ECE 375 LAB 6

External Interrupts

Lab Time: Tuesday 4-6

Alexander Uong

INTRODUCTION

The purpose of this lab is to understand how to use interrupts and implementing them into an assembly program. This lab resembles previous labs we've encountered such as lab 1 and lab 4, where we worked with the BumpBot program, as well as displaying contents on the LCD display and modifying the contents. In the case of Lab 6, the BumpBot program will be implemented using interrupts rather than polling, with two counters displayed on the LCD keeping track of the number of times the right and left whisker have been pressed. The program will also add functionality to pins 2 and 3 on the board, with pressing them resetting the respective right and left whisker counters.

PROGRAM OVERVIEW

In this program, we were asked to implement the basic BumpBot program using interrupts, while keeping track of the amount of times the right and left whisker have been pressed, displaying the two counters on the LCD display found on the AVR board. As stated in the introduction, added functionality will be added to the program, with the pressing of pins 2 and 3 on the AVR board resetting the right and left whisker counters respectively, displaying a value of 0 on the LCD.

Besides the INIT and MAIN routine, several subroutines are used, including HitRight, HitLeft, WaitFunc, UpdateRight, UpdateLeft, ClrRight, and ClrLeft. The HitRight and HitLeft routines provide functionality for handling a Right or Left whisker hit, respectively. The Wait routine was created to provide an accurate busy wait, allowing time for the TekBot to backup and turn. UpdateRight updates the right whisker counter and displays the new value of the counter on the LCD Display when the right whisker is pressed. UpdateLeft is identical to UpdateRight, except it updates the left whisker counter and displays the new value of the counter on the LCD Display when the left whisker is pressed. ClrRight resets the right whisker counter to zero, updating the value on the LCD Display. ClrLeft resets the left whisker counter to zero, updating the value on the LCD Display.

INITIALIZATION ROUTINE

Within the initialization routine, the stack pointer is initialized, PORT B is initialized for output, PORT D is initialized for input, external interrupts are initialized through writing values into EICRA and EIMSK registers, the LCD display is initialized, and the left and right whisker counters are initialized to display on the LCD display.

MAIN ROUTINE

The main routine is simple, loading MovFwd into mpr, which is declared in the internal register definitions and constants section. MPR is then outputted into PORTB, which turns on the LED's of pins 5 and 6 on the board. Rcall LCDWrite is a routine from the LCDDriver, writing the contents of the counters onto the LCD display.

SUBROUTINES

1. HITRIGHT ROUTINE

HitRight was implemented from Lab1, where the Tekbot moves backwards for 1 second, turns left for 1 second, and then moves forward again. The WaitFunc routine is called to allow the different functionalities to last a second each. The right whisker counter is incremented each time this function is called to keep track of the amount of times the right whisker is pressed on the board. The flag register is also cleared to prevent interrupts from queuing.

2. HITLEFT ROUTINE

HitLeft was implemented from Lab1, where the Tekbot moves backwards for 1 second, turns right for 1 second, and then moves forward again. The WaitFunc routine is called to allow the different functionalities to last a second each. The left whisker counter is incremented each time this function is called to keep track of the amount of times the left whisker is pressed on the board. The flag register is also cleared to prevent interrupts from queuing.

3. WAITFUNC ROUTINE

WaitFunc was implemented from Lab1, where a triple-nested loop is created that leads to 16+159975*wait cycles executed. This is called in HitRight and HitLeft, allowing us to load in 100 10ms intervals to wait for 1 second.

4. UPDATERIGHT ROUTINE

UpdateRight clears line 1 of the LCD display and copies the right whisker counter register into mpr. The X register is set to point at the address of line 1 of the LCD display and mpr is stored into the X register. The LCDDriver routine Bin2ASCII is then called to convert the value of the counter into an ASCII text string equivalent.

5. UPDATELEFT ROUTINE

UpdateLeft is nearly identical to UpdateRight, except it clears line 2 of the LCD display and copies the left whisker counter register into mpr. The X register is then set to point at the address of line 2 of the LCD display and stores mpr into the X register. Bin2ASCII is then called.

