ECE 3731: Microproc & Embedded Sys Lab

Lab 5: External Interrupts Parallel

July 26th, 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 setting up interrupts to be controlled by PORTH
- To become familiar with “polling” interrupt flags
- 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

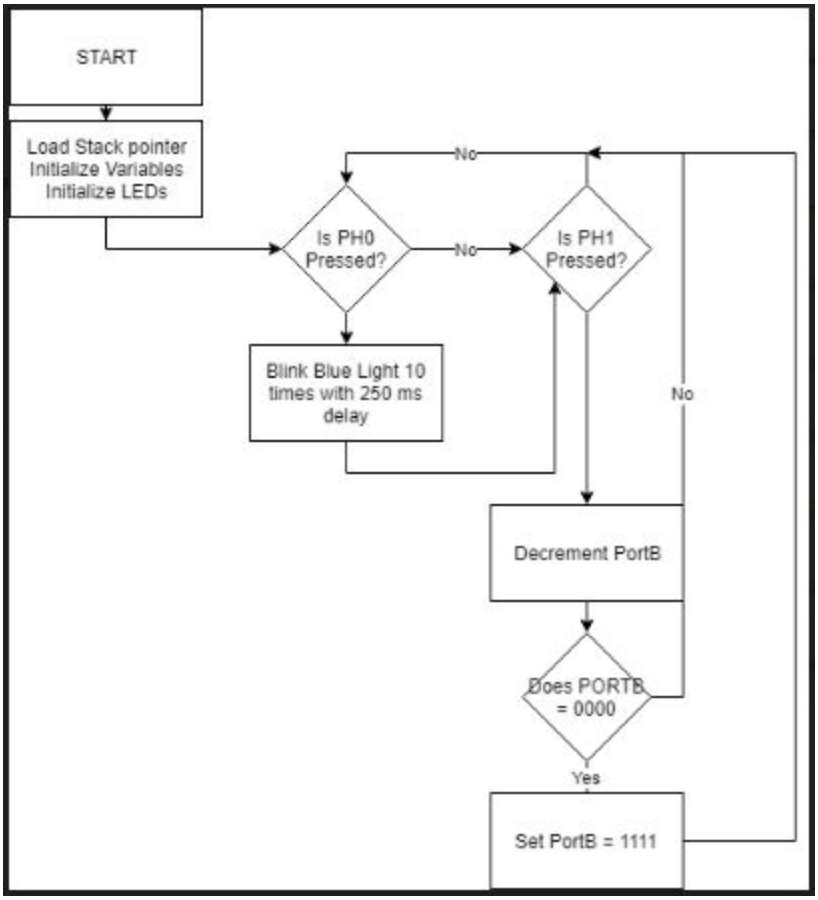
1. Pre-Lab Question 1
    a. The purpose of question 1 of the Pre-Lab was to write a program which flashes the Blue LED 10 times with on time of 0.25 seconds and an off time of 0.25 seconds. When the user pushes PH0, the code clear the RBG value, then sets it to blue, then delays it for 250ms, then clears it again, then delays for 250ms, then checks a counter set by register A, and repeats accordingly.
        i. Main Code for Part 1:

```
;-----------BLINK BLUE 10 FUNCTION----------------
PH0FCN
        LDAA   #$A
REPEAT:
        bclr   PTP, RED+GREEN+BLUE
        bset   PTP, BLUE
        jsr    DELAY_250_MS    ; delay for 250 milisecods
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS    ; delay for 250 milisecods
        DECA
        CMPA   #0
        BNE    REPEAT
        JSR    DELAY_1_S
        RTS
```

2. Pre-Lab Question 2
    a. The purpose of question 2 of the Pre-Lab was to write a program that decrements a 4-bit counter on PORTB, and when the value of zero is reached and the counter is decremented, it becomes a 1111. When the user pushes PH1, the code loads the value of #$Finto register A and sets that to PORTB, then delays by 250ms, then decreases the value until it hits 0, and if that happens it then reloads the A register with #$F, and repeats accordingly.
        i. Main Code for Part 2:

```
;-----------Decrement 4-bit counter on PORTB----------------
PH1FCN
        LDAA #$F ;decrement
YOLO
        STAA PORTB
        JSR DELAY_250_MS
        DECA
        BNE YOLO

        BRA PH1FCN
```

