ECE 3731: Microproc & Embedded Sys Lab

Lab 4: Parallel Input/Output

July 8th, 2018

Summer 2018

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Brian Vera-Burgos

x Srinivas Simhan

Table of Contents

Objective

- To become familiar with CodeWarrior
- To become familiar with HCS Dragon12-Light
- To become familiar with subroutines and parallel inputs/outputs
- To understand and create flowcharts to be used in explanations in terms of documentation for our code

Equipment Used

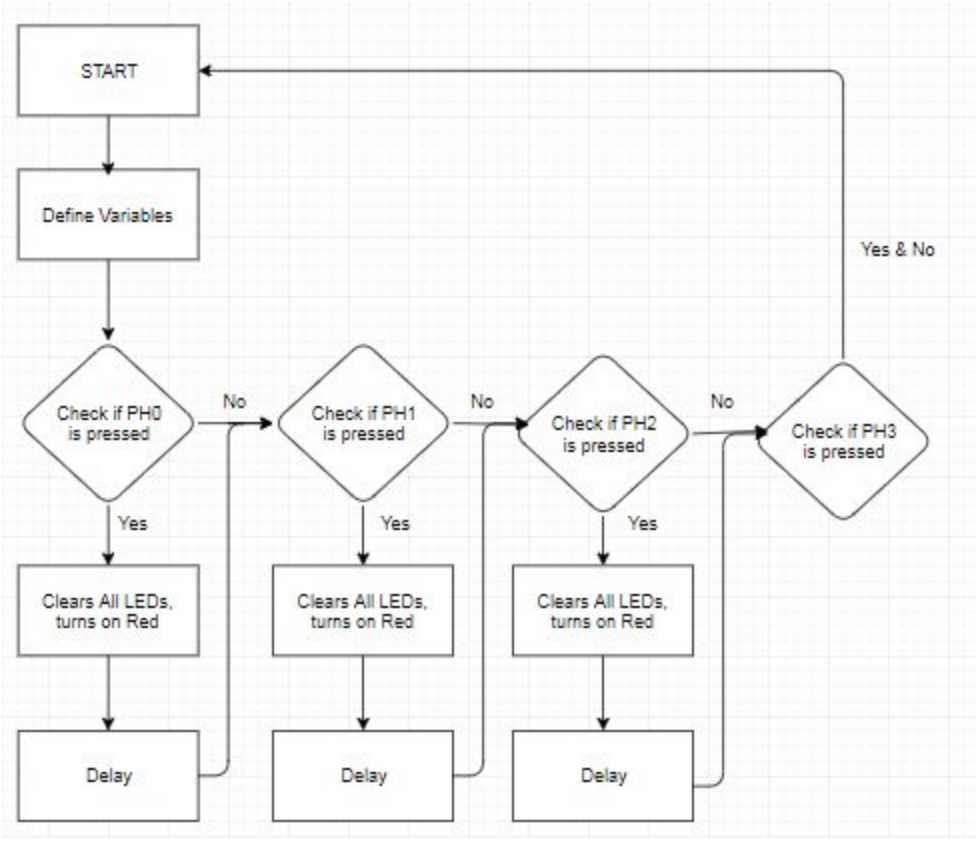
- CodeWarrior
- HCS Dragon12-Light

Pre-Lab

1. The purpose of question 1 of the Pre-Lab was to write a program which inputs a character from user, if the user enters a character ‘A’ from the keyboard. The value of the 4 - bit counter displayed on PORTB (LED's) is incremented by one. When a value of four ones is reached (1111), the counter rolls over to zero (0000).
    a. Full Code:

