$04,getcA
          jmp     scan_kB
getcA     ldaa    #$41                ; get the ASCII code of A
          rts
db_keyB   jsr     delay10ms
          brclr   keyboard,$08,getcB
          jmp     scan_r3
getcB     ldaa    #$42                ; get the ASCII code of B
          rts
```

```
db_keyC   jsr     delay10ms
          brclr   keyboard,$01,getcC
          jmp     scan_kD
getcC     ldaa    #$43              ; get the ASCII code of C
          rts
db_keyD   jsr     delay10ms
          brclr   keyboard,$02,getcD
          jmp     scan_kE
getcD     ldaa    #$44              ; get the ASCII code of D
          rts
db_keyE   jsr     delay10ms
          brclr   keyboard,$04,getcE
          jmp     scan_kF
getcE     ldaa    #$45              ; get the ASCII code of E
          rts
db_keyF   jsr     delay10ms
          brclr   keyboard,$08,getcF
          jmp     scan_r0
getcF     ldaa    #$46              ; get the ASCII code of F
          rts
```

```
delay10ms   movb    #$90,TSCR1          ; enable TCNT & fast flags clear
            movb    #$06,TSCR2          ; configure prescale factor to 64
            movb    #$01,TIOS           ; enable OC0
            ldd     TCNT
            addd    #3750               ; start an output compare operation
            std     TC0                 ; with 10 ms time delay
wait_lp2    brclr   TFLG1,$01,wait_lp2
            rts
```

Next…

- Interfacing with LCD
- Time-Multiplexing
- Class in BN 320 on Friday Nov 7th
- Read Chapter 7.6