6. CLRRIGHT ROUTINE

ClrRight clears line 1 of the LCD display, mpr, and the right whisker counter. It then sets the X register to point at the address of line 1 of the LCD display and stores the cleared mpr register into the X register. Bin2ASCII is then called to display the newly cleared value onto the LCD display.

7. CLRLEFT ROUTINE

ClrLeft is nearly identical to ClrRight, clearing line 2 of the LCD display, mpr, and the left whisker counter instead. The X register points at the address of line 2 of the LCD display instead of line 1. Other functionalities are identical to ClrRight.

ADDITIONAL QUESTIONS

1) As this lab, Lab 1, and Lab 2 have demonstrated, there are always multiple ways to accomplish the same task when programming (this is especially true for assembly programming). As an engineer, you will need to be able to justify your design choices. You have now seen the BumpBot behavior implemented using two different programming languages (AVR assembly and C), and also using two different methods of receiving external input (polling and interrupts). Explain the benefits and costs of each of these approaches. Some important areas of interest include, but are not limited to: efficiency, speed, cost of context switching, programming time, understandability, etc.

AVR assembly and C have their different advantages and disadvantages. Assembly language can be more efficient, taking up less memory. Programming in C is easier to understand in my opinion, leading to faster programming time. C is also often faster than assembly language as well. In regards to polling and interrupts, implementing the BumpBot program using polling is inefficient compared to interrupts, as it scans for external inputs by using a loop. With polling, it will continuously check for inputs, abandoning other works. However, with interrupts, the CPU is able to focus on other tasks and only changes its attention if an interrupt notifies the CPU. Once it has finished serving the interrupt, it can continue doing what it was doing prior.

2) Instead of using the Wait function that was provided in BasicBumpBot.asm, is it possible to use a timer/counter interrupt to perform the one-second delays that are a part of the BumpBot behavior, while still using external interrupts for the bumpers? Give a reasonable argument either way, and be sure to mention if interrupt priority had any effect on your answer.

Yes, instead of the Wait function, a timer/counter interrupt can be used to perform the one-second delays that are a part of the BumpBot behavior. Timers/counters are used for measuring some elapsed time and can generate a specified delay for each cycle. Timer/Counter 0 is a lower priority interrupt compared to other interrupts such as each of the external interrupts, so it's important the interrupt doesn't impact other interrupts such as the external interrupts.

CONCLUSION

In conclusion, this lab was a nice refresher in regards to the previous lab work we've done. I liked how we were able to implement numerous things we've learned and combine them together in one lab. Implementing interrupts in a program we've been exposed to before allowed me to focus on the implementation of interrupts while gaining experience with manipulating the LCD on the AVR board.