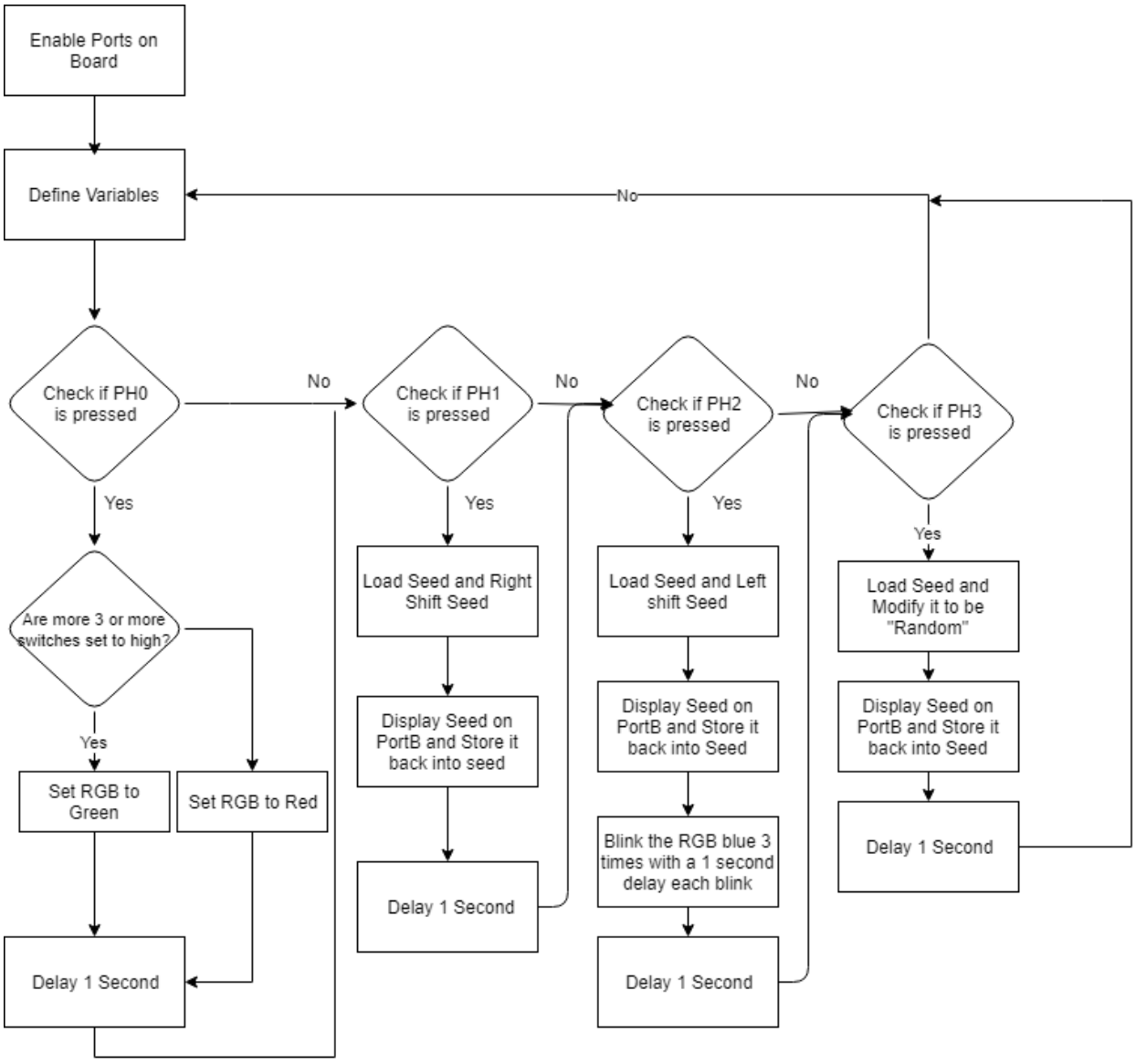
```
;---------------------------------------------------------
; Insert here your data definitions here
;---------------------------------------------------------
PROMPT  dc.b    $0A, $0D   ; CR LF
            dc.b    "Enter a A to increment: "
                    dc.b    0    ; using zero terminated strings

COUNTVAR        dc.b    0
;---------------------------------------------------------
; Insert your code here
;---------------------------------------------------------
       LDS   #ROMStart ; load stack pointer
       JSR TermInit
       JSR   led_enable   ; enable PORTB for LED's
MAINLOOP
      LDD #PROMPT
      JSR  printf
      JSR  getchar     ;Get input from user
      JSR putchar     ;Print out letter
      CMPB #$41     ;Check if value is the hex value of A
      BEQ GOA  ;If so branch to GOA
      BRA CONTINUE  ;Always go to continue unless A is pressed
GOA
      JSR INCFCN
CONTINUE
      BRA MAINLOOP
; Note: main program is an endless loop and subroutines follow
; (Must press reset to quit.)
;---------------------------------------------------------
; FUNCTIONS CALLED BY MAIN LOOP
;---------------------------------------------------------

INCFCN
    LDAB COUNTVAR  ;load countvar
    CMPB #$10
    BLO  CECS     ;Check if below $10
    LDAB #$00      ;if not set to $00
CECS
    STAB PORTB     ;turn on the LEDs on board
    jsr out2hex    ;print current number
    INCB           ;increment countvar
    STAB COUNTVAR  ;Store new number back into countvar
    RTS            ;return to main
```

