ECE3120: Computer Systems
Chapter 7: Interfacing with O/P devices

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

Today
- Interfacing with LEDs
- Interfacing with Seven-Segment Display
Figure 7.29 suggests three methods for interfacing with LEDs.

- Circuit (a) and (b) are recommended for LEDs that need only small current to light.
- Circuit (c) is recommended for LEDs that need larger current to light.
- Port J, Pin 1 is used to enable LEDs in case of our demo board.

![Illustration of LED interfacing methods](image-url)
Example 7.3 Use Port B to drive eight LEDs using the circuit shown in Figure 7.30. Light each LED for half a second in turn and repeat assuming the HCS12 has a 24-MHz E clock.

To turn on one LED at a time for half a second in turn, one should output the value $80, $40, $20, $10, $08,$04,$02, and $01 and stay on for half a second in each value.
The assembly program that performs the operation is as follows:

```assembly
#include "C:\miniide\hcs12.inc"

led_tab  dc.b $80,$40,$20,$10,$08,$04,$02,$01
         dc.b $01,$02,$04,$08,$10,$20,$40,$80
org      $1500
movb     #$FF,DDRB   ; configure port B for output
bset     DDRJ,$02    ; configure PJ1 pin for output
bclr     PTJ,$02    ; enable LEDs to light
forever  ldaa        #16        ; initialize loop count to 16
          ldx        #led_tab   ; use X as the pointer to LED pattern table
          movb        1,x+,PTB   ; turn on one LED
          ldy        #5         ; wait for half a second
          jsr        delayby100ms ;
          dbne        a,led_lp  ; reach the end of the table yet?
          bra        forever   ; start from beginning
#include "C:\miniide\delay.asm"
end
```
Driving a Single Seven-Segment Display

- A common cathode seven-segment display is driven by the 74HC244 via resistors.
- The output high voltage of the 74HC244 is close to 5V with a 5V power supply.
- The segment patterns for 0 to 9 are shown in Table 7.5.

Table 7.5 BCD to seven-segment decoder

<table>
<thead>
<tr>
<th>BCD digit</th>
<th>Segments</th>
<th>Corresponding Hex Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 1 1 1 1 1 0</td>
<td>$7E</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 0 0 0</td>
<td>$30</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 1 1 0 1</td>
<td>$6D</td>
</tr>
<tr>
<td>3</td>
<td>1 1 1 1 0 0 1</td>
<td>$79</td>
</tr>
<tr>
<td>4</td>
<td>0 1 1 0 0 1 1</td>
<td>$33</td>
</tr>
<tr>
<td>5</td>
<td>1 0 1 1 0 1 1</td>
<td>$5B</td>
</tr>
<tr>
<td>6</td>
<td>1 0 1 1 1 1 1</td>
<td>$5F</td>
</tr>
<tr>
<td>7</td>
<td>1 1 1 0 0 0 0</td>
<td>$70</td>
</tr>
<tr>
<td>8</td>
<td>1 1 1 1 1 1 1</td>
<td>$7F</td>
</tr>
<tr>
<td>9</td>
<td>1 1 1 1 0 1 1</td>
<td>$7B</td>
</tr>
</tbody>
</table>

Figure 7.31 Driving a single seven-segment display
Driving Multiple Seven-Segment Displays

- Time multiplexing technique is often used to drive multiple displays in order to save I/O pins.

- One parallel port is used to drive the segment pattern (B) and the other port turns on one display at a time (K).

Figure 7.32 Port B and Port K together drive six seven-segment displays (MC9S12DP256)
Example 7.4 Write a sequence of instructions to display 4 on the seven-segment display #4 in Figure 7.32.

Solution: To display the digit 4 on the display #4, we need to:
- Output the hex value $33 to port B
- Set the PK4 pin to 1
- Clear pins PK5 and PK3...P0 to 0

```c
#include <hcs12.inc>

four equ $33 ; seven-segment pattern of digit 4
movb #$3F,DDRK ; configure PORT K for output
movb #$FF,DDRB ; configure PORT B for output
bset PTK,$10 ; turn on seven-segment display #4
bclr PTK,$2F ; turn off seven-segment displays #5, #3...#0
movb #four,PTB ; output the seven-segment pattern to PORTP
```
Example 7.5 Write a program to display 123456 on the six seven-segment displays shown in Figure 7.32.

Solution: Display 123456 on display #5, #4, #3, #2, #1, and #0, respectively.

The values to be output to Port B and Port K to display one digit at a time is shown in Table 7.6.

<table>
<thead>
<tr>
<th>seven-segment display</th>
<th>displayed BCD digit</th>
<th>Port B</th>
<th>Port K</th>
</tr>
</thead>
<tbody>
<tr>
<td>#5</td>
<td>1</td>
<td>$30</td>
<td>$20</td>
</tr>
<tr>
<td>#4</td>
<td>2</td>
<td>$6D</td>
<td>$10</td>
</tr>
<tr>
<td>#3</td>
<td>3</td>
<td>$79</td>
<td>$08</td>
</tr>
<tr>
<td>#2</td>
<td>4</td>
<td>$33</td>
<td>$04</td>
</tr>
<tr>
<td>#1</td>
<td>5</td>
<td>$5B</td>
<td>$02</td>
</tr>
<tr>
<td>#0</td>
<td>6</td>
<td>$5F</td>
<td>$01</td>
</tr>
</tbody>
</table>

- The program logic is shown in Figure 7.33.
Figure 7.33 Time-multiplexed seven-segment display algorithm
```assembly
#include "c:\miniide\hcs12.inc"

pat_port  equ PTB       ; Port that drives the segment pattern
pat_dir   equ DDRB      ; direction register of the segment pattern
sel_port  equ PTK       ; Port that selects the digit
sel_dir   equ DDRK      ; data direction register of the digit select port
org       $1500
movb      #$FF,pat_dir  ; configure pattern port for output
movb      #$3F,sel_dir  ; configure digit select port for output
forever   ldx #disp_tab ; use X as the pointer
loop      movb 1,x+,pat_port ; output digit pattern and move the pointer
          movb 1,x+,sel_port ; output digit select value and move the pointer
          ldy #1           ; wait for 1 ms
          jsr delayby1ms  ; 
          cpx #disp_tab+12 ; reach the end of the table
          bne loop
          bra forever
#include "c:\miniide\delay.asm"
disp_tab  dc.b  $30,$20 ; seven-segment display table
          dc.b  $6D,$10
          dc.b  $79,$08
          dc.b  $33,$04
          dc.b  $5B,$02
          dc.b  $5F,$01
end
```
Next…

- Interfacing to LCD
- Read Chapter 7.6