EE260 – Lab 6

The objective of this lab is to learn:

1. how to use the timing system
2. how to drive multiple 7-segment LED displays using a single 8-bit output port.

In order to drive multiple 7-segment displays with only one output port, each display must be driven in turns. The frequency at which each display is driven must be higher than the eye latency (40Hz). In this way the eye does not notice the displays are not driven simultaneously. The timing subsystem allows managing the amount of time each display is driven.

Pre-Lab:
Find out the address and the meaning of each bit of the following registers: TOC2, TCTL1, TCNT, TMSK1, TMSK2, TFLG1, PORTB, PORTD, DDRD.
Read and understand the attached assembly program: 7seg.asm.

Lab:

a. Write a C program that display the word “done” on the four 7-segment displays of the Wytec board. Use the Timing subsystem to manage the amount of time each display is driven.
b. The 7 segments are driven by portB, while the individual displays are enabled by portD.
c. Write a report documenting your design. The report should have the following sections:
   • Objectives/Overview
   • Background Information
   • Design Description
   • Testing Strategy
   • Lab. Procedures
   • Conclusions
   • Appendix (attach your code, and the symbol table)

Hint:
Use the output capture interrupt mechanism to manage the amount of time each display is driven.
* 7seg.asm -- Program illustrating the use of 7-seg LED displays and of the
*    time capture interrupt mechanisms
*    
*    Function: Display the word "done" on the 7 segment LED displays
*    The 7-segment displays are multiplexed at 5ms rate per digit,
*    which is equivalent to a 50Hz (1/50 = 20ms) rate for 4 digits
*    
*    Hardware: the Wytec evplus2 board provides:
*    Four common cathod 7(+ decimal point) seg. LED displays
*    The LED segments are driven by portB
*    The display enables are driven by portD
*    LED segments to portB map:
*    dp,g,f,e,d,c,b,a  <=> PB7,PB6,PB5,PB4,PB3,PB2,PB1,PB0
*    Enables to portD map:
*    digit 3, digit 2, digit 1, digit 0  <=> PD5,PD4,PD3,PD2

vtoc2    equ $ffe6
vres     equ $fffe
main     equ $9000

stack    equ $FF

tb5ms:   equ 10000 ; time base of 10,000 clock cycles
*         ; T = clock cycle = 4*(1/8MHz) = 0.5 us
*         ; 10,000 x 0.5us = 5ms
regbase  equ $1000
portb    equ $04
portd    equ $08
ddrd     equ $09 ; portd direction data register (input=0, out=1)
toc2:    equ $18 ; portd output capture register
*         ; when the value in TCNT register (Timer count)
*         ; "reach" the value programmed in the TOC2
*         ; register the OC2 flag bit in TFLG1 register
*         ; is set.
*         ; At this point if the OC2 arm bit in the TMSK1
*         ; register is set an interrupt is generated

tctl1:   equ $20 ; timer control resigter 1
*         ; set action to take on the OC pins as a result of a
*         ; successful compare

tmsk1:   equ $22 ; timer mask register 1

tflg1:   equ $23 ; timer interrupt flag register 1

tcnt:    equ $0e  ; timer counter register (not used)

BIT6     equ $40 ; bit 6

org      $40
*
select:  rmb 1

time:    fcb 0 ; time flag, (initialized to 0)
disp_data: rmb 4 ; data to show on the four displays

org      main ; program code starts here

start:
lhs     #stack ; initialize the STACK bottom
ldx     #regbase
*    configuration ritual
sei ; temporary disable interrupts
ldaa #$3f ; disable 7-segment displays
* portD has only 6 bits, the 2 msb
  ; don’t matter
staa portd,x
lda #3c       ; 0 = input, 1 output
            ; ddrd has only 6 bits, the 2 msb
            ; don’t matter
staa ddrd,x
lda #00       ; do not take any action on the OC pins as a
            ; result of a successful output compare event
staa tctl1,x
lda #BIT6     ; arm OC2 interrupt
staa tmsk1,x
cli            ; enable interrupts back
* end of configuration ritual

begin: ldab #%01011110 ; binary code for the letter 'd'
        stab disp_data ; the first byte (digit) to display is stored
                          ; at disp_data
        ldab #%00111111 ; binary code for the letter 'o'
        stab disp_data+1 ; the second byte (digit) to display is stored
                           ; at disp_data+1
        ldab #%00110111 ; binary code for the letter 'n'
        stab disp_data+2 ; the third byte (digit) to display is stored
                           ; at disp_data+2
        ldab #%01111011 ; binary code for the letter 'e'
        stab disp_data+3 ; the fourth byte (digit) to display is stored
                           ; at disp_data+3

ldx #disp_data ; load the start address of the bytes into x
jsr sel_digit  ; select one digit at the time

wait:  tst  time        ; test if the time flag is zero
        beq wait          ; if so wait for interrupt. Every 5ms an
                              ; interrupt is generated.
                              ; When the interrupt occurs the Handler
                              ; toggle the time flag. The goal is to force
                              ; the program execution to break the
                              ; wait-loop and drive the next display’s
                              ; "digit"
        clr     time        
        jmp begin

* * multiplexing display one digit at a time *
*
sel_digit:
ldx #regbase
inc select          ; the value of select is irrelevant
        ldb select
        andb #3           ; true if accb(1:0) == 11
        tstb
        beq digit3
        decb
        beq digit2
        decb
        beq digit1

digit0:  ldaa disp_data+3 ; digit0 shows the fourth byte
        staa portb,x     ; send the byte to the segments
        bclr portd,x 4    ; turn ON digit 0
        bset portd,x 8    ; turn off digit 1
        bset portd,x $10  ; turn off digit 2
        bset portd,x $20  ; turn off digit 3
Name:______________________________________________________________

rts               ; exit from sel_digit

digit1:                       ; digit1 shows the third byte
  ldaa disp_data+2
  staa portb,x             ; send the byte to the segments
  bset portd,x 8            ; turn ON digit 1
  bset portd,x $10          ; turn off digit 2
  bset portd,x $20          ; turn off digit 3
  rts              ; exit from sel_digit

digit2:                          ; digit1 shows the second byte
  ldaa disp_data+1
  staa portb,x          ; send the byte to the segments
  bset portd,x 4         ; turn off digit 0
  bset portd,x 8         ; turn off digit 1
  bclr portd,x $10      ; turn ON digit 2
  bset portd,x $20       ; turn off digit 3
  rts                      ; exit from sel_digit

digit3:                          ; digit0 shows the third byte
  ldaa disp_data
  staa portb,x           ; send the byte to the segments
  bset portd,x 4         ; turn off digit 0
  bset portd,x 8         ; turn off digit 1
  bset portd,x $10       ; turn off digit 2
  bset portd,x $20       ; turn off digit 3
  rts              ; exit from sel_digit

* interrupt service routine for the timer output capture 2 interrupt
toc2Hdlr:

  ldx  #regbase
  inc  time               ; change the time iteration
  ldd  #tb5ms              ; reload the count for 5 ms time base
  addd toc2,x
  std  toc2,x
  ldaa #BIT6
  staa tflg1,x           ; to clear the flag "write" a 1 in the
                         ; associated bit
  rti

* Interrupt Vectors

  org  vtoc2
  fdb  toc2Hdlr
  org  vres
  fdb  main

end