2. Draw the flowchart of the sample program LAB4ASM:



- The purpose of the prelab was to get us familiar with displaying values of the 4-bit counter onto PORTB (LED's), modify the information after it's inserted onto PORTB, and to understand the flowchart given by the sample program that was provided to us (LAB4ASM). After this prelab, we have a better understanding of how to take output information onto the HCS Dragon12-Light Board.

Lab Flowchart



Procedure/ Full Code

```
;*****************************************************************
;* LAB4.ASM
;*
;*****************************************************************
; export symbols
          XDEF Entry, _Startup            ; export 'Entry' symbol
          ABSENTRY Entry        ; for absolute assembly: mark this as application entry point

; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'

;-------------------------------------------------
; Equates Section
;-------------------------------------------------
ROMStart    EQU  $2000  ; absolute address to place my code
RED:        EQU     $10    ; PP4
BLUE:       EQU     $20    ; PP5
GREEN:      EQU     $40    ; PP6

;-------------------------------------------------
; Variable/Data Section
;-------------------------------------------------
          ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here
SEED   dc.b   0

      INCLUDE 'utilities.inc'
      INCLUDE 'LCD.inc'

;-------------------------------------------------
; Code Section
;-------------------------------------------------
          ORG   ROMStart  ; loc $2000
Entry:
_Startup:
          ; remap the RAM &amp;amp; EEPROM here. See EB386.pdf
 ifdef _HCS12_SERIALMON
          ; set registers at $0000
          CLR   $11                ; INITRG= $0
          ; set ram to end at $3FFF
          LDAB  #$39
          STAB  $10                ; INITRM= $39

          ; set eeprom to end at $0FFF
          LDAA  #$9
          STAA  $12                ; INITEE= $9
          JSR   PLL_init      ; initialize PLL
  endif

;-------------------------------------------------
; Insert your code here
;-------------------------------------------------
```

```
 ;*Note: Make sure DIP sw's are all up
          JSR    led_enable    ; init. PORTB for LED's
          bset    DDRP,  RED+GREEN+BLUE  ; make pp4-pp6  outputs
          bset    DDRM, 4    ; make PM2 an output
          bclr    PTM, 4   ; make PM2 low to enable RGB

MAIN
*Check for PTH pin 0 SW press  (SW5)  if not check next one
CHECKPH0  BRSET   PTH, $01,CHECKPH1
                        JSR   PH0FCN  ; if button pressed call function for button
                        JSR   DELAY_200_MS
*Check for PTH pin 1 SW press (SW4)  if not check next one
CHECKPH1  BRSET   PTH, %00000010,CHECKPH2
                        JSR   PH1FCN  ; if button pressed call function for button
                        JSR   DELAY_200_MS
*Check for PTH pin 2 SW press  (SW3) if not check next one
CHECKPH2  BRSET   PTH, %00000100, CHECKPH3
                        JSR   PH2FCN  ; if button pressed call function for button
                        JSR   DELAY_200_MS
CHECKPH3  BRSET   PTH, %00001000, NEXT
                        JSR   PH3FCN  ; if button pressed call function for button
                        JSR   DELAY_200_MS
NEXT      JMP   MAIN


; Note: main program is an endless loop and subroutines follow
; (Must press reset to quit.)

;=====================================================================
; FUNCTIONS CALLED BY MAIN LOOP
;* <This function will simply call the ELECTION function and then delay for 1 second before restarting the program>
PH0FCN

        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        JSR ELECTION
        jsr    DELAY_1_S   ; delay for 1 second
 RTS
;---------------------------------------------------------------------
;*<This function will load the seed value and shift it right by one bit>
PH1FCN

        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        LDAA SEED
        LSRA
        STAA PORTB
        STAA SEED
        jsr    DELAY_1_S   ; delay for 1 second
        RTS