3. Flowchart for Pre-Labs
    a.



Procedure/ Full Code

- Part A: User presses PTH3
    - *Explanation of Code:*
        - *The Mainloop of the program continuously calls a subroutine that will blink the green RGB 10 times before restarting*

1) *If the first interrupt is called then the program will call a subroutine to look at the four dip switches and store their position (Up or down) on PORTB. It checks the position of the dip switches using the election subroutine from class. The program will also use the LCD on the dragon-12 board to print out “SW0 is Pressed”.*
2) *If the second interrupt is called then the program will call a subroutine to increment PORTB by one and then blink the blue RGB 4 times with a 250 ms delay. The program will overflow to zero if it increments 1111. The program will also use the LCD on the dragon-12 board to print out “SW1 is Pressed”.*
3) *If the third interrupt is called then the program will call a subroutine to decrement PORTB by one and then blink the RED RGB 5 times with a 200 ms delay. The program will overflow to 1111 if it decrements on 0000.The program will also use the LCD on the dragon-12 board to print out “SW2 is Pressed”.*
4) *If the fourth interrupt is called then the program will call a subroutine that will print out the multiplication table of 5 up to 5*10 onto the terminal. The program will also use the LCD on the dragon-12 board to print out “SW4 is Pressed”.*

- Part A Full Code

```
;*****************************************************************
;* Lab5A.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


SW0PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW0 is Pressed "
        dc.b    0    ; using zero terminated strings

SW1PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW1 is Pressed "
        dc.b    0    ; using zero terminated strings

SW2PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW2 is Pressed "
        dc.b    0    ; using zero terminated strings

SW3PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW3 is Pressed "
        dc.b    0    ; using zero terminated strings
;These are the variables used in the fourth button
INPUT_NUM   dc.b    $35   ;Permenant storage of users input number
COUNT_NUM   dc.b    $0   ;Temperary storage for count number
MULT_NUM    dc.b    $0   ;Starting value for multiplication
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 decimal value in DEC_PRINT

      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
* Port H interrupt program for Dragon12
* Lights LED 0 (and clears LED1) when sw5 is pressed (PH0)
* Lights LED 1 (and clears LED0) when sw4 is pressed (PH1)
        jsr led_enable    ;Initialize LEDS
```

```
        jsr TermInit      ;Initialize the code warrior terminal
        bset    DDRM, 4    ; make PM2 an output
        bclr    PTM, 4   ; make PM2 low to enable RGB
        bset    DDRP,  RED+GREEN+BLUE  ; make pp4-pp6  outputs
; note Port H is all inputs after reset
        jsr   lcd_init    ; initialize LCD (must be done first)
        BCLR    PPSH, #$0F  ; set Port H pins 0-1 for falling edge
        MOVB   #$0F, PIFH    ; clear interrupt flags initially
        BSET     PIEH,  $0F     ; enable interrupts on Port H  pins 0-1
        CLI                    ; enable interrupts
RESTRARTGRN
        JSR Flash_Green_Led

        BRA  RESTRARTGRN              ; endless loop waiting for reset (and for interrupts)


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

;*****************************************************************


; ISR must test to see which button was pressed, because there ;is only one ISR for the two enabled buttons

PTHISR:  ; the interrupt service routine
       BRSET  PIFH, %00000001,PUSHBTN0  ; test btn0 IF flag
       BRSET  PIFH, %00000010,PUSHBTN1  ; test btn1 IF flag
       BRSET  PIFH, %00000100,PUSHBTN2  ; test btn2 IF flag
       BRSET  PIFH, %00001000,PUSHBTN3  ; test btn3 IF flag
; NOTE:  Flags are tested –not the switches
       JMP  DONE
;---------------------------------------------------------------------
;----------------INTERRUPTS
;---------------------------------------------------------------------
;*****************************************************************
PUSHBTN0:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all

        LDD #SW0PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt

        JSR ELECTION
        jsr    DELAY_1_S   ; delay for 1 second
        JMP    DONE

;*****************************************************************
PUSHBTN1:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        LDD #SW1PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
        JSR INCPORT    ;Call INCPORT subroutine
       JMP    DONE
;*****************************************************************
