ECE 3731: **Introduction to Microprocessors**

Lab 5: External Interrupts

Fall, 2017

Honor Code:

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

X __Hussein El-Souri__

Contents

Objective

- Have an ongoing event that happens (flashing green)
- Interrupt that event through external inputs
- Use assembly language and interrupt-driven approach
- Implement interrupt through both flags and regular interrupt handling services

Equipment Used

- Codewarrior
- HCS12 Microprocessor
- Notepad ++ for importing code with format
- https://www.lucidchart.com/ for easily drawing flowcharts

Flowchart

Number 1 Pre-Lab

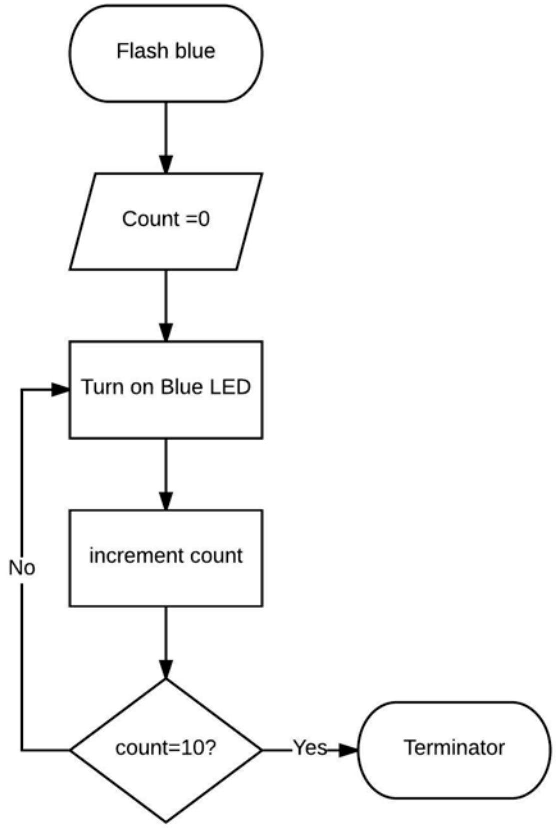


Figure 1: Part 1 pre-lab

Number 2 prelab

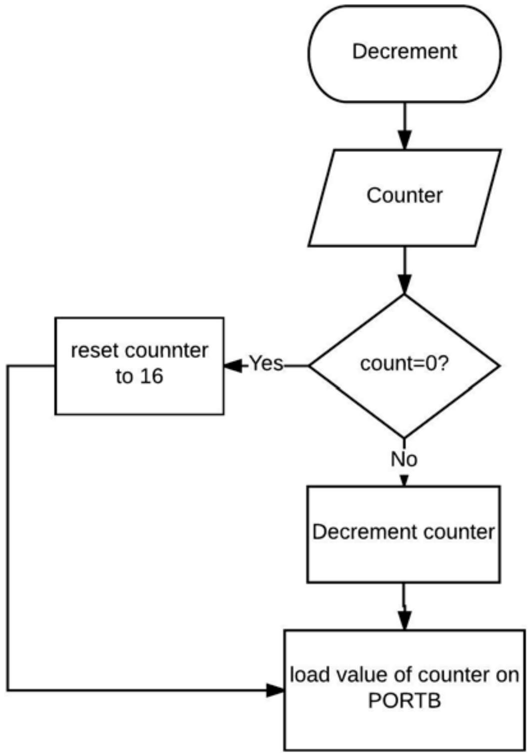
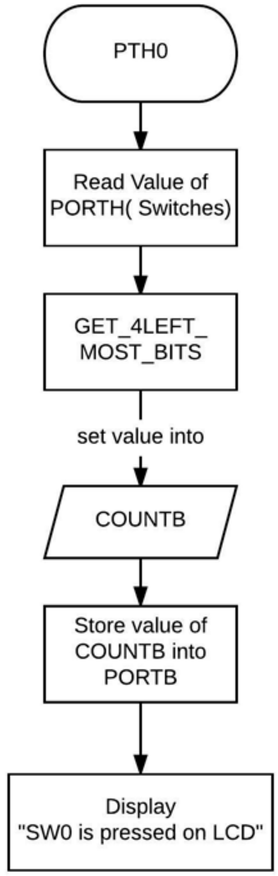