;---------------------------------------------------------------------
;*<This function will load in the seed value and shift it left by one bit, it will then blink the blue LED 3 times with a 1 second
delay>
PH2FCN
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        bset   PTP,  BLUE   ; turn on blue
        LDAA SEED
        LSLA
```

```
        STAA PORTB
        STAA SEED
        jsr     DELAY_1_S    ; delay for 1 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr     DELAY_1_S    ; delay for 1 second
        bset   PTP,  BLUE
        jsr     DELAY_1_S    ; delay for 1 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr     DELAY_1_S    ; delay for 1 second
        bset   PTP,  BLUE
        jsr     DELAY_1_S    ; delay for 1 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr     DELAY_1_S    ; delay for 1 second
        bset   PTP,  BLUE
        rts
;---------------------------------------------------------------------
;*<This function will call the rand function and store it into seed>
PH3FCN
         JSR RAND
         LDAA SEED
         STAA PORTB
         rts
;---------------------------------------------------------------------
;*<This function will load in the seed value and manipulate it to make a "random" number to return>
RAND    LDAA    SEED
       LSLA
       ADCA     #20
       STAA     SEED
       RTS

;---------------------------------------------------------------------
;*<This function will call the 1 millisecond delay to delay for 200 milliseconds>
DELAY_200_MS
        LDX #$00C8       ;load in 200
LOOP_200_MS
        JSR DELAY_1_MS    ;call delay 1 milisecond
        DEX             ;decrease X
        CPX #$0
        BNE LOOP_200_MS     ;branch until X = 0
        RTS             ;return
;---------------------------------------------------------------------
;*<This function will call the 1 millisecond delay to delay for 1000 milliseconds>
DELAY_1_S
        LDX #$03E8       ;load in 1000
LOOP_1_S
        JSR DELAY_1_MS    ;call delay 1 milisecond
        DEX             ;decrease X
        CPX #$0
        BNE LOOP_1_S     ;branch until X = 0
        RTS             ;return
;---------------------------------------------------------------------
;*<This function will run a simple program 6000 times to delay for 1 millisecond>

DELAY_1_MS
        LDY #$1770       ;load in 6000
LOOP_1_MS
        DEY             ;Decrement y
```

```
        CPY #$0
        BNE LOOP_1_MS     ;
        RTS
;-------------------------------------------------------------------
;*<This function will check the dip switches and set turn on either the red or green LED depending on whether more are
high or low>
ELECTION

        LDAA #0
        BRCLR PTH,%1000000,L1
        INCA
L1
        BRCLR PTH,%0100000,L2
        INCA
L2
        BRCLR PTH,%0010000,L3
        INCA
L3
        BRCLR PTH,%0001000,L4
        INCA
L4
        CMPA #$3
        BHI RESULTS
        bset   PTP,  RED   ; turn on red
        RTS
RESULTS
        bset   PTP,  GREEN   ; turn on green
        RTS
;-------------------------------------------------------------------
;END OF FUNCTIONS




;*************************************************************
;*               Interrupt Vectors                           *
;*************************************************************
          ORG   Vreset
          DC.W  Entry         ; Reset Vector
```

- Part A: User presses PTH3
    - *Explanation of Code:*
        - *The program loads SEED and modifies it to be a “random” value before displaying it on PORTB. This random variable is decided by a subroutine that was provided in the Sample Project. The program stores this new “random” value of Seed back into seed before delaying 1 second and restarting the program.*
- Part B: User presses PTH2
    - *Explanation of Code:*
        - *The program logically shifts left the bit positions by one position on PORTB & then flashes the blue LED 3 times in intervals (1 second on*

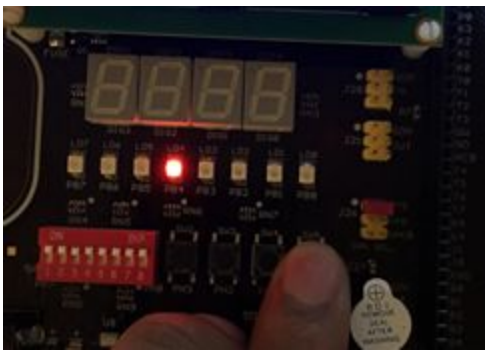
time and 1 second off time). It will then delay 1 second and restart the program.