SOURCE CODE

```
;***********************************************************
;*
;*     Alexander_Uong_Lab6_sourcecode.asm
;*
;*     BumpBot program supporting external interrupts. Added functionality includes
;*     displaying on LCD the amount of times rightwhisker and left whisker are pressed
;*     and adding functionality to pins 2 and 3, clearing the right and left whisker
respectively
;*     This is the skeleton file for Lab 6 of ECE 375
;*
;***********************************************************
;*
;*      Author: Alexander Uong
;*        Date: 2/21/2021
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*     Internal Register Definitions and Constants
;***********************************************************
.def   mpr = r16                             ; Multipurpose register
.def   ilcnt = r23                           ; Inner Loop Counter
.def   olcnt = r24                           ; Outer Loop Counter

.def   rightwhiskercounter = r5
.def   leftwhiskercounter =  r6
.def   botregister = r25             ;register used in bumpbot functions

.equ   WTime = 100                           ; Time to wait in wait loop

.equ   WskrR = 0                             ; Right Whisker Input Bit
.equ   WskrL = 1                             ; Left Whisker Input Bit

//.equ EngEnR = 4                            ; Right Engine Enable Bit
//.equ EngEnL = 7                            ; Left Engine Enable Bit
.equ   EngDirR = 5                           ; Right Engine Direction Bit
.equ   EngDirL = 6                           ; Left Engine Direction Bit

.equ   MovFwd = (1<<EngDirR|1<<EngDirL)      ; Move Forward Command
.equ   MovBck = $00                          ; Move Backward Command
.equ   TurnR = (1<<EngDirL)                  ; Turn Right Command
.equ   TurnL = (1<<EngDirR)                  ; Turn Left Command
//.equ Halt = (1<<EngEnR|1<<EngEnL)          ; Halt Command
;***********************************************************
;*     Start of Code Segment
;***********************************************************
.cseg                                               ; Beginning of code segment

;***********************************************************
;*     Interrupt Vectors
;***********************************************************
.org   $0000                                 ; Beginning of IVs
               rjmp   INIT                   ; Reset interrupt

.org   $0002                                 ;if pin 0 is pressed
               rcall HitRight
               reti

.org   $0004                                 ;if pin 1 is pressed
               rcall HitLeft
               reti
```

```
.org    $0006                               ;if pin 2 is pressed
               rcall ClrRight
               reti

.org    $0008                               ;if pin 3 is pressed
               rcall ClrLeft
               reti

.org    $0046                               ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:                                               ; The initialization routine
               ; Initialize Stack Pointer
               LDI mpr, LOW(RAMEND)   ; Low Byte of End SRAM Address
               OUT SPL, mpr                 ; Write Byte to SPL
               LDI mpr, HIGH(RAMEND)  ; High Byte of End SRAM Address
               OUT SPH, mpr                 ; Write Byte to SPH

               ; Initialize Port B for output
               ldi mpr, $FF   ;Set port b data direction register
               out DDRB, mpr  ;set for output
               ldi mpr, $00   ;initialize port b data register
               out PORTB, mpr ;all port b outputs are now low

               ; Initialize Port D for input
               ldi mpr, $00   ;set port d data direction register
               out DDRD, mpr  ;set for input
               ldi mpr, $FF   ;initialize port d data register
               out PORTD, mpr ;all port d inputs are tri-state

               ; Initialize external interrupts
               ; Set the Interrupt Sense Control to falling edge
               ldi mpr,0b10101010
               sts EICRA, mpr


               ; Configure the External Interrupt Mask
               ldi mpr, 0b00001111
               out EIMSK, mpr

               ;Initialize LCD Display
               rcall LCDInit

               ;Initialize leftwhiskercounter to 0 and display on first line of LCD
               clr rightwhiskercounter
               mov mpr, rightwhiskercounter          ;copy rightwhiskercounter into mpr

               ldi XL, LOW(LCDLn1Addr)                              ;X register points to address
of line1 of the LCD
               ldi XH, HIGH(LCDLn1Addr)

               st X, mpr                                                    ;store mpr into x
register
               rcall Bin2ASCII                                              ;Bin2ASCII converts
binary number of ascii equivalent


               ;Initialize rightwhiskercounter to 0 and display on second line of LCD
               clr    leftwhiskercounter
               mov mpr, leftwhiskercounter           ;copy leftwhiskercounter into mpr

               ldi XL, LOW(LCDLn2Addr)                      ;X register points to address of
line2 of the LCD
               Ldi XH, High(LCDLn2Addr)


               st X, mpr                                            ;store mpr into x register
               rcall Bin2ASCII                                      ;Bin2ASCII converts binary
number of ascii equivalent
```

```
               ; Turn on interrupts