PUSHBTN2:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
```

```
        bset   PTP,  RED   ; turn on RED

        LDD #SW2PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
        JSR DECPORT    ;Call DECPORT Subroutinr
        JMP    DONE
;=====================================================================
PUSHBTN3:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        LDD #SW3PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
        JSR PRINT5MULT

DONE
       MOVB  #$0F, PIFH   ; clear Port H interrupt flags
       RTI
;---------------------------------------------------------------------
;----------------FUNCTIONS
;---------------------------------------------------------------------

Flash_Green_Led
        LDAA #10 ;This will flash the green LED 10 times
GREENLOOP
        bset   PTP,  GREEN   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        DECA   ;Decrement the count of GREENLOOP
        BNE GREENLOOP ;If it hasnt flashed 10 times then go to GREENLOOP
        RTS
;---------------------------------------------------------------------
PRINT5MULT
        LDAA #$1      ;Count for the table loop
TABLE_LOOP
        LDAB #$A      ;Load Newline
        JSR putchar   ;Print Newline
        LDAB #$0D
        JSR putchar
        LDAB #$05 ;load in input number
        JSR out2hex    ;print input number
        LDAB #$20      ;Load a space
        JSR putchar    ;print the space
        LDAB #$58      ;Load in X
        JSR putchar    ;Print X

        CMPA #$A       ;Check if the count has reached 10, if so we need to do a special Print_count to print 10
        BLO  PRINT_COUNT;If < 9 then skip this
        LDAB #$10      ;Load a $10
        JSR out2hex
        BRA AFTER_COUNT

PRINT_COUNT

        STAA COUNT_NUM ;Store into COUNT_NUM
        LDAB COUNT_NUM ;Pull from COUNT_NUM
```

```
        JSR out2hex    ;Print COUNT_NUM

AFTER_COUNT

        LDAB #$20      ;Load a space
        JSR putchar    ;print the space
        LDAB #$3D      ;Load an = sign
        JSR putchar    ;Print the = sign
        LDAB #$20      ;Load a space
        JSR putchar    ;print the space

        STAA COUNT_NUM ;Store current count for later
        LDAB INPUT_NUM ;load the input number
        ADDB #$D0      ;subtract 30 so that the value is one digit
        STAB MULT_NUM  ;store value for use in multiplication
        LDAB #$0
        LDAA #$1

MULT_LOOP
        ADDB MULT_NUM  ;add stored value to zero
        CMPA COUNT_NUM ;check if enough additions have happened
        INCA
        BLO  MULT_LOOP ;Add again if not finished


        JSR HEX2DEC    ;go to hex to decimal function

        LDAA COUNT_NUM ;reload the count number to do the next number on the table
        INCA           ;increment to the next number on the table
        CMPA #$B       ;check if we have reached 10 yet
        BLO TABLE_LOOP ;if not go back to TABLE_LOOP to finish the table

        RTS
;---------------------------------------------------------------------
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
      JSR DELAY_1_S
      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
        STAA PORTB
        BRCLR PTH,%10000000,L1   ;Check if first Dip switch is low
        BSET PORTB, %00001000    ;Set fifth portb led to high
L1
        BRCLR PTH,%01000000,L2   ;check if second dip switch is low
        BSET PORTB, %00000100    ;set sixth portb LED to high
L2
        BRCLR PTH,%00100000,L3   ;check if third dip switch is low
        BSET PORTB, %00000010    ;set seventh portb LED to high
L3
        BRCLR PTH,%00010000,L4   ;check if fourth dit switch is low
        BSET PORTB, %00000001    ;set eighth portb LED to high
L4
        RTS
;-------------------------------------------------------------------
DECPORT
        LDAA PORTB
        DECA
        CMPA #%11111111
        BNE NOTSET2
        LDAA #%00001111
NOTSET2
        STAA PORTB
        ;This is 5 blinks at 2 HZ
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        RTS
;-------------------------------------------------------------------
INCPORT
        LDAA PORTB
        INCA
        CMPA #%00010000
        BNE NOTSET
        LDAA #%00000000
NOTSET
        STAA PORTB
        ;This is 4 blinks at 1 HZ
```

```
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .25 second
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second

        RTS
 ;-------------------------------------------------------------------
;*<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 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 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 250 milliseconds>
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
;---------------------------------------------------------------------
```