- Part C: User presses PTH1
    - *Explanation of Code:*
        - *The program loads SEED and arithmetically shifts right the bits by one position of PORTB, before storing the new value back into seed. The program will then delay 1 second and restart the program.*
- Part D: User presses PTH0
    - *Explanation of Code:*
        - *The program read the switches on PORT H (the 4 leftmost bits on the DIP switch), and based on if those switches are on/off decides if the LED will be GREEN or RED.*
        - *If the number of logic '1' bits (in the 4 leftmost bits) is greater than the number of logic '0' bits, the program lights the GREEN LED on the RGB (while the other LED's (colors) on the RGB are off).*
        - *Otherwise, the RED LED on the RGB is turned on (while the other LED's on the RGB are off).*
        - *The program will then delay for 1 second and restart.*

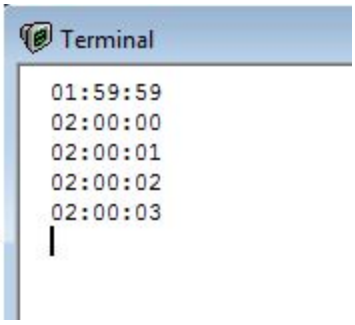
Post-Lab

We did a demo of all 4 questions from Homework 5 to our lab instructor and passed.

1. Question 1:
    a. Snapshot:



    b. Code is in Part 5.
2. Question 2:
    a. Yes, we heard the tone on the speaker.
    b. Code is in Part 5.
3. Question 3:
    a. Yes, we heard the tones on the speaker.
    b. Code is in Part 5.
4. Question 4:
    a. Snapshots:



b. Code is in part 6.