                      ; NOTE: This must be the last thing to do in the INIT function
               sei

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:                                               ; The Main program

               ; TODO: ???
               ldi botregister, MovFwd              ;load the MoveFwd functionality into the bot
register
               out PORTB, botregister       ;store this into output PORTB
               rcall LCDWrite                       ;LCD driver that writes contents stored onto
LCD Display

               rjmp    MAIN                 ; Create an infinite while loop to signify the
                                                            ; end of the program.

;***********************************************************
;*      Functions and Subroutines
;***********************************************************

;-----------------------------------------------------------
;       You will probably want several functions, one to handle the
;       left whisker interrupt, one to handle the right whisker
;       interrupt, and maybe a wait function
;------------------------------------------------------------

;----------------------------------------------------------------
; Sub:  HitRight
; Desc: Handles functionality of the TekBot when the right whisker
;               is triggered.
;----------------------------------------------------------------

HitRight:
               push    botregister                  ; Save mpr register
               push    wait                 ; Save wait register
               in              botregister, SREG    ; Save program state
               push    botregister                  ;

               ; Move Backwards for a second
               ldi             botregister, MovBck  ; Load Move Backward command
               out             PORTB, botregister   ; Send command to port
               ldi             wait, WTime  ; Wait for 1 second
               rcall   WaitFunc                     ; Call wait function

               ; Turn left for a second
               ldi             botregister, TurnL   ; Load Turn Left Command
               out             PORTB, botregister   ; Send command to port
               ldi             wait, WTime  ; Wait for 1 second
               rcall   WaitFunc                     ; Call wait function

               inc rightwhiskercounter              ;increments the rightwhiskercounter
               rcall UpdateRight                    ;call to updateright function below

               ;clear flag register
               ldi botregister, 0b00001111          ;Setting the respective bit places to 1
resets the flag register contents
               out EIFR, botregister                ;this is to avoid queued interrupts

               pop             botregister          ; Restore program state
               out             SREG, botregister    ;
               pop             wait         ; Restore wait register
               pop             botregister          ; Restore mpr
               ret                          ; Return from subroutine

;----------------------------------------------------------------
; Sub:  HitLeft
; Desc: Handles functionality of the TekBot when the left whisker
;              is triggered.
;----------------------------------------------------------------
HitLeft:
               push   botregister                    ; Save mpr register
               push   wait                   ; Save wait register
               in             botregister, SREG     ; Save program state
               push   botregister                    ;

               ; Move Backwards for a second
               ldi            botregister, MovBck   ; Load Move Backward command
               out            PORTB, botregister    ; Send command to port
               ldi            wait, WTime    ; Wait for 1 second
               rcall  WaitFunc                       ; Call wait function

               ; Turn right for a second
               ldi            botregister, TurnR    ; Load Turn Left Command
               out            PORTB, botregister    ; Send command to port
               ldi            wait, WTime    ; Wait for 1 second
               rcall  WaitFunc                       ; Call wait function

               inc leftwhiskercounter        ;incrmeents leftwhiskercounter
               rcall UpdateLeft                     ;call to updateleft function below

               ;clear flag register
               ldi botregister, 0b00001111          ;Setting the respective bit places to 1
resets the flag register contents
               out EIFR, botregister                ;this is to avoid queued interrupts


               pop            botregister           ; Restore program state
               out            SREG, botregister     ;
               pop            wait           ; Restore wait register
               pop            botregister           ; Restore mpr
               ret                           ; Return from subroutine

;----------------------------------------------------------------
; Sub:  WaitFunc
; Desc: A wait loop that is 16 + 159975*waitcnt cycles or roughly
;              waitcnt*10ms.  Just initialize wait for the specific amount
;              of time in 10ms intervals. Here is the general eqaution
;              for the number of clock cycles in the wait loop:
;                      ((3 * ilcnt + 3) * olcnt + 3) * waitcnt + 13 + call