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

;*************************************************************
        ORG     Vporth     ; setup  Port H interrupt Vector
        DC.W    PTHISR
```

- Part B: User presses PTH2
    - *Explanation of Code:*
        - *The Mainloop of the program continuously calls a subroutine that will blink the green RGB 10 times before restarting. This time however the global interrupt is disabled*

1) *If the first interrupt flag is set high, then the program will call a subroutine to look at the four dip switches and store their position (Up or down) on PORTB. It checks the position of the dip switches using the election subroutine from class. The program will also use the LCD on the dragon-12 board to print out “SW0 is Pressed”.*

2) *If the second interrupt flag is set high, then the program will call a subroutine to increment PORTB by one and then blink the blue RGB 4 times with a 250 ms delay. The program will overflow to zero if it increments 1111. The program will also use the LCD on the dragon-12 board to print out “SW1 is Pressed”.*

3) *If the third interrupt flag is set high, then the program will call a subroutine to decrement PORTB by one and then blink the RED RGB 5 times with a 200 ms delay. The program will overflow to 1111 if it decrements on 0000.The program will also use the LCD on the dragon-12 board to print out “SW2 is Pressed”.*

4) *If the fourth interrupt flag is set high then the program will call a subroutine that will print out the multiplication table of 5 up to 5*10 onto the terminal. The program will also use the LCD on the dragon-12 board to print out "SW4 is Pressed".*

- Part B Full Code

```
;*****************************************************************
;* KeyWakeup.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


SW0PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW0 is Pressed "
        dc.b    0    ; using zero terminated strings

SW1PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW1 is Pressed "
        dc.b    0    ; using zero terminated strings

SW2PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW2 is Pressed "
        dc.b    0    ; using zero terminated strings

SW3PROMPT      dc.b    $0A, $0D   ; CR LF
        dc.b    "SW3 is Pressed "
        dc.b    0    ; using zero terminated strings
;These are the variables used in the fourth button
INPUT_NUM   dc.b    $35   ;Permenant storage of users input number
COUNT_NUM   dc.b    $0   ;Temperary storage for count number
MULT_NUM    dc.b    $0   ;Starting value for multiplication
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 decimal value in DEC_PRINT

    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
* Port H interrupt program for Dragon12
* Lights LED 0 (and clears LED1) when sw5 is pressed (PH0)
* Lights LED 1 (and clears LED0) when sw4 is pressed (PH1)
        jsr led_enable    ;Initialize LEDS
        jsr TermInit      ;Initialize the code warrior terminal
        bset    DDRM, 4    ; make PM2 an output
        bclr    PTM, 4   ; make PM2 low to enable RGB
        bset    DDRP,  RED+GREEN+BLUE  ; make pp4-pp6  outputs
; note Port H is all inputs after reset
        jsr   lcd_init    ; initialize LCD (must be done first)
        BCLR    PPSH, #$0F  ; set Port H pins 0-1 for falling edge
        MOVB   #$0F, PIFH    ; clear interrupt flags initially
        BSET     PIEH,  $0F     ; enable interrupts on Port H  pins 0-1
        SEI                    ;Disable Interrupts
RESTRARTGRN
        JSR Flash_Green_Led

POLL
        BRCLR PIFH, %00000001, CLEAR
        JSR PUSHBTN0
CLEAR
        BRCLR PIFH, %00000010, CLEAR1
        JSR PUSHBTN1
CLEAR1
        BRCLR PIFH, %00000100, CLEAR2
        JSR PUSHBTN2
CLEAR2
        BRCLR PIFH, %00001000, CLEAR3
        JSR PUSHBTN3
CLEAR3
```

```
        BRA  RESTRARTGRN              ; endless loop waiting for reset (and for interrupts)


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

;*****************************************************************


; ISR must test to see which button was pressed, because there ;is only one ISR for the two enabled buttons