5. Full Code for Questions 1, 2, 3:

```
;*****************************************************************
;* HMWK5.ASM
;*
;*****************************************************************
; export symbols
            XDEF Entry, _Startup            ; export 'Entry' symbol
            ABSENTRY Entry                  ; for absolute assembly: mark this as application entry point


; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'


;-------------------------------------------------
; Equates Section
;-------------------------------------------------
ROMStartEQU  $2000  ; absolute address to place my code
RED:      EQU       $10    ; PP4  %00010000
BLUE:     EQU       $20    ; PP5  %00100000
GREEN:    EQU       $40    ; PP6  %01000000


;-------------------------------------------------
; Variable/Data Section
;-------------------------------------------------
            ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here
SEED   dc.b   0
TIMER  dc.b   0
LOGIC_ONE dc.b 0
LOGIC_ZERO dc.b 0


            INCLUDE 'utilities.inc'
```

```
            INCLUDE 'LCD.inc'


;-------------------------------------------------------
; Code Section
;-------------------------------------------------------
            ORG   ROMStart  ; loc $2000
Entry:
_Startup:
            ; remap the RAM &amp;amp; EEPROM here. See EB386.pdf
 ifdef _HCS12_SERIALMON
            ; set registers at $0000
            CLR   $11               ; INITRG= $0
            ; set ram to end at $3FFF
            LDAB  #$39
            STAB  $10               ; INITRM= $39

            ; set eeprom to end at $0FFF
             LDAA  #$9
            STAA  $12               ; INITEE= $9
            JSR   PLL_init      ; initialize PLL
 endif

;-------------------------------------------------------
; Insert your code here
;-------------------------------------------------------
 ;*Note: Make sure DIP sw's are all up
            JSR         led_enable              ; init. PORTB for LED's
            bset        DDRP,  RED+GREEN+BLUE  ; make pp4-pp6  outputs
            bset        DDRM, 4    ; make PM2 an output
            bclr        PTM, 4   ; make PM2 low to enable RGB
            bset        DDRT,%00100000

MAIN
```

```
*Check for PTH pin 0 SW press  (SW5)  if not check next one
CHECKPH0  BRSET   PTH, $01,CHECKPH1
                  JSR   PH0FCN  ; if button pressed call function for button
*Check for PTH pin 1 SW press (SW4)  if not check next one
CHECKPH1  BRSET   PTH, %00000010,CHECKPH2
                  JSR   PH1FCN  ; if button pressed call function for button
*Check for PTH pin 2 SW press  (SW3) if not check next one
CHECKPH2  BRSET   PTH, %00000100, NEXT
                  JSR   PH2FCN  ; if button pressed call function for button
NEXT      JMP   MAIN


; Note: main program is an endless loop and subroutines follow
; (Must press reset to quit.)

;===============================================================================
; FUNCTIONS CALLED BY MAIN LOOP
;* < Students should write function comment headers for these>

;----------------------BLINK 5TH LED-------------------------
PH0FCN
          BSET PORTB, %00010000
          JSR DELAY_250_MS
          BCLR PORTB, %00010000
          JSR DELAY_250_MS
          RTS

;----------------------SQUARE WAVE-------------------------
PH1FCN
          LDAB #%00100000

          EORB #%00100000
          STAB PTT
```

```
        jsr     DELAY_1_MS   ; delay for .01 second
        EORB #%00100000
        STAB PTT
        jsr     DELAY_1_MS  ; delay for .01 second
        RTS
;---------------------THREE TONE SIREN------------------
PH2FCN

        LDAA #100
        LDAB #%00100000

        LOOP200:
        EORB #%00100000
        STAB PTT
        jsr     DELAY_2.5_MS   ; delay for .01 second
        EORB #%00100000
        STAB PTT
        jsr     DELAY_2.5_MS  ; delay for .01 second
        DECA
        BNE     LOOP200

        LDAA #250

        LOOP500:
        EORB #%00100000
        STAB PTT
        jsr     DELAY_1_MS   ; delay for .01 second
            EORB #%00100000
        STAB PTT
        jsr     DELAY_1_MS  ; delay for .01 second
        DECA
        BNE     LOOP500
```

```
            LDY  #500
            STY TIMER

            LOOP1000:
            EORB #%00100000
            STAB PTT
            jsr         DELAY_.5_MS   ; delay for .01 second
            EORB #%00100000
            STAB PTT
            jsr         DELAY_.5_MS  ; delay for .01 second
            LDY TIMER
            DEY
            STY TIMER
            BNE       LOOP1000

            BRA PH2FCN

;---------------------RANDOM---------------------------------------
* <put banner comments for the function here >
RAND     LDAA      SEED
              LSLA
            ADCA      #20
            STAA      SEED
            RTS




;-------------------------DELAYS-------------------------------
DELAY_200_MS
            LDX #$00C8          ;load in 200
LOOP_200_MS
            JSR DELAY_1_MS ;call delay 1 milisecond
            DEX                     ;decrease X
```

```
            CPX #$0
            BNE LOOP_200_MS          ;branch until X = 0
            RTS                      ;return
;-------------------------------------------------------------
DELAY_250_MS
            LDX #$00FA               ;load in 250
LOOP_250_MS
            JSR DELAY_1_MS           ;call delay 1 milisecond
            DEX                      ;decrease X
            CPX #$0
            BNE LOOP_250_MS          ;branch until X = 0
            RTS             ;return
;-------------------------------------------------------------
DELAY_2.5_MS
            LDX #$5                  ;load in 5
LOOP_2.5_MS
            JSR DELAY_.5_MS          ;call delay .5 milisecond
            DEX                      ;decrease X
            CPX #$0
               BNE LOOP_2.5_MS       ;branch until X = 0
            RTS                      ;return
;-------------------------------------------------------------

DELAY_1_S
            LDX #$03E8               ;load in 1000
LOOP_1_S
            JSR DELAY_1_MS           ;call delay 1 milisecond
            DEX                      ;decrease X
            CPX #$0
            BNE LOOP_1_S     ;branch until X = 0
            RTS                      ;return
```

```
;---------------------------------------------------------------
DELAY_1_MS
        LDY #$1770          ;load in 6000
LOOP_1_MS
        DEY                 ;Decrement y
        CPY #$0
        BNE LOOP_1_MS  ;
        RTS
;---------------------------------------------------------------
DELAY_.5_MS
        LDY #$0BB8          ;load in 3000
LOOP_.5_MS
        DEY                 ;Decrement y
        CPY #$0
        BNE LOOP_.5_MS  ;
        RTS
;---------------------------------------------------------------

DELAY_TWOHUN_SEC PSHY
        LDY #200
        LOOP2:
        jsr DELAY_1_MS
        DEY
        BNE LOOP2
        PULY
        RTS
;---------------------------------------------------------------
RED_LIGHT
        bset PTP, RED
        jsr DELAY_TWOHUN_SEC
        RTS
```

```
;*************************************************************
;*            Interrupt Vectors                    *
;*************************************************************
            ORG   Vreset
        DC.W  Entry     ; Reset Vector
```