Figure 2: Part 2 pre-lab

PTH0: POST-LAB



Extra (get left most bits)

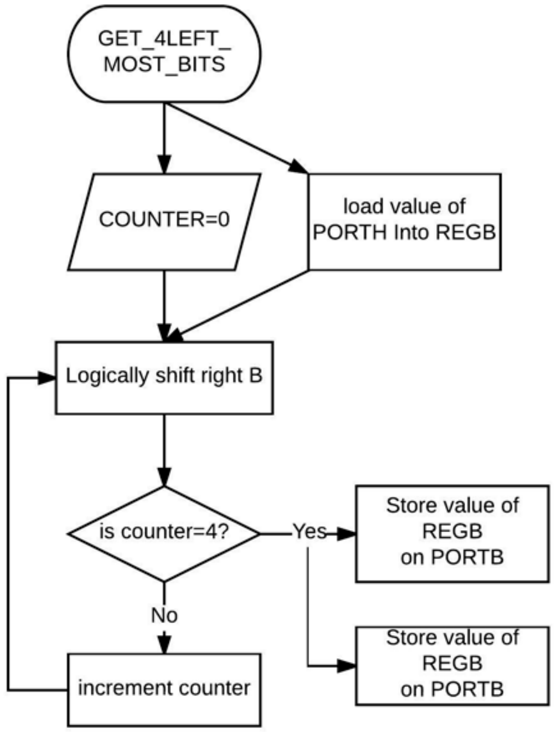


Figure 3: Get 4 left most bits

Procedure:

POST-LAB part 1

The main program loop is so that we flash the green LED 10 times this is done by following the approach presented in the flowchart Figure 1: Part 1 pre-lab. In this step it is important to reset the counter when we start or else the subroutine call will only work one time. Before starting this loop that is always running we must set bits 0, 1 and 2 on PORTH for falling edge and reset all the flags on those pins and enable interrupts on them. And then everything happens through an interrupt service subroutine.

If PUSHBTN0 is pressed the flag is set and the corresponding subroutine would execute. That subroutines reads the values of all the switches on PORTH and stores the value of the 4 left most bits into a counter that we called COUNTB and displays that value on PORTB and then the statement "SW0 is pressed" is displayed on the LCD. (check Figure 3: Get 4 left most bits for algorithm concerning getting 4 left most bits).

If PUSHBTN1 is pressed the value of COUNTB is incremented and displayed on PORTB. If the value of COUNTB reaches 16 and then incremented the value resets to zero. The statemen "SW1 is pressed" is displayed on the LCD the blue led is flashed twice. The Algorithm here is very similar to that in Figure 1: Part 1 pre-lab except the counter is incremented and reset when in reaches 16.

If PUSHBTN2 is pressed the value of COUNTB is decremented and displayed on PORTB. If the value of COUNTB reaches 0 and then decremented the value resets to 16. The statemen "SW2 is pressed" is displayed on the LCD. The red led is flashed twice. The Algorithm here is the one displayed in Figure 1: Part 1 pre-lab.

POST LAB part 2

All the subroutine and corresponding actions are the same the only difference is that interrupts aren't used the green LED flashed 10 times and then polling happens to which flag is set. The polling happens by checking every single flag.
Also in every subroutine the appropriate flag is reset for example if flag for bit 0 is set the subroutine for PUSHBTN0 is executed the flag is reset.

Code

Pre-Lab part 1

```
FLASH_BLUE
 JSR   clear_lcd     ;Clear the LCD
 CLR COUNT1      ;clear count 1
 bclr PTP, RED+GREEN+BLUE ; clear all
LOOP1
 bset PTP, BLUE ; turn on BLUE
 ldd #250 ; 250ms delay
 jsr ms_delay
 bclr PTP, RED+GREEN+BLUE ; clear all
 ldd #250 ; 250ms delay
 jsr ms_delay ; delay for 0.25 second
 ldaa COUNT1  ; load value of count1 into REG A
 inca         ;increment A
 STAA COUNT1  ;store result in count
 cmpa #10     ;compare to 10
 bne LOOP1    ; loop back if not equal
```