;----------------------------------------------------------------
WaitFunc:
               push   wait                   ; Save wait register
               push   ilcnt                  ; Save ilcnt register
               push   olcnt                  ; Save olcnt register

Loop:  ldi            olcnt, 224             ; load olcnt register
OLoop: ldi            ilcnt, 237             ; load ilcnt register
ILoop: dec            ilcnt                  ; decrement ilcnt
               brne   ILoop                  ; Continue Inner Loop
               dec            olcnt          ; decrement olcnt
               brne   OLoop                  ; Continue Outer Loop
               dec            wait           ; Decrement wait
               brne   Loop                   ; Continue Wait loop

               pop            olcnt          ; Restore olcnt register
               pop            ilcnt          ; Restore ilcnt register
               pop            wait           ; Restore wait register
               ret                           ; Return from subroutine

;----------------------------------------------------------------
; Sub:  UpdateRight
; Desc: Handles functionality of the TekBot when the right whisker
;              is triggered.
;----------------------------------------------------------------
UpdateRight:
       rcall LCDClrLn1               ;LCD driver clear line 1 of the display

       clr mpr                                              ;clear mpr register
```

```
        mov mpr, rightwhiskercounter  ;copy rightwhiskercounter into mpr register


        ldi XL, LOW(LCDLn1Addr)                      ;Set the X register to point to address of
line1 of LCD display
        ldi XH, HIGH(LCDLn1Addr)

        st X, mpr                                           ;store mpr into x register
        rcall Bin2ASCII                                     ;call to Bin2ASCII

        ret
;-----------------------------------------------------------------
; Sub:  UpdateLeft
; Desc: Handles functionality of the TekBot when the right whisker
;               is triggered.
;-----------------------------------------------------------------
UpdateLeft:
        rcall LCDClrLn2                                     ;LCD driver clear line 2 of the
display

        clr mpr                                             ;clear mpr register
        mov mpr, leftwhiskercounter          ;copy rightwhiskercounter into mpr register


        ldi XL, LOW(LCDLn2Addr)                      ;Set the X register to point to address of
line2 of LCD display
        ldi XH, HIGH(LCDLn2Addr)

        st X, mpr                                           ;store mpr into x register
        rcall Bin2ASCII                                     ;call to Bin2ASCII

        ret

;-----------------------------------------------------------------
; Sub:  ClrRight
; Desc: Handles functionality of the TekBot when the right whisker
;               is triggered.
;-----------------------------------------------------------------
ClrRight:
        push mpr                                            ;push mpr onto stack
        rcall LCDClrLn1                                     ;LCD driver clear line 1 of the
display

        clr rightwhiskercounter                      ;clear rightwhiskercounter
        clr mpr                                             ;clear mpr

        ldi XL, LOW(LCDLn1Addr)                      ;Set the X register to point to address of
line1 of LCD display
        ldi XH, HIGH(LCDLn1Addr)

        st X, mpr                                           ;store mpr into x register
        rcall Bin2ASCII                                     ;call to Bin2ASCII

        ldi mpr, 0b00001111           ;Setting the respective bit places to 1 resets the flag
register contents
        out EIFR, mpr                 ;this is to avoid queued interrupts
        pop mpr                              ;pop mpr to restore value

        ret


;-----------------------------------------------------------------
; Sub:  ClrLeft
; Desc: Handles functionality of the TekBot when the right whisker
;               is triggered.
;-----------------------------------------------------------------
ClrLeft:
        push mpr                                            ;push mpr onto stack
        rcall LCDClrLn2                                     ;LCD driver clear line 2 of the
display
```

```
        clr leftwhiskercounter                ;clear leftwhiskercounter
        clr mpr                                              ;clear mpr

        ldi XL, LOW(LCDLn2Addr)                     ;Set the X register to point to address of
line2 of LCD display
        ldi XH, HIGH(LCDLn2Addr)

        st X, mpr                                            ;store mpr into x register
        rcall Bin2ASCII                                      ;call to Bin2ASCII


        ldi mpr, 0b00001111           ;Setting the respective bit places to 1 resets the flag
register contents
        out EIFR, mpr                 ;this is to avoid queued interrupts
        pop mpr                               ;pop mpr to restore value


        ret

;***********************************************************
;*      Stored Program Data
;***********************************************************

; Enter any stored data you might need here

;***********************************************************
;*      Additional Program Includes
;***********************************************************
.include "LCDDriver.asm"              ; Include the LCD Driver
```