6. Full Code for Question 4:

```
;*************************************************************
;* HWK5.4.ASM
;
;Code Entry, Assembly, and Execution
;(Put your name and date here)
;-------------------------------------
;* -this is the sample code for Lab1
;* -for Full Chip Simulation or Board -- select your target
;* DO NOT DELETE ANY LINES IN THIS TEMPLATE
;* --ONLY FILL IN SECTIONS
;*************************************************************
; export symbols
            XDEF Entry, _Startup            ; export 'Entry' symbol
            ABSENTRY Entry                  ; for absolute assembly: mark this as application entry point


; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'


;-------------------------------------------------
; Equates Section
;-------------------------------------------------
ROMStartEQU  $2000  ; absolute address to place my code


;-------------------------------------------------
```

```
; Variable/Data Section
;-------------------------------------------------------
            ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

            ORG $1500

PROMPT              dc.b    $0A, $0D   ; CR LF
            dc.b        "Enter a number between 2 and 9: "
            dc.b        0    ; using zero terminated strings

QUIT_PROMPT dc.b              "End program. "
            dc.b        0    ; using zero terminated strings

SECS_NUM   dc.b    $3A   ;Permenant storage of seconds
MINS_NUM   dc.b    $3B   ;Permenant storage of minutes
HOURS_NUM  dc.b    $01   ;Permenant storage of hours
;TENS_VAL and ONES_VAL are used for the hex2dec convertor
TENS_VAL            dc.b    $0   ;Temperary storage for the tens place decimal value in DEC_PRINT
ONES_VAL            dc.b    $0   ;Temperary storage for the once place


            INCLUDE 'utilities.inc'
            INCLUDE 'LCD.inc'

;-------------------------------------------------------
; Code Section
;-------------------------------------------------------
            ORG   ROMStart  ; loc $2000
Entry:
_Startup:
            ; remap the RAM &amp;amp; EEPROM here. See EB386.pdf
 ifdef _HCS12_SERIALMON
```

```
            ; set registers at $0000
            CLR   $11               ; INITRG= $0
            ; set ram to end at $3FFF
               LDAB  #$39
            STAB  $10               ; INITRM= $39

            ; set eeprom to end at $0FFF
            LDAA  #$9
            STAA  $12               ; INITEE= $9
            JSR   PLL_init      ; initialize PLL
 endif

;---------------------------------------------------
; Insert your code here
;---------------------------------------------------
            lds   #ROMStart ; load stack pointer
            jsr TermInit

MAINLOOP
            JSR DELAY_1_S  ;delay 1 second
            LDAB SECS_NUM  ;load in seconds value
            INCB       ;increment seconds value

            CMPB #$3C           ;compare seconds value with 60
            BNE  NOTMIN         ;if it doesnt equal 60 go to notmin
            LDAB MINS_NUM  ;Load Minutes value
            INCB       ;Increment Minutes
            STAB MINS_NUM  ;Store Mintues Value
            LDAB #$0            ;load a 0 to reset seconds
NOTMIN
            STAB SECS_NUM  ;Store current secondsval
```

```
            LDAB MINS_NUM  ;Load current minutes Value
            CMPB #$3C      ;compare minutes value with 60
            BNE  NOTHR     ;if it doesnt equal 60 go to nothr
            LDAB HOURS_NUM ;If it does load hours value
            INCB       ;Increment Hours Value
            STAB HOURS_NUM ;Store Hours Value
            LDAB #$0       ;Load a zero to reset minutes
NOTHR
            STAB MINS_NUM


            LDAB  HOURS_NUM ; Load Hournum
            JSR HEX2DEC    ; Print hournum
            LDAB #$3A      ; Load a colon
            JSR putchar    ; Print :

            LDAB  MINS_NUM  ; Load MINS_NUM
            JSR HEX2DEC    ; Print MINS_NUM
            LDAB #$3A      ; Load a colon
            JSR putchar    ; Print :

            LDAB  SECS_NUM  ; Load SECS_NUM
            JSR HEX2DEC    ; Print SECS_NUM
            LDAB #$3A      ; Load a colon

            LDAB #$A       ;Load a newline
            JSR putchar    ;Print a newline

            BRA MAINLOOP

here: jmp here      ;Stay here forever to end program

;-------End Program HWK3-------------
```

```
;*****************************************************************
; FUNCTIONS CALLED BY MAIN LOOP
;---------------------------------------------------------------
DELAY_1_MS
        LDY #$1770          ;load in 6000
LOOP_1_MS
        DEY                 ;Decrement y
        CPY #$0
        BNE LOOP_1_MS  ;
        RTS
;---------------------------------------------------------------
DELAY_1_S
        LDX #$03E8      ;load in 1000
LOOP_1_S
        JSR DELAY_1_MS ;call delay 1 milisecond
        DEX                 ;decrease X
        CPX #$0
        BNE LOOP_1_S    ;branch until X = 0
        RTS                 ;return
;---------------------------------------------------------------
HEX2DEC
        LDAA #0             ;load zero in for the tens place
LOOP
        CMPB #$0A           ;check if value is already less than 10
        BLO DEC_PRINT  ;if so go to dec_print
        ADDB #$F6           ;subtract 10
        INCA       ;increase tens place
        CMPB #$0A           ;check if its below 10 now
        BHS LOOP            ;if not go back to loop

DEC_PRINT
        STAB ONES_VAL  ;store ones value
        STAA TENS_VAL  ;store tens value
```

```
            LDAB TENS_VAL  ;obtain tens value
            ADDB #$30      ;add $30 so the ascii value lines up
            JSR putchar    ;print tens value
            LDAB ONES_VAL  ;load ones value
            ADDB #$30      ;add $30 so the ascii value lines up
            JSR putchar    ;print ones value

            RTS
;---------------------------------------------------------------------



;*************************************************************
;*          Interrupt Vectors                    *
;*************************************************************
            ORG   Vreset
            DC.W  Entry       ; Reset Vector
```