PTHISR:  ; the interrupt service routine
        BRSET  PIFH, %00000001,PUSHBTN0  ; test btn0 IF flag
        BRSET  PIFH, %00000010,PUSHBTN1  ; test btn1 IF flag
        BRSET  PIFH, %00000100,PUSHBTN2  ; test btn2 IF flag
        BRSET  PIFH, %00001000,PUSHBTN3  ; test btn3 IF flag
; NOTE:  Flags are tested –not the switches
        JMP  DONE
;-------------------------------------------------------------------
;----------------INTERRUPTS
;-------------------------------------------------------------------
;*****************************************************************
PUSHBTN0:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all

        LDD #SW0PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt

        JSR ELECTION
        jsr    DELAY_1_S   ; delay for 1 second
        JMP    DONE

;*****************************************************************
PUSHBTN1:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        LDD #SW1PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
        JSR INCPORT    ;Call INCPORT subroutine
       JMP    DONE
;*****************************************************************
PUSHBTN2:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        bset   PTP,  RED   ; turn on RED

        LDD #SW2PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
        JSR DECPORT    ;Call DECPORT Subroutinr
        JMP    DONE
;*****************************************************************
PUSHBTN3:
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        LDD #SW3PROMPT    ;Load LCD Prompt
        JSR clear_lcd  ;Clear anything else off of LCD
        JSR lcd_prtstrg;Print LCD prompt
```

```
        JSR PRINT5MULT

DONE
        MOVB  #$0F, PIFH   ; clear Port H interrupt flags
        RTS
;---------------------------------------------------------------------
;----------------FUNCTIONS
;---------------------------------------------------------------------

Flash_Green_Led
        LDAA #10 ;This will flash the green LED 10 times
GREENLOOP
        bset   PTP,  GREEN   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        DECA   ;Decrement the count of GREENLOOP
        BNE GREENLOOP ;If it hasnt flashed 10 times then go to GREENLOOP
        RTS
;---------------------------------------------------------------------
PRINT5MULT
        LDAA #$1      ;Count for the table loop
TABLE_LOOP
        LDAB #$A      ;Load Newline
        JSR putchar   ;Print Newline
        LDAB #$0D
        JSR putchar
        LDAB #$05 ;load in input number
        JSR out2hex    ;print input number
        LDAB #$20      ;Load a space
        JSR putchar    ;print the space
        LDAB #$58      ;Load in X
        JSR putchar    ;Print X

        CMPA #$A       ;Check if the count has reached 10, if so we need to do a special Print_count to print 10
        BLO  PRINT_COUNT;If < 9 then skip this
        LDAB #$10      ;Load a $10
        JSR out2hex
        BRA AFTER_COUNT

PRINT_COUNT

        STAA COUNT_NUM ;Store into COUNT_NUM
        LDAB COUNT_NUM ;Pull from COUNT_NUM
        JSR out2hex    ;Print COUNT_NUM

AFTER_COUNT

        LDAB #$20      ;Load a space
        JSR putchar    ;print the space
        LDAB #$3D      ;Load an = sign
        JSR putchar    ;Print the = sign
        LDAB #$20      ;Load a space
        JSR putchar    ;print the space

        STAA COUNT_NUM ;Store current count for later
        LDAB INPUT_NUM ;load the input number
```

```
        ADDB #$D0     ;subtract 30 so that the value is one digit
        STAB MULT_NUM  ;store value for use in multiplication
        LDAB #$0
        LDAA #$1

MULT_LOOP
        ADDB MULT_NUM  ;add stored value to zero
        CMPA COUNT_NUM ;check if enough additions have happened
        INCA
        BLO  MULT_LOOP ;Add again if not finished


        JSR HEX2DEC    ;go to hex to decimal function

        LDAA COUNT_NUM ;reload the count number to do the next number on the table
        INCA           ;increment to the next number on the table
        CMPA #$B       ;check if we have reached 10 yet
        BLO TABLE_LOOP ;if not go back to TABLE_LOOP to finish the table