Pre-Lab part 2

```
        ldab COUNTB      ;load value of REG B into COUNB
        cmpb #$0              ; compare to 0
        BEQ RESET_COUNT1 ; branch if equal
        BNE DEC_COUNT         ; branch if not equal
DEC_COUNT
        DECB                 ;decrement balue of REGB
        STAB COUNTB        ;store contets of REGB in COUNTB
        stab PORTB              ;store onto PORTB
RESET_COUNT1
        LDAB #$F
        STAB COUNTB    ;store contets of REGB in COUNTB
        STAB PORTB     ;store onto PORTB
```

POST-LAB part 1

```
COUNT1 dc.b 0
COUNT2 dc.b 0
COUNT3 dc.b 0
COUNT4 dc.b 0

COUNTB dc.b 0

MSG1        FCC  "SW0 is pressed"
             dc.b  0
MSG2        FCC  "SW1 is pressed"
             dc.b 0
MSG3        FCC  "SW2 is pressed"
             dc.b  0

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

;---------------------------------------------------
; Code Section
;---------------------------------------------------
            ORG   ROMStart  ; loc $2000
Entry:
_Startup:
            ; remap the RAM &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
; note Port H is all inputs after reset
          jsr   lcd_init    ; initialize LCD (must be done first)
          BCLR    PPSH, #$07  ; set Port H pins 0-1-2 for falling edge
          MOVB   #$07, PIFH    ; clear interrupt flags initially
          BSET     PIEH,  #$07     ; enable interrupts on Port H  pins 0-1-2
          CLI                      ; enable interrupts
Flash_Green_Led
          JSR FLASH_GREEN
          BRA Flash_Green_Led

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

;==================================================================

;************FUNCTIONS

FLASH_GREEN
 JSR   clear_lcd     ;Clear the LCD
 CLR COUNT1     ;clear count 1
 bclr PTP, RED+GREEN+BLUE ; clear all
```

```
LOOP1
 bset PTP, GREEN ; turn on GREEN
 ldd #250 ; 250ms delay
 jsr ms_delay ; delay for 0.25 second
 bclr PTP, RED+GREEN+BLUE ; clear all
 ldd #250 ; 250ms delay
 jsr ms_delay ; delay for 0.25 second
 ldaa COUNT1  ; load value of count1 into REG A
 inca         ;increment A
 STAA COUNT1  ;store result in count
 cmpa #10     ;compare to 10
 bne LOOP1    ; loop back if not equal
 RTS
FLASH_BLUE
              CLR COUNT3    ;clear count3
              bclr PTP, RED+GREEN+BLUE ; clear all
LOOP2
              bset PTP, BLUE ; turn on blue
              ldd #500 ; 500ms delay
              jsr ms_delay ; delay for 0.25 second
              bclr PTP, RED+GREEN+BLUE ; clear all
              ldd #200 ; 200ms delay
              jsr ms_delay
              ldaa COUNT3 ;load value of count3 into reg A
              inca            ;inc value reg A
              STAA COUNT3   ;store baxk in count3
              cmpa #2
              BNE LOOP2
              LBRA DONE

FLASH_RED
              CLR COUNT4
              bclr PTP, RED+GREEN+BLUE ; clear all
LOOP3
              bset PTP, RED ; turn on RED
              ldd #500 ; 500ms delay
              jsr ms_delay ; delay for 0.25 second
              bclr PTP, RED+GREEN+BLUE ; clear all
              ldd #200 ; 200ms delay
              jsr ms_delay
              ldaa COUNT4
              inca
              STAA COUNT4
              cmpa #2
              BNE LOOP3
              LBRA DONE
; 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
; NOTE:  Flags are tested -not the switches
         lBRA  DONE
PUSHBTN0:
          bclr PTP, RED+GREEN+BLUE ; clear all
          JSR   clear_lcd     ;Clear the LCD
          CLR COUNT2          ;clear count 2
          ldab PTH
GET_BITS
          ldaa COUNT2         ;loads value of count2
          LSRB                ;shifts content of B to the right
          inca
          staa COUNT2
          cmpa #4             ; shifts 4 times tpo get 4 MSB
          BNE GET_BITS
          STAB PORTB          ; stores value of B on PORTB
          STAB COUNTB         ; stores Value of reg B into a count specific for portB
          ldab  #$0           ; set print position to top line
          jsr   set_lcd_addr  ;      "
```

```
          ldd   #MSG1       ; D is pointer to string
          jsr lcd_prtstrg;    ; print first string
          BRA   DONE
PUSHBTN1:
        bclr PTP, RED+GREEN+BLUE ; clear all
        JSR   clear_lcd     ;Clear the LCD
        ldab COUNTB
        cmpb #$F
        BEQ RESET_COUNT
        BMI INC_COUNT
INC_COUNT
        incb
        STAB COUNTB
        stab PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr
        ldd   #MSG2       ; D is pointer to string
        jsr lcd_prtstrg   ; print first string
        LBRA FLASH_BLUE
RESET_COUNT
        CLR COUNTB
        ldab COUNTB
        STAB PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr  ;       "
        ldd   #MSG2       ; D is pointer to string
        jsr lcd_prtstrg;    ; print first string
        LBRA FLASH_BLUE
PUSHBTN2
        bclr PTP, RED+GREEN+BLUE ; clear all
        JSR   clear_lcd     ;Clear the LCD
        ldab COUNTB
        cmpb #$0
        BEQ RESET_COUNT1
        BNE DEC_COUNT
DEC_COUNT
        DECB
        STAB COUNTB
        stab PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr
        ldd   #MSG3       ; D is pointer to string
        jsr lcd_prtstrg   ; print first string
        LBRA FLASH_RED
RESET_COUNT1
        LDAB #$F
        STAB COUNTB
        STAB PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr  ;       "
        ldd   #MSG3       ; D is pointer to string
        jsr lcd_prtstrg;    ; print first string
        LBRA FLASH_RED
DONE
         LDD #1000
         jsr ms_delay
         MOVB  #$07, PIFH   ; clear Port H interrupt flags
         RTI
```