Conclusion

- Pre-Lab:
    - We focused a lot on inputting a value into the HCS Dragon12-Light Board, and displaying the the outputs onto PORTB (LED's). What we learned from this was how to manipulate the input and outputs using the HCS Dragon12-Light Board and our code on CodeWarrior.
- In-Lab:
    - Part A (PTH 3)
        - We learned how to generate a "random" value and display it on PORTB using the "RAND" subroutine with the variable "SEED".
    - Part B (PTH 2)
        - We practiced how to implement the logic shift left and display it on PORTB and then store it back into Seed. We also learned how to make a primitive delay function for the first time and use it to blink the RGB LED.
    - Part C (PTH 1)
        - In Part C we practiced loading the Seed variable and performing a right shift before displaying it on PORTB.
    - Part D (PTH 0)
        - In part D we learned how to use the DIP switches on the Dragon12 board to have a voting program. This program would check if more DIP switches were up rather than down and switch a light based on this.
- Post- Lab:
    - We learned how to modify our code to adjust the outputs we received on the LEDs of PORTB.
    - We also learned how to use the speaker on the board by creating different square waves and a three-tone siren. The speaker functionality taught us how to modify the periods of sound wave to create different tones at 200Hz, 500Hz, and 1kHz.
    - There was an emphasized focus on learning to define variables to store information, rather than just using the registers to contain and manipulate data. As a result we were able to display a clock that counted up in seconds like a normal clock in the following format:
        - hh:mm:ss
            - hh is 2 digit hours display
            - mm is 2 digit minutes display
            - ss is 2 digit seconds display