        RTS
;-------------------------------------------------------------------------
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
      JSR DELAY_1_S  ;Delay 1 second between lines
      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
        STAA PORTB
        BRCLR PTH,%10000000,L1   ;Check if first Dip switch is low
        BSET PORTB, %00001000    ;Set fifth portb led to high
L1
        BRCLR PTH,%01000000,L2   ;check if second dip switch is low
        BSET PORTB, %00000100    ;set sixth portb LED to high
L2
        BRCLR PTH,%00100000,L3   ;check if third dip switch is low
        BSET PORTB, %00000010    ;set seventh portb LED to high
L3
        BRCLR PTH,%00010000,L4   ;check if fourth dit switch is low
```

```
        BSET PORTB, %00000001    ;set eighth portb LED to high
L4
        RTS
;-------------------------------------------------------------------
DECPORT
        LDAA PORTB
        DECA
        CMPA #%11111111
        BNE NOTSET2
        LDAA #%00001111
NOTSET2
        STAA PORTB
        ;This is 5 blinks at 2 HZ
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  RED   ; turn on red
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        RTS
;-------------------------------------------------------------------
INCPORT
        LDAA PORTB
        INCA
        CMPA #%00010000
        BNE NOTSET
        LDAA #%00000000
NOTSET
        STAA PORTB
        ;This is 4 blinks at 1 HZ
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .25 second
        jsr    DELAY_250_MS   ; delay for .25 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .25 second
        jsr    DELAY_250_MS   ; delay for .25 second
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bset   PTP,  BLUE   ; turn on blue
```

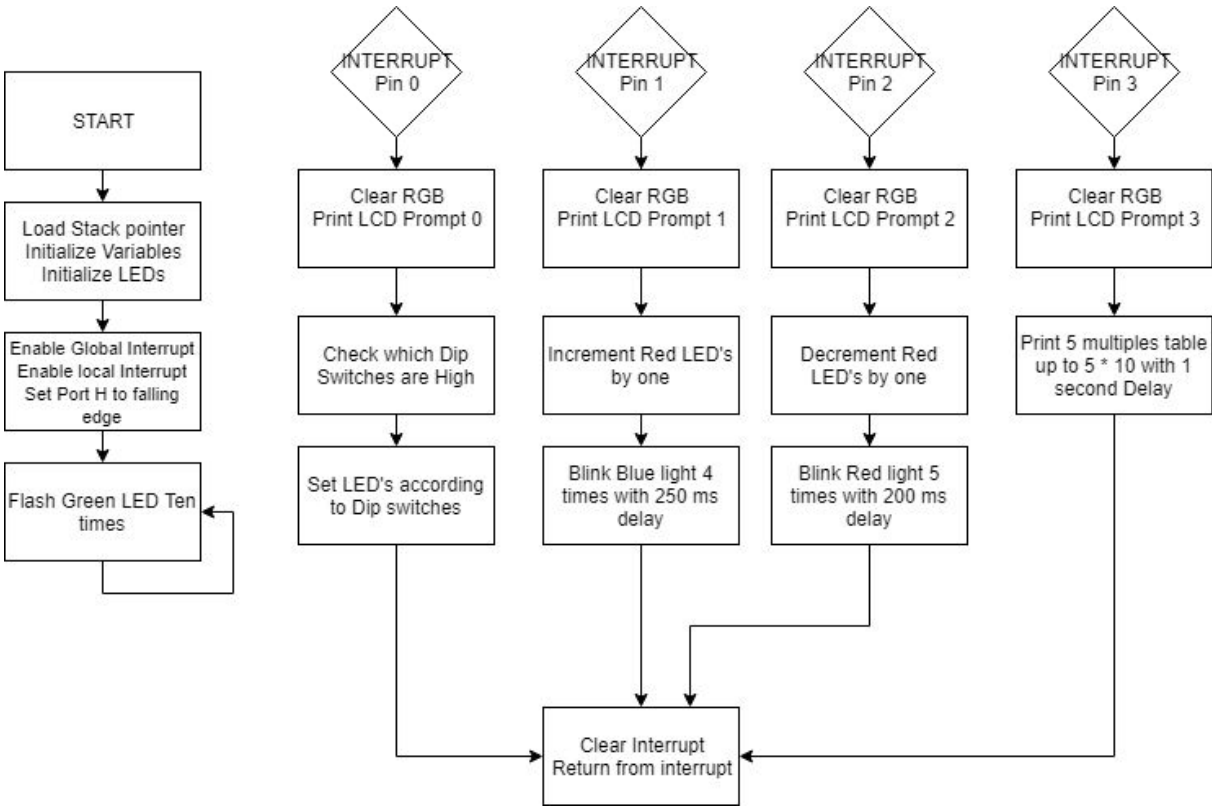
```
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bset   PTP,  BLUE   ; turn on blue
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second
        bclr   PTP,  RED+GREEN+BLUE  ; clear all
        jsr    DELAY_250_MS   ; delay for .2 second
        jsr    DELAY_250_MS   ; delay for .5 second