POST LAB part 2

```
COUNT1 dc.b 0
COUNT2 dc.b 0
COUNT3 dc.b 0
COUNT4 dc.b 0
COUNT5 DC.B 0

COUNTB dc.b 0

MSG1       FCC  "SW0 is pressed"
            dc.b  0
```

```
MSG2        FCC  "SW1 is pressed"
             dc.b 0
MSG3        FCC  "SW2 is pressed"
             dc.b  0

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

;-----------------------------------------------------
; Code Section
;-----------------------------------------------------
            ORG    ROMStart  ; loc $2000
Entry:
_Startup:
            ; remap the RAM &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
; note Port H is all inputs after reset
          jsr    lcd_init     ; initialize LCD (must be done first)
          BCLR    PPSH, #$07   ; set Port H pins 0-1-2 for falling edge
Flash_Green_Led
          JSR FLASH_GREEN
          BRA Flash_Green_Led

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

;=====================================================================

;************FUNCTIONS

FLASH_GREEN
 JSR    clear_lcd     ;Clear the LCD
 CLR COUNT1     ;clear count 1
 CLR COUNT5
 bclr PTP, RED+GREEN+BLUE ; clear all
LOOP1
 bset PTP, GREEN ; turn on GREEN
 ldd #250 ; 250ms delay
 jsr ms_delay ; delay for 0.25 second
 bclr PTP, RED+GREEN+BLUE ; clear all
 ldd #250 ; 250ms delay
 jsr ms_delay ; delay for 0.25 second
 ldaa COUNT1  ; load value of count1 into REG A
 inca         ;increment A
 STAA COUNT1  ;store result in count
 cmpa #10     ;compare to 10
 bne LOOP1    ; loop back if not equal
LOOP_HERE
 LDAA COUNT5
 jsr     PTHISR
```

```
 CMPA #3
 bne LOOP_HERE
 INCA
 STAA COUNT5
 RTS
FLASH_BLUE
              CLR COUNT3    ;clear count3
              bclr PTP, RED+GREEN+BLUE ; clear all
LOOP2
              bset PTP, BLUE ; turn on blue
              ldd #500 ; 500ms delay
              jsr ms_delay ; delay for 0.25 second
              bclr PTP, RED+GREEN+BLUE ; clear all
              ldd #200 ; 200ms delay
              jsr ms_delay
              ldaa COUNT3 ;load value of count3 into reg A
              inca            ;inc value reg A
              STAA COUNT3   ;store baxk in count3
              cmpa #2
              BNE LOOP2
              rts
FLASH_RED
              CLR COUNT4
              bclr PTP, RED+GREEN+BLUE ; clear all
LOOP3
              bset PTP, RED ; turn on RED
              ldd #500 ; 500ms delay
              jsr ms_delay ; delay for 0.25 second
              bclr PTP, RED+GREEN+BLUE ; clear all
              ldd #200 ; 200ms delay
              jsr ms_delay
              ldaa COUNT4
              inca
              STAA COUNT4
              cmpa #2
              BNE LOOP3
              rts
; 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
; NOTE:  Flags are tested -not the switches
         LBRA Flash_Green_Led
PUSHBTN0:
          bclr PTP, RED+GREEN+BLUE ; clear all
          JSR   clear_lcd      ;Clear the LCD
          CLR COUNT2           ;clear count 2
          ldab PTH
GET_BITS
          ldaa COUNT2          ;loads value of count2
          LSRB                 ;shifts content of B to the right
          inca
          staa COUNT2
          cmpa #4              ; shifts 4 times tpo get 4 MSB
          BNE GET_BITS
          STAB PORTB           ; stores value of B on PORTB
          STAB COUNTB          ; stores Value of reg B into a count specific for portB
          ldab  #$0            ; set print position to top line
          jsr   set_lcd_addr   ;      "
          ldd   #MSG1        ; D is pointer to string
          jsr lcd_prtstrg;    ; print first string
          MOVB  #$01, PIFH     ; CLEAR FLAG FOR BIT 1
          rts
```

```
PUSHBTN1:
        bclr PTP, RED+GREEN+BLUE ; clear all
        JSR   clear_lcd      ;Clear the LCD
        ldab COUNTB
        cmpb #$F
        BEQ RESET_COUNT
        BMI INC_COUNT
INC_COUNT
        incb
        STAB COUNTB
        stab PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr
        ldd   #MSG2        ; D is pointer to string
        jsr lcd_prtstrg    ; print first string
        MOVB  #$02, PIFH    ; CLEAR FLAG FOR BIT 2
        LBRA FLASH_BLUE
RESET_COUNT
        CLR COUNTB
        ldab COUNTB
        STAB PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr  ;      "
        ldd   #MSG2        ; D is pointer to string
        jsr lcd_prtstrg;    ; print first string
        MOVB  #$02, PIFH    ; CLEAR FLAG FOR BIT 2
        LBRA FLASH_BLUE
PUSHBTN2
        bclr PTP, RED+GREEN+BLUE ; clear all
        JSR   clear_lcd      ;Clear the LCD
        ldab COUNTB
        cmpb #$0
        BEQ RESET_COUNT1
        BNE DEC_COUNT
DEC_COUNT
        DECB
        STAB COUNTB
        stab PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr
        ldd   #MSG3        ; D is pointer to string
        jsr lcd_prtstrg    ; print first string
        MOVB  #$04, PIFH    ; CLEAR FLAG FOR BIT 3
        LBRA FLASH_RED
RESET_COUNT1
        LDAB #$F
        STAB COUNTB
        STAB PORTB
        ldab  #$0           ; set print position to top line
        jsr   set_lcd_addr  ;      "
        ldd   #MSG3        ; D is pointer to string
        jsr lcd_prtstrg;    ; print first string
        MOVB  #$04, PIFH    ; CLEAR FLAG FOR BIT 3
        LBRA FLASH_RED
```

Conclusion

We learned the difference between polling and using interrupts and the advantages of using interrupts since polling is time consuming. In the case of cars that drive themselves polling would result in many accidents as the system would have to poll all through a process and then press the brakes if an object is detected in front of a car. Whereas if interrupts are used brakes are pressed instantly.