        RTS
 ;-------------------------------------------------------------------
;*<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 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 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 250 milliseconds>
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
;-------------------------------------------------------------------


;*************************************************************
```

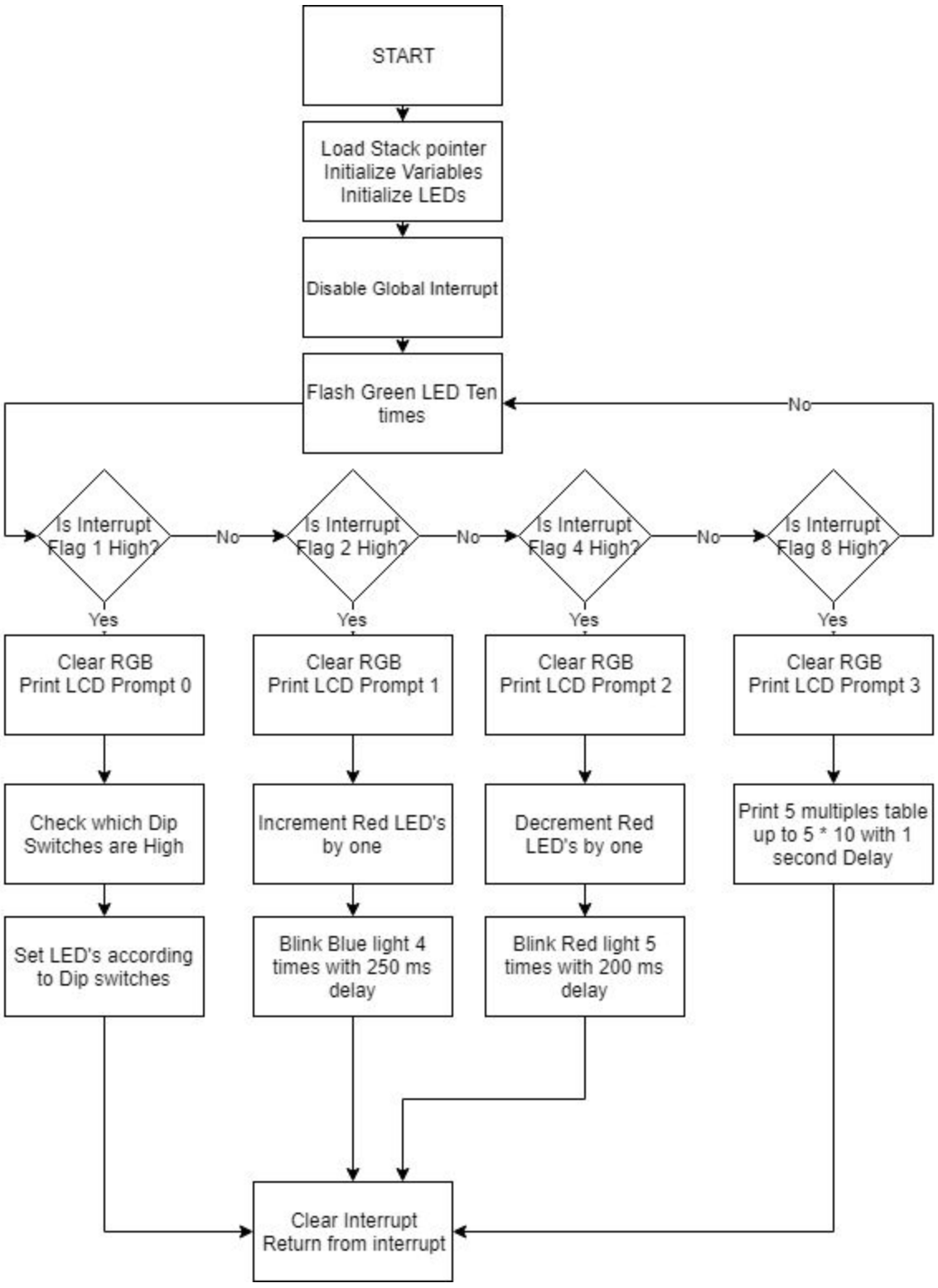
```
;*              Interrupt Vectors                       *
;*************************************************************
        ORG   Vreset
        DC.W  Entry         ; Reset Vector

;*************************************************************
        ORG     Vporth     ; setup  Port H interrupt Vector
        DC.W    PTHISR
```

Post-Lab

1. Question 1
    a. The push buttons can briefly stop the green led by calling an interrupt. They can do this by changing the interrupt flag status. In our program it is impossible to stop the green light permanently by holding the button down because the flag is set by falling edge logic.
2. Question 2
    a. The push buttons use falling edge logic. This means that when you push the push button the signal goes from 1 to 0. It is at the instant of this change that the interrupt flag bit is modified. Staying at 1 or at 0 will have no effect on the interrupt flag bit.
    b. BCLR PPSH, #$0F ; set Port H pins 0-1 for falling edge
    If you wanted to set it to rising edge you could use BSET instead of BCLR
3. Question 3
    a. If the interrupt flag bit is not cleared at the end of the interrupt then that interrupt will be called immediately after the interrupt ends, this will happen an infinite number of times.
4. Question 4
    a. It is impossible to set the interrupt flag bits directly using software, these bits are handled by hard wired logic.

5. Question 5


START
Load Stack pointer
Initialize Variables
Initialize LEDs
Enable Global Interrupt
Enable local Interrupt
Set Port H to falling edge
Flash Green LED Ten times
INTERRUPT Pin 0
Clear RGB
Print LCD Prompt 0
Check which Dip Switches are High
Set LED's according to Dip switches
INTERRUPT Pin 1
Clear RGB
Print LCD Prompt 1
Increment Red LED's by one
Blink Blue light 4 times with 250 ms delay
INTERRUPT Pin 2
Clear RGB
Print LCD Prompt 2
Decrement Red LED's by one
Blink Red light 5 times with 200 ms delay
INTERRUPT Pin 3
Clear RGB
Print LCD Prompt 3
Print 5 multiples table up to 5 * 10 with 1 second Delay
Clear Interrupt
Return from interrupt


START
Load Stack pointer
Initialize Variables
Initialize LEDs
Disable Global Interrupt
Flash Green LED Ten times
No
Is Interrupt Flag 1 High?
No
Is Interrupt Flag 2 High?
No
Is Interrupt Flag 4 High?
No
Is Interrupt Flag 8 High?
Yes
Yes
Yes
Yes
Clear RGB
Print LCD Prompt 0
Clear RGB
Print LCD Prompt 1
Clear RGB
Print LCD Prompt 2
Clear RGB
Print LCD Prompt 3
Check which Dip Switches are High
Increment Red LED's by one
Decrement Red LED's by one
Print 5 multiples table up to 5 * 10 with 1 second Delay
Set LED's according to Dip switches
Blink Blue light 4 times with 250 ms delay
Blink Red light 5 times with 200 ms delay
Clear Interrupt
Return from interrupt

Conclusion

- Pre-Lab
    - In the prelab we practiced blinking a blue light on the RGB and using our time delay functions to control the frequency of the blinks (PTH0). We also wrote a subroutine that would be used to increment the 4 rightmost bits on porth including an overflow scenario (PTH1).
- In-Lab
    - Part A (PTH 3)
        - We learned how to use interrupts to control the interruption (for lack of a better word) of a green LED and perform various subroutines from other labs.
    - Part B (PTH 2)
        - We learned how to “poll” the interrupt flag bits and use them as a determining factor on calling various subroutines.
- Post-Lab
    - The Postlab pointed out that our code had the interrupt flag triggering off of falling edge logic, as well as exploring what happens if you don't clear your interrupt bits.